

Konzeption und Implementierung einer Progressive Web App (PWA) als plattformunabhängiger Ansatz für mobile Endgeräte und Desktops

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science

Ernst-Abbe-Hochschule Jena

Fachbereich Wirtschaftsingenieurwesen
Bachelor-Studiengang E-Commerce

eingereicht von: Tine Pöhlmann

geboren am: 19.04.1994

Matrikelnummer: 644437

Betreuender der Hochschule: Prof. Dr.-Ing. Michael Stepping

Tag der Einreichung: 28.03.2022

Referat:

Ein Nachteil der nativen App-Entwicklung ist der entstehende Mehraufwand, der bei der Entwicklung einer App für verschiedene Plattformen entsteht. Alternative Entwicklungskonzepte, wie hybride Apps oder plattformunabhängige Web Apps weisen jedoch Einschränkungen in Performance und Funktionsumfang auf und dienen so nur bedingt als Alternative zu einer nativen App. Die vorliegende Bachelorarbeit befasst sich daher mit einem weiteren plattformunabhängigen Entwicklungskonzept. Für Progressive Web Apps wird versprochen, dass diese die Vorteile der zuvor genannten Konzepte kombinieren und die Nachteile beheben. In einer praktischen Umsetzung im Rahmen dieser Arbeit wird eine Anwendung für die Ernst-Abbe-Hochschule Jena unter Verwendung des Single-Page Web Application Frameworks Angular nach dem Konzept von Progressive Web Apps implementiert.

Gliederung

Gliederung	I
Abbildungsverzeichnis	III
Tabellenverzeichnis	IV
Code-Snippet-Verzeichnis	V
Abkürzungsverzeichnis	VI
1 Einleitung.....	1
1.1 Problemstellung	1
1.2 Zielsetzung.....	2
1.3 Aufbau der Arbeit	2
2 Grundlagen.....	3
2.1 App-Entwicklungskonzepte	3
2.1.1 Native Apps	3
2.1.2 Web Apps	4
2.1.3 Hybride App.....	5
2.1.4 Progressive Web App.....	6
3 Progressive Web App	7
3.1 Eigenschaften	7
3.1.1 Progressiv (Progressive)	7
3.1.2 Reaktionsfähig (Responsive)	8
3.1.3 App-ähnlich (App-like)	8
3.1.4 Netzwerkunabhängig (Connectivity Independent)	8
3.1.5 Aktuell (Fresh)	9
3.1.6 Sicher (Safe).....	9
3.1.7 Auffindbarkeit (Discoverable)	9
3.1.8 Installierbarkeit (Installable).....	10
3.1.9 Reaktivierbar (Re-engageable)	10
3.1.10 Verlinkbarkeit (Linkable).....	10
3.2 Funktionsweise	10
3.2.1 Service Worker	11
3.2.2 Web App Manifest	13
3.2.3 Web APIs.....	14
3.3 Unterstützung.....	15

3.4	Realisierungsoptionen.....	20
4	Technische Umsetzung der Hochschul-App	22
4.1	Anforderungen	22
4.1.1	Umzusetzende Features	22
4.1.2	Datenquelle	22
4.1.3	Funktionale und nicht-funktionale Anforderungen an die Anwendung.....	22
4.2	Konzeption	23
4.2.1	Framework-Auswahl.....	23
4.2.2	UI- und UX-Konzept	25
4.2.3	Clientseitige Speichertechnologien	29
4.2.4	Offline Support Konzept	29
4.2.5	Architekturkonzept.....	30
4.2.6	Deployment	33
4.3	Implementierung	33
4.3.1	Projekt Setup	33
4.3.2	UI	34
4.3.3	App Shell	38
4.3.4	Wiederverwendbare Komponenten.....	40
4.3.5	Service Implementierung.....	41
4.3.6	Indexed DB und Local Storage	43
4.3.7	Implementierung der PWA Komponenten.....	45
4.3.8	App-ähnliches Verhalten	48
4.4	Kritische Betrachtung der Umsetzung	49
4.4.1	Erfüllung der Anforderungen	49
4.4.2	Betrachtung der entwickelten Lösung für iOS	50
5	Fazit und Ausblick.....	52
5.1	Ausblick	53
Literaturverzeichnis		VII
Anhang		XI
Ehrenwörtliche Erklärung und Einverständniserklärung		XXI

Abbildungsverzeichnis

Abbildung 1: Funktionsweise des Service Workers	12
Abbildung 2: Funktionsweise von Push-Benachrichtigungen bei PWAs	13
Abbildung 3: Architektur und Funktionsweise einer PWA	14
Abbildung 4: Aufbau des Angular Frameworks	24
Abbildung 5: Angular Implementierung des MVC Patterns	25
Abbildung 6: Mockup und Layoutkonzept der Hauptansichten der Features.....	27
Abbildung 7: UX-Konzept zu Mein Stundenplan	27
Abbildung 8: Interaktionselemente mit Icons	28
Abbildung 9: High Level Architektur.....	30
Abbildung 10: App Shell Architektur	31
Abbildung 11: Architekturkonzept Module	32
Abbildung 12: Architekturkonzept Mensa Module	32
Abbildung 13: Architekturkonzept Stundenplan Modul.....	33
Abbildung 14: Abstufungen der Primärfarbe nach Corporate Design.....	XVI
Abbildung 15: Corporate Design Akzentfarben der Ernst-Abbe-Hochschule Jena	XVI
Abbildung 16: EAH-Logo Kurzversion mit Schutzzone	XVII

Tabellenverzeichnis

Tabelle 1: Übersicht über die App-Programmiersprachen der verschiedenen Plattformen	3
Tabelle 2: Ausgewählte Web-APIs und deren Verwendungszweck	15
Tabelle 3: Browserunterstützung für responsive Design	16
Tabelle 4: Browserunterstützung des Web App Manifests.....	16
Tabelle 5: Browserunterstützung für Web APIs zur Umsetzung von Offline Funktionalitäten	17
Tabelle 6: Browserunterstützung nativer Benachrichtigungsfunktionen	18
Tabelle 7: Browserunterstützung - native Funktionen und Gerätezugriff.....	18
Tabelle 8: Browserunterstützung beim Zugriff auf Gerätesensoren	19
Tabelle 9: Browserunterstützung für Schnittstellen zum Datenaustausch	19
Tabelle 10: Übersicht diverser Caching Strategien	30
Tabelle 11: Inhalt des Web Application Manifests nach dem W3C Standard	XII
Tabelle 12: Übersicht über verwendete API Endpoints	XIII
Tabelle 13: Übersicht der wichtigsten Befehle der Angular CLI	XVIII

Code-Snippet-Verzeichnis

Code-Snippet 1: Theme-Konfiguration	35
Code-Snippet 2: Einbindung der Icon Font.....	35
Code-Snippet 3: Erstellung eines Icon Buttons mit Angular Material	36
Code-Snippet 4: Grid-Template.....	36
Code-Snippet 5 Implementierung des abweichenden Desktop-Layouts	37
Code-Snippet 6: Responsive Design Implementierung.....	37
Code-Snippet 7: index.html	38
Code-Snippet 8: @Component-Decorator.....	38
Code-Snippet 9: App Shell - Template	39
Code-Snippet 10: Routing-Konfiguration	39
Code-Snippet 11: Beispiel für Kommunikation zwischen Komponenten	40
Code-Snippet 12: Beispiel für Data Binding.....	41
Code-Snippet 13: Implementierung des Stundenplanservice	41
Code-Snippet 14: Nutzung eines Services in einer Komponente.....	42
Code-Snippet 15: Konfiguration der Indexed DB Stores.....	44
Code-Snippet 16: Kommunikation mit der Indexed DB.....	44
Code-Snippet 17: Local Storage.....	45
Code-Snippet 18: Einbindung eines Splash Screens für ein iOS Endgerät.....	46
Code-Snippet 19: Service Worker Konfiguration - Data Groups	47
Code-Snippet 20: Service Worker Update.....	48
Code-Snippet 21: Unterbinden des Browser-Standardverhaltens.....	48
Code-Snippet 22: GET /timetable - Response Schema (gekürzt)	XIII
Code-Snippet 23: GET /timetable/events?timetableId=SPLUSD44A96 - Response Schema (gekürzt)	XIV
Code-Snippet 24: GET /canteens - Response Schema (gekürzt)	XIV
Code-Snippet 25: GET /canteens/58 - Response Schema (gekürzt)	XV
Code-Snippet 26: Response-Schema für einen neuen Mein-Stundenplan-Endpoint	XX

Abkürzungsverzeichnis

API	Application Programming Interface
App	Application (dt. Anwendung)
CORS	Cross-Origin Ressource Sharing
CSS3	Cascading Style Sheet Version 3
DB	Data Base (dt. Datenbank)
DOM	Document Object Model
EAH	Ernst-Abbe-Hochschule
HTML5	Hypertext Markup Language Version 5
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrated Development Environment
JS	JavaScript
JSON	JavaScript Object Notation
MB	Megabyte
npm	Node Package Manager
PWA	Progressive Web App
RxJS	Reactive Extensions for JavaScript (JavaScript Bibliothek)
SASS	Syntactically Awesome Stylesheets
SCSS	Kombination aus CSS und SASS
SDK	Software Development Kit
SPA	Single-Page Web Application
SQL	Structured Query Language
UI	User Interface
URL	Unified Ressource Locator
UX	User Experience
W3C	World Wide Web Consortium
Web App	Web Application

1 Einleitung

Die Digitalisierung ist nahezu in jedem Lebensbereich angekommen. Wir leben in einem digitalen Informationszeitalter und Apps sind ein ständiger Begleiter in unserem Alltag geworden. Sie dienen zum Teilen von Informationen oder als Werkzeug zur Unterstützung in verschiedensten Situationen. Sie unterstützen uns dabei, uns besser zu organisieren und effizienter zu leben. Eine aktuelle Herausforderung ergibt sich aus der Vielzahl an Endgeräten, die jeweils unterschiedliche Eigenschaften und Fähigkeiten aufweisen. Da ein Webbrowser zur Grundausstattung der Software der Endgeräte gehört, werden häufig plattformübergreifende Webanwendungen entwickelt. Diese haben jedoch das Problem, dass sie aus Gründen der Sicherheit nicht alle Funktionen des Endgeräts vollumfänglich nutzen können. Hierfür wäre eine Entwicklung von plattformabhängigen nativen Apps notwendig, die jedoch auch mit einem höheren Aufwand einhergeht. Diese Arbeit beschäftigt sich mit einem neuen Ansatz der Appentwicklung namens **Progressive Web App (PWA)**. Die Basisfunktionen dieser Anwendungen sind unabhängig von Betriebssystem und Fähigkeiten des Endgerätes nutzbar, wobei der Funktionsumfang mit der Leistungsstärke des eingesetzten Browsers progressiv ansteigt.

1.1 Problemstellung

Um den Studierenden der Ernst-Abbe-Hochschule Jena den Studienalltag zu erleichtern, wird von der Hochschule die App EAH Jena offeriert, welche zur Information über verschiedene Aspekte des Studienalltags wie Stundenplanung, Hochschul-News und Mensa-Speisepläne dient. Bisher wurde diese App ausschließlich für die Android Plattform entwickelt. Für die iOS Plattform gibt es keine Lösung in Form einer mobilen App. Neben der mobilen Android-App besteht die Möglichkeit allgemeine Informationen über die offizielle Webseite der Hochschule¹ zu beziehen, Mensaspisepläne u.a. über die Webseite des Studierendenwerks² und Stundenpläne über eine separate Webseite der Hochschule³ abzurufen. Insbesondere die Webseite zum Abruf der Stundenpläne ist dabei nicht für mobile Endgeräte optimiert und bietet daher nur wenig Komfort für den Zugriff über ein Smartphone. Während die Android-App teilweise ohne bestehende Netzwerkverbindung nutzbar bleibt, ist eine Offline-Nutzung der verschiedenen Webseiten nicht möglich.

Eine Möglichkeit das Problem zu lösen, wäre es eine weitere App für iOS Geräte zu entwickeln. Dies würde jedoch zu einem erhöhten Wartungsaufwand führen, da dann zwei Systeme

¹ Vgl. Ernst-Abbe-Hochschule Jena, o. D.

² Vgl. Einrichtungen - Studierendenwerk Thüringen, o. D.

³ Vgl. Ernst-Abbe-Hochschule Jena - Stundenplanung, o. D.

parallel gewartet werden müssten. Da es aktuell zudem kein Feature gibt, das den Zugriff auf native Funktionen des Endgeräts erfordert, bietet sich als eine Alternative die Entwicklung einer plattformunabhängigen Anwendung an, die die Android-App ablösen könnte. Diese könnte dann sowohl von mobilen Endgeräten wie Smartphones und Tablets als auch von Laptops und Desktop-Geräten genutzt werden. Diese Arbeit wird sich daher mit einem solchen Ansatz auseinandersetzen.

1.2 Zielsetzung

Ziel dieser Arbeit ist es ein Konzept sowie einen ersten Prototyp einer Anwendung zu erstellen, die sich an Mitarbeitende und Studierende der EAH Jena richtet. Dabei sollen die am häufigsten verwendeten Funktionen der bestehenden Android-App, der Abruf und das Zusammenstellen individueller Stundenpläne sowie die Abfrage des Speisenangebots der Mensen in Jena, in dem zu erstellenden Prototyp realisiert werden. Die Anwendung soll auf dem Entwicklungsprinzip einer Progressive Web App basieren und soll Nutzenden jeglicher Betriebssysteme und Endgeräte einen gleichermaßen guten Komfort bei der Nutzung der Anwendung bieten.

1.3 Aufbau der Arbeit

Nachdem in diesem einleitenden Kapitel die Problemstellung und Zielsetzung der Arbeit erläutert wurden, werden im folgenden Kapitel 2 zunächst die theoretischen Grundlagen zu verschiedenen Entwicklungskonzepten und Typen von Anwendungssoftware dargestellt.

In Kapitel 3 wird das Konzept von Progressive Web Apps im Detail betrachtet. Dabei wird auf deren Eigenschaften und Funktionsweise eingegangen. Zudem werden Vorteile gegenüber anderen App-Entwicklungskonzepten, die Unterstützung von verschiedenen Browsern und Möglichkeiten zur Umsetzung bzw. Entwicklung von PWAs beschrieben.

Kapitel 4 beschreibt die technische Umsetzung der Hochschulapp als Progressive Web App einschließlich einer kritischen Betrachtung der Umsetzung.

Es folgt ein abschließendes Fazit und ein Ausblick für zukünftige Möglichkeiten der Erweiterung und Verbesserung der App in Kapitel 5.

2 Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen erläutert. Insbesondere wird auf verschiedene Entwicklungskonzepte bzw. Typen von Apps eingegangen, um die Thematik von Progressive Web Apps einordnen zu können und das weitere Verständnis dieser Arbeit sicherzustellen.

2.1 App-Entwicklungskonzepte

2.1.1 Native Apps

Bei nativen Apps handelt es sich um Anwendungen, die explizit für eine bestimmte Plattform bzw. ein bestimmtes Betriebssystem entwickelt werden.⁴ Für die Entwicklung der nativen App wird das von der Plattform bereitgestellte Software Development Kit (SDK) genutzt. Darunter versteht sich ein Set verschiedener Werkzeuge und hilfreicher Ressourcen zur Erstellung der Anwendung. Standardmäßig enthält ein SDK dabei Compiler, Debugger und APIs. Zusätzlich enthalten sein können zudem Dokumentationen, Libraries, Editoren, Runtime-/Entwicklungsumgebungen, Test-/Analyse-Tools, Treiber sowie Netzwerkprotokolle.⁵ Zur Programmierung wird eine von der Plattform vorgegebene Programmiersprache verwendet. So können alle Funktionen des Betriebssystems ausgenutzt, die Performance optimiert und der Nutzungskomfort erhöht werden. Derzeit gibt es zwei Plattformen für die Entwicklung nativer Apps für Smartphones – Android und iOS. Doch auch für die Windows Plattform können native Desktop-Apps entwickelt werden. Eine Übersicht über die bevorzugten Programmiersprachen für die verschiedenen Plattformen, die jeweils vorgesehene Entwicklungsumgebung sowie das zu nutzende Software Development Kit wird in folgender Tabelle 1 aufgezeigt.

	Android	iOS	Windows
Programmiersprache	Java, Kotlin	Objective-C, Swift	C#, Visual Basic, C++, JavaScript
Entwicklungsumgebung (IDE)	Android Studio	Xcode	Visual Studio
Software Development Kit (SDK)	Android SDK	iOS SDK	Windows App SDK

Tabelle 1: Übersicht über die App-Programmiersprachen der verschiedenen Plattformen

Quelle: Eigene Darstellung. Datenquelle: Microsoft, 2022b; Chebbi, 2021.

⁴ Vgl. Tremp, 2021, S.109.

⁵ Vgl. Red Hat Inc., 2020.

Aus der Tabelle wird ersichtlich, dass keine Übereinstimmungen der zu nutzenden Programmiersprache und Tools zwischen den verschiedenen Plattformen vorliegen. Das bedeutet, dass bei der Entwicklung nativer Apps jeweils eine eigene Anwendung für jede Plattform programmiert werden muss, was zwangsläufig mit einem erhöhten Entwicklungs-, Wartungs- und damit auch Kostenaufwand einhergeht.⁶

Um eine native App auf dem Endgerät auszuführen, muss sie auf diesem installiert werden. Dies erfordert die Verfügbarkeit von Speicherressourcen. Bezogen werden native Apps über eine spezifische zentrale Stelle der jeweiligen Plattform. Für Android-Apps ist das der Google Play Store, für iOS der App Store von Apple und für Windows der Microsoft Store. Bevor die App im Store der jeweiligen Plattform publiziert wird, findet ein Prüfprozess statt, der sicherstellt, dass die definierten Richtlinien der jeweiligen Plattform eingehalten wurden.^{7 8 9} Außerdem erhalten die App Stores eine Beteiligung am Umsatz von In-App-Käufen.^{10 11 12}

2.1.2 Web Apps

Eine Web App ist eine plattformübergreifende Anwendung, welche auf einer Client-Server-Architektur basiert. Das bedeutet, dass die Anwendung von einem Server bereitgestellt wird und von einem Webbrowser als Client über einen Unified Resource Locator (URL) abgerufen werden kann.¹³ Eine Installation der Anwendung ist nicht notwendig. Die Verfügbarkeit der Web App ist durch die Ausführung in einem Webbrowser unabhängig vom Betriebssystem des Endgerätes.¹⁴ Technologisch basieren Web Apps auf den Webtechnologien HTML5, CSS3 und JavaScript. HTML dient dazu, den Inhalt bereitzustellen, während CSS genutzt wird, um das Layout und die Darstellung der Inhalte zu definieren. JavaScript ermöglicht es, die Inhalte dynamisch, z.B. durch Benutzerinteraktionen, zu verändern. Auch klassische Webseiten basieren auf diesem Prinzip.¹⁵ Die Übergänge zwischen einer klassischen Webseite und einer Web App sind fließend. Durch die Implementierung eines responsiven Designs ist die Benutzeroberfläche von Web Apps für die Nutzung auf mobilen Endgeräten optimiert. Im Vergleich

⁶ Vgl. Darji et al., 2021, S.91f.

⁷ Vgl. Apple Inc., o. D. a.

⁸ Vgl. Google, 2022a.

⁹ Vgl. Microsoft, 2022a.

¹⁰ Vgl. Google, 2022b.

¹¹ Vgl. Apple Inc., 2020.

¹² Vgl. Geuß, 2021.

¹³ Vgl. Malavolta, 2016, S.1.

¹⁴ Vgl. Adetunji et al., 2020, S.91f.

¹⁵ Vgl. Tosic, 2015, S.8f.

zum nativen Entwicklungsansatz können jedoch weder Offline-Funktionalitäten noch die nativen Funktionen des Betriebssystems des Endgerätes voll ausgeschöpft werden. JavaScript bietet allerdings für einige native Funktionen Schnittstellen bspw. für die Geolokalisierung, Kamera- und Mikrofonzugriff sowie für das Senden von Push-Benachrichtigungen.¹⁶ Die einzelnen Funktionalitäten werden je nach Browser unterschiedlich unterstützt. Aufgrund der uneinheitlichen Unterstützung eignen sich hardwarenahe Lösungen aktuell nicht für die Umsetzung als Web App.¹⁷

2.1.2.1 Single-Page Web Application (SPA)

Eine spezielle Form der Web-Anwendung ist die sogenannte Single-Page Web Application. Ebenso wie klassische Web Apps basiert deren Umsetzung auf den Technologien HTML5, CSS3 und JavaScript und der Ausführung im Browser. Im Gegensatz zu klassischen Webanwendungen werden beim Aufruf der SPA alle Quelldateien vom Server in den Browser des Benutzenden geladen. Weitere Anfragen an einen Webserver werden nur gestellt, um entfernt gespeicherte Daten zu beziehen oder zu manipulieren. Ansonsten funktioniert eine SPA ebenso wie eine native App autonom.¹⁸ Die SPA besteht aus nur einem einzigen initial geladenen HTML-Dokument. Der Inhalt dieses Dokuments wird mithilfe von JavaScript-Funktionen dynamisch modifiziert. Durch Benutzerinteraktionen werden dann nur einzelne Bestandteile der Anwendung, sogenannte Views, verändert, während der Rest der Anwendung davon unberührt bleibt und nicht neu geladen wird. Dies wirkt sich positiv auf die Performance und das Nutzungserlebnis (engl. User Experience) aus.¹⁹

2.1.3 Hybride App

Hybride Apps sind, wie native Apps, Anwendungen speziell für mobile Endgeräte. Bei dem Konzept einer hybriden App werden native und webbasierte Elemente kombiniert und so die Vorteile beider Ansätze ausgenutzt. Entwickelt wird die hybride App mit den Webtechnologien HTML, CSS und JavaScript. Diese Webinhalte werden dann mithilfe eines Frameworks in einen Web-View Container (auch nativer Container) eingebettet. Zur Darstellung wird die native Browser Rendering Engine des Endgeräts genutzt, nicht jedoch der Browser an sich. So kann eine benutzerfreundliche Oberfläche bereitgestellt und der Zugriff auf Hard- und Softwarekomponenten des Endgeräts gewährleistet werden, da die Frameworks, welche zur

¹⁶ Vgl. Bar, o. D. b.

¹⁷ Vgl. Rentrop/Augsten, 2021.

¹⁸ Vgl. Scott, 2015, S.4-13.

¹⁹ Vgl. Tremp, 2021, S.100.

Implementierung genutzt werden, die erforderlichen Schnittstellen bereitstellen.²⁰ Hinsichtlich ihrer Performance schneiden hybride Apps gegenüber nativen Apps schlechter ab, da die Ausführung dieser in der Browser-Engine erfolgt.²¹ Ist dies also ein wichtiges Anforderungskriterium, sollte eine native Entwicklung vorgezogen werden. Soll die Anwendung jedoch lediglich zum Abrufen, Speichern und Darstellen von Daten dienen, empfiehlt sich die weniger aufwendige Entwicklung einer Web App. Wie bei nativen Apps und im Gegensatz zu Web Apps, erfolgt die Distribution von hybriden Apps über die plattformspezifischen App Stores.

