hochschule mannheim

Entwicklung einer Progressive Web App zum Herunterladen, Verwalten und Abspielen von Audio-Medien zur Offlinenutzung mit Angular

Martin Schalter

Bachelor-Thesis

zur Erlangung des akademischen Grades Bachelor of Science (B.Sc.)

Studiengang Informatik

Fakultät für Informatik

Hochschule Mannheim

19.08.2020

Betreuer

Prof. Dr. Thomas Specht, Hochschule Mannheim

Christian Perian, biblepool gUG

Schalter, Martin:
Entwicklung einer Progressive Web App zum Herunterladen, Verwalten und Abspielen von Audio-Medien zur Offlinenutzung mit Angular / Martin Schalter. –
Bachelor-Thesis, Mannheim: Hochschule Mannheim, 2020. 17 Seiten.

Schalter, Martin:
Developement of a progressive web app to download, manage and play audio files for offline use with Angular / Martin Schalter. –
Bachelor Thesis, Mannheim: University of Applied Sciences Mannheim, 2020. 17 pages.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich bin damit einverstanden, dass meine Arbeit veröffentlicht wird, d. h. dass die Arbeit elektronisch gespeichert, in andere Formate konvertiert, auf den Servern der Hochschule Mannheim öffentlich zugänglich gemacht und über das Internet verbreitet werden darf.

Mannheim, 19.08.2020

Martin Schalter

Abstract

Entwicklung einer Progressive Web App zum Herunterladen, Verwalten und Abspielen von Audio-Medien zur Offlinenutzung mit Angular

—

Developement of a progressive web app to download, manage and play audio files for offline use with Angular

—

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Das Webportal CROSSLOAD bietet viele christliche Medien zum Download an. Zur Nutzergruppe gehören Leute, die es gewohnt sind mit dem Computer oder Smartphone zu arbeiten, aber auch solche mit wenig Erfahrung im Umgang mit technischen Geräten. Das Portal wird sowohl von daheim im eigenen WLAN als auch unterwegs mit mobilen Daten genutzt.

Durch die mancherorts schlechte Mobilfunkabdeckung oder das beschränkte Datenvolumen möglicher Nutzer, könnte die Nutzung des Portals an vielen Stellen nicht möglich sein. Wenn man die Medien unterwegs abrufen möchte, müsste man sie sich vorher herunterladen und anschließend auf dem Gerät suchen. Für einige Nutzer ist das eine zu große Hürde und muss deswegen vereinfacht werden.

1.2 Zielsetzung

Ziel dieser Arbeit ist es für CROSSLOAD ein Konzept zur Offline Nutzung zu entwickeln und anschließend zu implementieren. Allen Nutzern soll eine kompfortable Nutzung des Webportals angeboten werden, sowohl auf Desktop-Computern, Tables oder Smartphones.

Dafür werden verschiedene Funktionen und APIs von Webbrowsern untersucht und verglichen. Die passendsten Funktionen werden ausgewählt und in das vorhandene Portal eingebunden. Dabei muss noch kein fertiges Produkt entstehen, aber die Machbarkeit gezeigt werden.

Schließlich stellt sich noch die Frage, ob die Funktionen und APIs der Webbrowser ausreichen, um alle Anforderungen zu erfüllen. Ist eine Progressive Web App in der Lage für CROSSLOAD eine native App zu ersetzten?

1.3 Aufbau

Zuerst werden die Anforderungen an das Webportal im Blick auf die Offline Nutzung herausgearbeitet. Danach folgt die Konzeption zur Datenspeicherung und zum Herunteladen der Daten. Anschließend wird auf die Implementierung der ausgewählten Funktionen eingegangen. In Kapitel 6 folgt die Evaluation und kritische Beurteilung. Zuletzt werden noch einige Funktionen beleuchtet, die in Zukunft umgesetzt werden könnten, um eine noch bessere Nutzerfreundlichkeit zu bieten.

Kapitel 2

Grundlagen

2.1 Progressive Web App

Eine Progressive Web App (PWA) ist eine Anwendung, die in modernen Webbrowsern läuft und dem Nutzer dabei eine Erfahrung ähnlich zu bekannten nativen Apps bietet (Sheppard 2017) (Rojas 2020). Die wichtigsten Funktionen dabei sind:

- Schnell: Die App startet schnell und bietet schnell die Möglichkeit zur Interkation (Hajian 2019) (Sheppard 2017)
- Zuverlässig: auf älteren Geräten funktionsfähig und auch ohne Internet nutzbar. Außerdem passt sich das Design an verschiedene Bildschirmgrößen an (responsive design) (Hajian 2019) (Sheppard 2017)
- Installierbar: nach dem Installieren wird ein Icon auf dem Homescreen angezeigt (Hajian 2019) (Sheppard 2017) (Rojas 2020)
- Benachrichtigungen: der Nutzer kann Benachrichtigungen bekommen, obwohl er die App nicht offen hat und aktiv benutzt (Hajian 2019) (Sheppard 2017)
- Native-like Funktionen: Zugriff auf Hardware wie zum Beispiel die Kamera (Hajian 2019)

