



Möglichkeiten zur Umsetzung Barrierefreier Web-Anwendungen mithilfe Aktueller Web-Frameworks

Bachelorarbeit

im Studiengang Angewandte Informatik der Fakultät Wirtschaftsinformatik und
Angewandte Informatik der Otto-Friedrich-Universität Bamberg

Lehrstuhl für Medieninformatik

Verfasser: Leonard Paul Siegmayer

Prüfer: Prof. Dr. Andreas HENRICH

20.08.2020

Gender Erklärung

Die in dieser Arbeit gewählte männliche Form bezieht sich immer zugleich auf weibliche, männliche und diverse Personen. Auf eine Mehrfachbezeichnung wird in der Regel aus Gründen der besseren Lesbarkeit verzichtet.

Zusammenfassung

Menschen mit Behinderungen muss es ermöglicht werden, trotz besonderer physischer oder psychischer Merkmale, am alltäglichen Leben in der Gesellschaft teilnehmen zu können. Mit der zunehmenden Relevanz des Internets im Alltag ist es zwingend notwendig, auch bei der Entwicklung von Web-Anwendungen einen erhöhten Fokus auf die Barrierefreiheit zu legen. Frameworks sind ein wesentlicher Bestandteil der modernen Web-Entwicklung und können je nach Art einen unterschiedlich großen Einfluss auf die Barrierefreiheit haben. Dieser ist bei CSS-Frameworks wie Bootstrap besonders groß, da Barrierefreiheit stark mit der Interaktion und Wahrnehmung der vorhandenen Inhalte zusammenhängt. Bei SPA-Frameworks sind die Möglichkeiten zwar etwas geringer, trotzdem zeichnet sich vor allem Angular, zusammen mit der UI-Bibliothek Angular Material, durch einen besonders hohen Fokus auf Barrierefreiheit aus. Im Kontrast dazu steht das Python-Framework Django, das durch die Abwesenheit expliziter Unterstützung auffällt.

Allgemein haben Web-Frameworks zwar das Potenzial, Entwickler bei der Umsetzung der Barrierefreiheit zu unterstützen, es wird für diese aber immer notwendig sein, sich über Besonderheiten der Nutzer zu informieren, um aktiv den Aufbau von Barrieren zu vermeiden.

Inhaltsverzeichnis

1 Motivation und Aufgabenstellung	1
2 Arten von Beeinträchtigungen und allgemeine Lösungsansätze	3
2.1 Sehbehinderungen	3
2.2 Hörbehinderungen	5
2.3 Motorische Behinderungen	6
2.4 Kognitive Beeinträchtigungen	7
2.5 Psychische Krankheiten	8
2.6 Web Content Accessibility Guidelines	9
3 Barrierefreiheit in modernen Web-Frameworks	11
3.1 Boostrap: Ein CSS-Framework	11
3.1.1 Grid-System	11
3.1.2 Klassen für Screenreader	12
3.1.3 Animationen und Farben	13
3.1.4 Komponenten	14
3.1.5 Vergleich mit weiteren CSS-Frameworks: Foundation und PUXL-Framework	16
3.2 Angular: Ein SPA-Framework	17
3.2.1 Komponentensystem, Data-Binding und Routing	17
3.2.2 Codelyzer	18
3.2.3 Angular Material	19
3.2.4 Animationen	21
3.2.5 Vergleich mit einem verwandten Framework: Vue.js	21
3.3 Django: Ein Python-Framework	22
3.3.1 Template-System	22
3.3.2 Verbesserte Lesbarkeit durch Filter	22
3.3.3 Admin-Interface	22
3.3.4 Formulare	24
3.3.5 Suche	24
3.3.6 Vergleich mit einem verwandten Framework: Ruby on Rails	25
3.4 Gegenüberstellung der betrachteten Web-Frameworks	25
4 Demonstration der Umsetzungsmöglichkeiten	27
4.1 Thema und Anforderungen	27
4.1.1 Inhaltliche Anforderungen	27

4.1.2	Anforderungen an die Barrierefreiheit	27
4.2	Architektur der Anwendungen	28
4.3	Umsetzung der Barrierefreiheit in den Anwendungen	30
4.3.1	Aspekte mit Bezug zu Persona #1: Parkinson	30
4.3.2	Aspekte mit Bezug zu Persona #2: Katarakt	34
4.3.3	Aspekte mit Bezug zu Persona #3: Erblindung	34
4.3.4	Aspekte mit Bezug zu Persona #4: Demenz	35
4.3.5	Weitere Aspekte	36
5	Evaluierung	38
5.1	Evaluierung mithilfe der WCAG 2.1-Kriterien	38
5.2	Auswertung der Ergebnisse	38
6	Ausblick und Fazit	43
	Bibliographie	44

Tabellenverzeichnis

1	Die 13 Richtlinien der WCAG 2.1	10
2	Codelyzer-Regeln zur Überprüfung der Barrierefreiheit	19
3	Filter des 'Humanize'-Pakets	23
4	WCAG Evaluierung - 'Prinzip 1: Wahrnehmbar'	39
5	WCAG Evaluierung - 'Prinzip 2: Bedienbar'	40
6	WCAG Evaluierung - 'Prinzip 3: Verständlich'	41
7	WCAG Evaluierung - 'Prinzip 4: Robust'	42

Abbildungsverzeichnis

1	Wahrnehmung mit unterschiedlichen Sehbehinderungen	4
2	Auswirkungen der <code>.order</code> -Klasse	12
3	<code>.sr-only</code> -Klassen für Skip-Links, Badges und Spinner	13
4	Hinweise auf Mängel inkl. Verbesserungsvorschläge im PUXL-Framework .	17
5	Heuristiken zur Verwendung der Angular Komponenten in Kombination mit ARIA und Skip-Links	18
6	Personae	29
7	Übersicht über die Angular-Komponenten und das Routing	30
8	ER-Diagramm der Klassen aus dem Angular-Projekt	31
9	Fokus-Management mit dem <code>NavigationEnd</code> -Event des Angular-Routers .	32
10	Implementierung eines Overlays mit der Anguar 'FocusTrap'	32
11	Auswahl der Optionen der Auto vervollständigung mit Hilfe der Schnittstelle zur Datenbank in Django	33
12	Beispielanwendung der 'SkipLinkComponent' in Angular	34
13	ARIA- und 'Autocomplete'-Attribute in Django-Formularen (vereinfachter Ausschnitt)	34
14	Typischer Aufbau der Hauptkomponenten von Unterseiten	35
15	Template-Vererbungssystem in Django für Breadcrumbs	36

Kapitel 1

Motivation und Aufgabenstellung

Große Teile des alltäglichen Lebens werden durch die Digitalisierung zunehmend in das Internet verschoben. Home-Office, Online-Banking und Home-Schooling gewinnen immer mehr an Relevanz. Zuletzt sorgte die COVID-19-Pandemie zudem dafür, dass diese Konzepte zum Teil vorübergehend alternativlos wurden. In der 'Allgemeinen Erklärung der Menschenrechte' der UN-Vollversammlung wird jedem Menschen das Recht auf Bildung, Arbeit und Kultur zugesprochen [UN-Vollversammlung, 1948]. Grundsätzlich hat auch jeder das Recht auf Internetzugang [UNHRC, 2016]. Doch um diese Rechte für jeden gewährleisten zu können, darf bei der Entwicklung von Web-Anwendungen nicht der Aspekt der Barrierefreiheit vernachlässigt werden. In der WebAIM Analyse der Barrierefreiheit von 2020 wurden auf 98,1 % der insgesamt eine Million betrachteten Webseiten Mängel entdeckt. Durchschnittlich waren es 60,9 pro Webseite. Im Vergleich zum Vorjahr sind das etwas schlechtere Ergebnisse, was durch den Anstieg der Komplexität der Anwendungen erklärt werden kann. Die drei häufigsten Probleme stellten hierbei zu geringer Kontrast, fehlende Alternativtexte und leere Links dar [WebAIM, 2020].