2.1.4 Progressive Web App

Eine Progressive Web App ist eine spezielle Form der Single-Page Web Application. Sie wird demzufolge mit Webtechnologien (HTML5, CSS3, JavaScript) entwickelt und über einen Webbrowser ausgeführt. Der Unterschied zu gewöhnlichen Web Apps bzw. Single-Page Web Applications beruht auf dem Einsatz verschiedener zusätzlicher Technologien wie dem Service Worker, dem Web App Manifest und diversen Web APIs, welche in Kapitel 3.2 im Detail erläutert werden. Dies ermöglicht die Unterstützung von Funktionen wie Offline-Fähigkeit und das Senden von Push-Benachrichtigungen, welche zuvor nur installierten Apps vorbehalten waren.²² Außerdem wird eine Möglichkeit der Installation realisiert, um die PWA in ihrem Aussehen und Verhalten einer nativen App anzugleichen. Diese und weitere besondere Eigenschaften von PWAs werden in Kapitel 3.1 ausführlich beschrieben. Durch die Ausführung der Anwendung über einen Webbrowser verfolgt das Entwicklungskonzept der PWA, wie das der klassischen Webanwendung, einen plattformunabhängigen Ansatz. Die Nutzung ist somit für alle Endgeräte mit einem installierten Webbrowser möglich. Dabei können PWAs Anwendenden ein Nutzungserlebnis ähnlich dem einer nativen App bieten. Die PWA kann sowohl über eine URL im Browser aufgerufen werden als auch über die App Stores der Plattformen bereitgestellt werden.^{23 24 25}

²⁰ Vgl. ANEXIA Deutschland GmbH, o. D.

²¹ Vgl. Latif et al., 2016.

²² Vgl. Tremp, 2021, S.101.

²³ Vgl. Windows Platform, 2021.

²⁴ Vgl. Publish your PWA to the iOS App Store, 2021.

²⁵ Vgl. Android Platform, 2021.

3 Progressive Web App

In diesem Kapitel wird das Konzept von Progressive Web Apps im Detail betrachtet. Es werden deren Eigenschaften und Funktionsweise beschrieben. Zudem wird auf die aktuelle Unterstützung der verschiedenen Webbrowser eingegangen und Möglichkeiten für eine technische Umsetzung vorgestellt.

3.1 Eigenschaften

Der Begriff Progressive Web App wurde erstmals 2015 von dem Google Entwickler Alex Russell vorgestellt. In seinem Blogbeitrag nennt er für diese neue Art von Webapplikationen folgenden Eigenschaften:

- Responsive
- Connectivity independent
- App-like Interactions
- Fresh
- Safe
- Discoverable
- Re-engagable
- Installable
- Linkable ²⁶

In einer Ausarbeitung von Liebel wird zusätzlich das Merkmal Progressive genannt, welches sich bereits im Namen des Konzepts wiederfindet. In den folgenden Unterkapiteln werden diese zehn Eigenschaften von PWAs inklusive ihrer Ansätze für eine technische Umsetzung erläutert.

3.1.1 Progressiv (Progressive)

Diese Eigenschaft ist auf zwei Arten zu betrachten.

1. Die PWA kann sich durch ihre Installation schrittweise, progressiv von einer Webanwendung, welche im Browser geöffnet und genutzt werden kann, zu einer nativen Anwendung entwickeln, welche in einem eigenen Fenster bzw. im Vollbildmodus ausgeführt wird.
2. Nach dem Prinzip von Progressive Enhancement sind die Kernfunktionen der PWA auf jedem Endgerät ausführbar, wobei der Funktionsumfang der Anwendung mit der Stärke des eingesetzten Browsers ansteigt. So werden nicht alle PWA-Schnittstellen

²⁶ Vgl. Russell, 2015.

in allen Browsern gleichermaßen unterstützt. Der Safari mobile Browser unterstützt z.B. keine Push-Benachrichtigungen auf Betriebssystemebene. Die Anwendung ist dennoch auch in diesem Browser funktionsfähig, lediglich ohne die Benachrichtigungsfunktion. Technisch basiert dies auf dem Prinzip der Feature Detection.²⁷ Dabei prüft die Anwendung zunächst, ob eine bestimmte Schnittstelle zur Verfügung steht, bevor diese angefragt wird.

3.1.2 Reaktionsfähig (Responsive)

Diese Eigenschaft beschreibt die Fähigkeit von Webseitenlayouts, sich flexibel an das Endgerät der Benutzenden anzupassen. Es dient dazu, dass die Inhalte und Steuerelemente für Benutzende der Webseite stets eine komfortable Größe und Erreichbarkeit auf dem Display aufweisen und so ein bestmögliches Nutzungserlebnis erzielt werden kann. Die technische Basis dafür bilden CSS3 Media Queries, das CSS Flexible Box Layout Module (Flexbox) und das Grid Layout Module. Zudem muss im Kopf des HTML-Dokuments das Metatag `viewport` integriert werden.

3.1.3 App-ähnlich (App-like)

PWAs unterliegen den gleichen Gütekriterien wie ihr natives Pendant bezüglich ihrer Bedienung, Funktion, Aussehen und Performance. Die Umsetzung beginnt bereits im Architekturkonzept der PWA. Hier wird auf das Prinzip von Single-Page Web Applications zurückgegriffen, welches in Abschnitt 2.1.2.1 dieser Arbeit erläutert wurde. Auf Funktionsebene kommunizieren PWAs mit Web-Schnittstellen, um von nativen Funktionen wie dem Kamera- und Mikrofongriff, dem Gerätestandort oder dem Senden von Push-Benachrichtigungen Gebrauch zu machen. Durch die Anpassung von Navigationsstrukturen, Steuerelementen, Effekten und Animationen wird die Benutzeroberfläche einer PWA dem nativen Vorbild angeglichen. Die Performance wird durch das Konzept der App Shell optimiert. Dabei werden Quelldateien für den statischen Anwendungsrahmen offline auf dem Gerät gehalten und nicht redundant über das Netzwerk bezogen.

3.1.4 Netzwerkunabhängig (Connectivity Independent)

Die PWA bleibt auch in einem Zustand mit einer instabilen Netzwerkverbindung oder Offline funktionsfähig. Dafür werden mindestens die Quelldateien offline zur Verfügung gestellt. Optimaler Weise werden auch die Anwenderdaten offline gehalten, um eine umfangreiche Nutzung der Anwendung auch im Offline Modus zu gewährleisten. Ersteres wird ermöglicht durch die

²⁷ Vgl. Steiner, 2018, S. 792.

Einführung von Service Workern und der Cache API, zweiteres durch clientseitige Speichertechnologien wie der Indexed DB oder Local Storage.

3.1.5 Aktuell (Fresh)

Die Eigenschaft fresh steht unmittelbar in Zusammenhang mit der zuvor beschriebenen Eigenschaft der Netzwerkverbindungsunabhängigkeit. Trotz einer lokal zwischengespeicherten Offlinekopie der Quelldateien wird eine neue Version der Anwendung schnellstmöglich für Benutzende bereitgestellt. Dies wird mithilfe des Service-Worker-Update Prozesses umgesetzt.

3.1.6 Sicher (Safe)

Grundvoraussetzung für die Installation eines Service Workers, auf welchem das Konzept der PWA basiert, ist die Übertragung der Quelldateien und Anwenderdaten über eine gesicherte Verbindung. Eine PWA wird daher ausschließlich über das Hypertext Transfer Protocol Secure (HTTPS) ausgeliefert. Dadurch können die Anwenderdaten verschlüsselt zur Gegenseite übertragen werden und sind vor dem Ausspähen und vor Manipulation geschützt. Da die PWA in einem Webbrowser ausgeführt wird, unterliegt sie zudem auch dessen Sicherheitsanforderungen. Die PWA läuft isoliert von zeitgleich geöffneten Webseiten in der Sandbox des Browsers. Im Webbrowser finden zudem diverse Sicherheitskonzepte Anwendung, wie bspw. das Konzept der Same-Origin-Policy, welches verhindert, dass Elemente aus fremden Quellen ohne zusätzliche Maßnahmen geladen werden können. Auch die URL ist als Sicherheitsmerkmal einer PWA zu betrachten, da sie der eindeutige Identifikator der Anwendung ist und somit ein Imitat der Anwendung für Anwendende durch die divergente URL erkennbar wäre.

3.1.7 Auffindbarkeit (Discoverable)

PWAs und deren Inhalte lassen sich von Suchmaschinen vollständig indexieren und auffinden, da sie wie klassische Webseiten auf Webtechnologien basieren und im Browser ausgeführt werden.²⁸ Ebenso kann eine Distribution über die App Stores der verschiedenen Plattformen erfolgen.²⁹ PWAs sind außerdem vom Browser als solche identifizierbar und so von klassischen Webseiten differenzierbar. Dies basiert auf dem Web-App-Manifest als Bestandteil einer PWA. PWAs können so auch von speziellen Crawlern gefunden werden und automatisiert in App Stores bereitgestellt werden.³⁰

²⁸ Vgl. Microsoft, 2022c.

²⁹ Vgl. Tremp, 2021, S.102.

³⁰ Vgl. Rubino, 2018.

3.1.8 Installierbarkeit (Installable)

Die Eigenschaft Installierbarkeit einer PWA beschreibt die Möglichkeit zur Erstellung einer Verknüpfung in der Programmliste des Betriebssystems bzw. des Homescreens eines Mobilgeräts. Die PWA ist aber auch ohne eine Installation voll funktionsfähig. Die Installation dient dazu, die Anwendung dem Look and Feel einer nativen App anzugleichen. Die Anwendung kann dadurch über ein Icon auf dem Homescreen oder Desktop des Endgeräts in einem eigenen nativen Fenster im Vollbildmodus geöffnet werden.

3.1.9 Reaktivierbar (Re-engageable)

Diese Eigenschaft zielt darauf ab, Benutzende der PWA, dazu zu animieren, die Anwendung erneut zu nutzen. Umgesetzt wird dies durch das Senden von Push-Benachrichtigungen, um Benutzende über externe Ereignisse oder abgeschlossene Vorgänge innerhalb der Anwendung zu informieren. Eine weitere Umsetzungsmöglichkeit besteht darin, Benutzende über ein Badge Icon zu informieren, dass neue Informationen innerhalb der Anwendung zur Verfügung stehen. Das Badge wird dabei an dem Icon der Anwendung auf dem Home Screen angezeigt. Technisch realisiert wird dies durch den Einsatz des Service Workers in Verbindung mit Push API und Notifications API unter der Nutzung des Web Push Protokolls, als einheitliche Schnittstelle zum Versand von Push-Benachrichtigungen, bzw. der App Badge API für die Nutzerinformation über Badges.³¹

3.1.10 Verlinkbarkeit (Linkable)

Die Eigenschaft Verlinkbarkeit beschreibt die Fähigkeit auf eine Anwendung mittels einer URL zu verweisen. Dabei kann durch den Deep Linking Mechanismus bis zu einer bestimmten Zielsicht oder einem Zielzustand der Anwendung verwiesen werden. Die technische Umsetzung basiert auf der Implementierung verschiedener clientseitiger Routingstrategien für Single-Page Web Applications. Dabei wird einer bestimmten Sicht der Anwendung eine spezifische URL zugewiesen.

3.2 Funktionsweise

PWAs bauen maßgeblich auf der Nutzung diverser Kernkomponenten auf. Durch diese erhält die PWA die in Abschnitt 3.1 genannten Eigenschaften. Die Betrachtung dieser Kernkomponenten erfolgt in den folgenden Unterabschnitten.

³¹ Vgl. Microsoft Edge, 2022.

3.2.1 Service Worker

Der Service Worker ist eine Sonderform des Web Workers. Web Worker ermöglichen Multithreading, also die parallele Ausführung von Programmcode, im Kontext des Webbrowsers. Dies hat den Vorteil, dass rechenintensive Operationen in einem separaten Thread im Hintergrund ausgeführt werden können, ohne den Hauptthread zu verlangsamen oder zu blockieren. Ein Web Worker ist ein Objekt, das mit einem Konstruktor erzeugt wird und eine JavaScript-Datei ausführt. Diese Datei enthält Code, der in einem eigenen Worker-Thread im Hintergrund ausgeführt wird.³² Der Service Worker, als eine Sonderform des Web Workers, ist im Gegensatz zu diesem nicht an die Webanwendung gebunden. Er wird im Webbrowser registriert und läuft im Hintergrund weiter, auch wenn die Anwendung geschlossen wurde. Es ist dem Service Worker nicht möglich, das Document Object Model (DOM) zu manipulieren und er hat keinen direkten Zugriff auf die Webanwendung. Er kann Requests der Anwendung und Responses des Web Servers abfangen und manipulieren und über ein Nachrichtensystem Daten mit dem Hauptthread austauschen. Er fungiert so als programmierbarer Netzwerk-Proxy zwischen dem Webserver und dem Browser.³³ Aus Sicherheitsgründen ist die Nutzung des Service Workers nur über eine gesicherte, verschlüsselte Verbindung über das HTTPS-Protokoll möglich.³⁴ Der Service Worker wird insbesondere dafür verwendet, Webanwendungen offline verfügbar zu machen und Push-Benachrichtigungen zu senden. Die Funktionsweise dieser zwei Anwendungsfälle wird im Folgenden betrachtet.

3.2.1.1 Offline-Funktionalität

Abbildung 1 veranschaulicht, wie der Service Worker in Kombination mit der Cache API für die Bereitstellung von Offline-Funktionalität einer Webanwendung eingesetzt wird. Beim ersten Aufruf der Anwendung über die URL wird der Service Worker vom Hauptthread registriert und installiert. Nach einer erfolgreichen Installation kann der Service Worker Inhalte in Form von Request-Response-Paaren im Cache speichern. Bei erneutem Aufruf der Webseite kann der Service Worker auch ohne eine bestehende Verbindung zum Internet die Anwendungsdateien aus dem Cache laden. Generell können verschiedene Caching Strategien implementiert werden, um Daten präferiert entweder zuerst aus dem Cache oder vom Web Server zu verwenden.³⁵ Es können sowohl die statischen Quelldateien der Anwendung als auch dynamische Daten von API-Anfragen im Cache gespeichert werden.

³² Vgl. MDN, 2022b.

³³ Vgl. Reuter, 2021.

³⁴ Vgl. Rytte, o. D.

³⁵ Vgl. Siegrist, 2021.

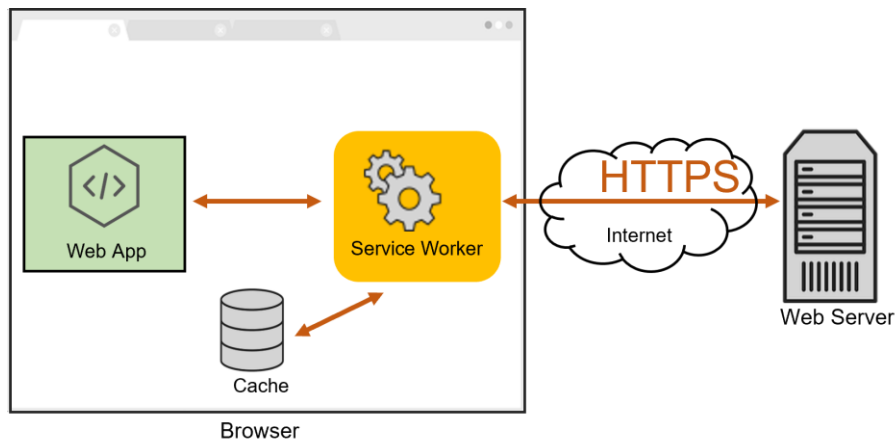


Abbildung 1: Funktionsweise des Service Workers
Quelle: Eigene Darstellung in Anlehnung an Siegrist, 2021.

3.2.1.2 Push-Benachrichtigungen

Da der Service Worker auch nach dem Schließen der Anwendung im Hintergrund aktiv bleibt, wird er in Kombination mit der Push API und der Notifications API für das Senden von Push-Benachrichtigungen eingesetzt. Die Notifications API ist dabei die Schnittstelle zwischen dem Browser bzw. der Anwendung und dem Betriebssystem und dient dazu dem Benutzenden Benachrichtigungen außerhalb der Anwendung anzuzeigen. Sie kann auch unabhängig von Service Worker und Push API verwendet werden, um Benachrichtigungen anzuzeigen, während die Anwendung aktiv ist. Über die Notifications API erfolgt zudem die Berechtigungsanfrage an den Benutzenden für das Senden der Benachrichtigungen. Um Benachrichtigungen über einen Webserver zu empfangen, unabhängig davon, ob die Anwendung im Vordergrund aktiv ist, im Hintergrund läuft oder geschlossen ist, benötigt es die Anbindung an die Push API. Die Push API dient als Schnittstelle zwischen Webbrowser bzw. Service Worker und einem Push Dienst der jeweiligen Plattform. Für Google Chrome ist dieser Push Dienst *Firebase Cloud Messaging*, für Mozilla Firefox *Mozilla Web Push* und Microsoft Edge nutzt den *Windows Push Notification Service*. Die Kommunikation erfolgt verschlüsselt über das Web Push Protokoll. Der Service Worker kann über die Push API Benachrichtigungen bei einem Push Dienst abonnieren. Der Server der Anwendung sendet Benachrichtigungen dann an den Push Dienst und der Push Dienst übermittelt die Nachricht an den Service Worker. Der Service Worker gibt die Benachrichtigung an die Anwendung weiter. Die Anwendung wiederum nutzt dann die Notifications API, um die Nachricht über das Betriebssystem anzuzeigen.³⁶ Abbildung 2 veranschaulicht die beschriebenen Zusammenhänge.

³⁶ Vgl. Liebel, 2019, S.125f.

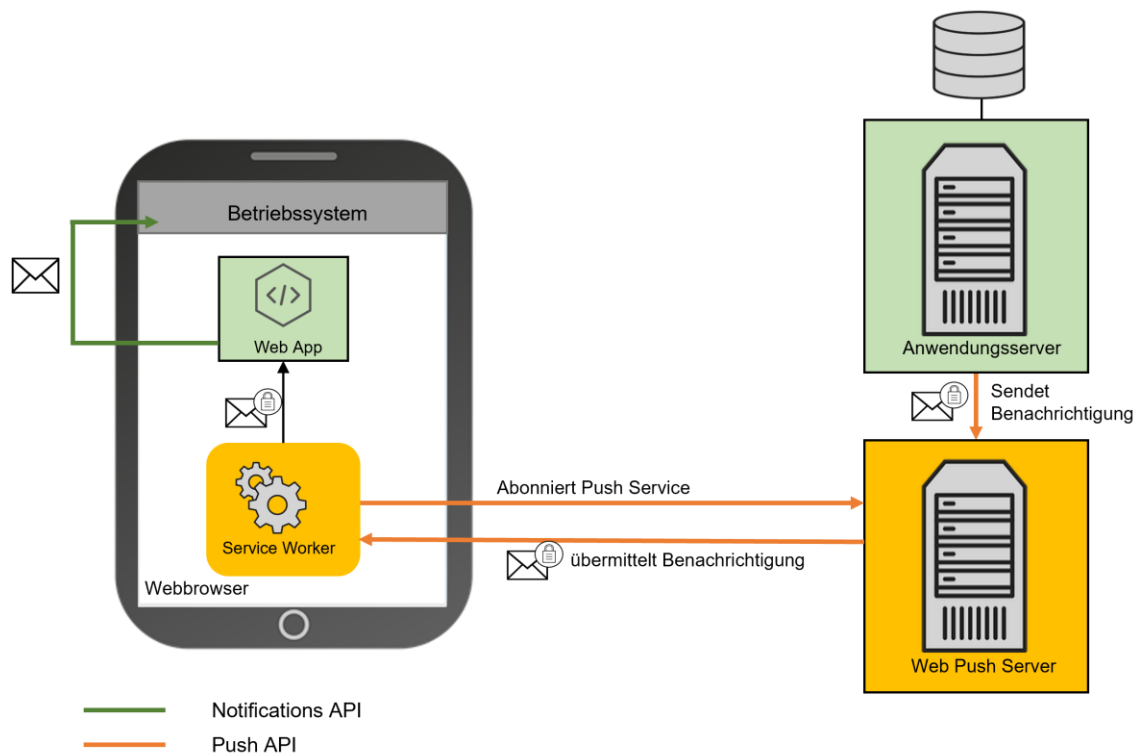


Abbildung 2: Funktionsweise von Push-Benachrichtigungen bei PWAs
Quelle: Eigene Darstellung.

3.2.2 Web App Manifest

Bei dem Web App Manifest handelt sich um eine Spezifikationsdatei im JSON-Format, in der Metadaten der Anwendung ausgewiesen werden. Konkret enthalten sind Informationen wie Name, Beschreibung und Icons der Anwendung, Informationen zur Theme-Farbe und zum Anzeigemodus sowie die Start-URL. Eine vollständige Liste aller Attribute kann dem *Anhang A: Inhalt des Web App Manifest nach dem W3C Standard* entnommen werden. Das Web App Manifest wird für die Installation der App benötigt und dient dazu, die PWA in ihrem Aussehen und Verhalten einer nativen Anwendung anzugleichen.³⁷ Auf die Manifest-Datei wird aus der HTML-Datei der PWA heraus über ein `link` Element verwiesen.³⁸ Der Browser kann die PWA so von klassischen Web Apps unterscheiden und Benutzende auf die Möglichkeit der Installation der App hinweisen. Durch die Installation wird ein Icon auf der Betriebssystemoberfläche des Endgeräts erstellt, über welches die App gestartet werden kann. Beim Aufruf der installierten PWA über das Icon startet der Webbrowser anhand der Informationen im Web App Manifest.³⁹ Die Anwendung wird über die für das Attribut `start_url` definierte URL

³⁷ Vgl. Sheppard, 2017, S.95.

³⁸ Vgl. W3C, 2022.

³⁹ Vgl. Tremp, 2021, S.101.

aufgerufen. Die Attribute `name`, `icons` und `background_color` können dafür genutzt werden einen Splash Screen anzuzeigen, während die App lädt. Wird das Attribut `display` mit dem Wert "standalone" oder "minimal-ui" definiert, startet die Anwendung in einem unsichtbaren Container und Toolbar sowie Navigationselemente des Browsers sind für Benutzende unsichtbar. Vom Look and Feel verhält sich die PWA dadurch für Benutzende ähnlich einer nativen App. Nach dem Prinzip von Progressive Enhancement wird die Manifest-Datei bei der Ausführung der PWA ignoriert, wenn der verwendete Browser diese Funktion nicht unterstützt. Abbildung 3 veranschaulicht die Architektur und Funktionsweise einer PWA mit dem Web App Manifest und dem Service Worker als Kernkomponenten.

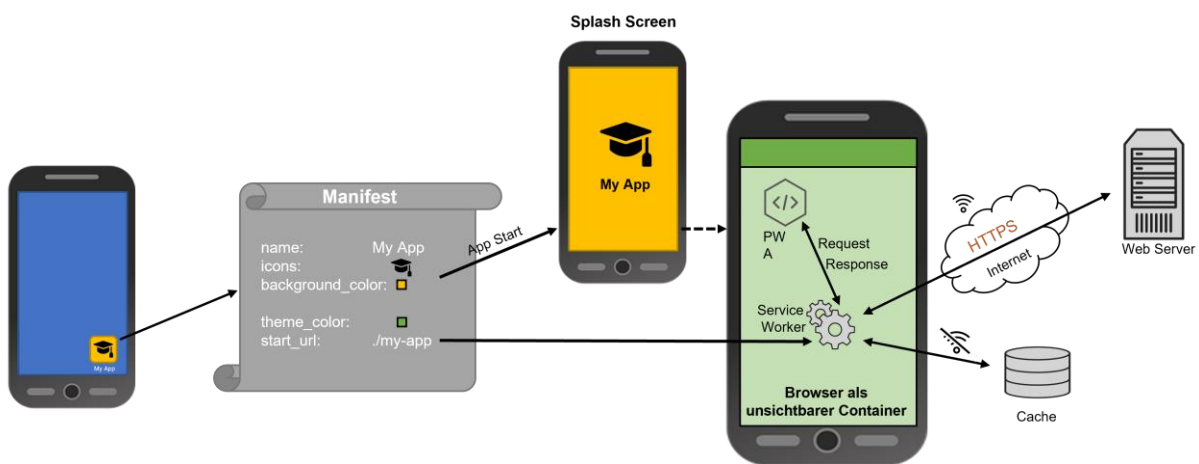


Abbildung 3: Architektur und Funktionsweise einer PWA
Quelle: In Anlehnung an Tremp, 2021.