Bei der Nutzung bzw. der Entwicklung einer PWA entstehen viele Vorteile für die Entwickler und Nutzer. Webtechnologien sind weit verbreitet und für jedes Betriebssystem sind moderne Browser verfügbar, dadurch kann die Anwendung sehr leicht an eine große Nutzergruppe ausgeliefert werden (Rojas 2020). Das Ausliefern an die Nutzer ist auch sehr einfach, weil diese nur eine URL benötigen (Khan,

Al-Badi und Al-Kindi 2019) und eine neue Version der App automatisch beim Start der App heruntergeladen wird (Rojas 2020). Außerdem ist die Entwicklung mit Webtechnologien weit verbreitet, man findet viele Ressourcen und Tools, die bei der Entwicklung helfen (Rojas 2020). Beim Vergleichen von nativen Apps zu einer PWA fällt zudem auf, dass die PWA sehr wenig Speicherplatz nach der Installation benötigt (Biørn-Hansen, Majchrzak und Grønli 2017) (Khan, Al-Badi und Al-Kindi 2019). Die Geschwindigkeit einer PWA auf Android kann im Vergleich zu anderen Cross-Platform Ansätzen mithalten oder ist sogar schneller (Biørn-Hansen, Majchrzak und Grønli 2017).

Eine PWA hat nicht nur Vorteile, sondern auch Nachteile. Einer davon ist die Limitierung auf die APIs der Webbrowser. Hinzu kommt, dass einige Funktionen noch nicht standardisiert sind und / oder noch nicht von allen gängigen Browsern unterstützt werden (Majchrzak, Biørn-Hansen und Grønli 2018) (Biørn-Hansen, Majchrzak und Grønli 2017). Wenn die Performance der App eine große Rolle spielt, wie etwas bei Spielen, ist eine PWA nicht sehr gut geeignet (Biørn-Hansen, Majchrzak und Grønli 2017).

Damit aus einer Website eine PWA wird sind mindestens zwei Dinge erforderlich. Erstens wird eine Manifest Datei benötigt, die Informationen über das Icon, den Namen und vielen mehr enthält (Hajian 2019) (Rojas 2020). Als zweites wird ein Service Worker benötigt, der unter anderem für die Offline-Fähigkeit verantwortlich ist (Rojas 2020).

2.2 Angular und Typescript

Angular ist ein Framework zum Entwickeln von Webanwendungen für alle Plattformen (Google LLC 2020). Es ist Open-Source, getrieben von einer großen Community aber auch von Firmen wie Google weiterentwickelt und selbst viel genutzt (Google LLC 2020). Als Programmiersprache ist Typescript vorgesehen.

Typescript erweitert JavaScript um einige Funktionen wie zum Beispiel Typisierung (Microsoft 2020). Typescript kann vor dem Ausliefern in reines JavaScript kompiliert werden und ist somit in allen gängigen Browsern oder auch auf Servern mit Node.js ausführbar (Microsoft 2020).

Die Kombination aus Angular und Typescript bietet eine sehr gute Basis zum Entwickeln von großen, schnellen und skalierbaren Anwendungen (Google LLC 2020).

Durch die statische Typisierung sind zum Beispiel Refactorings mit den vielen verfügbaren Tools schnell und sicher durchzuführen (Microsoft 2020) (Google LLC 2020). In der großen Community findet man auf sehr viele Fragen sofort eine Antwort.

2.3 CROSSLOAD

CROSSLOAD ist ein Webportal das christliche Medien, im Moment hauptsächlich Predigten, zur Verfügung stellt (biblepool gUG 2020). Diese Medien können durchsucht, angehört und heruntergeladen werden. Die Entwicklung befindet sich im Moment in der Beta-Phase (https://beta.crossload.org) und wird vor allem von Ehrenamtlichen vorangetrieben. In Abbildung 2.1 ist die mobile Seite mit den neusten Inhalten auf CROSSLOAD zu sehen. Abbildung 2.2 zeigt die Detailansicht einer Predigt.