Dabei gibt es neben ethischen, moralischen und gesetzlichen Gründen noch weitere Anreize, um einen erhöhten Fokus auf Barrierefreiheit bei der Entwicklung zu rechtfertigen. Einer davon ist die erweiterte Zielgruppe, durch die der Gewinn gesteigert werden kann. Schätzungsweise 15 % der gesamten Weltbevölkerung, also mehr als eine Milliarde Menschen, leiden an einer Art von Behinderung. Bei 2,2 bis 3,8 % der über 15-Jährigen liegt eine als 'schwerwiegend' eingestufte vor. Aufgrund des Alterns der Bevölkerung und deren steigenden Wachstums wird erwartet, dass diese Zahlen in den kommenden Jahren noch weiter ansteigen werden [WHO, 2011]. Zahlreiche Fallstudien unterstreichen die Annahme, dass Barrierefreiheit einen maßgeblichen positiven Einfluss auf den Erfolg einer Web-Anwendung hat. 'This American Life', ein amerikanischer Radiosender, musste aufgrund neuer gesetzlicher Regulierungen Audio-Transkriptionen zu allen archivierten Aufzeichnungen erstellen. Als Folge dessen stieg die Zahl der 'Unique Visitors' um 4,18 % [3PlayMedia, 2014].

Außerdem haben Anwendungen, die großen Wert auf Barrierefreiheit legen, auch einen positiven Einfluss auf das Erlebnis derjenigen, die nicht unbedingt darauf angewiesen sind. Von höherem Kontrast, alternativen Eingabemethoden, verständlicherer Sprache oder einfacherer Bedienung kann jeder Nutzer profitieren. Des Weiteren befindet sich jede Person einmal in einer Situation, in der sie temporär beeinträchtigt ist. Beispielsweise in der U-Bahn, in der durch das Schaukeln des Waggons das Smartphone nur schwer zu bedie-

nen ist, oder durch vorübergehende Behinderungen, wie etwa einen gebrochenen Arm. Entwickler, die sich früh in der Konzeption mit Barrierefreiheit auseinandersetzen, können bereits mit wenig Aufwand viel erreichen. Häufig ist schon das Bewusstsein über mögliche Barrieren zu Beginn der Entwicklung ausreichend, um deren Aufbau ohne großen zusätzlichen Aufwand zu vermeiden. Inzwischen gibt es zahlreiche assistive Technologien wie 'Screenreader', 'Braille Displays' oder 'Eye-Tracker', die Nutzer mit Behinderungen bei der Interaktion mit dem Web unterstützen. Diese Hilfen funktionieren bereits sehr gut, können aber nur bestmöglich genutzt werden, wenn die besuchten Webseiten auch dafür optimiert wurden. Hinsichtlich der Barrierefreiheit sehen laut einer Umfrage 85,3 % der Nutzer eher ein Problem bei der Gestaltung der Webseiten als bei den assistiven Technologien [WebAIM, 2017a]. Aufgrund dessen stellt sich nicht nur die Frage, was Entwickler tun können, um dieses Problem zu beheben, sondern auch wie moderne Technologien bei der Entwicklung barrierefreier Web-Anwendungen Unterstützung bieten können. Web-Frameworks wie Bootstrap, Angular oder Django sind aus dem modernen Technologie-Stack nicht mehr wegzudenken. Deshalb soll in der vorliegenden Arbeit für diese untersucht werden, welche Features und Werkzeuge sie Entwicklern an die Hand geben, um zu einem Endprodukt ohne Barrieren zu verhelfen. Außerdem wird untersucht, wie sie im Vergleich zueinander abschneiden und sich gegenseitig ergänzen können. Auch wird gezeigt, an welchen Stellen sie keine Unterstützung bieten können.

Kapitel 2 befasst sich zunächst allgemein mit Behinderungen und den Barrieren, auf die Betroffene im Internet stoßen können. Dabei werden auch grundlegende Lösungsansätze zu deren Abbau vorgestellt.

Das anschließende Kapitel wird daraufhin drei verschiedene Arten von Web-Frameworks hinsichtlich der darin enthaltenen Unterstützung für Barrierefreiheit untersuchen: Bootstrap, ein Cascading-Style-Sheets-Framework (CSS-Framework), Angular, ein Single-Page-Application-Framework (SPA-Framework), und Django, ein in der Programmiersprache Python geschriebenes Backend-Framework.

Die in Kapitel 3 erarbeiteten Konzepte werden in Kapitel 4 anhand einer Beispielanwendung demonstriert. Thema dieser ist die Web-Anwendung einer fiktiven Fluggesellschaft namens 'Ally-Airlines'. Die Umsetzung erfolgt in zwei Projekten. Eines davon kombiniert Bootstrap und Django, das andere Angular und die dazugehörige Erweiterung für User-Interface-Komponenten (UI-Komponenten), Angular Material.

In Kapitel 5 werden die Ergebnisse mithilfe der 'Web Content Accessibility Guidelines' (WCAG) und automatisierter Tools evaluiert.

Kapitel 6 rundet die Arbeit mit einer Zusammenfassung der Kernaspekte und einem Ausblick auf mögliche weitere Themen der Barrierefreiheit, allgemein in der Informatik und speziell im Internet, ab.

Kapitel 2

Arten von Beeinträchtigungen und allgemeine Lösungsansätze

Bevor untersucht werden kann, wie aktuelle Web-Frameworks die Entwicklung barrierefreier Web-Anwendungen unterstützen können, soll in diesem Kapitel aufgezeigt werden, welche Beeinträchtigungen unterschiedliche Nutzer haben, auf welche Barrieren sie bei der Verwendung des Webs stoßen und wie diese abseits von Frameworks abgebaut werden können. Hierbei wird unter anderem auch sichtbar, dass Web-Entwickler trotz vorhandener technischer Unterstützungen ein umfangreiches Verständnis ihrer Nutzer besitzen müssen, um sich über mögliche Auswirkungen, teilweise trivial erscheinender Entscheidungen, bewusst werden zu können. Dies wird vor allem bei den psychischen Krankheiten deutlich werden. Am Ende des Kapitels wird zudem ein Blick auf die WCAG geworfen.

2.1 Sehbehinderungen

Mindestens 2,2 Milliarden Menschen weltweit leiden an einer eingeschränkten Sehfähigkeit oder an Blindheit [WHO, 2019b]. Bei denjenigen mit Kurz- und Weitsichtigkeit ist das Tragen einer Brille oder von Kontaktlinsen ausreichend, um die Sehschwäche zu korrigieren. Doch es existiert eine Vielzahl weiterer Erkrankungen, für die keine derart einfache Lösung existiert. Dazu zählen

- die Makuladegeneration, die zu Sehverlust im Zentrum des Sichtbereichs führt,
- die Katarakt (Grauer Star), die zu einer Trübung in manchen Bereichen des Sichtbereichs führt,
- die diabetische Retinopathie, die zu Verzerrung und Verschwommenheit in manchen Bereichen des Sichtbereichs führt,
- das Glaukom (Grüner Star), das zu peripherem Sehverlust führt,
- und verschiedene Formen von Farbenblindheit, die zu eingeschränkten Farbwahrnehmungen führen [WebAIM, 2013].

Wie diese Krankheiten die Wahrnehmung beeinflussen, kann anhand Abbildung 1 veranschaulicht werden. Einige Erkrankungen am Auge können auch zu vollständiger, irreversibler Erblindung führen [WebAIM, 2013].

KAPITEL 2. ARTEM VON BEEINTRÄCHTIGUNGEN UND ALLGEMEINE LÖSUNGANSÄTZE

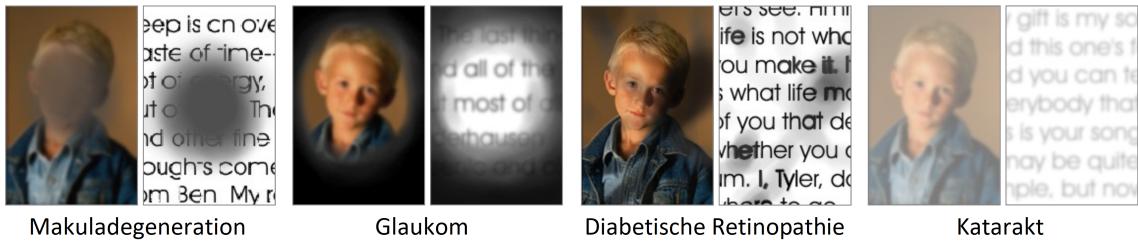


Abbildung 1: Wahrnehmung mit unterschiedlichen Sehbehinderungen [WebAIM, 2013]

Blindheit kann leicht durch das Schließen der Augen nachgeahmt werden, um eine Web-Anwendung daraufhin zu testen. Auch andere Sehbehinderungen, wie beispielsweise Farbenblindheit, können mithilfe entsprechender Tools simuliert werden. Im Gegensatz zu einigen der nachfolgend behandelten Beeinträchtigungen, ist es demnach vergleichsweise einfach, sich für Testzwecke in Nutzer mit Sehbehinderungen hineinzuversetzen.