3.2.3 Web APIs

Damit Webanwendungen Funktionen bereitstellen können, die den Zugriff auf native Schnittstellen des Betriebssystems des Endgeräts erfordern, werden diverse Webschnittstellen entwickelt. Tabelle 1 zeigt eine Auswahl bestehender Schnittstellen, die Entwickler:innen zur freien Nutzung bereitstehen, und deren Verwendungsmöglichkeiten. Die Browserunterstützung variiert dabei. Eine erweiterte Betrachtung der Browserunterstützung folgt in Kapitel 3.3. Es kommt auch hier wieder das Prinzip von Progressive Enhancement zum Einsatz. Sollte eine Schnittstelle in einem verwendeten Browser nicht verfügbar sein, kann dieses Feature in der Anwendung nicht genutzt werden. Die Basisfunktionen der Anwendung stehen dem Benutzenden dennoch zur Verfügung.

Schnittstelle	Verwendung
Audio Output Devices API	Audioausgabe auf verbundenen Geräten (z.B. Bluetooth-Headset)
Badging API	Benachrichtigungsfunktion über ein Badge am App-Icon
Battery Status API	Auslesen des Akkustands
Contact Picker API	Zugriff auf Kontaktinformationen des Nutzers
Generic Sensor API	Zugriff auf Sensoren (Accelerometer, Gyroskop, Magnetometer) zum Auslesen von Beschleunigung, Drehungen, magnetischer Flussdichte und Umgebungsbeleuchtung
Geolocation API	Auslesen des Gerätestandorts
IndexedDB API	clientseitige Speicherung strukturierter Daten inkl. Dateien
Media Capture and Stream API	auf Kamera und Mikrofon zugreifen
Notifications API	Darstellung von Push-Benachrichtigungen auf Level des Betriebssystems
Push API	Senden von Push-Benachrichtigungen
Vibration API	Auslösen des Vibrationsmechanismus des Endgeräts
Web Bluetooth API	Interaktion mit Bluetooth Geräten
Web NFC	Kommunikation mit NFC-Tags
Web Share API	Inhalte über einen nativen Teilen-Dialog mit anderen Apps teilen
Web Speech API	Spracheingaben in Text übersetzen und vice versa
Web Storage API	clientseitiges Speichern von Schlüssel-Wert-Paaren
Web XR Device API	Zugriff auf Virtual Reality (VR) und Augmented Reality (AR) Geräte

Tabelle 2: Ausgewählte Web-APIs und deren Verwendungszweck

Quelle: Eigene Darstellung. Datenquelle: MDN 2020.

3.3 Unterstützung

Wie in Kapitel 3.2 erörtert, basieren PWAs auf modernen HTML5 APIs, zur Umsetzung verschiedener Eigenschaften und Funktionen. Um diese nutzen zu können, müssen die APIs vom verwendeten Browser und Betriebssystem unterstützt werden. Die folgenden Tabellen stellen die Browserunterstützung der Eigenschaften und Funktionen von PWAs durch HTML5 APIs auf Basis der Daten von Can I use übersichtlich dar. Dargestellt werden die Daten für die jeweils neuste Version der Browser Chrome, Firefox, Edge, Opera und Internet Explorer und die mobilen Browserversionen Chrome for Android, Firefox for Android, Opera Mobile und Safari on iOS. Die Unterstützung wird dabei mit einem grünen Haken (✓) gekennzeichnet, die Nicht-Unterstützung mit einem roten X (✗). In einigen Fällen werden nur Teilfunktionalitäten einer Schnittstelle unterstützt. Dies wird mit einem orangenen Kreis (○) dargestellt. Ist die Unterstützung unbekannt, wird dies durch ein schwarzes Minus (–) markiert.











	Desktop Browser						Mobile Browser			
	 V. 99	 V. 98	 V. 98	 V. 83	 V. 15.3	 V. 11	 for And- roid V. 98	 for And- roid V. 96	 Mobile V. 64	 on iOS V. 15.3
CSS Flexible Box	✓	✓	✓	✓	✓	○	✓	✓	✓	✓
CSS Grid Layout	✓	✓	✓	✓	✓	○	✓	✓	✓	✓
CSS3 Media Queries	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Tabelle 3: Browserunterstützung für responsive Design
Quelle: Eigene Darstellung. Datenquelle: Can I use, 2022.

Die Umsetzung der Eigenschaft responsive wird grundlegend von allen Browsern unterstützt (siehe Tabelle 3). Einschränkungen gibt es für den Internet Explorer beim CSS Flexible Box Layout durch ungelöste Bugs.⁴⁰ Beim CSS Grid Layout wird vom Internet Explorer nur eine veraltete Spezifikation aus dem Jahr 2011 unterstützt.⁴¹











	Desktop Browser						Mobile Browser			
	 V. 99	 V. 98	 V. 98	 V. 83	 V. 15.3	 V. 11	 for And- roid V. 98	 for And- roid V. 96	 Mobile V. 64	 on iOS V. 15.3
Installierbarkeit	✓	✗	✓	✗	✗	✗	✓	✓	✓	○
Web App Manifest Attribute										
name	✓	–	✓	–	–	✗	✓	○	–	✓
icons	✓	–	✓	–	–	✗	✓	○	–	✗
background_color	–	–	✓	–	–	✗	✓	–	–	✗
theme_color	✓	–	✓	–	–	✗	✓	○	–	✓

Tabelle 4: Browserunterstützung des Web App Manifests
Quelle: Eigene Darstellung. Datenquelle: Can I use, 2022.

In Tabelle 4 wird die Unterstützung des Web App Manifests dargestellt. Dabei variiert die Unterstützung für einzelne Attribute des Manifests. Die in Tabelle 4 gelisteten Attribute sind daher von Bedeutung, insofern sie die Anwendung mit einem nativen Aussehen und Verhalten wie bspw. dem Splash Screen bei Start der App versehen (siehe Kapitel 3.2.2). Im Browser Firefox für Android wird aktuell nur experimentelle Unterstützung für diese Funktion angeboten. Die Unterstützung muss daher in dem Firefox für Android Browser durch zusätzliche Konfiguration

⁴⁰ Vgl. CSS Flexible Box Layout Module, o. D.

⁴¹ Vgl. CSS Grid Layout (level 1), o. D.

erst aktiviert werden.⁴² Für mobile iOS Endgeräte wird eine Installation nur über den Safari on iOS Browser direkt und nicht für WebViews von Chrome und Firefox unterstützt.⁴³ Zudem werden für die aktuelle Version 15.3 des Safari on iOS Browsers die Manifestattribute `background_color` und `icons` nicht unterstützt. Daher ist es über Safari on iOS nicht möglich, ein Home Screen Icon oder einen Splash Screen anhand der definierten Attributwerte des Web App Manifests anzuzeigen. Es gibt jedoch Alternativen für die Implementierung der Funktionalitäten für iOS, welche in Kapitel 4.3.7.1 vorgestellt werden. Eine Unterstützung für das Attribut `icons` ist für die Version 15.4 vorgesehen.⁴⁴











	Desktop Browser						Mobile Browser			
	 V. 99	 V. 98	 V. 98	 V. 83	 V. 15.3	 V. 11	 for Android V. 98	 for Android V. 96	 Mobile V. 64	 on iOS V. 15.3
Service Worker API	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Cache API	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
IndexedDB	✓	✓	✓	✓	✓	○	✓	✓	✓	✓
Local Storage	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Tabelle 5: Browserunterstützung für Web APIs zur Umsetzung von Offline Funktionalitäten
Quelle: Eigene Darstellung. Datenquelle: Can I use, 2022.

Wie Tabelle 5 zeigt, werden Schnittstellen zur Umsetzung der Offlinefähigkeit einer PWA von allen Browsern mit Ausnahme des Internet Explorers vollständig unterstützt. Dies betrifft die Service Worker Schnittstelle in Verbindung mit der Cache API als auch clientseitige Speichertechnologien. Die clientseitigen Speichertechnologien Indexed DB und Local Storage werden zumindest teilweise auch vom Internet Explorer unterstützt.

⁴² Vgl. MDN, 2022a.

⁴³ Vgl. Add to home screen (A2HS), o. D.

⁴⁴ Vgl. Manifest: icons, o. D.











	Desktop Browser						Mobile Browser			
	 V. 99	 V. 98	 V. 98	 V. 83	 V. 15.3	 V. 11	 for And- roid V. 98	 for And- roid V. 96	 Mobile V. 64	 on iOS V. 15.3
Notifications API: badge	✓	✗	✓	✓	✗	✗	✓	✗	✓	✗
Notifications API	✓	✓	✓	○	✗	✗	✓	✓	✓	✗
Push API	✓	✓	✓	✓	✗	✗	✓	✓	✓	✗

Tabelle 6: Browserunterstützung nativer Benachrichtigungsfunktionen

Quelle: Eigene Darstellung. Datenquelle: Can I use, 2022.

Die Unterstützung der Schnittstellen für Push-Benachrichtigungen ist bei den Browsern Chrome, Edge, Opera und Firefox sowohl für die Desktopversionen der Browser als auch die mobilen Versionen gegeben (siehe Tabelle 6). Benachrichtigungen durch Badges werden dabei von Firefox nicht unterstützt. Safari und Internet Explorer bieten keine Unterstützung für Push-Benachrichtigungen und Badges. Eine, vorerst experimentelle, Unterstützung für die Push API und Notifications API ist Firtman zufolge für zukünftige Versionen des Safari Browsers vorgesehen.⁴⁵











	Desktop Browser						Mobile Browser			
	 V. 99	 V. 98	 V. 98	 V. 83	 V. 15.3	 V. 11	 for And- roid V. 98	 for And- roid V. 96	 Mobile V. 64	 on iOS V. 15.3
Audio Output Devices API	✓	✓	✓	✓	✓	○	✓	✓	✓	✓
Battery Status API	✓	✗	✓	✓	✗	✗	✓	✗	✓	✗
Contacts Manager API	✗	✗	✗	✓	✗	✗	✓	✓	✓	✗
Geolocation API	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Kamera- und Mikrofon- zugriff	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Spracheingabe	○	✗	✗	✗	○	✗	○	✗	✗	○
Text zu Sprache Ausgabe	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Vibration API	✓	✓	✓	✓	✗	✗	✓	✓	✓	✗
Web Share API	○	✗	○	✗	✓	✗	✓	✓	✗	✓

Tabelle 7: Browserunterstützung - native Funktionen und Gerätezugriff

Quelle: Eigene Darstellung. Datenquelle: Can I use, 2022.

⁴⁵ Vgl. Firtman, 2022.

In Tabelle 7 wird die Browserunterstützung für Web APIs betrachtet, über die auf native Funktionen des Betriebssystems zugegriffen werden kann. Im mobilen Chrome Browser für Android werden diese APIs am umfangreichsten unterstützt. Einzig die Schnittstelle zur Spracheingabe steht in diesem Browser nur partiell zur Verfügung. Auch die Desktopversion des Browsers bietet eine gute Unterstützung der Web APIs an. Den geringsten Support für die aufgezeigten APIs weist der Browser Internet Explorer auf, der nur die Schnittstelle für die Geolokalisierung unterstützt. Auch der Safari Browser bietet deutlich weniger Unterstützung für die aufgezeigten APIs im Vergleich zu den anderen Browsern.











	Desktop Browser						Mobile Browser			
	 V. 99	 V. 98	 V. 98	 V. 83	 V. 15.3	 V. 11	 for Android V. 98	 for Android V. 96	 Mobile V. 64	 on iOS V. 15.3
Accelerometer	✓	✗	✓	✓	✗	✗	✓	✗	✗	✗
Gyroskop	✓	✗	✓	✓	✗	✗	✓	✓	✓	✗
Magnetometer	✓	✗	✓	✓	✗	✗	✓	✗	✗	✗

Tabelle 8: Browserunterstützung beim Zugriff auf Gerätesensoren

Quelle: Eigene Darstellung. Datenquelle: Can I use, 2022.

Schnittstellen für den Zugriff auf Sensoren des Endgeräts, zum Auslesen von Beschleunigung, Neigung und magnetischer Flussdichte, werden vollständig von den Browsern Chrome, Edge und dem Opera Desktop Browser unterstützt (siehe Tabelle 8). Die Browser Internet Explorer, Safari und Firefox in der Desktopversion bieten bisher keinen Support für diese APIs.











	Desktop Browser						Mobile Browser			
	 V. 99	 V. 98	 V. 98	 V. 83	 V. 15.3	 V. 11	 For Android V. 98	 For Android V. 96	 Mobile V. 64	 on iOS V. 15.3
Web Bluetooth API	✓	✗	✓	✓	✗	✗	✓	✗	✓	✗
Web NFC	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗

Tabelle 9: Browserunterstützung für Schnittstellen zum Datenaustausch

Quelle: Eigene Darstellung. Datenquelle: Can I use, 2022.

Abschließend wird in Tabelle 9 die Browserunterstützung der Bluetooth- und NFC-Technologie zum drahtlosen Datenaustausch mit anderen Geräten betrachtet. Die NFC API wird zurzeit nur im Chrome Browser für Android unterstützt. Die Unterstützung für die Bluetooth-API fällt

etwas umfangreicher aus. Sie wird von den Browsern Chrome, Edge und Opera für die Desktopversionen und die mobilen Versionen der Browser unterstützt.

3.4 Realisierungsoptionen

Für die technische Realisierung einer PWA bestehen verschiedene Möglichkeiten. Für die Erstellung einer PWA muss insbesondere die Implementierung der Kernkomponenten Service Worker und Web App Manifest erfolgen. Für diese gibt es Ansätze von unterschiedlicher Komplexität.

Einfaches JavaScript (Vanilla JS)

Bei diesem Ansatz wird der Service Worker manuell anhand der Spezifikationsdokumente⁴⁶ auf Basis von klassischem JavaScript erstellt und konfiguriert. Das Web App Manifest wird ebenfalls manuell erstellt und in das index.html Dokument eingebunden.

JavaScript Bibliothek Workbox

Workbox ist eine von Google bereitgestellte frameworkunabhängige JavaScript Bibliothek, die die Entwicklung eines Service Workers erleichtert. Mit Workbox lässt sich ein Service Worker generieren, der durch eine Konfigurationsdatei spezifiziert wird: Für mehr Gestaltungsfreiheiten besteht zudem die Möglichkeit einen eigenen Service Worker zu entwickeln. Dafür stellt Workbox nützliche APIs zur Verfügung, die das Caching verschiedener Assets erleichtern.⁴⁷

SPA-Frameworks wie Angular, ReactJS oder Vue.js

Oft ist die Entwicklung auf Basis eines Single-Page Application Frameworks sinnvoll. Diese enthalten eine Reihe an Tools und Bibliotheken, die die Entwicklung einer Anwendung vereinfachen, die Produktion von Code-Duplikaten vermeiden und so die Effizienz bei der Entwicklung steigern. Für die PWA-Unterstützung gibt es diverse Node Packages, welche in das Projekt eingebunden werden können. Bei Angular ist dies das Package `@angular/pwa`⁴⁸, ReactJS nutzt `create-react-app`⁴⁹ und Vue.js stellt das Plugin `@vue/cli-plugin-pwa` bereit, welches wiederum auf der Workbox Bibliothek basiert.⁵⁰ Durch die Einbindung der Packages wird sowohl der Service Worker als auch das Web App Manifest in das Projekt eingebunden. Der Vorteil bei der Nutzung der Node-Packages ist es, dass bei der Installation des Packages im Framework über ein Command Line Interface Tool (CLI) bereits Code-Fragmente, sogenannter

⁴⁶ Vgl. W3C, 2021.

⁴⁷ Vgl. Workbox | Google Developers, o. D.

⁴⁸ Vgl. Getting started with service workers, o. D.

⁴⁹ Vgl. Facebook/Karrrys, 2021.

⁵⁰ Vgl. `@vue/cli-plugin-pwa`, o. D.

Boilerplate-Code, vom Framework automatisiert vorgeneriert werden. Dabei wird die Datei für den Service Worker als auch die Datei für das Web App Manifest automatisiert erstellt und an den entsprechenden Stellen im Framework registriert. Anpassungen sind über eine separate Konfigurationsdatei einfach umzusetzen ohne zusätzlichen Kodierungsaufwand. Zudem wird das Prinzip von Progressive Enhancement in den Implementierungen des Packages bereits berücksichtigt und erfordert auch in diesem Bereich keinen zusätzlichen Kodierungsaufwand. Die genannten Node Packages können jederzeit zu einer bestehenden Anwendung hinzugefügt werden insofern diese mit dem Framework entwickelt wurde. So lassen sich auch bestehende Webanwendungen einfach und schnell in eine PWA überführen.

PWA Builder

Bei PWA Builder handelt es sich um ein Tool von Microsoft, mit dem die Umwandlung einer Website zu einer PWA durchgeführt werden kann. Durch Angabe der Website-URL prüft PWA Builder, welche zusätzlichen Dateien für die Umwandlung in eine PWA benötigt werden und geleiten Entwickler:innen im Anschluss durch den Prozess die fehlenden Dateien zu erstellen. Insbesondere für die Erstellung des Service Workers gibt es dabei zahlreiche Optionen, die passend für den individuellen Anwendungsfall gewählt werden können.⁵¹ Über den PWA Builder können die Apps zudem auch für eine Veröffentlichung in den plattformspezifischen App Stores vorbereitet werden. Dafür stellt PWA Builder verschiedene Store Packages zum Download zur Verfügung inklusive einer Anleitung für weitere notwendige Schritte zur Veröffentlichung der PWA in den jeweiligen App Stores.^{52 53}

⁵¹ Vgl. Kennedy et al., 2022.

⁵² Vgl. Build Your iOS App, 2021.

⁵³ Vgl. Generating your Android package, 2021.

4 Technische Umsetzung der Hochschul-App

In diesem Kapitel werden die Konzeption und Implementierung des Prototyps der Progressive Web App für die Ernst-Abbe-Hochschule Jena beschrieben. Dafür werden zunächst die Anforderungen an die Anwendung definiert. Es folgen Erläuterungen zur Konzeption und Implementierung und anschließend eine kritische Betrachtung der Umsetzung.

4.1 Anforderungen

4.1.1 Umzusetzende Features

Die umzusetzende Anwendung orientiert sich an der bestehenden Android-App EAH Jena. Umzusetzen sind die Features Stundenplan, Mein Stundenplan und Mensa. Das Feature Stundenplan umfasst die Darstellung von Stundenplandaten basierend auf der Auswahl eines Studiengangs, Semesters und einer Setgruppe. Das Feature Mein Stundenplan dient ebenfalls zur Visualisierung von Stundenplandaten. Jedoch kann hier ein individueller Stundenplan erstellt werden. Dafür kann aus dem gesamten Kursangebot der Hochschule eine individuelle Selektion getroffen werden. Für das Feature Mensa sollen Daten zum Speisenangebot der verschiedenen Mensen in Jena abgerufen und dargestellt werden.

4.1.2 Datenquelle

Als Datenquelle dient eine REST-API, die von der Fachhochschule Erfurt bereitgestellt wird. Diese wird bereits von der Android-App konsumiert. Für die Umsetzung der in Abschnitt 4.1.1 genannten Features stehen vier Endpoints zur Verfügung. Eine Dokumentation dieser vier Endpoints der Schnittstelle wird in *Anhang B: Dokumentation der REST-API der Fachhochschule Erfurt* bereitgestellt.

4.1.3 Funktionale und nicht-funktionale Anforderungen an die Anwendung

- (1) Anwenderdaten zu erstellten Stundenplänen sollen sessionübergreifend verfügbar bleiben.
- (2) Zu speichernde Daten des Benutzenden sollen dabei dezentral am Client gespeichert werden. Dadurch wird eine zentrale serverseitige Speicherung personenbezogener Daten vermieden, die mit erhöhten Anforderungen an den Datenschutz einherginge.
- (3) Die Anwendung soll offline funktionsfähig sein.
- (4) Die dargestellten Daten in der Anwendung sollen aktuell sein.
- (5) Die Benutzeroberfläche (UI) der Anwendung muss den Richtlinien des Corporate Designs der Ernst-Abbe-Hochschule Jena entsprechen. Eine Zusammenstellung der relevanten Richtlinien kann in *Anhang C: Corporate Design Richtlinien der Ernst-Abbe-Hochschule Jena* eingesehen werden.

- (6) Die Anwendung soll einen Nutzungskomfort für Benutzende bieten, der vergleichbar mit dem einer nativen App ist.
- (7) Dies umfasst auch eine angemessen gute Performance der Anwendung. Die Anwendung sollte sich flüssig bedienen lassen.
- (8) Die Anwendung soll alle Aspekte einer PWA erfüllen, die über den Google Lighthouse Test geprüft werden.

4.2 Konzeption

4.2.1 Framework-Auswahl

Für die weitere Konzeption der Anwendung wurde zunächst eine Technologie für die Umsetzung gewählt. Wie in Kapitel 3.4 erläutert wurde, ist die Nutzung eines SPA Frameworks für die Implementierung von PWAs von Vorteil. Daher wird die Anwendung im Angular Framework entwickelt, nicht zuletzt auch, weil die Autorin im Umgang mit diesem Framework bereits vertraut ist und so auf Erfahrungswerte bei der Entwicklung der Anwendung zurückgegriffen werden kann. Im Folgenden wird der grundlegende Aufbau des Frameworks beschrieben.

Angular ist ein von Google entwickeltes TypeScript-basiertes Frontend Framework zur Entwicklung von Single-Page Web Applications. Bei TypeScript handelt es sich um ein Superset von JavaScript und somit um eine Implementierung von ECMAScript. JavaScript wird dabei insbesondere um Features im Bereich Objektorientierung, eine umfassende Typisierung und Annotations erweitert. Die Typsicherheit unterstützt Entwickler:innen und sichert eine höhere Code-Qualität. Kompiliert wird TypeScript vom TypeScript-Compiler zu JavaScript, da nur dies im Browser ausführbar ist.⁵⁴ Das Angular Framework ist modular aufgebaut. Angular Module fassen zusammengehörigen Code in funktionalen Sets zusammen. Eine Anwendung besteht aus mindestens einem Modul, dem Root-Modul, das den Bootstrap-Mechanismus bereitstellt, der die Anwendung startet. Weitere Funktionsmodule können optional je nach Bedarf eingebunden werden. Ein Modul besteht aus einer oder mehreren Komponenten. Zudem können Services mit den Komponenten verknüpft werden, um funktionale Einheiten zu bilden. Auch können Funktionalitäten anderer Module in ein Modul importiert werden.⁵⁵

Zwei Kernkonzepte bilden die Angular CLI und Components. Abbildung 4 veranschaulicht den Aufbau des Frameworks und zeigt zudem einige von Angular bereitgestellte Module, die zusätzlich eingebunden werden können. Im weiteren Verlauf dieses Kapitels werden einige der

⁵⁴ Vgl. Tremp, 2021, S.99.