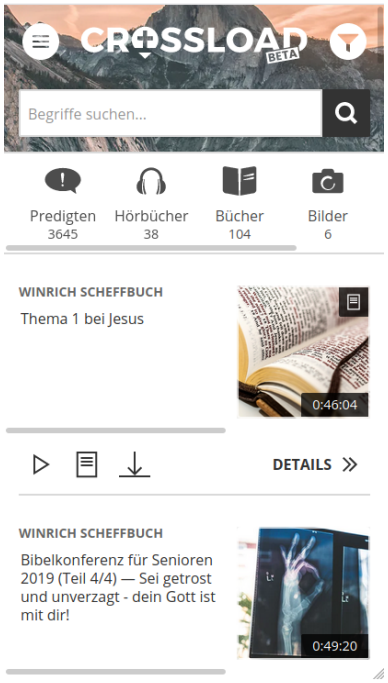

Abbildung 2.1: CROSSLOAD - Mobile Ansicht der neusten Inhalte

Im Frontend wird Angular mit Typescript zum Entwickeln verwendet und das Backend besteht aus vielen verschiedenen Services, die mit unterschiedlichen Sprachen und Frameworks entwickelt werden. Für diese Arbeit ist nur die Technologie für das Frontend interessant. Das Webportal besitzt schon über eine Manifest-Datei

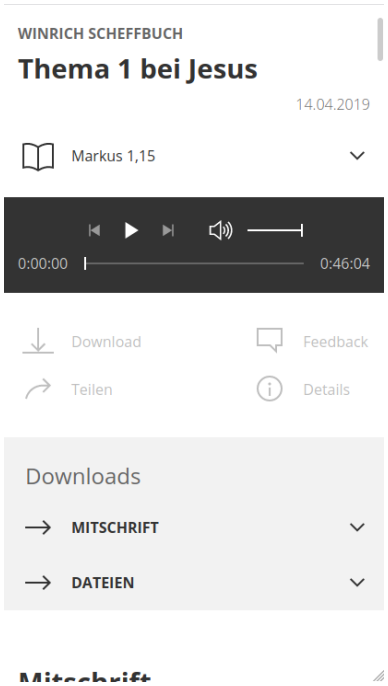

Abbildung 2.2: CROSSLOAD - Dateilansicht einer Predigt

und kann somit auf den Startbildschirm des Smartphones hinzugefügt werden. Statische Inhalte werden bereits gecached und sind somit auch offline Verfügbar.

Kapitel 3

Anforderungsanalyse

3.1 Randbedingungen

3.2 Funktionale Anforderungen

3.3 Use Cases

Kapitel 4

Konzeption

4.1 Lokale Datenspeicherung

4.1.1 Web Storage

SessionStorage: Bleibt erhalten solange der Browser nicht geschlossen wird. LocalStorage: Wie SessionStorage aber bleibt immer erhalten. Nachteil: Nur Synchron, nur strings, kein Web Worker Support Hajian 2019

4.1.2 WebSQL

Asynchron (callback-based) Nachteil: kein Web Worker Support, nicht in Firefox und depracated (Hajian 2019)

4.1.3 File System API

Asynchron (callback-based), bietet ein virtuelles Dateisystem zum ablegen von Daten. Läuft aber in einer Sandbox und kann nicht auf das "richtige"Dateisystem zugreifen. Vorteil: funktioniert in Web Workers Nachteil: (Fast) nur von Chrome unterstützt (Hajian 2019)

4.1.4 Indexed DB

- Umfangreiche Objektdatenbank um große Datenmengen auf dem Client zu speichern. Unterstützt Indexe zum Durchsuchen. Unterstützt auch Transaktionen (Sheppard 2017)

- key-value pari NoSQL Datenbank zum Speichern von vielen Daten (bis zu 20-50 % des verfügbaren Speicherplatzes). Unterstützt viele Datentypen, ist asynchron und kann überall verwendet werden (Hajian 2019)

- Verfügbare Bibliotheken: LocalForage, Dexie.js, ... (Hajian 2019)

Transaktionelle Datenbank - Gut für viel Daten, die Durchsucht werden müssen - Asynchron

4.1.5 Cache API

- Service Worker: Laufen im Hintergrund (im eigenen Thread) ohne Zugriff aufs DOM (Sheppard 2017)

- Service Worker ist ein Mittler zwischen APP und Internet. Es übernimmt Aufgaben wie Caching, Syncing, Benachrichtigungen (Sheppard 2017)

Service Worker: - Ärbeiterïn JavaScript die im Hintergrund laufen - Ermöglicht es Netzwerk-Request zu unterbrechen (Proxy) - Benachrichtigungen zu senden - Cache Verwalten und vieles mehr - Laufen im eigenen Thread ohne DOM Zugriff, HTTPS-Only - Website sollte nicht auf Service Worker angewiesen sein (Hajian 2019)

Angular bietet einen einfachen Weg Service Workers zu verwenden: - ng add @angular/pwa (Hajian 2019)