Ohne Sehen ist ein unabhängiges Leben schwer umsetzbar, die Möglichkeit, trotzdem mit dem Internet interagieren zu können, kann aber dabei unterstützen. Besonders hilfreich sind dabei 'Screenreader', die Inhalte laut vorlesen und bei der Navigation helfen. 'Braille Displays' funktionieren ähnlich, die Ausgabe erfolgt hierbei aber nicht auditiv, sondern textuell in Brailleschrift. Eine Herausforderung für Entwickler besteht darin, Informationen, die ausschließlich visuell vorhanden sind, und Kontexte, die durch die Linearität dieser Technologien verloren gehen, dennoch zu vermitteln [Yesilada et al., 2019].

HTML-Semantik

Screenreader bieten die Möglichkeit, alle Überschriften, Links und Seitenregionen ('Landmarks') der aktuellen Seite wiederzugeben, um diese zur Navigation und zur Verschaffung eines Überblicks zu verwenden. Damit diese Funktion sinnvoll genutzt werden kann, müssen Überschriften und Links auch ohne Kontext verständlich sein. Ein Link mit der Beschriftung 'klicken Sie bitte hier' ist das nicht. Außerdem sollten Heading-Elemente in einer logischen Hierarchie zueinander stehen [WebAIM, 2017b].

Eine konsistente und korrekte Verwendung semantischer Hypertext-Markup-Language-Elemente (HTML-Elemente) erleichtert die Navigation bei der Verwendung von Screenreadern. Wenn ein entsprechendes Element bereits existiert, sollte dieses auch genutzt werden, anstatt ein eigenes mithilfe von Elementen ohne Semantik zu erstellen. Die beiden in HTML integrierten Attribute 'alt', für textuelle Alternativen visueller Inhalte, und 'lang', zur Definition der verwendeten Sprache, spielen ebenfalls eine bedeutende Rolle [Firth, 2019].

ARIA

Zusätzliche semantische Informationen können durch 'ARIA' ergänzt werden. Dabei handelt es sich um eine vom 'World Wide Web Consortium' (W3C) bereit gestellte Spezifikation, die durch die Anwendung bestimmter Attribute integriert werden kann. Die wichtigsten Konzepte dabei sind Rollen für die Beschreibung der Art von Elementen und Seitenregionen, Label für deren Beschriftung und das Setzen von Beziehungen zwischen mehreren Elementen. Außerdem können durch 'ARIA-Live' Benachrichtigungen und Pop-ups, die

KAPITEL 2. ARDEN VON BEEINTRÄCHTIGUNGEN UND ALLGEMEINE LÖSUNGANSÄTZE

nicht automatisch fokussiert werden und somit für Nutzer von Screenreadern verborgen bleiben, wahrnehmbar gemacht werden [Cooper, 2016]. ARIA wird auch im weiteren Verlauf der Arbeit eine wesentliche Rolle spielen. Dabei wird zu sehen sein, wie es in Kombination mit Frameworks funktioniert. Anwendungsszenarien werden im praktischen Teil der Arbeit demonstriert.

Styling

Für Menschen mit eingeschränkter Sehfähigkeit müssen die zu vermittelnden Inhalte deutlich präsentiert werden. Für den verwendeten Text bedeutet das, dass eine leicht leserliche Schriftgröße und -art gewählt werden sollten. Außerdem sollten Schriftgröße und Zeilenabstand über relative Einheiten, wie beispielsweise 'em' anstelle von 'px'¹, definiert werden. Dadurch passt sich die Größe an die von Nutzern vordefinierten Skalierungen im Browser an. Für diejenigen mit Glaukom, Katarakt oder ähnlichen Krankheiten sollten die Kontraste groß genug sein. Farbschemata sollten so gewählt werden, dass sie für möglichst viele Arten von Farbenblindheit funktionieren [Firth, 2019]. Für viele ist es notwendig, Inhalte zu vergrößern, um sie ausreichend wahrnehmen zu können. Damit die Struktur auch bei großen Zoomstufen ordnungsgemäß beibehalten wird, müssen Webseiten responsiv gestaltet werden [Yesilada et al., 2019].

Navigation und Interaktion

Die bisher beschriebenen Ansätze werden genutzt, um bei der Wahrnehmung der Inhalte zu helfen. Doch auch deren Navigation und Interaktion muss optimiert werden, da die Verwendung einer Maus für blinde Nutzer nicht und für sehschwache nicht immer möglich ist. Wie dieses Problem angegangen werden kann, wird in Kapitel 2.3 'Motorische Behinderungen' beschrieben. Dabei ist die effektive und effiziente Nutzung des 'Fokus' von besonderer Bedeutung.

2.2 Hörbehinderungen

Hörbehinderungen können angeboren sein oder erst im Laufe des Lebens auftreten. Dabei kann zwischen Taubheit und verschiedenen Arten der Schwerhörigkeit unterschieden werden. Etwa 5 % der Weltbevölkerung leiden an Hörverlust. Bis 2050 wird erwartet, dass diese Zahl auf 10 % ansteigen wird [WHO, 2020].

Untertitel und Gebärdensprache

Eine notwendige Maßnahme, um auch hörgeschädigten Menschen ein barrierefreies Erlebnis bieten zu können, ist die Verwendung von Untertiteln und bestenfalls einer zusätzlichen Alternative in Gebärdensprache für Video- und Audioinhalte. Letzteres ist notwendig, da Menschen, die bereits von Geburt an taub sind, häufig Schwierigkeiten mit dem Lesen haben. Generell bevorzugen die meisten Nutzer mit Hörbehinderungen Gebärdensprache gegenüber Lautsprachen [Firth, 2019].

¹Die Einheit 'em' beschreibt die Größe relativ zum Eltern-Element. Die Einheit 'px' hingegen entspricht der Größe eines Bildpunkts auf dem anzeigenenden Gerät [W3C, 2010].

KAPITEL 2. ARTEM VON BEEINTRÄCHTIGUNGEN UND ALLGEMEINE LÖSUNGANSÄTZE

Untertitel haben zudem weitere Vorteile. So können sie Nutzer beim Erlernen neuer Sprachen oder in Umgebungen mit lauten Hintergrundgeräuschen unterstützen. Außerdem profitieren Betreiber von Web-Anwendungen von einer verbesserten Auffindbarkeit durch Suchmaschinen [Firth, 2019].

Soziale Aspekte

Auch auf sozialer Ebene gibt es Aspekte, die bei der Erstellung einer Web-Anwendung beachtet werden müssen. So ist das automatische Abspielen von Videos oder Audiodateien nicht nur allgemein eher störend, sondern kann auch eine Person, die das aufgrund einer Hörbehinderung nicht mitbekommt, in eine sozial unangenehme Situation bringen. Befindet sie sich beispielsweise in einem Zug, kann der Sound andere Reisende belästigen, ohne dass dies beabsichtigt ist.

Des Weiteren sollte es für Kontaktaufnahmen, beispielsweise mit dem Support, neben der telefonischen eine weitere Option für Hörgeschädigte Nutzer geben. Dafür bieten sich Web-Chats oder E-Mails an. Auch sogenannte 'Text-Relay-Dienste' sind eine Option, bei denen ein Assistent als Vermittler dient, indem Gesprochenes in Schrift bzw. Schrift in Sprache umgewandelt wird [Firth, 2019].

2.3 Motorische Behinderungen

Menschen mit motorischen Behinderungen leiden an einem vollen oder partiellen Verlust der Kontrolle über Funktionen des Körpers und an chronischen Schmerzen, geringer Ausdauer, Muskelschwäche oder Lähmung. Damit Nutzer mit Erkrankungen, wie Multipler Sklerose, Parkinson oder Infantiler Zerebralparese, die solche Behinderungen verursachen, trotzdem mit dem Web interagieren können, müssen alternative Eingabegeräte unterstützt werden. Die Maus ist heutzutage Standard, doch sie erfordert einen hohen Grad an Hand-Auge-Koordination. Barrierefreie Web-Anwendungen sollten deshalb für eine Interaktion, die ausschließlich mit Tastatur, Touch-Screen, Joystick, Eye-Tracker oder Mundstab erfolgt, optimiert sein [WebAIM, 2014].