⁵⁵ Vgl. Introduction to Angular concepts, o. D.

Module noch genauer betrachtet, die für die Konzeption und Implementierung der zu erstellenden Anwendung von Relevanz sind.

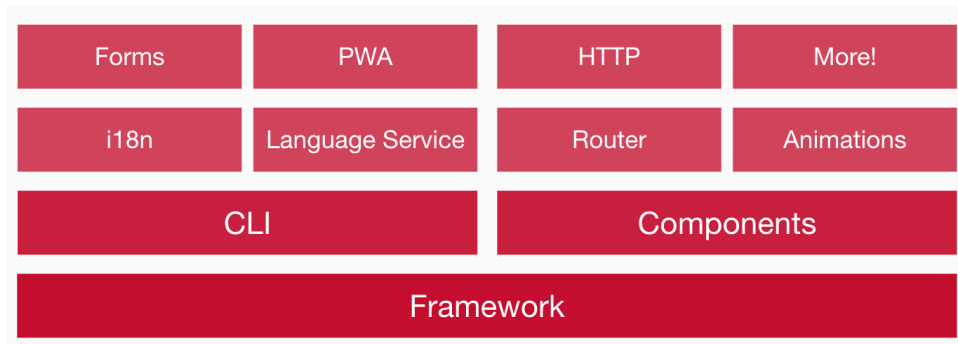


Abbildung 4: Aufbau des Angular Frameworks
Quelle: Böhm, 2020.

Angular-CLI

Die Angular-CLI ermöglicht es mit dem Kommandozeilenwerkzeug `ng` zu arbeiten. Dies soll Entwickler:innen bei der Initialisierung, Entwicklung, dem Scaffolding und der Wartung eines Angular-Projekts unterstützen.⁵⁶ Unter Scaffolding wird dabei die automatisierte Erzeugung eines Code-Grundgerüsts verstanden. Mithilfe von Angular Schematics und Angular CLI lassen sich so automatisiert Dateien mit einem Code-Gerüst für neue Artefakte wie Komponenten und Services generieren. Eine Übersicht der ausführbaren Befehle wird in *Anhang D: Übersicht über grundlegende Befehle der Angular-CLI* beigefügt.

Komponenten und Services

Komponenten sind die sogenannten Building Blocks einer Angular-Anwendung, also logische Bausteine, aus denen sich die Anwendung zusammensetzt. Sie sind Anzeigeelemente und werden als eigene HTML-Elemente definiert. „Abhängig von der definierten Anzeige-Logik und den aktuellen Daten stellen diese Elemente den Zustand der Anwendung dar“⁵⁷, führt Böhm aus. Jede Komponente besteht aus einem HTML-Template, einer TypeScript-Klasse, einem CSS-Selektor und optionalen Stylesheets. In dem HTML-Template wird die Darstellung der Komponente definiert. Die TypeScript Klasse beschreibt das Verhalten der Komponente und der CSS-Selektor beschreibt die Komponente als HTML-Element. Über diesen Selektor kann die Komponente in andere Templates eingebunden werden.

Services hingegen „definieren [die] Daten, Logik und Algorithmen der Anwendung. Sie sind modular und wiederverwendbar“⁵⁷ und „unabhängig von der Anzeige [der] Anwendung.“⁵⁷

⁵⁶ Vgl. CLI Overview and Command Reference, o. D.

⁵⁷ Böhm, 2020.

Entwurfsmuster: Model View Controller

Angular verwendet das Model View Controller (MVC) Pattern, wobei von Angular eine etwas abweichende Terminologie verwendet wird. In Abbildung 5 stellt Freeman eine clientseitige Implementierung des MVC Patterns und die verwendete Terminologie von Angular dar. Dabei stellt die Komponente (Component) den Controller dar und das Template die View. Die Daten im Model kommen dabei von einer serverseitigen Komponente, etwa über eine REST-API. Controller und View arbeiten dann mit den Daten im Model, um DOM-Manipulationen durchzuführen und HTML-Elemente zu erstellen oder zu manipulieren, mit denen Benutzende interagieren können. Diese Interaktionen werden an den Controller zurückgegeben, wodurch eine Schleife entsteht und damit eine interaktive Anwendung, beschreibt Freeman.⁵⁸

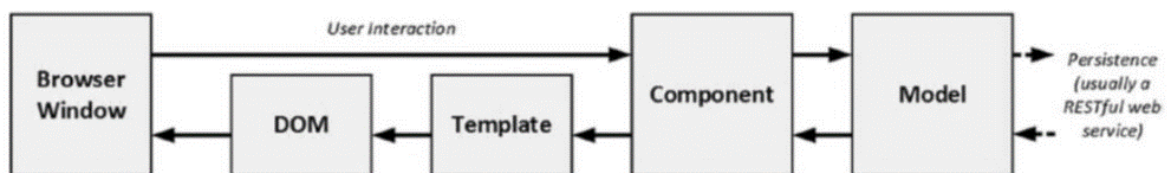


Abbildung 5: Angular Implementierung des MVC Patterns
Quelle: Freeman, 2020, S.39.

Dieser Aufbau des Frameworks ermöglicht eine gute Wartbarkeit, Testbarkeit und Wiederverwendbarkeit des Quellcodes.

4.2.2 UI- und UX-Konzept

Für die Umsetzung des User Interface wird Angular Material genutzt. Dies ist eine Bibliothek von UI-Komponenten, die sich an Googles Material Design orientiert. Die Komponentenbibliothek ist an das Framework angepasst und bietet die Möglichkeit mit verhältnismäßig geringem Aufwand optisch ansprechende und funktionale Benutzeroberflächen zu erstellen. Bei der Konzeption der UI wurden daher hauptsächlich Angular Material Komponenten verwendet.

Unter der Annahme, dass die App häufiger mit mobilen Endgeräten genutzt werden wird, wurde das User Interface (UI) Konzept nach der Mobile First Strategie entwickelt. Dabei wird zuerst die Benutzeroberfläche für mobile Endgeräte entworfen. So wird der Darstellung der Anwendung auf Smartphones eine höhere Priorität zugewiesen, um die User Experience (UX) gezielt für diese zu optimieren.

⁵⁸ Vgl. Freeman, 2020, S.40.

Das Farbkonzept der Anwendung basiert auf den Corporate Design Richtlinien der Ernst-Abbe-Hochschule Jena (siehe *Anhang C: Corporate Design Richtlinien der Ernst-Abbe-Hochschule Jena*). Demnach wird die Basisfarbe türkis mit dem Hexadezimal Farbcodex #009999 als primäre Farbe der Anwendung genutzt. Die App-Bar und Interaktionselemente der Anwendung sollen daher in dieser Farbe dargestellt werden. Für zusätzliche Farbakzente wird die Farbe tiefseeblau verwendet, die im Corporate Design der Hochschule als Akzentfarbe mit dem Hexadezimalcode #34495B definiert ist. Diese wird jedoch nur sparsam eingesetzt, um die Anwendung nicht farblich zu überladen.

Aus Gründen der Benutzerfreundlichkeit wird ein einheitliches Layout für die Anwendung verwendet. Grundlegend sind alle Ansichten in zwei Bereiche aufgeteilt. Die App Bar als Header der Anwendung und der Content-Bereich. Die App Bar besteht dabei aus einem Titel und einem Button, über den das Navigationsmenü erreichbar ist. Als Titel wird der Name des Features angezeigt in dessen Ansicht der Benutzende sich gerade befindet. Auch für die Hauptansichten der verschiedenen Features wurde ein einheitliches Layout konzipiert, sodass jede Ansicht ähnlich strukturiert ist. Im oberen Bereich der Ansicht werden die Interaktionselemente platziert, durch die Benutzende die anzuzeigenden Informationen konfigurieren können. Im Falle der Ansicht des Mensa-Features ist dies ein Auswahlfeld, um eine spezifische Mensa zu wählen, deren Speisenangebot angezeigt werden soll. Bei dem Stundenplan-Feature werden in diesem Bereich der selektierte Studiengang und das Semester angezeigt mit einem rechts daneben platzierten Button, über welchen sich diese Auswahl modifizieren lässt. Für das Mein Stundenplan-Feature sind in dem Interaktionsbereich Buttons vorgesehen, um die Kursauswahl zu konfigurieren oder um zu einer Ansicht mit den aktuell ausgewählten Kursen zu wechseln. Unter dem Interaktionsbereich ist für jedes der Feature ein Date Picker vorgesehen über den Benutzende einen Tag selektieren können. Darunter wiederum werden dann die jeweiligen Informationen für den ausgewählten Tag dargestellt. Abbildung 6 veranschaulicht den Aufbau des Layouts.

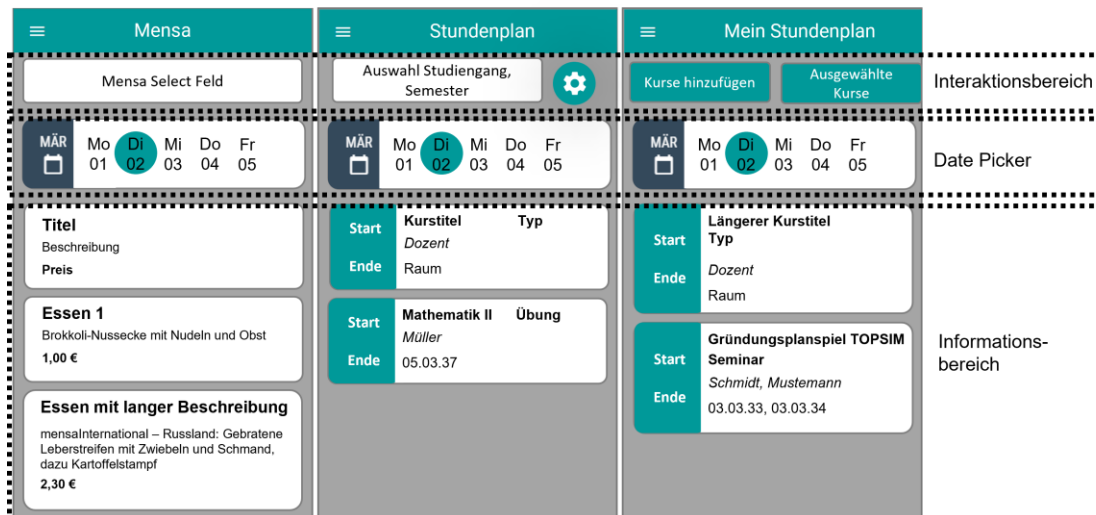


Abbildung 6: Mockup und Layoutkonzept der Hauptansichten der Features
Quelle: Eigene Darstellung.

Während die Features über die Hauptnavigation in Form eines sogenannten Navigation Drawers erreichbar sind, ist für das Mein Stundenplan Feature eine zusätzliche Navigation vorgesehen. Basierend darauf, ob bereits Kurse ausgewählt wurden oder nicht, erfolgt bei dem Aufruf von Mein Stundenplan über den Navigation Drawer der Einstieg entweder in der Ansicht der Kursauswahl (siehe Abbildung 7 rechts), wenn noch keine Auswahl getroffen wurde oder in der Stundenplanansicht (siehe Abbildung 7 links). Zudem gibt es noch eine weitere Ansicht, die die getroffene Kursauswahl (siehe Abbildung 7 Mitte) abbildet. Die Navigation zwischen den Ansichten innerhalb des Features erfolgt über Buttons im Interaktionsbereich des Layouts.

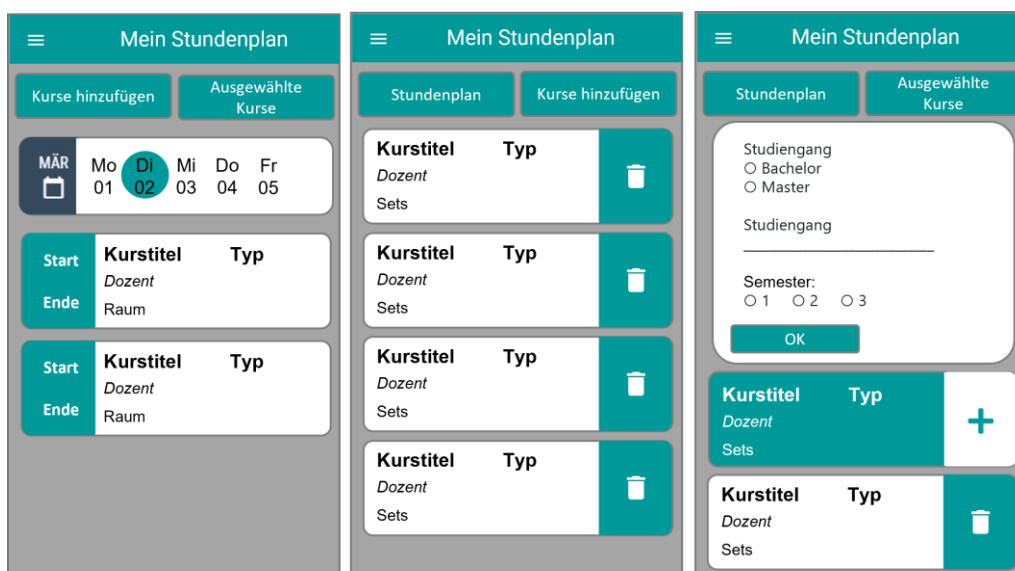


Abbildung 7: UX-Konzept zu Mein Stundenplan
Quelle: Eigene Darstellung.

Anpassungen für größere Displays

Für größere Displays wurde das Layoutkonzept angepasst, um Benutzenden eines Desktop-Computers oder eines Laptops ein gleichermaßen gutes Nutzungserlebnis zu bieten. Die Navigation ist nicht wie bei der mobilen Variante über einen Button aufrufbar, sondern permanent im linken Bereich des Ansichtsfensters geöffnet und somit zugänglicher für Benutzende. Auch ist der horizontal scrollende Date Picker ohne Touchdisplay weniger komfortabel zu bedienen. Daher wurde dieser durch einen Date Picker in Form einer Kalenderansicht ersetzt. In den Stundenplanansichten wird der Date Picker durch einen weiteren Button im Interaktionsbereich des Layouts aufgerufen. Der Bereich für den Date Picker im Layout der mobilen Ansichten (siehe Abbildung 6) entfällt dadurch. Weiterhin erfolgt die Anzeige des Stundenplans aufgrund des vergrößerten Anzeigebereichs wochenweise. Im Mensafeature werden die Informationen zum Speisenangebot weiterhin tageweise angezeigt. Dafür ist der Date Picker hier permanent geöffnet und so für Benutzende gut erreichbar. Aufgrund des Mehrangebotes an Platz wird zur optischen Aufwertung der Ansicht ein Foto der ausgewählten Mensa eingefügt. Benutzende können sich so ein Bild von der Lokalität machen, insofern sie diese noch nicht kennen. Zugleich dient es auch als Kontrollfunktion, dass die gewünschte Mensa korrekt ausgewählt wurde.

Icons

Für einige Interaktionselemente wurden Icons anstelle von Text eingesetzt (siehe Abbildung 8), um die dahinterliegende Aktion zu beschreiben. Die Auswahl der Icons wurde dabei so getroffen, dass die Funktion für den Benutzenden erkennbar ist und die Anwendung intuitiv bedienbar ist.

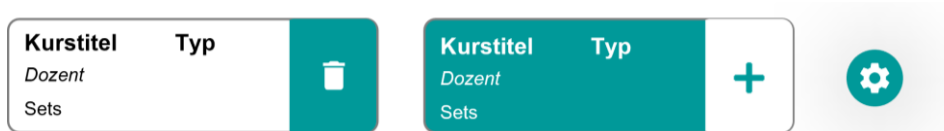


Abbildung 8: Interaktionselemente mit Icons
Quelle: Eigene Darstellung.

Responsivität

Unabhängig vom Layout sollen UI-Elemente sich responsive verhalten und sich an die jeweilige Größe des Anzeigefensters anpassen. UI-Elemente sollen stets eine angemessene und komfortable Größe und Erreichbarkeit aufweisen, um eine bestmögliche User Experience zu gewährleisten.

4.2.3 Clientseitige Speichertechnologien

Damit Daten zur Auswahl von Stundenplänen bzw. Kursen sitzungsübergreifend erhalten bleiben, müssen diese gespeichert werden. Wie in Abschnitt 4.1.3 zu den Anforderungen an die Anwendung beschrieben wurde, soll die Speicherung der Daten clientseitig erfolgen. Von Cookies soll dabei abgesehen werden, da viele Benutzende diese ablehnen und mittlerweile geeignetere Technologien existieren. Für die Speicherung komplexer Datenstrukturen steht die Indexed DB zur Verfügung. Dies ist eine NoSQL HTML5-Datenbank innerhalb des Browsers. Das Limit der zu speichernden Datenmenge, ist abhängig vom jeweiligen Browser. Die niedrigste Speicherkapazität weist aktuell der Internet Explorer mit 250MB auf.⁵⁹ Dies ist für die zu speichernden Daten der Anwendung ausreichend. Daten der Anwendenden, in Zusammenhang mit der Stundenplankonfiguration, sollen in der Indexed DB gespeichert werden, da hierfür das Speichern eines komplexeren Datensatzes erforderlich ist.

Einfache Schlüssel-Wert-Paare in Form von Strings können im Local Storage (auch Web Storage) gespeichert werden. Auch bei Local Storage handelt es sich um einen Speicherbereich im Browser. Im Local Storage soll die zuletzt geöffnete Ansicht hinterlegt werden, damit diese bei erneutem Aufruf der App direkt geöffnet werden kann. Dies soll insbesondere Benutzenden, die häufig dieselbe Funktion der Anwendung verwenden, einen höheren Komfort bieten, da sie sich bei App Start nicht erst durch die Navigation klicken müssen.

4.2.4 Offline Support Konzept

Für den Offline Support stellt Angular ein PWA-Modul bereit, welches für die Umsetzung der Anwendung eingesetzt werden soll. Darin enthalten ist auch eine Implementierung des Service Workers inklusive verschiedener Caching-Strategien für die Absicherung der Offline-Fähigkeit der Anwendung. Eine Übersicht über verschiedene Caching Strategien zeigt Tabelle 10. Der Angular Service Worker unterstützt ausschließlich die Strategien „Cache falling back to network“ und „Network falling back to Cache“. Innerhalb der Anwendung sollen beide Strategien zum Einsatz kommen. Für eine umfangreiche Nutzbarkeit der Anwendung im Offline-Zustand sollen sowohl die statischen Quelldateien der Anwendung als auch die dynamischen Daten aus API-Requests bzw. -Responses gespeichert werden. Die statischen Anwendungsdaten sollen für einen schnellen Start der App zuerst aus dem Cache geladen werden. Die Daten des Mensaangebots ändern sich selten. Daher soll für Requests an diesen Endpoint die Strategie „Cache falling back to network“ zum Einsatz kommen, um den Traffic über das Netz gering zu halten und die Performance zu verbessern. Bei den Stundenplandaten kann es allerdings auch kurzfristige Änderungen geben und es ist für Benutzende von höherer Relevanz

⁵⁹ Vgl. LePage, 2022.

hier aktuelle Daten zu erhalten. Daher soll für diese Daten die Strategie „Network falling back to cache“ genutzt werden.

Strategie	Auswirkung
Cache Only	Anfragen werden ausschließlich über den Cache bedient.
Network Only	Anfragen werden ausschließlich über das Netz bedient.
Cache falling back to network	Die Anfrage soll zunächst aus dem Cache bedient werden. Ist dort keine Antwort vorhanden, wird sie über das Netz geleitet.
Generic Fallback	Kann eine Anfrage weder über das Netz noch den Cache bedient werden, wird sie mit einer generischen Antwort beantwortet.
Network falling back to cache	Die Anfrage wird zunächst über das Netz geleitet. Schlägt das fehl, wird die Antwort aus dem Cache bedient.
Cache then network	Die Anwendung befragt zunächst den Cache, löst zusätzlich aber auch eine Netzabfrage aus. Bei der Rückkehr der Antwort wird der Cache aktualisiert.
Cache & network race	Cache und Netz werden parallel angefragt. Die erste Antwort gewinnt.
Service-Worker-Side Templating	Der Service Worker befüllt beim Beantworten der Anfrage Platzhalter im Nachrichtenrumpf.

Tabelle 10: Übersicht diverser Caching Strategien

Quelle: In Anlehnung an Liebel, 2019.

4.2.5 Architekturkonzept

Abbildung 9 gibt einen Gesamtüberblick über die beteiligten Systeme und Komponenten im Kontext der Anwendung zwischen denen ein Datenaustausch stattfindet. In Abschnitt 4.1.2 wurde bereits beschrieben, dass die Daten, die in der Anwendung dargestellt werden sollen, von einem Service der Fachhochschule Erfurt bezogen werden. Der Implementierungsaufwand beschränkt sich somit auf die Entwicklung des Frontends der Anwendung. Daher befasst sich dieser Abschnitt im Folgenden mit der Frontendarchitektur der Anwendung.

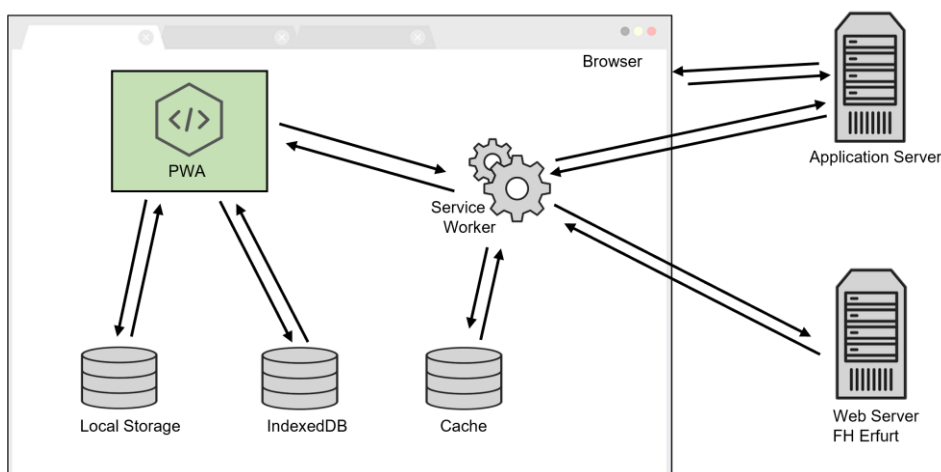


Abbildung 9: High Level Architektur

Quelle: Eigene Darstellung.

Für einen performanten Start der PWA basiert die grundlegende Architektur auf dem Ansatz der App Shell. Dabei wird das statische Grundgerüst der Anwendung von den dynamischen Inhalten separiert (siehe Abbildung 10). Dies ermöglicht das Zwischenspeichern des Grundgerüsts im Cache, sodass dieses nicht bei jedem Start der App redundant vom Server geladen wird. Die Anwendung kann so schneller starten was das Nutzungserlebnis verbessert. Weiterhin ist dies dienlich für die umzusetzende Offline-Funktionalität der Anwendung. In der App Shell der umzusetzenden App soll die App Bar sowie die Navigation enthalten sein.