Cache Strategien: - Cache only: Es wird nur der Inhalt des Caches geliefert. Wenn nicht verfügbar wird nichts geliefert - Netzwerk only: Es wird nur die Antwort des Requests gesendet - Cache-First: Cache und wenn nicht verfügbar -> Netzwerk - Netzwerk-First: Netzwerk und wenn nicht verfügbar -> Cache - Cache und Netzwerk (stale-while-revalidate): Nimm erst den Wert vom Cache und sobald das Netzwerk geantwortet hat, die Antwort vom Netzwerk. Gut für sich häufig ändernde Daten (Facebook) - Generic-Fallback: Zum Beispiel für ein Bild ein Standardbild liefern, wenn Netzwerk und Cache keins haben (Hajian 2019) (Rojas 2020)

Bibliothek für Service Worker: - Workbox (Rojas 2020)

Cache storage != browser cache (Rojas 2020)

4.2 Herunterladen der Audiodaten

4.2.1 Fetch API

Einfachere XMLHttpRequests, Promise basierend (Rojas 2020)

4.2.2 Background Fetch

4.2.3 Background Sync

- Speichert den API Call bis eine stabile Internetverbindung besteht und schickt ihn dann raus. Sogar wenn die APP nicht aktiv ist oder läuft (Sheppard 2017)

- Versendet Request sobald eine Internetverbindung besteht, auch wenn die APP nicht offen ist (Rojas 2020)

- Man kann diese API mit IndexedDB kombinieren. Daten können in der Datenbank zwischengespeichert werden, bis eine Verbindung besteht (Rojas 2020)

4.3 Verbindungsstaus auslesen

- navigator.onLine: ist nicht 100% genau. Wird manchmal sagen, dass das Gerät online ist wenn es nur mit einem Netzwerk verbunden ist (Sheppard 2017)

Kapitel 5

Implementierung

5.1 Grafische Oberfläche

5.2 Lokale Datenspeicherung

5.3 Herunterladen der Audiodaten

Kapitel 6

Evaluation und Reflexion

6.1 Kann durch die PWA eine native APP ersetzt werden?

6.2 Nutzerfreundlichkeit

Kapitel 7

Zusammenfassung und Ausblick

7.1 Zusammenfassung

7.2 Ausblick

Abkürzungsverzeichnis

PWA Progressive Web App

Tabellenverzeichnis

Abbildungsverzeichnis

Literatur

biblepool gUG (2020). *Crossload - Deine Tankstelle für Wachstum im Glauben*. URL: https://crossload.org/ (besucht am 20. 05. 2020).

Biørn-Hansen, Andreas, Tim A Majchrzak und Tor-Morten Grønli (2017). „Progressive Web Apps: The Possible Web-native Unifier for Mobile Development.“ In: *WEBIST*, S. 344–351.

Google LLC (2020). *Angular*. URL: https://angular.io/ (besucht am 20. 05. 2020).

Hajian, Majid (2019). „Safety Service Worker“. In: *Progressive Web Apps with Angular: Create Responsive, Fast and Reliable PWAs Using Angular*. Berkeley, CA: Apress, S. 283–288. DOI: 10.1007/978-1-4842-4448-7_11. URL: https://doi.org/10.1007/978-1-4842-4448-7_11.

Khan, Asharul Islam, Ali Al-Badi und Mahmood Al-Kindi (2019). „Progressive Web Application Assessment Using AHP“. In: *Procedia Computer Science* 155. The 16th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2019),The 14th International Conference on Future Networks and Communications (FNC-2019),The 9th International Conference on Sustainable Energy Information Technology, S. 289–294. DOI: https://doi.org/10.1016/j.procs.2019.08.041. URL: http://www.sciencedirect.com/science/article/pii/S187705091930955X.

Majchrzak, Tim A, Andreas Biørn-Hansen und Tor-Morten Grønli (2018). „Progressive web apps: the definite approach to cross-platform development?“ In:

Microsoft (2020). *Typescript - JavaScript that scales.* URL: https://www.typescriptlang.org/ (besucht am 20. 05. 2020).

Rojas, Carlos (2020). „Service Workers“. In: *Building Progressive Web Applications with Vue.js : Reliable, Fast, and Engaging Apps with Vue.js*. Berkeley, CA: Apress, S. 55–65. DOI: 10.1007/978-1-4842-5334-2_3. URL: https://doi.org/10.1007/978-1-4842-5334-2_3.

Sheppard, Dennis (2017). „Background Sync for Offline Apps with Service Workers“. In: *Beginning Progressive Web App Development: Creating a Native App Experience on the Web*. Berkeley, CA: Apress, S. 73–93. DOI: 10.1007/978-1-4842-3090-9_5. URL: https://doi.org/10.1007/978-1-4842-3090-9_5.