Fokus

Ein Element wird als 'fokussiert' bezeichnet, wenn es ausgewählt ist und ihm die Eingabe des Nutzers zugeordnet wird. Viele Menschen mit motorischen Behinderungen sind auf die Navigation durch das Wechseln des Fokus mithilfe der Tabulator-Taste angewiesen. Um dieses Verfahren so einfach wie möglich zu gestalten, müssen einige Aspekte berücksichtigt werden. Generell gilt die Regel, dass interaktive Elemente, wie Buttons oder Textfelder, fokussierbar, und nicht interaktive, wie reiner Text, nicht fokussierbar sein sollten. Die visuelle Hervorhebung des aktuell ausgewählten Elements muss deutlich sein und darf nicht etwa aus ästhetischen Gründen deaktiviert werden. Auch die Reihenfolge der Elemente spielt eine entscheidende Rolle. Nutzer sollten vorhersehen können, welches Element als nächstes angesprungen wird. Die Reihenfolge des Fokus entspricht der der Elemente im Document Object Model (DOM)². Deshalb ist darauf zu achten, die visuelle Abfolge im Browserfenster an die logische im DOM anzupassen [Yesilada et al., 2019].

²Programmierschnittstelle für HTML-Dokumente, die deren logische Struktur und die Zugriffs- und Manipulationsmöglichkeiten darauf definiert [W3C, 1998].

KAPITEL 2. ARDEN VON BEEINTRÄCHTIGUNGEN UND ALLGEMEINE LÖSUNGSANSÄTZE

In HTML steht jedem Element das Attribut 'tabindex' zur Verfügung, mit dem festgelegt wird, ob und in welcher Reihenfolge Elemente fokussiert werden können. Das kann für Overlays und Pop-ups genutzt werden, die beim Erscheinen automatisch fokussiert werden sollten und die der Fokus nicht verlassen darf, um dahinterliegende, eventuell verdeckte Elemente zu erfassen. Nutzer dürfen aber auch nicht in solchen Overlays gefangen sein und sollten sie ohne Maus wieder verlassen können [Kearney et al., 2019].

Zeigerbasierte Eingabegeräte, Autocomplete und Bewegungssteuerung

Zeigerbasierte Eingabegeräte, wie die Maus, Touch-Screens oder Eye-Tracker, können bei zu kleinen Elementen dazu führen, dass versehentlich ein anderes naheliegendes ausgewählt wird. Nicht nur Menschen mit motorischen Behinderungen profitieren von ausreichend großen Flächen mit genügend Abstand voneinander.

Eine Funktion zur Autovervollständigung ('Autocomplete') von Eingabefeldern kann dabei helfen, die Anzahl der notwendigen Eingaben zu reduzieren und somit zu einer Erleichterung der Nutzer führen. Doch eine fehlerhafte Anwendung, bei der Daten mit falschen Feldern assoziiert werden, kann einen genau umgekehrten Effekt erzeugen, bei dem zuerst die automatisch ausgefüllten Felder zurückgesetzt und neu ausgefüllt werden müssen. Deshalb muss immer darauf geachtet werden, den Typ der Eingabe korrekt und so konkret wie möglich über das 'type'-Attribut festzulegen.

Zuletzt soll auch die Gefahr der Bewegungssteuerung betrachtet werden. Ist diese standardmäßig aktiviert, kann das zu unbeabsichtigten Eingaben von Nutzern mit motorischen Behinderungen führen. Außerdem sollte sie immer nur optional, mit einer Alternative vorhanden sein [Firth, 2019].

2.4 Kognitive Beeinträchtigungen

Menschen mit kognitiven Beeinträchtigungen haben mehr Schwierigkeiten mit einer oder mehreren mentalen Aufgaben als die durchschnittliche Person [WebAIM, 2018b]. Manche Betroffene können ihr Leben dennoch unabhängig weiterführen, bei anderen kann es aber dazu führen, dass sie auf die Hilfe anderer angewiesen sind. Krankheiten, die solche Beeinträchtigungen hervorrufen, sind unter anderem Autismus, Aufmerksamkeitsdefizit-/Hyperaktivitätsstörung (ADHS), Demenz und das Down-Syndrom. An Demenz, die hauptsächlich bei älteren Menschen über 60 auftritt, sind weltweit etwa 50 Millionen Menschen erkrankt und jedes Jahr kommen fast 10 Millionen neue Fälle hinzu [WHO, 2019a].

Sprache

Für Nutzer mit eingeschränktem Sprachverständnis ist es wichtig, dass der verwendete Text in einer einfachen Sprache verfasst wurde. Um das zu erreichen, sollten Sätze und Abschnitte kurz sein, unbekannte Wörter durch häufig verwendete Synonyme ersetzt und Redewendungen vermieden werden. Weitere Guidelines und Hilfen zur Verfassung barrierefreier Texte werden unter anderem von der US-Regierung zur Verfügung gestellt [PLAIN, 2011].

Definitionen unbekannter Begriffe können in HTML durch das <dfn>-Element bereitgestellt werden. Durch die Verwendung des <abbr>-Elements können Akronymen die vollständigen, ausgeschriebenen Bedeutungen hinzugefügt werden.

KAPITEL 2. ARTEM VON BEEINTRÄCHTIGUNGEN UND ALLGEMEINE LÖSUNGANSÄTZE

Wie auch für Menschen mit motorischen Behinderungen kann die 'Autocomplete'-Funktion Nutzern mit Rechtschreibschwäche helfen. Besonders hilfreich ist diese dann, wenn sie eine 'Fuzzy'-Suche erlaubt, bei der trotz Rechtschreibfehler passende Ergebnisse gefunden werden [Firth, 2019].

Design

Ein übersichtliches Design hilft Nutzern mit visuellen, räumlichen und Gedächtnisproblemen. Dies kann unter anderem dadurch realisiert werden, dass wiederkehrende Elemente, zum Beispiel Navigationsmenüs, konsistent platziert und bekannte sowie weit verbreitete Icons verwendet werden. Die Konsistenz bezieht sich dabei nicht nur auf die Inhalte und Struktur innerhalb der Anwendung selbst, sondern auch auf allgemein verbreitete Muster, die im Web zu finden sind [Yesilada et al., 2019].

Zwei Features, die häufig vorzufinden sind und bei der Orientierung helfen, sind Breadcrumbs und Sitemaps. Erstere sind Textzeilen, die den aktuellen Pfad anzeigen und zur Navigation zur Verfügung stellen. Sie können auch dazu verwendet werden, den jeweiligen Fortschritt einer mehrschrittigen Aktion anzuzeigen. Sitemaps sind Auflistungen aller verfügbaren Pfade. Bei zu komplexen Web-Anwendungen kann dieses Feature jedoch unübersichtlich werden [W3C, 2015].

2.5 Psychische Krankheiten

In den USA leidet etwa einer von vier Amerikanern über 18 Jahren an einer diagnostizierbaren psychischen Krankheit [JHM, 2019]. Im Web sind immer wieder Taktiken zu finden, die bewusst oder unbewusst verwendet werden, um Nutzer zu manipulieren. Solche sogenannten 'Dark Patterns' können die Symptome von Krankheiten wie Depressionen, Angststörungen, Psychosen, Phobien oder Posttraumatischer Belastungsstörung (PTBS) verschlimmern. Kurzfristig können sie Unternehmen zwar zu höherem Umsatz verhelfen, sind aber ethisch nicht vertretbar, weshalb sie sich negativ auf das Image auswirken sowie auf lange Sicht schaden können [Firth, 2019]. Nicht immer muss die Anwendung von Dark Patterns beabsichtigt sein. Web-Frameworks und andere Technologien können deren versehentliches Anwenden nicht verhindern. Stattdessen müssen Designer sich über deren Existenz bewusst sein.