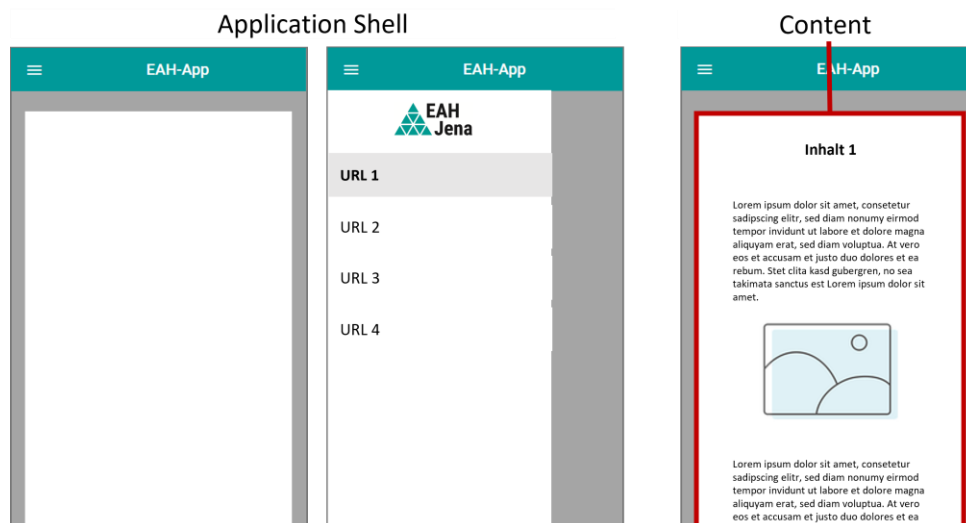


Abbildung 10: App Shell Architektur
Quelle: In Anlehnung an Osmani, o.D.

Der Aufbau der Anwendung soll modular erfolgen, um zusammengehörige Funktionsblöcke für eine bessere Wartbarkeit und Wiederverwendbarkeit zu bündeln. Im Falle der umzusetzenden App sollen dabei unabhängige Module für Mensa und Stundenplan erzeugt werden. Wie in Abschnitt 4.2.1 erläutert, umfassen Module ein oder mehrere Komponenten und können weitere Module importieren, um bereits implementierte Funktionalitäten nutzen zu können. Die App wird über ein Root-Modul, hier App Modul genannt, gestartet. In der Architektur ist daher vorgesehen, dass das App Modul die Module Mensa und Stundenplan importiert und die App Shell als Komponente bereitstellt (siehe Abbildung 11). In den folgenden Abbildungen zur Beschreibung der Architektur (Abbildung 11, Abbildung 12 und Abbildung 13), werden die Feature-Module der Anwendung blau dargestellt, Komponenten orange und Funktionsmodule, die von Angular bereitgestellt werden, sind rot gekennzeichnet. Grün dargestellt werden Services, welche wiederum modular implementiert werden und deren Dienste Komponenten beziehen können. Das App Modul soll demnach, wie Abbildung 11 aufzeigt, mit Funktionen zum Routing ausgestattet werden. Unter Routing wird dabei die Aktivierung von Views verstanden, wobei einer Ansicht eine logische URL zugewiesen werden kann. Dadurch wird die Eigenschaft Verlinkbarkeit von PWAs, die in Abschnitt 3.1.10 dieser Arbeit beschrieben wurde, abgesichert. Das HTTP Modul stellt Funktionen zur Kommunikation mit dem Server über das HTTP-

Protokoll bereit. Dies wird für Anfragen an den Webserver benötigt, der die Daten der Anwendung über einen RESTful-Service bereitstellt. Zudem werden diverse Module der Angular Material Komponentenbibliothek für die Erstellung der Ansichten genutzt und die PWA Funktionalitäten, insbesondere für die Offlinefähigkeit der Anwendung, aus dem PWA Modul in das Wurzelmodul der Anwendung integriert.

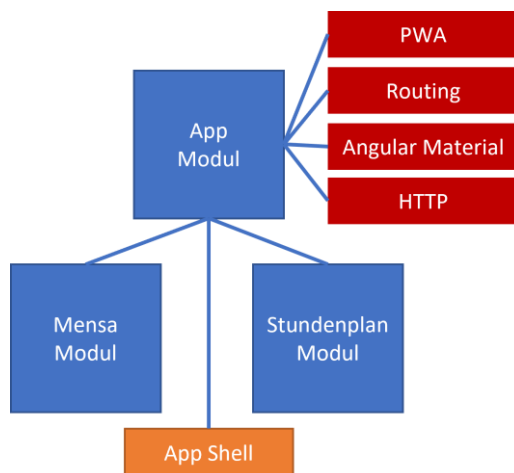


Abbildung 11: Architekturkonzept Module
Quelle: Eigene Darstellung.

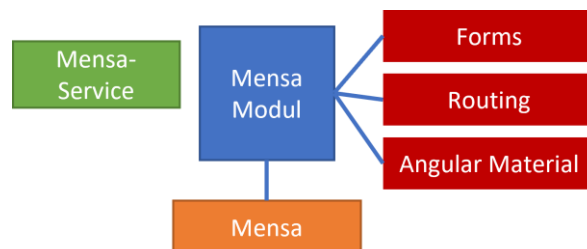


Abbildung 12: Architekturkonzept Mensa Module
Quelle: Eigene Darstellung.

Im Modul Mensa ist nur eine Ansicht vorgesehen. Daher ist, wie in Abbildung 12 dargestellt, nur eine Komponente Mensa für das Mensa Modul vorgesehen. Die Konzeption des User Interface für das Mensa-Feature beinhaltet ein Auswahlfeld für die Selektion einer Mensa (siehe Abschnitt 4.2.2, Abbildung 6). Die dafür benötigten Funktionalitäten werden über das Angular Modul Forms bereitgestellt.

Das Stundenplan-Modul setzt sich zusammen aus den Komponenten für das Stundenplan-Feature und das Mein Stundenplan-Feature. Insgesamt sind fünf Ansichten in diesem Zusammenhang vorgesehen. Innerhalb dieser fünf Ansichten treten sich wiederholende Strukturen auf. Um möglichst wenig Code-Duplikate zu erzeugen, werden diese Strukturen in Komponenten ausgelagert, um sie wiederverwendbar zu machen. Abbildung 13 soll dies verdeutlichen. Für die Umsetzung der Kursauswahlkomponenten, werden auch hier Funktionalitäten des Forms Modul eingebunden. Die Daten sollen den Komponenten über Services bereitgestellt werden. Wie die Abbildungen Abbildung 12 und Abbildung 13 ebenfalls aufzeigen sollen hierfür unabhängige Dienste für jedes Modul bereitgestellt werden.

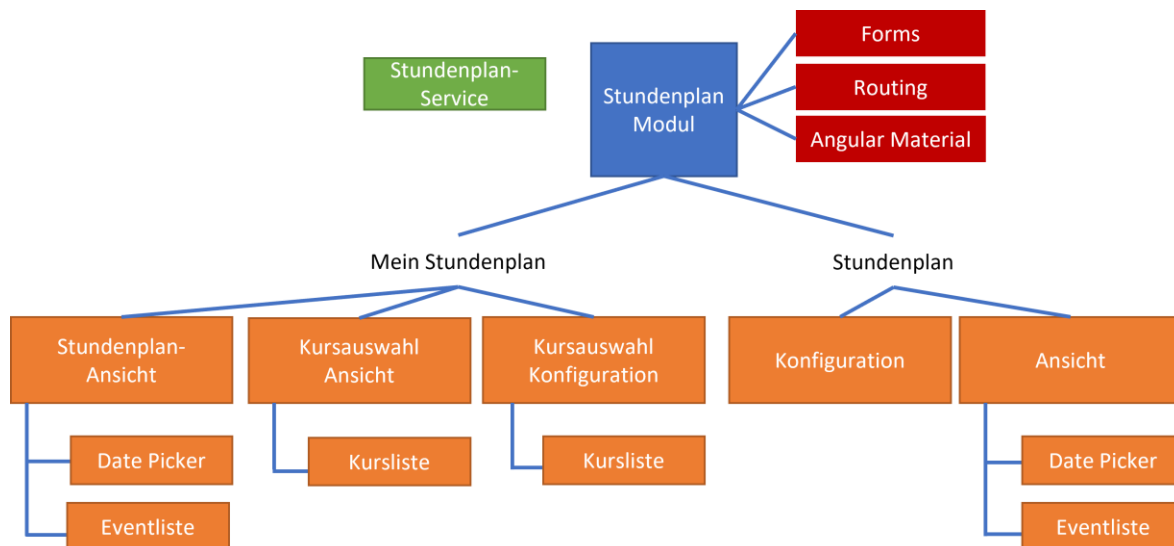


Abbildung 13: Architekturkonzept Stundenplan Modul
Quelle: Eigene Darstellung.

4.2.6 Deployment

Einige PWA Funktionalitäten, wie die Registrierung des Service Workers, werden nur in einem produktiven Build der Anwendung unterstützt und nicht im Entwicklungsmodus. Daher soll die Anwendung für erste manuelle Tests über GitHub Pages bereitgestellt werden, bevor sie über einen Server der Hochschule veröffentlicht wird.

4.3 Implementierung

In den nachfolgenden Abschnitten werden Details der Implementierung betrachtet. Die Implementierung erfolgt anhand der Konzeption, die im vorausgegangenen Abschnitt 4.2 beschrieben wurde. Um mit dem Angular Framework zu arbeiten, muss eine aktuelle Node.js-Version, als Laufzeitumgebung für JavaScript, installiert sein. Zudem wird der Node Package Manager (npm) benötigt. Als Entwicklungsumgebung wurde IntelliJ IDEA eingesetzt.

4.3.1 Projekt Setup

Vor dem Erstellen eines neuen Projekts, wird zunächst die Angular CLI durch das Ausführen von `npm install -g @angular/cli` in der Kommandokonsole installiert. Nach der Installation ist die Angular CLI einsatzbereit und das Werkzeug `ng` kann für die Initialisierung des Projekts genutzt werden. Über die Kommandokonsole wird dafür `ng new EAH-PWA` ausgeführt, wobei EAH-PWA der Name des zu erstellenden Projekts ist. Über das Kommandozeileninterface wird nun abgefragt, ob das Angular Routing Modul in das Projekt eingebunden werden soll. Da das Routing Modul in der zu erstellenden App verwendet werden soll, wird dies bestätigt. Außerdem kann eine Sprache für die Erstellung von Stylesheets gewählt werden. Es wurde die Sprache SCSS gewählt. Diese Sprache erweitert den Funktionsumfang von

CSS und ermöglicht es einen verschachtelten Syntax zu verwenden. SCSS wird von der Angular CLI nach CSS kompiliert, um es im Browser ausführen zu können. Durch die Projektinitialisierung über die Angular CLI werden nun einige Dateien für das Grundgerüst der Anwendung generiert. Diese umfassen die Dateien `index.html` und `main.ts` als Startpunkt der Anwendungsausführung, ein globales Stylesheet, eine Datei für Polyfills, das Root-Modul `AppModule` mit einer Komponente `AppComponent`, ein Routing Modul (`AppRoutingModule`) sowie diverse Konfigurationsdateien.

4.3.2 UI

Nach der Initialisierung des Projekts wird in einem nächsten Schritt die Angular Material Komponentenbibliothek in das Projekt eingebunden. Dies erfolgt mithilfe der Angular CLI durch die Ausführung von `ng add @angular/material`. Über die Angular CLI wird dann abgefragt, welches Theme eingesetzt werden soll. Hier wurde Custom gewählt, da die Erstellung eines eigenen Themes vorgesehen ist. Weiterhin wird abgefragt, ob globale Typografie Styles und Browseranimationen mit eingerichtet werden sollen. Beides wurde bestätigt. Die Angular CLI installiert nun die erforderlichen Pakete, registriert die Abhängigkeiten in der Datei `package.json`, importiert die erforderlichen Module im `AppModule`, bindet Schriftarten in der `index.html` Datei ein und generiert Code, für die Erstellung eines Themes in der globalen `style.scss` Datei.

Nach der Installation des Packages wird das Theme für die Anwendung konfiguriert. Dafür muss zunächst eine Farbpalette erstellt werden, die die Theme-Farben definiert. Zur Erstellung der Farbpalette wurde ein Generator von Material Design genutzt. Nach der Eingabe der primären Farbe werden Abstufungen von dieser generiert und zusammengestellt. Diese bilden die Farbpalette. Es wurden jeweils eine Farbpalette für die Primärfarbe türkis und eine Farbpalette für die Akzentfarbe tiefseeblau erstellt. Diese Farbpaletten wurden dann an eine Funktion übergeben (siehe Code-Snippet 1, Zeile 1-3). Anschließend wird ein Theme-Objekt mit den entsprechenden Farbpaletten erzeugt (siehe Code-Snippet 1, Zeile 6-12). Neben den Farben soll auch die Typografie angepasst werden. Dafür wurde die Schriftart Roboto-Condensed des Corporate Designs der Ernst-Abbe-Hochschule Jena in das Projekt eingebunden (siehe Code-Snippet 1, Zeile 15-22). Die Einbindung der definierten Theme Styles erfolgte dann über eine spezielle Funktion, ein sogenanntes Mixin (siehe Code-Snippet 1, Zeile 25), sodass alle Angular Material Komponenten die vereinbarten Theme-Styles anwenden können. Da das erstellte Theme in ein separates Stylesheet ausgelagert wurde, muss dieses in der Datei `angular.json` im `styles`-Array registriert werden, damit es beim Start der Anwendung ausgeführt wird.

```
1  $EAH-PWA-primary: mat.define-palette($turquoise);
2  $EAH-PWA-accent: mat.define-palette($navy);
3  $EAH-PWA-warn: mat.define-palette(mat.$red-palette);
4
5  // Theme Object
6  $EAH-PWA-theme: mat.define-light-theme((
7    color: (
8      primary: $EAH-PWA-primary,
9      accent: $EAH-PWA-accent,
10     warn: $EAH-PWA-warn,
11   )
12 ));
13
14 // Typography
15 @font-face {
16   font-family: 'Roboto- Condensed';
17   src: url('~src/assets/font/roboto_condensed_regular.ttf');
18 }
19
20 $custom-typography: mat.define-typography-config(
21   $font-family: 'Roboto-Condensed',
22 );
23
24 // Include Theme Styles
25 @include mat.all-component-themes($EAH-PWA-theme);
```

Code-Snippet 1: Theme-Konfiguration

Um Angular Material Komponenten im Projekt zu verwenden, wird das jeweilige Modul einer Material-Komponente in das Modul des Projekts importiert, in dem es genutzt wird. In der App Shell wird beispielsweise die Toolbar Komponente genutzt. Die App Shell ist eine Komponente im Modul AppModule. Daher wird das Modul MatToolbarModule in der Liste der Imports des AppModule registriert. Auf die Verwendung der einzelnen UI-Komponenten wird in dieser Arbeit nicht im Detail eingegangen. Die einzelnen Komponenten und deren Schnittstellen sind umfangreich dokumentiert. Die Dokumentation kann über material.angular.io eingesehen werden.

In der UI-Konzeption ist auch die Verwendung von Icons vorgesehen. Eingesetzt werden Icons von Googles Material Design Ressourcen. Sie werden von Google kostenlos zur Verfügung gestellt und können als Schriftart in das Projekt eingebunden werden. Die Einbindung erfolgt in der Datei index.html über ein `link`-Tag (siehe Code-Snippet 2).

```
1  <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
2      rel="stylesheet">
```

Code-Snippet 2: Einbindung der Icon Font

Für den Einsatz der Icons in Templates werden Ligaturen genutzt. Dabei wird das Icon durch einen textuellen Namen ersetzt. In der Darstellung im Browser wird der Text dann durch das Icon ersetzt. Dafür wird der Text zwischen den Inhalt der Angular Material Icon Komponente `mat-icon` gesetzt (siehe Code-Snippet 3, Zeile 2). Code-Snippet 3 zeigt die Erstellung eines Icon Buttons mit Angular Material Komponenten. Dafür wird die Direktive `mat-fab` genutzt,

wodurch der runde Button Style angewendet wird. Der Eigenschaft `color` können die Werte `primary`, `accent` oder `warn` gegeben werden. Diese repräsentieren die im Theme definierten Farbpaletten.

```
1 <button mat-fab color="primary" (click)="goToConfig()">
2   <mat-icon>settings</mat-icon>
3 </button>
```



Code-Snippet 3: Erstellung eines Icon Buttons mit Angular Material

Für die Implementierung des Layouts wurde eine CSS-Klasse mit einem Raster bzw. einem Grid Template definiert (siehe Code-Snippet 4). Für die im Layout definierten Sektionen wurden Bereiche angelegt (siehe Zeile 4). Die einzelnen Bereiche passen sich dabei an die Größe ihres Inhaltes an (siehe Zeile 5). Um die Sektionen voneinander abzutrennen, wird eine Lücke zwischen zwei angrenzenden Sektionen definiert (siehe Zeile 6). Den Komponenten, die das Layout verwenden, wird in ihrem HTML-Template die erstellte CSS-Klasse `grid-container` zugewiesen. Die jeweiligen Layoutblöcke erhalten den entsprechenden Wert für das Attribut `grid-area`. In der Mensa-Komponente erhält bspw. der HTML-Block, der das Auswahlfeld für eine Mensa beinhaltet, den `grid-area`-Wert `controls`.

```
1 // grid layout
2 .grid-container {
3   display: grid;
4   grid-template-areas: "controls" "picker" "list";
5   grid-template-rows: auto auto minmax(0, 1fr);
6   grid-row-gap: 2rem;
7 }
```

Code-Snippet 4: Grid-Template

Für die Umsetzung des abweichenden Layouts für die Desktopvariante wurde ein Service eingesetzt, der die Größe des Ansichtsfensters prüft. In einer Komponente können durch die Nutzung von `ngTemplate` mehrere Templates erstellt werden. Über die Direktive `ngIf` kann im Template eine Bedingung geprüft werden, wobei das Template auf die globalen Variablen der Klasse zugreifen kann, insofern diese nicht mit dem Schlüsselwort `private` erstellt wurden. Wenn die Bedingung erfüllt ist, wird das Template an das DOM gebunden. So können ebenfalls wiederverwendbare Komponenten erzeugt werden. Code-Snippet 5 zeigt den Einsatz von `ng-template` für die Implementierung der abweichenden Layouts. Es wird zunächst geprüft, ob ein kleines Ansichtsfenster vorliegt (siehe Zeile 2). Wenn diese Bedingung nicht erfüllt wird, soll ein alternatives Template (siehe Zeile 12-17) an das DOM gebunden werden auf welches referenziert wird (siehe Zeile 3).

```

1  <ng-template
2  [ngIf]="(rs.mobileQuery.matches || rs.tabletPortraitModeQuery.matches)"
3  [ngIfElse]="isDesktop">
4  <div class="grid-container">
5      <!-- CONTROLS -->
6      <!-- DATE PICKER MOBILE -->
7      <!-- EVENT LIST MOBILE -->
8  </div>
9  </ng-template>
10
11 |
12 <ng-template #isDesktop>
13     <div class="grid-container">
14         <!-- DATE PICKER + CONTROLS FOR DESKTOP -->
15         <!-- EVENT LIST DESKTOP (WEEK) -->
16     </div>
17 </ng-template>

```

Code-Snippet 5 Implementierung des abweichenden Desktop-Layouts

Für die Implementierung des responsiven Designs kamen verschiedene Methoden zum Einsatz. Zum einen wurden über die `ngClass`-Direktive, im Falle einer eintretenden Bedingung, zusätzliche CSS-Klassen an ein Element gebunden (siehe Code-Snippet 6 links, Zeile 2-3). Zum anderen wurden media-Queries (siehe Code-Snippet 6 rechts, Zeile 9) und das Flexible Box Layout Module genutzt, um Elemente entsprechend des verfügbaren Platzes anzuordnen. Zudem wurde die CSS-Funktion `clamp` genutzt, um Schriften dynamisch zu vergrößern (siehe Code-Snippet 6 rechts, Zeile 22). Dieser Funktion wird ein Minimalwert, ein bevorzugter Wert und ein Maximalwert übergeben. Der bevorzugte Wert in der Mitte definiert dabei wie schnell die Größe skaliert.

HTML	SCSS
<pre> 1 <div class="course-selection" 2 [ngClass]= 3 "'desktop-row' : rs.largeScreenQuery.matches"> 4 5 <!-- SELECTED COURSE ---> 6 <p id="course-title"> 7 {{courseSelection.courseTitle}}, 8 9 Semester {{courseSelection.semester}} 10 11 </p> 12 13 <!-- SETTINGS BUTTON ---> 14 <button mat-fab> 15 ... 16 </button> 17 18 19 <ng-template 20 [ngIf]="rs.largeScreenQuery.matches"> 21 <!-- DESKTOP DATE RANGE PICKER ---> 22 </ng-template> 23 24 </div> </pre>	<pre> 1 .course-selection { 2 display: flex; 3 justify-content: space-between; 4 align-items: baseline; 5 grid-area: controls; 6 font-weight: bold; 7 margin-bottom: 1.5rem; 8 9 @media (max-width: 390px) { 10 align-items: flex-start; 11 12 p { 13 margin-top: 1rem; 14 display: flex; 15 flex-direction: column; 16 } 17 } 18 } 19 20 21 button { 22 font-size: clamp(1.6rem, 2vw, 2rem); 23 } </pre>

Code-Snippet 6: Responsive Design Implementierung

4.3.3 App Shell

Wie in Abschnitt 4.2.5 beschrieben, soll in der App Shell das Grundgerüst der Anwendung bereitgestellt werden. Die Ausführung der Anwendung startet in der `index.html` Datei, welche in gekürzter Form in Code-Snippet 7 dargestellt wird. In Zeile 6 wird ein Element `app-root` eingebunden. Dies ist der Selektor für die Komponente `AppComponent`. Der Selektor einer Komponente, mit dem diese in ein beliebiges Template eingebunden werden kann, wird in dem `@Component`-Dekorator der Klasse der Komponente definiert (siehe Code-Snippet 8).

```
1 <!doctype html>
2 <html lang="de">
3 <head>
4 </head>
5 <body class="mat-typography">
6 <app-root></app-root>
7 </body>
8 </html>
```

Code-Snippet 7: `index.html`

```
1 @Component({
2   selector: 'app-root',
3   templateUrl: './app.component.html',
4   styleUrls: ['./app.component.scss']
5 })
6 export class AppComponent {
7   ...
8 }
```

Code-Snippet 8: `@Component`-Dekorator

In der Komponente `AppComponent` wird das Grundgerüst der Anwendung definiert. Das Template der Komponente ist in Code-Snippet 9 zu sehen. Es beinhaltet die Toolbar (siehe Zeile 3-8), die Navigation (siehe Zeile 11-22) und einen Platzhalter für die dynamischen Inhalte (siehe Zeile 26). Durch den Text-Interpolation Syntax können dynamische Werte aus Variablen der Klasse in das Template eingebunden werden. Eine Klassenvariable wird dabei zwischen doppelte geschweifte Klammern geschrieben. Im Template der App Shell wird dies genutzt um den Titel, der in der App Bar angezeigt wird (siehe Zeile 7), dynamisch zu ändern, wenn ein Navigationselement von Benutzenden ausgewählt wird (siehe Zeile 16). Durch die Nutzung der Angular Direktive `*ngFor` kann das Template verkürzt werden. Direktiven sind Befehle, durch die das Template modifiziert werden kann. Die Direktive `*ngFor` wendet eine Vorlage wiederholt für jedes enthaltene Element eines iterierbaren Objekts an. In diesem Fall wird das Array `navigationList` durchlaufen (siehe Zeile 17) um die Listenelemente (siehe Zeile 14-20) zu erstellen und an das DOM zu binden. Auch hier werden die Werte durch Text-Interpolation an das Template gebunden (siehe Zeile 18, Zeile 19).

```

1  <div id="root">
2    <!--- TOOLBAR --->
3    <mat-toolbar>
4      <button mat-icon-button (click)="sidenav.toggle()">
5        <mat-icon>menu</mat-icon>
6      </button>
7      <span> {{ title }} </span>
8    </mat-toolbar>
9
10   <!--- SIDE NAVIGATION --->
11   <mat-sidenav-container>
12     <mat-sidenav #sidenav>
13       <mat-nav-list>
14         <a mat-list-item
15           [routerLink]="menuItem.url"
16           (click)="title=menuItem.title"
17           *ngFor="let menuItem of navigationList">
18           <mat-icon> {{ menuItem.icon }} </mat-icon>
19           <span> {{ menuItem.title }} </span>
20         </a>
21       </mat-nav-list>
22     </mat-sidenav>
23
24   <!--- CONTENT --->
25   <mat-sidenav-content>
26     <router-outlet></router-outlet>
27   </mat-sidenav-content>
28
29 </mat-sidenav-container>
30 </div>

```

Code-Snippet 9: App Shell - Template

In Zeile 26 von Code-Snippet 9 ist das Tag `router-outlet` zu sehen. Dies ist der Platzhalter für die dynamischen Inhalte. Das Angular Router Modul stellt u.a. Funktionen bereit, die es ermöglichen logische URLs für Ansichten zu definieren. Jeder URL wird dabei eine Komponente zugeordnet. Die Definition der Routenkonfiguration ist in Code-Snippet 10 dargestellt. Über die `routerLink`-Direktive kann eine Ansicht, die im Routing-Modul registriert wurde, über das Template aktiviert werden (siehe Zeile 15, Code-Snippet 9). Die Aktivierung erfolgt durch das Klicken auf das Element mit der `routerLink`-Direktive.

```

1  const routes: Routes = [
2    {path: '', component: InstallInstructionsComponent},
3    {path: 'start', component: StartComponent},
4    {path: 'mensa', loadChildren: () =>
5      import('./mensa/mensa.module').then(m => m.MensaModule)},
6    {path: 'stundenplan', loadChildren: () =>
7      import('./timetable/timetable.module').then(m => m.TimetableModule)},
8  ];

```

Code-Snippet 10: Routing-Konfiguration

4.3.4 Wiederverwendbare Komponenten

Im Architekturkonzept wurde das Ziel definiert Code-Duplikate zu vermeiden und daher für wiederkehrende Strukturen wiederverwendbare Artefakte zu erzeugen.

