



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. 15 Seiten.

Schalter, Martin:

Development 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. 15 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

| | |
|---|-----------|
| 1. Einleitung | 1 |
| 1.1. Motivation | 1 |
| 1.2. Zielsetzung | 1 |
| 1.3. Aufbau | 1 |
| 2. Grundlagen | 3 |
| 2.1. Typescript | 3 |
| 2.2. Angular | 3 |
| 2.3. Progressive Web App | 3 |
| 2.4. CROSSLOAD | 3 |
| 3. Anforderungsanalyse | 5 |
| 3.1. Randbedingungen | 5 |
| 3.2. Funktionale Anforderungen | 5 |
| 3.3. Use Cases | 5 |
| 4. Konzeption | 7 |
| 4.1. Lokale Datenspeicherung | 7 |
| 4.1.1. Web Storage | 7 |
| 4.1.2. WebSQL | 7 |
| 4.1.3. File System API | 7 |
| 4.1.4. Indexed DB | 8 |
| 4.1.5. Cache API | 8 |
| 4.2. Herunterladen der Audiodaten | 9 |
| 4.2.1. Fetch API | 9 |
| 4.2.2. Background Fetch | 9 |
| 4.2.3. Background Sync | 9 |
| 4.3. Verbindungsstatus auslesen | 9 |
| 5. Implementierung | 11 |
| 5.1. Grafische Oberfläche | 11 |
| 5.2. Lokale Datenspeicherung | 11 |
| 5.3. Herunterladen der Audiodaten | 11 |

| | |
|---|-------------|
| 6. Evaluation und Reflexion | 13 |
| 6.1. Kann durch die PWA eine native APP ersetzt werden? | 13 |
| 6.2. Nutzerfreundlichkeit | 13 |
| 7. Zusammenfassung und Ausblick | 15 |
| 7.1. Zusammenfassung | 15 |
| 7.2. Ausblick | 15 |
| Abkürzungsverzeichnis | vii |
| Tabellenverzeichnis | ix |
| Abbildungsverzeichnis | xi |
| Quellcodeverzeichnis | xiii |
| Literatur | xv |
| Index | xvii |
| A. Erster Anhang | xvii |
| B. Zweiter Anhang | xix |

Kapitel 1

Einleitung

1.1. Motivation

1.2. Zielsetzung

1.3. Aufbau

Kapitel 2

Grundlagen

2.1. Typescript

2.2. Angular

2.3. Progressive Web App

- Liste von Strategien, Techniken und APIs um dem Nutzer eine Erfahrung ähnlich zu nativen APPs zu bieten Sheppard 2017

- PWAs sind: Schnell, Zuverlässig (auch ohne Internet und auf alten Geräten), Engaging? Sheppard 2017 Hajian 2019

- Features von PWAs: Offline support (zumindest für die Hauptseite), Schnell (auch für mobiles Internet), Home screen icon und splash screen, Benachrichtigungen Sheppard 2017

Features von PWAs: - Sehr schnell geladen und Interaktion möglich - offline fähig - Responsive: mobile-first (weil es schlechtere Hardware hat, offline-first) - Benachrichtigungen - Native-like Features: hardware apis (kamera, bluetooth, ...) - Sicher (HTTPS) - Installierbar - Fortschrittlich (progressive) Hajian 2019

PWAs haben ein APP Manifest mit - name, icon, usw. Hajian 2019

2.4. CROSSLOAD

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 deprecated 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

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: - Arbeiterin 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: 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

4.2. Herunterladen der Audiodaten

4.2.1. Fetch API

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

4.3. Verbindungsstatus 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

Tabellenverzeichnis

Abbildungsverzeichnis

Listings

Literatur

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.

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.

Anhang A

Erster Anhang

Hier ein Beispiel für einen Anhang. Der Anhang kann genauso in Kapitel und Unterkapitel unterteilt werden, wie die anderen Teile der Arbeit auch.

Anhang B

Zweiter Anhang

Hier noch ein Beispiel für einen Anhang.