Kontrolle und Sicherheit vermitteln

Das Dark Pattern 'Roach Motel' beschreibt Situationen, in die Nutzer zwar leicht gelangen, aber aus denen sie nur sehr schwer wieder herauskommen. Ein Beispiel hierfür ist das Abonnieren eines Service, bei dem die Option, zu kündigen, schwer zu finden ist. Generell sollten alle Optionen und Features leicht auffindbar sein. Zu tief verschachtelte Menüs sollten vermieden werden, Aktionen, die häufig ausgeführt werden, wie etwa das Anmelden, sollten so schnell und einfach wie möglich ablaufen. Damit werden Menschen mit Depressionen unterstützt, die eine geringe Motivation aufweisen und somit schnell aufgeben, ihr Ziel zu erreichen. Wer an Panikstörungen leidet, kann aufgrund auftretender Verwirrung in Panik verfallen und Schizophrenieerkrankte haben häufig Schwierigkeiten, die Beziehungen mehrerer Unterseiten zueinander zu verstehen. Auch Frequently Asked

KAPITEL 2. ARDEN VON BEEINTRÄCHTIGUNGEN UND ALLGEMEINE LÖSUNGANSÄTZE

Questions (FAQs) und Suchfunktionen sind wichtige Bestandteile einer Webseite, um eine möglichst entspannte Interaktion mit ihr zu ermöglichen.

Nutzer erhalten ein Gefühl der Kontrolle und der Sicherheit, wenn sie zu jedem Zeitpunkt wissen, welche Auswirkungen ihre Aktionen haben. Deshalb sollten Links und Buttons aussagekräftig beschriftet sein. Ein Button mit der Beschriftung „Weiter“ ist weniger aussagekräftig als einer auf dem „Weiter zur Bezahlung“ steht. Fortschrittsbalken mit bereits erledigten und noch ausstehenden Aktionen können zur Motivation beitragen und Nutzern mit eingeschränkten Gedächtnisleistungen beim Erinnern unterstützen [Firth, 2019].

Auslöser für Angst vermeiden

„Forced Urgency“ ist ein weiteres Beispiel für ein Dark Pattern. Dieses beschreibt Situationen, in denen ein falsches Gefühl von Dringlichkeit vermittelt wird. Aussagen wie „Der Preis ist in den letzten 3 Tagen gestiegen“ oder „20 weitere Personen schauen sich das Angebot gerade an“, kann dazu führen, dass Entscheidungen getroffen werden, die im Nachhinein bereut werden. Besonders anfällig für solche Taktiken sind Nutzer mit Psychosen, Paranoia, Angststörungen und anderen psychischen Krankheiten, die Angst verursachen. Um dem entgegenzuwirken, sollten Nutzer so viel Zeit wie möglich erhalten, um Entscheidungen zu treffen. Außerdem sollten so viele Entscheidungen wie möglich rückgängig gemacht werden können.

Nicht nur Hörgeschädigte Menschen profitieren davon, wenn nicht nur telefonisch Kontakt aufgenommen werden kann. Wer an einer Sozialphobie leidet, bevorzugt häufig die Kommunikation per E-Mail oder Webchat. Auch für Verifikationen, die normalerweise per Telefonat ablaufen, sollte es Alternativen geben [Firth, 2019].

Eine Vielzahl weiterer Dark Patterns kann im Web ausfindig gemacht werden. Zusammenfassend sollte bei der Entwicklung mit Rücksicht auf Menschen mit psychischen Erkrankungen immer bedacht werden, wie sich bestimmte Designentscheidungen auf die Emotionen, Kontrolle und Orientierung der Nutzer auswirken.

2.6 Web Content Accessibility Guidelines

Um eine Web-Anwendung auf den Grad der Barrierefreiheit hin zu überprüfen, wurden vom W3C 13 Richtlinien für barrierefreie Webinhalte, die WCAG, definiert (siehe Tabelle 1). Diese wurden in folgende vier Prinzipien unterteilt [W3C, 2018]:

- **Wahrnehmbar:** Informationen und UI-Komponenten müssen so präsentiert werden, dass sie von den Nutzern wahrgenommen werden können.
- **Bedienbar:** UI-Komponenten und Navigationen müssen bedienbar sein.
- **Verständlich:** Nutzer müssen die Informationen und das Interface verstehen können.
- **Robust:** Die Inhalte müssen zuverlässig von allen Benutzeragenten, einschließlich assistiver Technologien, interpretierbar sein.

Für jede Richtlinie gibt es mehrere testbare Erfolgskriterien mit unterschiedlichen Konformitätsstufen, die den Umfang der Barrierefreiheit festlegen. Stufe A stellt dabei die

KAPITEL 2. ARDEN VON BEEINTRÄCHTIGUNGEN UND ALLGEMEINE LÖSUNGANSÄTZE

Tabelle 1: Die 13 Richtlinien der WCAG 2.1 [W3C, 2018]

Prinzip	Richtlinie	Beschreibung
1. Wahrnehmbar	1.1 Textalternativen	Für alle Nichttext-Inhalte werden Textalternativen zur Verfügung gestellt.
	1.2 Zeitbasierte Medien	Alternativen für zeitbasierte Medien werden zur Verfügung gestellt.
	1.3 Anpassbar	Inhalte können auf verschiedene Arten dargestellt werden, ohne dass Informationen oder Struktur verloren gehen.
	1.4 Unterscheidbar	Nutzern wird es erleichtert, Inhalte sehen und hören zu können.
2. Bedienbar	2.1 Per Tastatur zugänglich	Alle Funktionalitäten sind per Tastatur zugänglich.
	2.2 Ausreichend Zeit	Nutzer haben genügend Zeit, Inhalte lesen und benutzen zu können.
	2.3 Anfälle	Inhalte sind nicht auf Arten, von denen bekannt ist, dass sie zu Anfällen führen, gestaltet.
	2.4 Navigierbar	Es werden Mittel zur Verfügung gestellt, um Nutzer dabei zu unterstützen, zu navigieren, Inhalte zu finden und zu bestimmen, wo sie sich befinden.
	2.5 Eingabe-Modalitäten	Nutzern wird es erleichtert, Funktionalitäten mit zahlreichen anderen Eingaben abseits der Tastatur zu verwenden.
3. Verständlich	3.1 Lesbar	Inhalte sind lesbar und verständlich.
	3.2 Vorhersehbar	Die Webseite funktioniert und sieht vorhersehbar aus.
	3.3 Hilfestellung bei der Eingabe	Nutzern wird dabei geholfen, Fehler zu vermeiden und zu korrigieren.
4. Robust	4.1 Kompatibel	Die Kompatibilität mit aktuellen und zukünftigen Benutzeragenten, einschließlich assistiver Technologien, wurde maximiert.

geringste, AA die mittlere und AAA die höchste Stufe dar.

Damit beispielsweise der Kontrast AA-konform ist, wird ein minimales Kontrastverhältnis von 4,5:1 benötigt. Für die AAA-Konformität liegt dieser Wert bei 7:1. Eine Ausnahme hierbei stellt großer Text (mindestens 18 pt oder 14 pt in Fett) dar, für den geringere Werte ausreichend sind.

Die aktuelle Version ist WCAG 2.1. Jede Version ist abwärtskompatibel zur vorherigen [W3C, 2018]. Die Richtlinien werden in Kapitel 5, bei der Evaluierung, Anwendung finden.

Kapitel 3

Barrierefreiheit in modernen Web-Frameworks

Frameworks sind ein wesentlicher Bestandteil der modernen Web-Entwicklung. Sie sind in einer großen Menge vorhanden, für die unterschiedlichsten Zwecke und in zahlreichen Programmiersprachen geschrieben. Ziel dieses Kapitels ist es, für drei verschiedene zu zeigen, inwiefern sie Entwickler bei der Umsetzung von Barrierefreiheit unterstützen können. Bei diesen drei gewählten Frameworks handelt es sich um Boostrap, ein CSS-Framework, Angular, ein Framework für die Entwicklung von SPA, und Django, ein in Python geschriebenes Backend-Framework. Außerdem wird es zu jedem einen kurzen Vergleich mit mindestens einem weiteren Framework der gleichen Art geben.

3.1 Bootstrap: Ein CSS-Framework

Bootstrap wurde 2011 von Twitter entwickelt und gehört heute mit zu den beliebtesten GitHub-Repositories. Es handelt sich dabei um ein CSS-Framework, mit Fokus auf responsivem Design und einem 'Mobile-First'-Ansatz. Die aktuelle und hier betrachtete Version ist Bootstrap 4 [Bootstrap, 2020a]. Da die Interaktion der Nutzer mit der Webseite überwiegend über die UI-Komponenten stattfindet, haben CSS-Frameworks, bei den in dieser Arbeit betrachteten, mit den größten Einfluss auf die Barrierefreiheit einer Web-Anwendung. Die Anwendung von Bootstrap findet zu einem großen Teil über vordefinierte Klassen statt, es wird aber auch eine Schnittstelle in JavaScript definiert. Die zwei wesentlichen Bestandteile sind das Grid-System und die bereitgestellten UI-Komponenten, wie Buttons, Fortschrittsbalken und Dialogfenster ('Modals'). Beides wird in diesem Kapitel, neben einigen weiteren Bestandteilen, näher betrachtet.