Für die Umsetzung der UI wurden größtenteils Angular Material Komponenten eingesetzt. Durch ihre modulare Implementierung lassen sich die Komponenten einzeln je nach Bedarf in ein Modul importieren. Für den horizontal scrollbaren Date Picker, der für die mobilen Ansichten zum Einsatz kommt, gab es keine passende Angular Material Komponente. Daher wurde dieser selbst entwickelt. Da er in mehreren Modulen eingesetzt wird, wurde nach dem Prinzip der Material Komponenten für diese neu entwickelte Komponente ebenfalls ein Modul angelegt, um sie in den entsprechenden Modulen wiederverwenden zu können. Dieses Modul kann, wie die Angular Material Module, in Module importiert werden, in welchen es eingesetzt werden soll.

Am Beispiel der Listenansicht der auszuwählenden Kurse bzw. der ausgewählten Kurse wird im Folgenden die Erzeugung einer wiederverwendbaren Komponente betrachtet. Wie bereits erläutert können Komponenten über ihren Selektor in ein Template eingebunden werden. Zunächst wird eine neue Komponente `CourseListComponent` durch die Angular CLI mit dem Kommando `ng generate component CourseList` erstellt. In der Komponente wird nun ihr Template und die Darstellungslogik definiert. Die Komponente soll dabei die Daten verarbeiten und darstellen, die durch die Elternkomponente, in die sie eingebettet wird, vorgegeben werden. Daher ist es essenziell, dass die Komponenten untereinander Daten austauschen können. In der Komponente `CourseListComponent` (siehe Code-Snippet 11) kommt eine `@Input`-Annotation (siehe Zeile 7) zum Einsatz. Dadurch wird die Komponente `CourseListComponent` befähigt von einer Elternkomponente, in welche sie eingebettet ist, Eingabewerte entgegenzunehmen. Diese Eingabewerte können dann nach der in der Komponente definierten Logik verarbeitet werden (siehe Zeile 11-14).

```
1  @Component({
2    selector: 'app-course-list',
3    templateUrl: './course-list.component.html',
4    styleUrls: ['./course-list.component.scss']
5  })
6  export class CourseListComponent {
7    @Input() groupedEvents: MyTimeTableEvent[];
8
9    ...
10
11    remove(title: string) {
12      this.groupedEvents = this.groupedEvents
13        .filter(ev => ev.title !== title);
14    }
15  }
```

Code-Snippet 11: Beispiel für Kommunikation zwischen Komponenten

Code-Snippet 12 zeigt wie die Daten im Template der Elternkomponente an die Kindkomponente `CourseListComponent` übergeben werden. Hierfür wird der Property Binding Syntax genutzt, bei dem das Attribut `groupedEvents` der Komponente `CourseListComponent` in eckige Klammern gesetzt wird. Der Wert aus der Variable `groupedEvents` der Klasse der Elternkomponente wird so über das Template an die Variable aus der Kindklasse gebunden.

```
1 <app-course-list [groupedEvents]="groupedEvents"></app-course-list>
```

Code-Snippet 12: Beispiel für Data Binding

4.3.5 Service Implementierung

Um die Komponenten mit Daten zu versorgen, die sie handhaben und ggf. darstellen, kommen Services zum Einsatz. Durch die Verwendung von Services wird die Datenlogik von der Darstellungslogik getrennt, was wiederum die Modularität und Wiederverwendbarkeit erhöht. Für die Anwendung werden, wie im Architekturkonzept vorgesehen, zwei unabhängige Services `MensaService` und `TimetableService` (siehe Code-Snippet 13) implementiert. Beide der Services erben von einem Basis-Service, der ihnen die Implementierung der GET Methode für die Schnittstellenkommunikation über das HTTP-Protokoll bereitstellt.

```
1 @Injectable({
2   providedIn: 'root'
3 })
4 export class TimetableService extends BaseRestService {
5   PROXY = 'https://cors-everywhere.herokuapp.com/';
6   API_TIMETABLE = 'http://app.fh-erfurt.de:8080/fheapp/api/eah/timetable';
7
8   constructor(
9     protected http: HttpClient,
10    private indexedDBService: NgxIndexedDBService
11  ) {
12    super(http);
13  }
14
15  getCourses(): Observable<Courses> {
16    ...
17  }
18
19  getTimetable(timetableId: string): Observable<Timetable[]> {
20    return this.get<Timetable[]>({
21      this.PROXY + this.API_TIMETABLE + "/events",
22      {params: {timetableId: timetableId}});
23    }
24    ...
25  }
```

Code-Snippet 13: Implementierung des Stundenplanservice

Services besitzen einen `@Injectable`-Dekorator (siehe Code-Snippet 13, Zeile 1) vor der Definition ihrer Klasse. Dadurch kann das Framework zwischen Komponenten und Services

unterscheiden.⁶⁰ Um den Service in Komponenten bereitzustellen, wird das im Angular Framework implementierte Prinzip Dependency Injection angewandt. Angular nutzt Constructor Injection, bei der die Abhängigkeit der Klasse über den Konstruktor zur Verfügung gestellt wird. Der Service wird daher im Konstruktor der Komponente deklariert (siehe Code-Snippet 14, Zeile 3), um eine für die Komponente nutzbare Instanz von ihm zu erstellen. Die Rückgabewerte der implementierten Methoden zum Abruf der Daten über die REST-Schnittstelle sind vom Datentyp Observable (siehe Zeile 15, Zeile 19). Dieser Datentyp wird in der reaktiven Programmierung verwendet und kann abstrakt als ein Daten-Stream betrachtet werden.⁶¹ Ein Observable kann erst Daten empfangen, wenn es abonniert wird. Dafür wird die `subscribe`-Methode auf dem Observable-Objekt angewandt (siehe Code-Snippet 14, Zeile 12). Weiterhin wird für den Umgang mit diesem Datentyp die Bibliothek RxJS verwendet, die bereits im Angular-Framework integriert ist, da das Observable-Konzept in weiten Teilen des Frameworks Verwendung findet. Diese Bibliothek stellt eine Reihe an Funktionen bzw. sogenannte Operatoren bereit, um die Daten aus dem Daten-Stream zu manipulieren. Diese Operatoren können innerhalb der `pipe`-Methode eines Observable eingesetzt werden (siehe Code-Snippet 14, Zeile 6-11).

```
1  export class MyTimetableConfigComponent {
2
3      constructor(private timetableService: TimetableService) {}
4
5      this.timetableService.getCourses()
6          .pipe(
7              map((courses: Courses) => courses.courseOfStudies),
8              tap(courses => {
9                  // sort courses alphabetically
10                     ...
11             })
12          ).subscribe()
13 }
```

Code-Snippet 14: Nutzung eines Services in einer Komponente

Aufgrund des im Browser implementierten Sicherheitskonzepts der Same-Origin-Policy (siehe 3.1.6), musste für den Abruf der Daten über den REST-Service ein Proxy verwendet werden, über den der Request an den Webserver erfolgt, da es sich hierbei um eine Cross-Origin Resource Sharing (CORS)-Anfrage handelt. Aus Sicherheitsgründen wird dafür ein spezieller Header, der Access-Control-Allow-Origin Header, benötigt, der der Server-Response beigelegt wird. Da allerdings kein Zugriff auf das System bestand, wurde das Problem mit der Umleitung des Requests über einen Proxy gelöst, dessen URL in Zeile 5 des Code-Snippet 13

⁶⁰ Vgl. Garg, 2019.

⁶¹ Vgl. Weiße, 2016.

angegeben ist. Dieser Proxy fügt dem Request den erforderlichen Header bei. Für die Umleitung über den Proxy wird die Proxy-URL der Request-URL vorangestellt (siehe Zeile 21).

4.3.6 Indexed DB und Local Storage

Die Indexed DB und Local Storage werden genutzt, um Anwenderdaten clientseitig zu speichern (siehe 4.2.3).

Um eine bestehende Implementierung der Indexed DB zu verwenden, wurde die Bibliothek `ngx-indexed-db` durch das Ausführen von `npm install ngx-indexed-db` in das Projekt eingebunden. Diese Bibliothek enthält einen Angular Service der wie in Abschnitt 4.3.5 beschrieben, über die Deklaration im Konstruktor einer Komponente bezogen werden kann. In der Indexed DB wird die getroffene Anwender-Konfiguration für die Stundenplankomponente und die ausgewählten Kurse für Mein Stundenplan gespeichert. Aufgrund der vorliegenden Datenstruktur in der Response der REST-API, ist es erforderlich verschiedene Datensätze zu speichern, die es ermöglichen die Response-Daten so zu filtern, dass die getroffene Auswahl der Anwendenden repräsentiert wird. Für die beiden genannten Anwendungsfälle wird jeweils ein Store erstellt (siehe Code-Snippet 15). Jedem Datensatz wird automatisch eine fortlaufende ID zugeordnet (siehe Zeile 7 und Zeile 16), über die sich der Datensatz eindeutig identifizieren lässt. In dem Store `generalTimetable` soll immer nur ein einziger Datensatz gespeichert sein. Daher kann das Feld `timetableId` als `unique` gekennzeichnet werden (siehe Zeile 9). `timetableId` ist aber prinzipiell keine eindeutige ID, sondern eine Repräsentation der Setgruppe. Daher kann diese im Store `myTimetable` nicht als `unique` gekennzeichnet werden, da es in diesem Store möglich sein soll, mehrere Objekte mit der gleichen `timetableId` zu speichern, wenn verschiedene Kurse desselben Studiengangs und Semesters mit derselben Setgruppe ausgewählt werden. Um die Daten für die Stundenplanansicht des Features Stundenplan abzurufen, benötigt es nur die `timetableId`. Die zusätzlichen Werte der Felder `courseTitle` (Zeile 10) und `semester` (Zeile 11) werden gespeichert, um den Anwendenden die getroffene Auswahl über die UI anzuzeigen (siehe Abschnitt 4.2.2, Abbildung 6). Das Speichern dieser Daten ist erforderlich, da sie über den Endpoint der API nicht mit zurückgegeben werden. Auch im Store `myTimetable` wurden Felder erstellt deren Werte nicht für den Request bzw. das Filtern der Daten benötigt werden, die jedoch in der UI dargestellt werden sollen. Dies betrifft in diesem Store die Felder `set` und `lecturer` (Zeile 21-22).

```

1  const dbConfig: DBConfig = {
2    name: 'eahDB',
3    version: 7,
4    objectStoresMeta: [
5      {
6        store: 'generalTimetable',
7        storeConfig: {keyPath: 'id', autoIncrement: true},
8        storeSchema: [
9          {name: 'timetableId', keypath: 'timetableId', options: {unique: true}},
10         {name: 'courseTitle', keypath: 'courseTitle', options: {unique: false}},
11         {name: 'semester', keypath: 'semester', options: {unique: false}}
12       ]
13     },
14     {
15       store: 'myTimetable',
16       storeConfig: {keyPath: 'id', autoIncrement: true},
17       storeSchema: [
18         {name: 'timetableId', keypath: 'timetableId', options: {unique: false}},
19         {name: 'courseTitle', keypath: 'courseTitle', options: {unique: false}},
20         {name: 'semester', keypath: 'semester', options: {unique: false}},
21         {name: 'set', keypath: 'set', options: {unique: false}},
22         {name: 'lecturer', keypath: 'lecturer', options: {unique: false}}
23       ]
24     }
25   ]
26 };

```

Code-Snippet 15: Konfiguration der Indexed DB Stores

Die genutzte Bibliothek stellt verschiedene Methoden für die Kommunikation zwischen der Indexed DB und der Anwendung bereit. Die genutzten Methoden werden in Code-Snippet 16 dargestellt. Die Rückgabewerte der Methoden sind vom Datentyp Observable. Daher müssen diese zur Aktivierung des Datenstreamings abonniert werden (siehe Zeile 2, Zeile 4). Beim Hinzufügen eines Datensatzes über die `add` Methode, wird der Name des Stores, in dem die Daten gespeichert werden sollen und der zu speichernde Datensatz als JavaScript-Objekt übergeben (siehe Zeile 4). Das Löschen eines Datensatzes erfolgt anhand dessen ID (siehe Zeile 6) und zum Abrufen aller Daten aus einem Store wird nur der Name des Stores als Parameter benötigt (siehe Zeile 1).

```

1  this.indexedDBService.getAll('myTimetable')
2    .subscribe(result => { this.indexedDBEvents = result });
3
4  this.indexedDBService.add('myTimetable', courseSelection).subscribe();
5
6  this.indexedDBService.delete('myTimetable', course.id);

```

Code-Snippet 16: Kommunikation mit der Indexed DB

Local Storage wurde im `AppComponent`, also in der Wurzelkomponente der Anwendung, eingesetzt, um die zuletzt gesehene Ansicht zu speichern (siehe Code-Snippet 17). Bei einem erneuten Start der Anwendung können Benutzende wieder dort einsteigen, wo sie in der Session zuvor aufgehört haben. Die Kommunikation mit Local Storage erfolgt über die Web Storage API. Im Local Storage können Schlüssel-Wert-Paare gespeichert werden, wobei der Wert vom Datentyp String sein muss. Über die Methode `setItem` wird ein Schlüssel-Wert-Paar

gespeichert. Schlüssel und Wert werden dabei als Eingabeparameter an die Methode übergeben (siehe Code-Snippet 17, Zeile 11). Anhand des Schlüssels können Werte aus dem Speicher durch die Methoden `getItem` und `removeItem` abgerufen bzw. gelöscht werden.

```
1  let lastVisitedUrl: string | null = localStorage.getItem('last_visited_url');
2  localStorage.removeItem('last_visited_url');
3
4  if (lastVisitedUrl && !window.location.href.split('/').reverse()[0]) {
5    this.router.navigateByUrl(lastVisitedUrl).then(_ => this.setTitle());
6  }
7  router.events.pipe(
8    filter(e => e instanceof RouterEvent)
9  ).subscribe(ev => {
10   if (ev instanceof NavigationEnd) {
11     localStorage.setItem('last_visited_url', ev.url)
12   }
13 });
```

Code-Snippet 17: Local Storage

4.3.7 Implementierung der PWA Komponenten

Für die Implementierung der PWA-Features Offline-Fähigkeit und Installierbarkeit wird eine Bibliothek von Angular bereitgestellt. Diese wird über die Angular CLI durch das Ausführen von `ng add @angular/pwa` in das Projekt eingebunden. Dadurch wird das Web App Manifest (siehe Abschnitt 3.2.2) generiert und mittels eines `link`-Elements in die `index.html` Datei eingebunden, es wird ein Modul mit einer Implementierung des Service Workers (siehe Abschnitt 3.2.1) bereitgestellt, der Service Worker wird im `AppModule`, dem Wurzelmodul der Anwendung, registriert und eine Konfigurationsdatei für den Service Worker wird generiert.

4.3.7.1 Einbindung der Web App Manifest Funktionen für iOS

Wie in Abschnitt 3.3 erläutert, werden vom Safari Browser für iOS nicht alle Attribute des Web App Manifests unterstützt. Es gibt jedoch alternative Implementierungsmöglichkeiten, um die entsprechenden Funktionalitäten dennoch anbieten zu können. Im Kopf der Datei `index.html` werden dafür zusätzliche Elemente eingebunden. Die Bereitstellung von Icons für iOS Geräte erfolgt durch die Einbindung über `link`-Elemente mit dem Wert `apple-touch-icon` für das Attribut `rel`. Um einen Splash Screen anzuzeigen, wird das Meta-Tag `<meta name="apple-mobile-web-app-capable" content="yes">` eingefügt. Es müssen dann für jedes Gerät Bilddateien in der passenden Größe über ein `link`-Element unter Angabe der korrekten Höhe, Breite und Skalierung des Displays im `media`-Attribut, wie in Code-Snippet 18, eingebunden werden.⁶²

⁶² Vgl. Bacon, 2019.

```
1 <link href="assets/splashscreens/iphone5_splash.png"
2       media="(device-width: 320px) and (device-height: 568px)
3           and (-webkit-device-pixel-ratio: 2) "
4       rel="apple-touch-startup-image">
```

Code-Snippet 18: Einbindung eines Splash Screens für ein iOS Endgerät

4.3.7.2 Service Worker: Caching

Über die Datei `ngsw-config.json` kann das Verhalten des Service Workers konfiguriert werden. In einem Array `assetGroups` wird festgelegt, wie Ressourcen, die Teil der Anwendungsversion sind, im Cache hinterlegt werden. Innerhalb des Arrays können mehrere Datensätze mit unterschiedlichen Konfigurationen hinterlegt werden. Für Datensätze in diesem Array können die Felder `installMode` und `updateMode` definiert werden, um festzulegen, wie bzw. wann die Quelldaten der Anwendung im Cache zwischengespeichert werden. Für das Feld `installMode` gibt es zwei mögliche Werte. Der Wert `prefetch` wird genutzt, um den Service Worker anzuweisen, alle angegebenen Ressourcen direkt abzurufen und sie im Cache zu speichern. Unter Verwendung des Werts `lazy` ruft der Service Worker eine Ressource erst ab und speichert diese im Cache, wenn sie von der Anwendung angefordert wird. Im Feld `updateMode` wird das Verhalten beim Update auf eine neue Version der Anwendung bestimmt. Auch hier werden die Werte `prefetch` oder `lazy` genutzt, um die veränderten Dateien der neuen Anwendungsversion entweder direkt oder auf Anfrage im Cache zwischenzuspeichern. Wenn statische Anwendungsdateien im Cache hinterlegt sind und es keine neue Version der Anwendung gibt, werden diese Daten vom Angular Service Worker immer zuerst aus dem Cache geladen.

Dynamische Daten aus API-Requests, die unabhängig von der Anwendungsversion sind, können ebenfalls im Cache zwischengespeichert werden. Die Konfiguration erfolgt im Array `dataGroups` (siehe Code-Snippet 19). In diesem Array werden Datensätze mit den Request-URLs und diversen Konfigurationsmöglichkeiten für das Caching angegeben. Zudem wird jedem Datensatz ein Name zugeordnet, über den der Datensatz identifiziert werden kann. Im Angular Service Worker sind zwei Caching Strategien implementiert, zwischen denen gewählt werden kann. Diese wurden bereits in Abschnitt 4.2.4 beschrieben. Für die Daten des Mensa Endpoints soll die Strategie „Cache falling back to network“ angewandt werden. Die Konfiguration erfolgt über das Feld `strategy` durch die Zuweisung des Werts `performance` (siehe Zeile 10). Zudem kann ein maximal zulässiges Alter für die Cachedaten über das Feld `maxAge` vereinbart werden (siehe Zeile 9). Außerdem wird die maximale Anzahl der zu speichernden Request-Response-Paare definiert, um das Speicherkontingent nicht zu überschreiten. Um die Stundenplandaten aktuell zu halten, sollten die Daten zuerst über das Netzwerk bezogen werden und nur im Falle einer unzureichenden Netzwerkverbindung aus dem Cache

abgerufen werden. Dazu wird der Wert für das Feld `strategy` auf `freshness` gesetzt. Es kann hier zudem eine `timeout`-Angabe erfolgen. Wenn innerhalb der definierten Zeit keine Response über das Netzwerk eintrifft, folgt die Abfrage der Daten aus dem Cache. Da die Performance der Stundenplanansichten durch die Anwendung dieser Caching Strategie nicht zufriedenstellend war, erfolgte hier eine Abweichung vom Konzept und der Wechsel zu der performanteren Strategie, bei der die Daten zuerst aus dem Cache geladen werden.

```
1  "dataGroups": [  
2    {  
3      "name": "mensa",  
4      "urls": [  
5        "https://cors-everywhere.herokuapp.com/  
6        http://app.fh-erfurt.de:8080/fheapp/api/eah/canteens/*",  
7      ],  
8      "cacheConfig": {  
9        "maxAge": "5d",  
10       "strategy": "performance",  
11       "maxSize": 18  
12     }  
13   },  
14   {  
15     "name": "timetable",  
16     "urls": [  
17       "https://cors-everywhere.herokuapp.com/http://app.fh-  
18     erfurt.de:8080/fheapp/api/eah/timetable/*"  
19     ],  
20     "cacheConfig": {  
21       "strategy": "freshness",  
22       "timeout": "5s",  
23       "maxSize": 25  
24     }  
25   }  
26 ]
```

Code-Snippet 19: Service Worker Konfiguration - Data Groups

4.3.7.3 Versionsupdate

Wenn eine neue Version der Anwendung über den Server bereitgestellt wird, enthält diese auch einen neuen Service Worker. Da zuerst die Anwendungsdateien inkl. des Service Workers aus dem Cache geladen werden, wird zunächst die alte Anwendungsversion beim Start der App geladen. Der Browser prüft in periodischen Abständen im Hintergrund, ob eine neue Service Worker Version vorhanden ist. Sobald eine neue Version erkannt wird, wird diese registriert und installiert, jedoch noch nicht aktiviert.⁶³ Zur Handhabung des Versionsupdates und der Aktivierung des neuen Service Workers stellt Angular über das Service Worker Modul einen Service `swUpdate` zur Verfügung. Über diesen Service kann geprüft werden, ob ein neuer Service Worker, und damit eine neue Version der Anwendung, verfügbar ist. Der Service

⁶³ Vgl. Bar, o. D. a.

wurde wie in Code-Snippet 20 verwendet, um Benutzende mittels einer Toast-Benachrichtigung in der UI (Zeile 13-19) zu signalisieren, dass eine neue Version der Anwendung verfügbar ist. Die Aktivierung des neuen Service Workers erfolgt, sobald die Anwendung erneut geladen wird. Benutzende können dafür einen Button in der Toastbenachrichtigung klicken, über welchen der Reload der Anwendung angestoßen wird (siehe Zeile 18).

```
1  export class AppComponent {
2      constructor( private serviceWorkerUpdates: SwUpdate,
3                  private _snackbar: MatSnackBar, ...
4                  ) { ... }
5
6      ngOnInit(): void {
7
8          this.subscription.add(
9              this.serviceWorkerUpdates.versionUpdates.pipe(
10                 filter((evt): evt is VersionReadyEvent =>
11                     evt.type === 'VERSION_READY'),
12                 mergeMap(_ => {
13                     return this._snackbar.open(
14                         'Eine neue Version der App ist verfügbar!',
15                         'Update',
16                         {verticalPosition: "bottom"}).onAction()
17                 })).subscribe(() => {
18                     document.location.reload();
19                 });
20      }
21  }
```

Code-Snippet 20: Service Worker Update

4.3.8 App-ähnliches Verhalten

Um die App in ihrem Verhalten dem einer nativen App weiter anzupassen, wurde das Anwenden des Standardverhaltens der Webbrowser durch die Anpassung einiger CSS-Eigenschaften unterbunden. Da es untypisch für native Apps ist, dass Elemente oder Texte innerhalb der Anwendung selektierbar sind, wie dies im Browser der Fall ist wurde das Verhalten über die user-select Eigenschaft deaktiviert (siehe Code-Snippet 21, Zeile 2-5). Im Safari Browser wird eine Information in einer Textbox über einem Element angezeigt, wenn dieses berührt und gehalten wird. Wenn ein Element doppelt angetippt wird, zoomt der Safari-Browser in das Element. Da dieses Verhalten ebenfalls untypisch für native Apps ist, wurde es ebenfalls angepasst (siehe Code-Snippet 21, Zeile 6-7).

```
1  body {
2      -webkit-user-select: none;
3      -moz-user-select: none;
4      -ms-user-select: none;
5      user-select: none;
6      -webkit-touch-callout: none;
7      touch-action: manipulation;
8  }
```

Code-Snippet 21: Unterbinden des Browser-Standardverhaltens

4.4 Kritische Betrachtung der Umsetzung

4.4.1 Erfüllung der Anforderungen

In diesem Abschnitt wird geprüft, ob die in Abschnitt 4.1 definierten Anforderungen an die Anwendung erfüllt werden konnten. Die in Abschnitt 4.1.1 definierten Features Stundenplan, Mein Stundenplan und Mensa konnten unter Nutzung der in Abschnitt 4.1.2 beschriebenen Datenquelle implementiert werden. Durch die Implementierung eines clientseitigen Speicherkonzepts gelten die in Abschnitt 4.1.3 definierten Anforderungen (1) und (2) als erfüllt. Durch die Bereitstellung eines Service Workers in Verbindung mit diversen Cache-Konfigurationen, wird Anforderung (3) erfüllt. Das implementierte Web App Manifests bzw. die alternative Implementierung für den Safari Browser macht die App installierbar und dadurch über ein Icon auf dem Home Screen aufrufbar. Zudem wird ein Splash Screen beim Start der Anwendung angezeigt. Durch die Umsetzung des Architekturkonzepts für Single-Page Web Applications, verhält sich die Anwendung beim Wechseln der Ansichten wie eine native App. Es konnten zudem zusätzliche Anpassungen getroffen werden, die das Standardverhalten des Browsers unterbinden und dadurch zu einem nativeren Verhalten der App beitragen. Die bereitgestellten UI-Komponenten der verwendeten UI-Komponenten-Bibliothek, Angular Material, orientieren sich an Googles Material Design. Da das iOS UI-Design abweichende Richtlinien verfolgt, fühlt sich die App vom optischen Eindruck für Benutzende im Android-Ökosystem wohl nativer an als in iOS. Insgesamt kann Anforderung (6) somit als größtenteils erfüllt betrachtet werden. Das User Interface wurde auf Basis des Corporate Designs der Ernst-Abbe-Hochschule entwickelt. Somit wurde auch Anforderung (5) erfüllt. In Anforderung (4) wurde definiert, dass die Daten der Anwendung aktuell sein sollen. Daher sollte, insbesondere für die Stundenplanfeatures, der Abruf der Daten zuerst von der Schnittstelle erfolgen und nur im Ausweichfall auf die zwischengespeicherten Daten im Cache zurückgegriffen werden. Dies wurde zunächst implementiert. Jedoch kam es dadurch zu starken Performanceeinbußen und die Ladezeiten für die Stundenplanansichten waren sehr hoch. Daher wurde die Caching Strategie zugunsten des Nutzungserlebnis angepasst, sodass die Daten zuerst aus dem Cache bezogen werden. Das maximal zulässige Alter der Daten wurde auf einen Tag beschränkt. Kurzfristige Stundenplanänderungen innerhalb eines Tages werden aber so möglicher Weise nicht angezeigt. Da diese Lösung nicht optimal ist, müssen hier weitere Schritte erfolgen, um das Problem der Performance bei API-Requests zu verbessern. Abschließend wurde der Test mit dem Analysetool Google Lighthouse durchgeführt. Google Lighthouse ist in den Entwicklertools des Google Chrome Browsers integriert. Es lassen sich die fünf verschiedenen Bereiche Performance, Accessibility, Best Practices, SEO und PWA analysieren. Für jeden Bereich werden verschiedene Tests durchgeführt deren Ergebnisse unterschiedlich gewichtet werden. Die gewichteten Ergebnisse werden dann zusammengefasst und bilden einen Score. Dieser kann zwischen 0

und 100 liegen. In Anforderung (8) wurde definiert, dass alle Aspekte einer PWA, die über Google Lighthouse getestet werden, erfüllt sein sollen. Es wurden alle Aspekte erfüllt. In Anforderung (7) wurde eine angemessene Performance der Anwendung definiert. Für die Analyse der Performance wurde auch hier Google Lighthouse eingesetzt. Beim initialen Laden der App wird ein Performance Score von 71 im Test für mobile Endgeräte und von 99 für Desktops erreicht. Beim zweiten Laden der Anwendung, wenn die Anwendungsdaten aus dem Cache bezogen werden, liegt der Performance Score für mobile Endgeräte bei 99 und für Desktops bei 100. Wie bereits erläutert ist die Performance der Anwendung insbesondere dann nicht zufriedenstellend, wenn Daten von der REST-API bezogen werden. Der Grund für die hohen Performanceeinbußen liegt zum einen darin begründet, dass die Requests über einen Proxy umgeleitet werden, der den fehlenden Header für die CORS-Anfrage beifügt. Dadurch kommt es zu Latenzen und einer damit verbundenen Verzögerung beim Eintreffen der Response. Zum anderen müssen die Response-Daten im Frontend der Anwendung aufbereitet werden, bevor sie in der UI angezeigt werden. Dies führt zu weiteren Verzögerungen für das Rendern der Seite. Abgesehen davon fühlt sich die Bedienung der Anwendung im manuellen Test beim Wechsel zwischen den Ansichten flüssig an. Wenn Daten vom Netzwerk geladen werden, wird zudem ein Ladeindikator angezeigt der Benutzenden signalisiert, dass der Aufbau der Ansicht in Arbeit ist.

Weiterhin wurde ein manueller Cross-Browser-Test der Anwendung durchgeführt, da, nach dem Prinzip von Progressive Enhancement, in jedem Browser mindestens die Grundfunktionalitäten der Anwendung nutzbar sein sollten. Bei diesem Test wurde festgestellt, dass die aktuelle Version des eingesetzten Frameworks Angular, den Internet Explorer nicht mehr unterstützt und die Anwendung in diesem Browser somit nicht zur Verfügung steht. Von einem Downgrade auf eine vorausgegangene Version des Frameworks, in der der Internet Explorer noch unterstützt wird, wurde aber aus Gründen der Relevanz abgesehen. Der Marktanteil des Internet-Explorers liegt im Januar 2022 bei nur 0,95%.⁶⁴ Zudem wurde auch von Microsoft bereits ein Ende des Supports für den Internetexplorer angekündigt.⁶⁵ In den Browsern Firefox, Chrome, Edge, Opera und Safari ist die Anwendung ausführbar.

4.4.2 Betrachtung der entwickelten Lösung für iOS

Das ursprüngliche Problem und die Motivation dieser Arbeit lagen auch darin begründet, dass keine benutzerfreundliche mobile Anwendung, insbesondere zum Abruf der Stundenplandaten, für Endgeräte mit dem Betriebssystem iOS vorhanden ist. Daher wurde die Funktionalität

⁶⁴ Vgl. Statista, 2022.

⁶⁵ Vgl. Microsoft, 2021.

der erstellten Anwendung gesondert auf einem iPhone 11 und einem iPad getestet. Im Gegensatz zum Google Chrome Browser animiert der Safari Browser Benutzende nicht zum Installieren der App. Deshalb wurde eine zusätzliche Ansicht erstellt, die Benutzende beim ersten Start der Anwendung über die Möglichkeit zur Installation über das Browsermenü informiert. Durch das Hinzufügen der Anwendung zum iOS Home Screen wird das implementierte Verhalten adaptiert. Die App lässt sich über ein Icon starten und zeigt den alternativ eingebundenen Splash Screen beim Start der App. Da keine hardwareabhängigen Features implementiert wurden, funktioniert die Anwendung auch auf iOS-Geräten ohne Einschränkungen. Wie bereits im vorausgegangenen Abschnitt aufgegriffen wurde, ist das UI-Design etwas abweichend vom Look einer nativen iOS-App. Apples Human Interface Elemente⁶⁶ begründen sich auf einem flachen und minimalistischen Ansatz. Im Gegensatz dazu steht das eingesetzte dreidimensionale Material Design von Google⁶⁷, was durch animierte Elemente und das Setzen von Schatten die Nutzerinteraktion unterstützt. Zudem gibt es aktuell noch keine Implementierung für die Rückwärtsnavigation, die bei Android über die globale Navigationsleiste möglich ist. Dennoch bietet die aktuelle Implementierung der Anwendung auch unter iOS ein gewisses Maß an Benutzerfreundlichkeit, eine gute Alternative zur Webseite der Stundenplanung und eine Möglichkeit für das Anlegen eines individuellen Stundenplans.

⁶⁶ Vgl. Apple Inc., o. D. b.

⁶⁷ Vgl. Google, o. D.

5 Fazit und Ausblick

Das Ziel dieser Arbeit war es, einen ersten Prototyp für eine plattformunabhängige Anwendung für die Ernst-Abbe-Hochschule Jena nach dem Entwicklungskonzept einer Progressive Web App zu erstellen. Dabei sollte die Anwendung für Benutzende mobiler Endgeräte und für Desktop-Anwender:innen einen gleichermaßen hohen Nutzungskomfort bieten.

In dieser Arbeit erfolgte zunächst die Auseinandersetzung mit den Grundlagen verschiedener App-Entwicklungskonzepte. Im Anschluss wurde das Konzept von Progressive Web Apps umfangreich erläutert. Für den praktischen Teil der Ausarbeitung wurden zunächst die Anforderungen an die zu erstellende Anwendung definiert und ein Konzept für die Umsetzung erarbeitet. Die Implementierung erfolgte unter Nutzung des Single-Page Application Frameworks Angular, was einige Vorteile in Bezug auf die Effizienz der Umsetzung mit sich brachte. Das Framework stellt eine Struktur bereit, die die Programmierarbeit vereinfacht. Zudem konnte auf bestehende Implementierungen häufig verwendeter Artefakte, durch die Einbindung von Bibliotheken, zurückgegriffen werden. Auch Konzepte des Frameworks wie Direktiven und Datenbindungsmechanismen unterstützten maßgeblich dabei den Code sauber und lesbar zu halten. Durch den Einsatz der Angular CLI bei der Initialisierung des Projekts, dem Einbinden von Bibliotheken und bei der Erstellung neuer Artefakte kann der zeitliche Aufwand durch die automatisierte Erzeugung von Codefragmenten weiter reduziert werden. So konnte die Implementierung der Progressive Web App innerhalb weniger Wochen erfolgen. Durch Eigenschaften wie Installierbarkeit und Offline-Fähigkeit, aber auch durch Darstellungselemente des User Interfaces weist die Anwendung Merkmale einer nativen App auf und bietet somit für Benutzende mobiler Endgeräte einen hohen Anwendungskomfort. Durch die Implementierung eines responsiven Layouts und eines erweiterten Darstellungskonzepts für größere Displays bleibt der Nutzungskomfort auch auf diesen Endgeräten erhalten. Da bei der Umsetzung einer Progressive Web App nur eine Codebasis entsteht, werden gegenüber dem nativen Ansatz sowohl der Entwicklungsaufwand als auch der Wartungsaufwand reduziert. Durch die Auffindbarkeit über Suchmaschinen wird die Sichtbarkeit der Anwendung erhöht und es bestehen keine Restriktionen bezüglich der Publizierung der App. Dennoch ist es möglich, die App für eine erweiterte Sichtbarkeit auch in den App-Stores der Plattformen anzubieten. Aktuell bestehen noch einige Einschränkungen bezüglich der Nutzung hardwarebasierter Funktionalitäten, aber auch in diesem Bereich wird an Lösungen durch die Bereitstellung performanterer HTML5 Schnittstellen gearbeitet und viele Browser unterstützen bestehende Schnittstellen bereits. Durch die Implementierung des Prinzips von Progressive Enhancement kann die Anwendung mit einem Grundumfang an Funktionen universell bereitgestellt werden. Abschließend lässt sich sagen, dass das Konzept der Progressive Web App ein geeigneter Ansatz ist, um die Hochschulapp plattformunabhängig anzubieten. Ein letzter Abschnitt dieser Arbeit soll

einen Ausblick auf zukünftige Schritte und Weiterentwicklungsmöglichkeiten der erstellten Anwendung geben.

5.1 Ausblick

Um den Wartungs- und Weiterentwicklungsaufwand zu verringern, ist es sinnvoll die Android-App durch die plattformunabhängige Progressive Web App abzulösen. Dafür muss der erstellte Prototyp zunächst um die Funktionen der Android-App erweitert werden. Die Anwendung kann dann in einer Testphase für Anwender:innen bereitgestellt werden. Anschließend sollte evaluiert werden, ob die Android-App durch die plattformunabhängige App abgelöst werden kann. Da App Stores ein häufiger und gewohnter Einstiegspunkt bei der Suche nach Apps sind, wäre die zusätzliche Bereitstellung der Anwendung über die Stores von Vorteil.

Vor einer ersten Veröffentlichung der Anwendung sollten jedoch zunächst die aktuell auftretenden Performance-Probleme bei Anfragen an die REST-API der Fachhochschule Erfurt behoben werden. Dafür muss der benötigte Header für CORS-Anfragen der Server-Response beigefügt werden. Zudem sollte die Datenlogik größtenteils ins Backend ausgelagert werden, sodass im Frontend die Daten nur konsumiert und angezeigt werden. Dafür müssten neue Endpoints entwickelt werden. Ein Vorschlag für die Konzeption eines neuen Endpoints für das Mein Stundenplan Feature wird in *Anhang E: Vorschlag zur Neugestaltung des REST-Endpoints für Mein Stundenplan* beigefügt. Um die User Experience für Nutzende von iOS-Endgeräten zu steigern, sollte zudem eine Rückwärtsnavigation unter Berücksichtigung der für iOS typische Gestensteuerung implementiert werden.

In Hinblick auf zukünftige Features der Anwendung wird die Implementierung eines Push-Service empfohlen, um Benutzende bspw. über Stundenplanänderungen, zu informieren. Das Angular Framework stellt die erforderlichen Schnittstellenimplementierungen in dem eingebundenen Service Worker Modul über den Service `SwPush` bereit. Weiterhin empfiehlt es sich die aktuell hartkodierte Texte der Anwendung, für eine bessere Wartbarkeit, in eine separate Datei, etwa im JSON-Format, auszulagern. Dafür kann bspw. das Angular Modul `I18N` eingebunden werden, was zusätzliche Funktionen in Bezug auf eine Lokalisierung bzw. Übersetzung der Anwendung bereitstellt.

Literaturverzeichnis

- Add to home screen (A2HS): in: Can I use, o. D., <https://caniuse.com/web-app-manifest> (abgerufen am 08.03.2022).
- Adetunji, Oluwatofunmi/Chigozirim Ajaegbu/Otuneme Nzechukwu: Dawning of Progressive Web Applications (PWA): Edging Out the Pitfalls of Traditional Mobile Development, in: American Scientific Research Journal for Engineering, Technology, and Sciences, Nr. 68, 2020, https://www.researchgate.net/publication/343472764_Dawning_of_Progressive_Web_Applications_PWA_Edging_Out_the_Pitfalls_of_Traditional_Mobile_Development, S. 85–99.
- Android Platform: in: PWA Builder, 27.10.2021, <https://blog.pwabuilder.com/docs/android-platform/> (abgerufen am 25.03.2022).
- ANEXIA Deutschland GmbH: Hybride App Entwicklung, in: anexia, o. D., <https://anexia.com/de/softwareentwicklung/app-entwicklung/hybride-app-entwicklung> (abgerufen am 23.03.2022).
- Apple Inc.: App Store Review Guidelines, in: App Store Developer, o. D., <https://developer.apple.com/app-store/review/guidelines/> (abgerufen am 22.02.2022a).
- Apple Inc.: Apple announces App Store Small Business Program, in: Apple Newsroom, 18.11.2020, <https://www.apple.com/newsroom/2020/11/apple-announces-app-store-small-business-program/> (abgerufen am 23.03.2022).
- Apple Inc.: Human Interface Guidelines, in: Apple Developer, o. D., <https://developer.apple.com/design/human-interface-guidelines/> (abgerufen am 25.03.2022b).
- Bacon, Evan: Enabling iOS Splash Screens for Progressive Web Apps, in: Medium, 18.04.2019, <https://blog.expo.dev/enabling-ios-splash-screens-for-progressive-web-apps-34f06f096e5c> (abgerufen am 25.03.2022).
- Bar, Adam: Handling Service Worker updates – how to keep the app updated and stay sane, in: What Web Can Do Today, o. D., <https://whatwebcando.today/articles/handling-service-worker-updates/> (abgerufen am 25.03.2022a).
- Bar, Adam: What Web Can Do Today, in: What Web Can Do Today, o. D., <https://whatwebcando.today/> (abgerufen am 23.02.2022b).
- Böhm, Robin: Angular-Tutorial für Einsteiger, in: Angular.DE, 19.10.2020, <https://angular.de/artikel/angular-tutorial-deutsch/> (abgerufen am 10.03.2022).
- Build Your iOS App: in: PWA Builder, 27.10.2021, <https://blog.pwabuilder.com/docs/build-your-ios-app> (abgerufen am 09.03.2022).
- Chebbi, Ajay: Best programming languages for mobile app development, in: IBM Developer, 04.06.2021, <https://developer.ibm.com/articles/choosing-the-best-programming-language-for-mobile-app-development/> (abgerufen am 20.03.2022).
- CLI Overview and Command Reference: in: Angular, o. D., <https://angular.io/cli> (abgerufen am 10.03.2022).
- CSS Flexible Box Layout Module: in: Can I use, o. D., <https://caniuse.com/flexbox> (abgerufen am 08.03.2022).
- CSS Grid Layout (level 1): in: Can I use, o. D., <https://caniuse.com/css-grid> (abgerufen am 08.03.2022).
- Darji, Vaidehi/Afifah Khan/Abhijit Dongaonkar: Mobile Application Development - Native App, in: International Research Journal of Innovations in Engineering and Technology (IRJIET), Bd. 5, Nr. 12, 2021, doi:10.47001/IRJIET/2021.512018, S. 91–93.

- Einrichtungen - Studierendenwerk Thüringen: in: Studierendenwerk Thüringen, o. D., <https://www.stw-thueringen.de/mensen/einrichtungen.html> (abgerufen am 20.02.2022).
- EAH Jena: Corporate Design: Die Grundelemente und Ihre Anwendung, 2021.
- Ernst-Abbe-Hochschule Jena: Ernst-Abbe-Hochschule Jena, in: Ernst-Abbe-Hochschule Jena, o. D., <https://www.eah-jena.de/> (abgerufen am 20.02.2022).
- Ernst-Abbe-Hochschule Jena - Stundenplanung: in: Ernst-Abbe-Hochschule Jena - Stundenplanung, o. D., <https://stundenplanung.eah-jena.de/studentset/> (abgerufen am 20.02.2022).
- Facebook/Luke Karrys: Getting Started, in: Create React App, 01.09.2021, <https://create-react-app.dev/docs/getting-started/> (abgerufen am 09.03.2022).
- Firtman, Maximiliano: Push Notifications, WebXR, and better PWA support coming to iOS, in: FIRT.DEV, 31.01.2022, <https://firt.dev/ios-15.4b#web-push-notifications-on-ios%E2%8Dwith-a-catch> (abgerufen am 08.03.2022).
- Freeman, Adam: Pro Angular 9: Build Powerful and Dynamic Web Apps, 4. Aufl., New York, USA: Apress, 2020.
- Garg, Bhavika: Angular Architecture Overview, in: Medium, 26.01.2019, <https://medium.com/@bhavikagarg8/angular-architecture-overview-1e7cc7483a0> (abgerufen am 25.03.2022).
- Generating your Android package: in: PWA Builder, 17.09.2021, <https://blog.pwabuilder.com/docs/generating-your-android-package> (abgerufen am 11.03.2022).
- Getting started with service workers: in: Angular, o. D., <https://angular.io/guide/service-worker-getting-started> (abgerufen am 09.03.2022).
- Geuß, Martin: Neue Richtlinien für den Microsoft Store: Standardprovision für Apps bei 12 Prozent, keine Updates für Win32 Apps ›, in: Dr. Windows, 02.06.2021, <https://www.drwindows.de/news/neue-richtlinien-fuer-den-microsoft-store-standard-provision-fuer-apps-bei-12-prozent-keine-updates-fuer-win32-apps> (abgerufen am 24.03.2022).
- Google: App veröffentlichen, in: Google Play Console-Hilfe, 2022, <https://support.google.com/googleplay/android-developer/answer/9859751?hl=de#zippy=> (abgerufen am 22.02.2022).
- Google: Design, in: Material Design, o. D., <https://material.io/design> (abgerufen am 23.03.2022).
- Google: Servicegebühren, in: Play Console-Hilfe, 2022b, <https://support.google.com/googleplay/android-developer/answer/112622?hl=de> (abgerufen am 23.03.2022).
- Gorczak, Nadja/Phillipp Ringleb/Stephan Druskat/Michael Stepping: Stundenplaner.FHE_EAH, in: Github, 01.03.2022, https://github.com/avinotec/Stundenplaner.FHE_EAH/blob/master/doc/EAH-API-Calls.docx (abgerufen am 03.03.2022).
- Introduction to Angular concepts: in: Angular, o. D., <https://angular.io/guide/architecture> (abgerufen am 25.03.2022).
- iOS Platform: in: PWA Builder, 28.10.2021, <https://blog.pwabuilder.com/docs/ios-platform/> (abgerufen am 25.03.2022).
- Kennedy, John/Alesandro Ortiz/Quinn Radich: Verwandeln Ihrer Website in eine hochwertige PWA, in: Microsoft Docs, 05.03.2022, <https://docs.microsoft.com/de-de/windows/uwp/publish/pwa/turn-your-website-pwa> (abgerufen am 09.03.2022).