Bootstrap behauptet von sich selbst, barrierefrei 'out-of-the-box' zu sein, gesteht aber auch Defizite bei der Farbpalette ein [Bootstrap, 2020b]. Tatsächlich kann die Verwendung von Bootstrap zu einem barrierefreieren Ergebnis führen. Trotzdem sind normalerweise weitere Anpassungen vorzunehmen.

3.1.1 Grid-System

Das bereitgestellte Grid-System untergliedert das Layout der Anwendung in Zeilen und Spalten. So kann abhängig von der Displaygröße festgelegt werden, wie die einzelnen Be-

<pre><div class="container"> <div class="row"> <div class="col"> Position im DOM: 1
 Visuelle Position: 1 </div> <div class="col order-12"> Position im DOM: 2
 Visuelle Position: 3 </div> <div class="col order-1"> Position im DOM: 3
 Visuelle Position: 2 </div> </div> </div></pre>	Position im DOM: 1 Visuelle Position: 1	Position im DOM: 3 Visuelle Position: 2	Position im DOM: 2 Visuelle Position: 3
--	--	--	--

Abbildung 2: Auswirkungen der .order-Klasse

standteile angeordnet werden sollen [Bootstrap, 2020c]. Dadurch wird ein responsives Design ermöglicht, das nicht nur dazu beiträgt, die Anwendung auf unterschiedlichen Geräten anwendbar zu machen, sondern auch die Zoom-Funktionalität, auf die viele Nutzer mit eingeschränkter Sehfähigkeit angewiesen sind, effektiver zu unterstützen.

Außerdem gibt es die Option, die Reihenfolge, in der Spalten angezeigt werden, mithilfe der .order-Klasse zu verändern. Deren Verwendung sollte allerdings möglichst vermieden werden, denn dadurch ergibt sich eine Diskrepanz zwischen der visuellen Anordnung im Browser und der logischen im DOM (siehe Abbildung 2). Die Reihenfolge, in der die Elemente fokussiert werden, bleibt somit bestehen und wird kontraintuitiv. Auch die Reihenfolge, in der Screenreader die Inhalte präsentieren, wird durch diese Klasse nicht verändert. Ihre Anwendung ist deshalb höchstens dann sinnvoll, wenn die Reihenfolge aus ästhetischen Gründen verändert wird und die Inhalte nicht fokusierbar sind. Wird sie verwendet, um die Reihenfolge unabhängig vom Gerät und der Displaygröße zu verändern, sollte stattdessen die Anordnung im DOM angepasst werden.

3.1.2 Klassen für Screenreader

Bootstrap stellt zwei Klassen zur Verfügung, die speziell für Screenreader entwickelt wurden sind: .sr-only und .sr-only-focusable. Elemente, die zu Ersterer gehören, können nur mit Screenreadern wahrgenommen werden. Die zweite unterscheidet sich dahingehend von dieser, dass zu ihr gehörende Elemente fokusierbar sind und visuell sichtbar werden, sobald sie den Fokus erhalten. Die beiden Klassen können verwendet werden, um visuelle Inhalte für Screenreader-Nutzer zu beschreiben oder um Skip-Links umzusetzen [Bootstrap, 2020b]. Abbildung 3 zeigt dies anhand von drei Beispielen: Die 'Spinner'-Komponente dient dazu, den Ladeprozess zu visualisieren. Durch die .sr-only-Klasse kann diese Information in einer für assistive Technologien zugänglichen Form hinzugefügt wer-

```
<!-- Skip-Link -->
<a href="#content" class="sr-only sr-only-focusable">Skip to Main Content</a>

<!-- Badge -->
<button type="button" class="btn btn-primary">
  Profile <span class="badge badge-light">9</span>
  <span class="sr-only">unread messages</span>
</button>

<!-- Spinner -->
<div class="text-primary spinner-border" role="status">
  <span class="sr-only">Loading...</span>
</div>
```

Abbildung 3: .sr-only-Klassen für Skip-Links, Badges und Spinner

den. Die 'status'-Rolle wird zusätzlich verwendet, um diese Information automatisch ohne vorherige Fokussierung an Nutzer von Screenreadern zu übergeben. Wie zu sehen ist, kann auch die 'Badge'-Komponente von der .sr-only-Klasse Gebrauch machen, um zusätzliche Kontextinformationen bereitzustellen.

3.1.3 Animationen und Farben

Animationen können bei Menschen mit vestibulären Störungen und Epilepsie Symptome, wie Übelkeit, Schwindel und epileptische Anfälle hervorrufen. Generell sollte möglichst auf diese verzichtet und im Falle der Verwendung zumindest darauf geachtet werden, dass sie nicht zu schnell ablaufen oder zu häufig auftreten [WebAIM, 2017c]. Über die Media-Query 'prefers-reduced-motion' lassen sich in CSS Geräte adressieren, auf deren Betriebssystem Animationen deaktiviert wurden. Bootstrap nutzt diese Funktion automatisch, um standardmäßig die Animationen der Komponenten auf solchen Geräten zu deaktivieren [Bootstrap, 2020b]. Betroffen davon sind unter anderem das 'Carousel' und das 'Accordion'. Die 'Spinner'- und 'Progress Bar'-Komponenten sind Ausnahmen, auf die diese Option keinen Einfluss hat.

Wie in Kapitel 2.1 beschrieben, ist bei Farben auf Nutzer mit Farbenblindheit und auf angemessene Kontraste zu achten. Die meisten in Bootstrap verfügbaren Komponenten gibt es in mehreren vordefinierten Farbschemata. Auch für Texte und Hintergründe allgemein können Bootstrap-Klassen verwendet werden. Bei Farben, die semantische Informationen vermitteln (z.B. Rot als Warnung), muss darauf geachtet werden, dass die vorhandenen Texte aussagekräftig genug sind, um die Information auch an Nutzer von Screenreadern zu vermitteln [Bootstrap, 2020d]. Alternativ kann hier auch die .sr-only-Klasse Verwendung finden, jedoch hilft diese nicht bei Farbenblindheit.

Hinsichtlich des Kontrastes erfüllen viele Farben der Bootstrap-Palette die AA-Kriterien der WCAG. AAA-Konformität wird oft nur dann erreicht, wenn für die Schriftgröße mindestens 18 pt verwendet werden.

Die 'Alert'-Komponente gibt es beispielsweise in sieben verschiedenen Farben. Vier da-

von sind vollständig AAA-konform. Zwei (.alert-success und .alert-info) sind AA-konform, aber nur bei ausreichend großer Schriftgröße AAA-konform. Auch die .alert-light-Klasse ist nur bei großer Schrift AA-konform.

Ein besonderes Problem besteht bei der .btn-outline-* -Klasse, bei der zwei Farbschemata nicht und zwei weitere nur beschränkt AA-konform sind.

Zum Zeitpunkt der Erstellung dieser Arbeit befindet sich Bootstrap 5 in einem fortgeschrittenen Entwicklungsstadium. Dort soll es durch eine erweiterte Farbpalette ermöglicht werden, hohe Kontraste leichter zu erreichen [Bootstrap Blog, 2020].

Nutzer mit Fotophobie leiden an einer Überempfindlichkeit des Auges gegenüber Licht. Ein Dunkelmodus kann mögliche Symptome mindern, sollte aber optional bleiben, da es hierbei umgekehrt bei Astigmatismus zu einem sogenannten 'Haloeffekt' kommen kann [Locke, 2020].

Die Umsetzung kann in Bootstrap über das Ändern der Farben, die mit den jeweiligen Klassen assoziiert werden, oder durch das Einfügen eines der zahlreichen, im Internet existierenden Bootstrap-Themes geschehen.

3.1.4 Komponenten

Die verfügbaren Komponenten wurden so entworfen, dass sie vollständig mit der Tastatur bedienbar sind [Bootstrap, 2020b]. Bei der Verwendung müssen entsprechende ARIA-Attribute manuell hinzugefügt werden. Im Gegensatz zu anderen Frameworks werden diese nämlich nur in wenigen Ausnahmefällen automatisch von Bootstrap generiert. Viele der Komponenten können dazu verwendet werden, Nutzern mit kognitiven- und psychischen Störungen zu unterstützen, indem die Orientierungsmöglichkeiten verbessert werden und ein Gefühl der Sicherheit vermittelt wird. Welche Komponenten das sind und auf welche konkreten Besonderheiten zu achten ist, wird im Folgenden genauer betrachtet.

Komponenten zur Strukturierung der Inhalte

Die Komponenten 'Card', 'Media Object', 'Jumbotron' und 'Button Group' symbolisieren eine visuelle Zusammengehörigkeit der darin gekapselten Elemente. Unabhängig vom verwendeten Gerät und der dargestellten Zoom-Stufe bleiben sie somit in einer räumlichen Nähe. Im Gegensatz zum 'Media Object', das speziell mit Medieninhalten verwendet werden sollte, ist eine 'Card' allgemeiner anwendbar. 'Jumbotrons' heben wesentliche Inhalte hervor und eine 'Button Group' verknüpft zusammengehörige Buttons. Vor allem bei Letzterem ist darauf zu achten, passende semantische HTML-Elemente (hier: <button>) einzusetzen, auch wenn die Komponenten durch die Verwendung der Bootstrap-Klassen theoretisch auf jedes Element anwendbar sind. Jede der vier genannten Komponenten sollte zusätzlich, abhängig vom Anwendungsfall, durch ARIA-Rollen und -Attribute beschrieben werden.

Durch die Verwendung der Komponenten 'Dropdown' und 'Collapse', durch die auch Accordions¹ umgesetzt werden können, werden vor allem Nutzer mit kognitiven Beeinträchtigungen unterstützt, da sich so die Menge der gleichzeitig sichtbaren Inhalte reduzieren lässt. Beispielsweise können dadurch für Navigationselemente zuerst nur all-

¹ UI-Komponente, die Inhalte in einem oder mehreren ein- und ausklappbaren Bereichen darstellt.

gemeine Menüpunkte angezeigt werden, die per Klick auf eine der Optionen durch ein Dropdown-Menü mit konkreteren Unterpunkten erweitert werden. Die Aktualisierung des 'aria-expanded'-Attributs muss manuell hinzugefügt werden, wird dann aber automatisch von Bootstrap aktualisiert.

Komponenten zur Vermittlung von Kontrolle und Sicherheit

Die Rolle von Breadcrumbs wurde bereits in Kapitel 2.4 angesprochen. In Bootstrap gibt es hierfür eine eigene Komponente. Die aktive Seite sollte durch das Attribut 'aria-current=„page“' gekennzeichnet werden und mit der .active-Klasse visuell erkennbar gemacht werden [Bootstrap, 2020e]. Die Verwendung Letzterer führt jedoch zu einem Kontrast, der die AA-Kriterien der WCAG erst ab einer Größe von 18 pt erfüllt und nicht AAA-konform ist. Trennzeichen zwischen den Links fügt Bootstrap via CSS hinzu, wodurch verhindert wird, dass diese von Screenreadern mit vorgelesen werden.

Eine ähnliche Funktion wie Breadcrumbs erfüllt die 'Scrollspy'-Komponente. Diese aktualisiert ein Navigationsmenü automatisch, während Nutzer durch die dazugehörigen Inhalte navigieren. Somit kann dieses als Orientierungspunkt verwendet werden. Nicht optimal ist dabei die Navigation mit der Tastatur. Sobald einer der Links in der Navigation aktiviert wird, springt der Fokus zum entsprechenden Inhalt. Um anschließend wieder zurück zur Navigation zu gelangen, muss der gesamte Inhalt rückwärts durchlaufen werden. Ein Vorschlag zur Verbesserung ist die Verwendung von Skip-Links, die zurück auf die Navigation verweisen.

Bei längeren Ladezeiten können die Komponenten 'Progress' und 'Spinner' verwendet werden, um den Ladeprozess zu visualisieren und zusätzliche Sicherheit zu vermitteln. Erstere ist ein Fortschrittsbalken, der auch dafür Anwendung finden kann, den aktuellen Progress einer mehrschrittigen Aktion zu repräsentieren.

An dieser Stelle soll auch die 'Tooltip'-Komponente erwähnt werden. Durch diese können komplexe Elemente um weitere Informationen, wie etwa eine genauere Beschreibung der Funktion, ergänzt werden. Tooltips werden sowohl angezeigt, wenn sich der Mauszeiger über dem entsprechenden Element befindet, als auch wenn es fokussiert ist. Damit Nutzer ohne Maus auf alle Tooltips zugreifen können, dürfen sie nur mit fokussierbaren Elementen verwendet werden. Die 'WAI-ARIA Authoring Practices' empfehlen das Schließen von Tooltips durch das Drücken der 'Esc'-Taste [W3C, 2019]. Diese Funktion ist nicht von vornherein verfügbar, kann aber durch die JavaScript-Schnittstelle selbst implementiert werden. Screenreader kündigen Tooltips automatisch an, ohne dass weitere ARIA-Attribute notwendig sind. Soll ein Tooltip nicht von selbst, sondern erst nach der Aktivierung durch 'Enter', die Leertaste oder durch Klicken umgeschaltet werden, kann die ansonsten größtenteils identische Komponente 'Popover' verwendet werden.

Besonderheiten weiterer Komponenten

Das Karussell in Bootstrap erfüllt einen Großteil, aber nicht alle der Anforderungen der 'WAI-ARIA Authoring Practices' [W3C, 2019]. Falls das automatische Rotieren der Inhalte aktiviert ist stoppt dieses, sobald sich der Mauszeiger über dem Karussell befindet, aber nicht wenn der Inhalt oder die Kontrollelemente fokussiert werden. Letztere zum Wechseln

des aktuell angezeigten Elements sind optional, für einen barrierefreien Webauftritt aber wesentlich. Auch eine Funktion zum Pausieren der Rotation wäre notwendig, doch hierfür bietet Bootstrap keine Lösung, weshalb diese mithilfe der JavaScript-Schnittstelle selbst implementiert werden muss. Zur Orientierung können Indikatoren, die die Position und die Anzahl der Seiten im Karussell anzeigen, verwendet werden.

Für Nutzer mit visuellen Beeinträchtigungen muss zudem auf den Kontrast der Interaktionselemente geachtet werden. Empfehlenswert ist es, diese mit einfarbigen Hintergründen zu hinterlegen, da sich bei jeder Rotation die Hintergrundfarbe ändern kann.

Die 'Toast'- und 'Alert'-Komponente zeigen Inhalte an, die zumeist an einer anderen Stelle des Displays auftauchen, ohne fokussiert zu werden. Vor allem Nutzer mit kognitiven Beeinträchtigungen, zum Beispiel Leseschwächen, benötigen ausreichend Zeit, um die Information verarbeiten zu können, bevor sie wieder verschwindet. Diejenigen mit Gedächtnisproblemen profitieren davon, wenn sie so lange bestehen bleiben, bis sie manuell wieder geschlossen werden oder die Information veraltet ist. Außerdem müssen auch Nutzer von Screenreadern informiert werden. Konkret bedeutet dies die Anwendung von ARIA-Live oder der 'alert'-Rolle. Da zu viele Meldungen auf einmal überfordern können, dürfen diese Komponenten nicht exzessiv verwendet werden.

Die 'Modal'-Komponente erfüllt alle Anforderungen an die Bedienbarkeit durch die Tastatur. Beim Öffnen werden dahinterliegende Elemente für den Fokus deaktiviert. Nachdem der Fokus am untersten Element angekommen ist, springt er wieder zum obersten. Die 'Esc'-Taste kann zum Schließen verwendet werden. Grundsätzlich muss jedem Modal die Rolle 'dialog' zugewiesen werden. Diese Aufgabe wird von Bootstrap automatisch übernommen.