- Latif, Mounaim/Younes Lakhrissi/El Habib Nfaoui/Najia Es-Sbai: Cross platform approach for mobile application development: A survey, in: 2016 International Conference on Information Technology for Organizations Development (IT4OD), 2016, doi:10.1109/it4od.2016.7479278, S. 1–5.
- LePage, Pete: Storage for the web, in: web.dev, 08.03.2022, <https://web.dev/storage-for-the-web/> (abgerufen am 20.03.2022).
- Liebel, Christian: Progressive Web Apps: Das Praxisbuch, 1. Aufl., Bonn, Deutschland: Rheinwerk Verlag, 2019.
- Malavolta, Ivano: Beyond native apps: web technologies to the rescue! (keynote), in: Proceedings of the 1st International Workshop on Mobile Development, 2016, doi:10.1145/3001854.3001863, S. 1.
- Manifest: icons: in: Can I use, o. D., https://caniuse.com/mdn-html_manifest_icons (abgerufen am 08.03.2022).
- MDN: Web API Referenz, in: MDN, 08.12.2020, <https://developer.mozilla.org/de/docs/Web/API> (abgerufen am 25.03.2022).
- MDN: Web App Manifest, in: mdn web docs, 09.03.2022a, <https://developer.mozilla.org/de/docs/Web/Manifest> (abgerufen am 09.03.2022).
- MDN: Web Workers API, in: mdn web docs, 18.02.2022b, https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API (abgerufen am 02.03.2022).
- Microsoft: Der App-Zertifizierungsprozess - UWP applications, in: Microsoft Docs, 05.03.2022a, <https://docs.microsoft.com/de-de/windows/uwp/publish/the-app-certification-process> (abgerufen am 22.02.2022).
- Microsoft: Die Internet Explorer 11-Desktopanwendung beendet die Unterstützung für bestimmte Betriebssysteme, in: Microsoft Docs, 24.09.2021, <https://docs.microsoft.com/de-de/lifecycle/announcements/internet-explorer-11-end-of-support> (abgerufen am 25.03.2022).
- Microsoft: Entwickeln von Apps für die universelle Windows-Plattform (UWP), in: Microsoft Docs, 12.01.2022b, <https://docs.microsoft.com/de-de/visualstudio/cross-platform/develop-apps-for-the-universal-windows-platform-uwp?view=vs-2022> (abgerufen am 20.02.2022).
- Microsoft: Übersicht über Progressive Web Apps (PWAs), in: Microsoft Docs, 08.03.2022c, <https://docs.microsoft.com/de-de/microsoft-edge/progressive-web-apps-chromium/> (abgerufen am 25.03.2022).
- Microsoft Edge: Re-engage users with badges, notifications, and push messages, in: Microsoft Docs, 24.02.2022, <https://docs.microsoft.com/en-us/microsoft-edge/progressive-web-apps-chromium/how-to/notifications-badges> (abgerufen am 25.02.2022).
- Osmani, Addy: The App Shell Model, in: Google Developers - Web Fundamentals, o. D., <https://developers.google.com/web/fundamentals/architecture/app-shell> (abgerufen am 20.03.2022).
- Red Hat Inc.: Was ist ein SDK?, in: RedHat, 10.06.2020, <https://www.redhat.com/de/topics/cloud-native-apps/what-is-SDK> (abgerufen am 20.02.2022).
- Rentrop, Christian/Stephan Augsten: Die App als PWA – Pro und Contra, in: Dev-Insider, 15.04.2021, <https://www.dev-insider.de/die-app-als-pwa--pro-und-contra-a-986616/> (abgerufen am 20.02.2022).
- Reuter, Benedikt: ServiceWorker – Offline First, in: Computer Science Blog, 24.02.2021, <https://blog.mi.hdm-stuttgart.de/index.php/2021/02/24/serviceworker-offline-first/> (abgerufen am 03.03.2022).

- Rubino, Daniel: First Windows 10 Progressive Web Apps (PWA) published by Microsoft hit the Store, in: Windows Central, 07.04.2018, <https://www.windowscentral.com/first-batch-windows-10-progressive-web-apps-here> (abgerufen am 26.02.2022).
- Russell, Alex: Progressive Web Apps: Escaping Tabs Without Losing Our Soul, in: Infrequently Noted, 15.06.2015, <https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul/> (abgerufen am 23.02.2022).
- Ryte: Service Worker, in: Ryte Wiki, o. D., https://de.ryte.com/wiki/Service_Worker (abgerufen am 02.03.2022).
- Scott, Emmit A. jr.: SPA Design and Architecture: Understanding Single Page Web Applications, New York, USA: Manning Publications Co., 2015.
- Sheppard, Dennis: Beginning Progressive Web App Development: Creating a Native App Experience on the Web, New York, USA: Springer Science+Business Media New York, 2017.
- Siegrist, Katharina: Service Worker in Web-Anwendungen, in: Informatik Aktuell, 29.06.2021, <https://www.informatik-aktuell.de/entwicklung/methoden/service-worker-in-web-anwendungen.html> (abgerufen am 04.03.2022).
- Statista: Marktanteile führender Browser in Deutschland bis Januar 2022, in: Statista, 01.02.2022, <https://de.statista.com/statistik/daten/studie/13007/umfrage/marktanteile-der-browser-bei-der-internetnutzung-in-deutschland-seit-2009/> (abgerufen am 25.03.2022).
- Steiner, Thomas: What is in a Web View: An Analysis of Progressive Web App Features When the Means of Web Access is not a Web Browser, in: Companion of the The Web Conference 2018 on The Web Conference 2018 - WWW '18, 2018, doi:10.1145/3184558.3188742, S. 789–796.
- Tosic, Marko: Apps für KMU: Praktisches Hintergrundwissen für Unternehmer, Wiesbaden, Deutschland: Springer Gabler, 2015.
- Tremp, Hansruedi: Architekturen Verteilter Softwaresysteme: SOA & Microservices - Mehrschichtenarchitekturen - Anwendungsintegration, Wiesbaden, Deutschland: Springer Vieweg, 2021.
- @vue/cli-plugin-pwa: in: Vue CLI, o. D., <https://cli.vuejs.org/core-plugins/pwa.html> (abgerufen am 09.03.2022).
- W3C: Service Workers Nightly, in: W3C Editor's Draft, 02.08.2021, <https://w3c.github.io/ServiceWorker/> (abgerufen am 08.03.2022).
- W3C: Web Application Manifest, in: W3.org, 17.02.2022, <https://www.w3.org/TR/appmanifest/#using-a-link-element-to-link-to-a-manifest> (abgerufen am 04.03.2022).
- Weiß, Bengt: Angular - Asynchronität von Callbacks zu Observables, in: Angular.DE, 27.05.2016, <https://angular.de/artikel/angular2-observables/> (abgerufen am 25.03.2022).
- Windows Platform: in: PWA Builder, 26.10.2021, <https://blog.pwabuilder.com/docs/windows-platform/> (abgerufen am 25.03.2022).
- Workbox | Google Developers: in: Workbox, o. D., <https://developers.google.com/web/tools/workbox> (abgerufen am 09.03.2022).

Anhang

Anhang A: Inhalt des Web App Manifest nach dem W3C Standard

Attribut	Beschreibung	mögliche Werte
dir	Textrichtung für lokalisierbare Felder der Manifestdatei (name, short_name, description)	"ltr" left-to-right "rtl" right-to-left "auto" (default)
lang	Sprache der Werte in der Manifestdatei	String in Form eines IETF BCP 47 Language-Tag, z.B. "de-DE"
scope	URL-Bereich, auf den sich das Manifest auswirken soll	String mit URL
start_url	URL, die beim Start der Anwendung aufgerufen werden soll	String mit URL
id	Repräsentiert die Identität der Anwendung	String
name	Name der Anwendung, der auf dem Splash Screen angezeigt wird	String
short_name	Kurze Version des Namens der Anwendung	String
description	Allgemeine Beschreibung zum Zweck der Anwendung	String
icons	Bilder zur Repräsentation der Anwendung in verschiedenen Kontexten, z.B. zur Darstellung der Anwendung im Task-Switcher, in der Anwendungsliste eines Betriebssystems oder für den Splash Screen	Liste von Bildobjekten Ein Bildobjekt kann folgende Attribute und Werte besitzen: src: String mit Pfad der Bilddatei sizes: String mit Bildmaßen type: String mit Medientyp des Bildes
display	Vom Entwickler präferierter Anzeigemodus für die Anwendung	"fullscreen" "standalone" "minimal-ui" "browser"
orientation	Standardausrichtung der Anwendung	"any" "natural" "landscape" "landscape-primary" "landscape-secondary" "portrait" "portrait-primary" "portrait-secondary"
theme_color	Primäre Farbe der Anwendung	String mit CSS-Farbwert oder hexadezimalen RGB-Farbwert
background_color	Hintergrundfarbe der Anwendung	String mit CSS-Farbwert oder hexadezimalen RGB-Farbwert

Attribut	Beschreibung	mögliche Werte
related_applications	Liste von Anwendungen, die mit dieser Anwendung verwandt ist	Liste von Anwendungsobjekten Ein Anwendungsobjekt kann folgende Attribute und Werte besitzen: platform: String mit Plattform url: String mit URL id: String
prefer_related_applications	Gibt an, ob dem Benutzenden mitgeteilt werden soll, dass verwandte Anwendungen verfügbar sind	Boolean
shortcuts	Liste von Shortcut-Elementen, für den schnellen Zugriff auf wichtige Bestandteile (Views) der Anwendung	Liste von Shortcut-Objekten Ein Anwendungsobjekt kann folgende Attribute und Werte besitzen: name: String short_name: String description: String url: String mit URL icons: Liste von Bildobjekten

Tabelle 11: Inhalt des Web Application Manifests nach dem W3C Standard

Quelle: Eigene Darstellung. Datenquelle: W3C, 2022.

Anhang B: Dokumentation der REST-API der Fachhochschule Erfurt

Base URL: <http://app.fh-erfurt.de:8080/fheapp/api/eah>

Endpoint	Methode	Beschreibung	Beispiel
/timetable	GET	<ul style="list-style-type: none"> - Rückgabe aller Studiengänge und Setgruppen - für jede Setgruppe steht eine timetableId 	siehe Code-Snippet 22 Code-Snippet 22: GET /timetable - Response Schema
/timetable/events? timetableId=<timetableId>	GET	<ul style="list-style-type: none"> - Rückgabe aller in der Zukunft liegenden Stundenplandaten - timetableId = Set (SPLUSnnnnnn) - UID = Veranstaltung (ID eines TimeTableVo) - Veranstaltungsnamen setzen sich zusammen aus Fach, Veranstaltungstyp, Gruppe (siehe Anhang B.1 Zusätzliche Hinweise zu den Attributen) 	siehe Code-Snippet 23
/canteens	GET	<ul style="list-style-type: none"> - Rückgabe einer Liste der Mensen in Jena 	siehe Code-Snippet 24
/canteens/<id>	GET	<ul style="list-style-type: none"> - Rückgabe des Speisenangebots einer Mensa 	Code-Snippet 25

Tabelle 12: Übersicht über verwendete API Endpoints

Quelle: Eigene Darstellung

```
{
  "courseOfStudies": [
    {
      "title": "Bachelor: Augenoptik",
      "id": "AO (BA) ",
      "numberOfTerms": 8,
      "terms": [
        {
          "title": "AO (BA) 2",
          "id": "2",
          "studyGroups": [
            {
              "title": "AO (BA) 2.01",
              "timetableId": "SPLUSD44A96",
              "semesterGroup": false
            },
            {
              "title": "AO (BA) 2.02",
              "timetableId": "SPLUSD44A99",
              "semesterGroup": false
            }
          ]
        }
      ]
    }
  ],
  "title": "EAH Jena Stundenplan",
  "modified": "Thu, 24 Mar 2022 02:49:36 CET"
}
```

Code-Snippet 22: GET /timetable - Response Schema (gekürzt)

```
[
  {
    "weekInYear": 14,
    "year": 2022,
    "weekdays": [
      {
        "dayInWeek": 2,
        "name": "Montag",
        "events": [
          {
            "startDate": 1649057400000,
            "title": "AO(BA)OTS/S/01/01",
            "shortTitle": "OTS/S/01/01",
            "date": "04.04.2022",
            "dayOfWeek": "Montag",
            "weekOfYear": 14,
            "startTime": "09:30",
            "endTime": "11:00",
            "lecturer": "Gebhardt",
            "room": "05.00.03",
            "uid": "SPLUSC27388s-0"
          }
        ]
      },
      {
        "dayInWeek": 3,
        "name": "Dienstag",
        "events": [ ... ]
      }
    ]
  }
]
```

Code-Snippet 23: GET /timetable/events?timetableId=SPLUSD44A96 - Response Schema (gekürzt)

```
[
  {
    "id": "58",
    "name": "Mensa Carl-Zeiss-Promenade",
    "city": "Jena",
    "urlPath": "https://www.stw-thueringen.de/mensen/jena/mensa-carl-zeiss-promenade.html"
  },
  {
    "id": "61",
    "name": "Cafeteria EAH",
    "city": "Jena",
    "urlPath": "https://www.stw-thueringen.de/mensen/jena/cafeteria-eah.html"
  }
]
```

Code-Snippet 24: GET /canteens - Response Schema (gekürzt)

```
[
  {
    "title": "Essen 1",
    "description": "Bandnudeln mit Zucchini und Tomate, dazu Parmesan",
    "ingredients": " T2",
    "price": "1,60 €",
    "date": 1648076400000,
    "dateAsString": "24.03.2022",
    "mensaName": "Mensa Carl-Zeiss-Promenade",
    "mensaId": "58"
  },
  {
    "title": "Essen 2",
    "description": "veganer Grießbrei mit Sauerkirschragout, Vorsuppe",
    "ingredients": "",
    "price": "1,85 €",
    "date": 1648076400000,
    "dateAsString": "24.03.2022",
    "mensaName": "Mensa Carl-Zeiss-Promenade",
    "mensaId": "58"
  }
]
```

Code-Snippet 25: GET /canteens/58 - Response Schema (gekürzt)

Anhang B.1 Zusätzliche Hinweise zu den Attributen

„

Beispiel:

WI/WIEC(BA)Ma/Ü/04.2

- WI/WIEC(BA) → Studiengang (Bachelor)
- Ma → Fach
- /Ü → Übung
- /04 → Gruppe (Übungs- oder Praktikumsgruppe (in die das Set eingeteilt ist), bei VL normalerweise für alle 01)
- .2 → Nr. der Eintragung - entsteht, wenn es mehrere Termine pro Woche oder Änderungen gibt

Das heißt die gleiche Veranstaltung hat ein regelmäßiges Muster für die Namen:

WI/WIEC(BA)Ma/Ü/04.n

bzw. zu Anfang bis es einen weiteren oder geänderten Termin gibt:

WI/WIEC(BA)Ma/Ü/04

[...]

es existieren folgende **Veranstaltungskürzel**:

APL, B, E, Kon, Lehre, mdl. Prfg., P, PL, S, T, V, Ü, Wdh.-Prfg., Wdh.-APL

⁶⁸
„

⁶⁸ Vgl. Gorczak et al., 2022.

Anhang C: Corporate Design Richtlinien der Ernst-Abbe-Hochschule Jena

Farben:

- **Basisfarbe:**

Türkisgrün mit Hexadezimalcode #009999

Mögliche Abstufungen durch weniger Deckkraft und höheren Grauteil (siehe Abbildung 14)

CMYK		70%	50%	20%
Cyan	100 %	70 %	50 %	20 %
Magenta	0 %	0 %	0 %	0 %
Gelb	45 %	32 %	23 %	9 %
Schwarz	0 %	0 %	0 %	0 %
Schwarz	100 %	70 %	50 %	20 %

Abbildung 14: Abstufungen der Primärfarbe nach Corporate Design
Quelle: Ernst-Abbe-Hochschule Jena (2021)

- **Akzentfarben:**

Lindgrün - Hex-Code: #C9DD03

Verkehrsblau - Hex-Code: #0039A6

Lichtblau - Hex-Code: #00B9E4

Tiefseeblau - Hex-Code: #34495B



Abbildung 15: Corporate Design Akzentfarben der Ernst-Abbe-Hochschule Jena
Quelle: Ernst-Abbe-Hochschule Jena (2021)

Typografie:

- **Hauptschrift:** Roboto Condensed
- **Alternativschrift:** Arial
 - Wird genutzt, wenn Roboto Schrift nicht verfügbar ist

Logo:

- **Kurzversion:**

- Zur Verwendung auf allen Arten von Werbe- und Informationsmaterialien.
- Die Schutzzone muss eingehalten werden (siehe Abbildung 16)
- Sollte nach Möglichkeit in der Farbversion verwendet werden. Alternativ steht eine Schwarz-Weiß-Version, eine weiße und eine weiß-türkis-grüne Version zur Verfügung

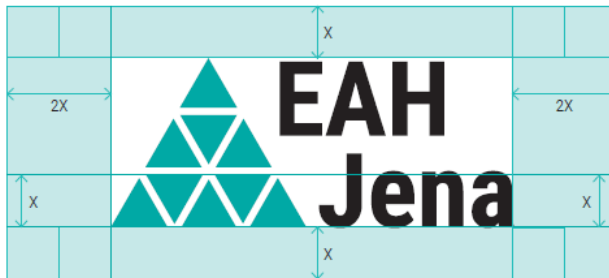


Abbildung 16: EAH-Logo Kurzversion mit Schutzzone
Quelle: Ernst-Abbe-Hochschule Jena (2021)

Anhang D: Übersicht über grundlegende Befehle der Angular-CLI

Befehl	Beschreibung
ng new <name> [options]	Erstellt und initialisiert eine neue Angular-Anwendung in einem neuen Angular Workspace
ng serve <project> [options]	Baut die App und stellt sie für eine Ausführung im Browser bereit. Bei Codeänderungen wird neu gebaut und die Änderungen werden direkt angezeigt.
ng build <project> [options]	Kompiliert eine Angular-Anwendung in ein Ausgabeverzeichnis namens dist/. Der Inhalt ist das Angular Artefakt zur Veröffentlichung.
ng generate <schematics>	Erzeugt und modifiziert Dateien basierend auf einem Angular Schema. Angular Schemata sind bspw. Klassen, Komponenten, Module, Pipes und Services.
ng add <collection> [options]	Fügt das npm-Paket für eine externe Bibliothek zum Workspace hinzu und konfiguriert das Projekt im aktuellen Arbeitsverzeichnis für die Verwendung der Bibliothek, wie im Schema der Bibliothek spezifiziert.
ng lint <project> [options]	Führt eine statische Codeanalyse durch und zeigt ggf. Schwachstellen im Code auf.
ng test <project> [options]	Führt Unit Tests des Projekts aus.
ng e2e <project> [options]	Baut die Anwendung, stellt sie zur Ausführung bereit und führt Ende-zu-Ende-Tests durch
ng update [options]	Führt Updates des Frameworks und der Abhängigkeiten durch.
ng version	Gibt die aktuelle Version der Angular CLI aus.

Tabelle 13: Übersicht der wichtigsten Befehle der Angular CLI

Quelle: Eigene Darstellung in Anlehnung an CLI Overview and Command Reference (o. D.)

Anhang E: Vorschlag zur Neugestaltung des REST-Endpoints für Mein Stundenplan**aktuell:**

- `http://app.fh-erfurt.de:8080/fheapp/api/eah/timetable`
- Ein Request ist ausschließlich anhand der Setgruppe über SPLUS-Nummer möglich
- `title` und `shortTitle` sind zusammengesetzte Konstrukte aus verschiedenen Attributen, die im Frontend aufgrund der Logik der App zerlegt werden müssen
 - *Zusammensetzung shortTitle:*
`<Veranstaltungsname>/<Veranstaltungs-Typ>/<Gruppennummer>`
z.B. *Opto I/V/01*
 - Teilweise treten Unregelmäßigkeiten auf, die ein valides Parsen erschweren und so zu Fehlern bei der Ausführung führen

neu:

Endpoint zur Abfrage der verfügbaren Kurse mit Auswahl von Studiengang und Semester für Mein Stundenplan

- `/timetable/events/<study-course-id>/<semester>`
BSP: Anfrage aller Kurse für das zweite Semester des Bachelorstudiengangs Augenoptik: `/timetable/events/AO(BA)/2`
- Rückgabe einer Liste mit Events für diese Auswahl
 - Die Liste sollte gruppiert sein: nach Set-Liste, Veranstaltungsname und Veranstaltungstyp, d.h. in einem Objekt sind Veranstaltungsname, Veranstaltungstyp und die Setliste gleich
 - Ein Beispiel für ein mögliches Response-Schema zeigt Code-Snippet 26

```
[
  {
    "name": "Ana/Phys.",
    "type": "Ü",
    "set": ["1", "3", "5", "7", "9"],
    "lecturer": "Wicher",
    "whereAndwhen": [
      {
        "date": "01.05.2022",
        "startTime": "12:00",
        "endTime": "15:00",
        "room": "01.01.01"
      },
      {
        "date": "08.05.2022",
        "startTime": "12:00",
        "endTime": "15:00",
        "room": "01.01.01"
      }
    ]
  },
  {
    "name": "Ana/Phys.",
    "type": "Ü",
    "set": ["2", "4", "6", "8", "10"],
    "lecturer": "Wicher",
    "whereAndwhen": [
      {
        "date": "01.05.2022",
        "startTime": "15:00",
        "endTime": "18:00",
        "room": "01.01.01"
      },
      {
        "date": "18.05.2022",
        "startTime": "11:30",
        "endTime": "13:00",
        "room": "01.01.01"
      }
    ]
  },
  {
    "name": "Ma I",
    "type": "T",
    "set": ["1", "2", "3", "4", "6"],
    "lecturer": "Tutor XX",
    "whereAndwhen": [
      {
        "date": "01.05.2022",
        "startTime": "12:00",
        "endTime": "15:00",
        "room": "01.01.01"
      }
    ]
  }
]
```

Code-Snippet 26: Response-Schema für einen neuen Mein-Stundenplan-Endpoint

Ehrenwörtliche Erklärung und Einverständniserklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig und ohne unerlaubte Hilfe Dritter verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle Stellen, die inhaltlich oder wörtlich aus Veröffentlichungen stammen, sind kenntlich gemacht. Diese Arbeit lag in der gleichen oder ähnlichen Weise noch keiner Prüfungsbehörde vor und wurde bisher noch nicht veröffentlicht. Hiermit erkläre ich mich mit der Einsichtnahme in meine Abschlussarbeit im Archiv der Bibliothek der EAH Jena nicht einverstanden.

Ort, Datum

Unterschrift