3.1.5 Vergleich mit weiteren CSS-Frameworks: Foundation und PUXL-Framework

Hinsichtlich der Barrierefreiheit bieten die meisten der verbreiteten CSS-Frameworks vergleichbare Vor- und Nachteile wie Bootstrap. Dazu gehört auch 'Foundation'. Auch hier liegt der Fokus auf Responsiveness, ebenfalls durch ein Grid-System umgesetzt, und der Bereitstellung zahlreicher UI-Komponenten, die die Verwendung ausschließlich mit Maus, Tastatur oder Touchscreen erlauben. Zusätzlich wurden diese für Screenreader optimiert, sodass häufig bereits die wichtigsten ARIA-Attribute automatisch generiert werden. Außerdem gibt es auch hier Klassen, um Inhalte nur Screenreadern zugänglich zu machen, und auch eine, um Inhalte vor ihnen zu verstecken. Eine Option um Animationen zu deaktivieren existiert ebenfalls [Foundation, o.D.].

Ein CSS-Framework, das heraussticht, ist das sich noch in der Beta befindende PUXL-Framework [PUXL, o.D.]. Dieses wurde explizit mit dem Ziel der Unterstützung von Barrierefreiheit entwickelt und bietet hierbei große Hilfsmöglichkeiten an:

Nach jeder Änderung sucht es die Anwendung automatisch nach Mängeln ab, die anschließend im Browserfenster hervorgehoben werden. Vorschläge zu deren Behebung sind dabei inklusive. Zu den auffindbaren Mängeln gehören unter anderem fehlende Attribute und zu geringe Kontraste (siehe Abbildung 4).

Das Framework wird zudem mit der Schriftart 'OpenDyslexic' ausgeliefert, die sich beson-

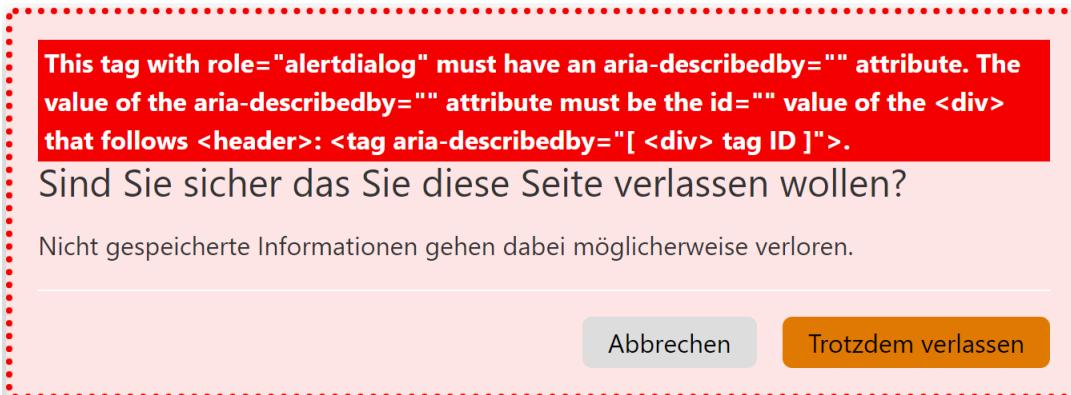


Abbildung 4: Hinweise auf Mängel inkl. Verbesserungsvorschläge im PUXL-Framework

ders gut für Menschen mit Lesestörungen eignet.

Die enthaltenen UI-Komponenten wurden mit dem Ziel, WCAG-konform zu sein, entwickelt, sodass diese vollständig mit der Tastatur bedienbar sind. Fokussierte Elemente werden durch eine gut sichtbare, leuchtende Umrandung hervorgehoben.

Animationen lassen sich zwar deaktivieren, eine automatische Adaption an die Einstellungen des verwendeten Gerätes ist im Gegensatz zu Bootstrap aber nicht implementiert.

3.2 Angular: Ein SPA-Framework

Bei Angular handelt es sich um ein von Google entwickeltes Open-Source Framework für die Entwicklung von SPA [Angular, o.D.a]. Das dynamische Austauschen bestimmter Inhalte, die Wiederverwendung mehrmals auftretender Komponenten und das Routing sind die typischen Merkmale dieser Art von Framework. Ein Abschnitt des Kapitels wird der UI-Bibliothek 'Angular Material' gewidmet, die einerseits die Arbeit mit ARIA erleichtert und zudem ein umfangreiches Paket mit Features für die Barrierefreiheit bereitstellt [Angular Material, o.D.a].

3.2.1 Komponentensystem, Data-Binding und Routing

Die bedeutendste Besonderheit bei SPA ist, dass bei Anfragen an den Server nicht die ganze Seite neu geladen wird, sondern nur relevante Komponenten dynamisch ausgetauscht werden. Beziiglich der Barrierefreiheit gibt es hierbei mehrere Aspekte zu beachten.

Erstens sorgt dies für einen gewissen Grad an Konsistenz, da statische Bestandteile erhalten und an derselben Position bleiben. Dazu gehört beispielsweise das Navigationsmenü. Auch die Wiederverwendbarkeit der Komponenten trägt zur Konsistenz bei.

Außerdem muss, vor allem für Nutzer von Screenreadern, darauf geachtet werden, dass der Austausch der Komponenten nicht zu Orientierungsverlusten führt. Angenommen der Nutzer eines Screenreaders navigiert von einem Abschnitt zum nächsten und möchte anschließend wieder zurück zum vorherigen: Befindet sich an derselben Stelle nun ein anderer Inhalt, kann dies für Verwirrung sorgen. In solchen Fällen bietet sich deshalb die Anwendung von ARIA-Live an. Ähnliches gilt für sich ändernde Inhalte durch das Verwenden des 'Data-Bindings', das Daten in HTML-Templates an solche im Model bindet.

```
<a class="skip-link" href="#mainComponent">Skip to Main Content</a>
<NavComponent role="navigation"></NavComponent>

<a class="skip-link" href="#footer">Skip to Footer</a>
<MainComponent role="main" id="mainComponent"></MainComponent>

<VisualComponent aria-hidden="true"></VisualComponent>
```

Abbildung 5: Heuristiken zur Verwendung der Angular Komponenten in Kombination mit ARIA und Skip-Links

Für die Kombination der Komponenten können verschiedene Heuristiken angewendet werden: Große, für einen bestimmten Inhalt erstellte Komponenten sind für gewöhnlich in sich abgeschlossen und sollten somit durch eine ARIA-Rolle oder entsprechende HTML-Elemente beschrieben werden. Vor solchen ist meist das Platzieren eines Skip-Links sinnvoll. Enthält eine Komponente ausschließlich visuelle Inhalte ohne semantische Informationen, so kann diese durch ein ARIA-Hidden-Attribut vor Nutzern assistiver Technologien versteckt werden. Abbildung 5 veranschaulicht diese drei Möglichkeiten anhand eines Beispiels.

Auch das Routing, wodurch Bestandteile der Uniform Resource Locator (URL) an bestimmte Komponenten gebunden werden, ist von Relevanz für die Barrierefreiheit. Wird dieses System genutzt, kann das zu übersichtlicheren URLs führen, die eine vergleichbare Rolle spielen wie Breadcrumbs. Außerdem kann so die 'Zurück'-Funktion effektiv unterstützt werden. Besonders ärgerlich ist es für Nutzer mit motorischen Behinderungen, wenn durch das Zurückgehen auf die vorherige Seite unerwarteterweise die vorherige Anwendung geladen wird und somit noch benötigte Eingaben verloren gehen, die anschließend erneut eingetippt werden müssen. Durch die Möglichkeit, auch kleine Schritte rückgängig machen zu können, wird das Gefühl der Sicherheit und Kontrolle gestärkt.

Nach der Navigation zu einer neuen Route muss für gewöhnlich Fokus-Management betrieben werden. Hilfreich dafür ist das 'NavigationEnd'-Event vom 'Router'-Service, das nach Beendigung der Navigation ausgelöst wird [Angular, o.D.b].

Da der Titel das erste ist, was Screenreader beim Betreten einer Seite vorlesen, sollte dieser für jede Unterseite aussagekräftig sein. In Angular kann er mithilfe des 'Title'-Service jederzeit aktualisiert werden.

3.2.2 Codelyzer

Codelyzer ist ein Werkzeug zur Überprüfung des Codes auf Best-Practice-Regeln. Beim Erstellen eines neuen Angular-Projekts mithilfe des Command Line Interfaces (CLI) von Angular wird Codelyzer diesem automatisch hinzugefügt. Zehn der vorhandenen Regeln sind der Barrierefreiheit gewidmet. Sie sind aber noch experimentell und standardmäßig deaktiviert. Um sie zu aktivieren, müssen sie in der 'tslint.json'-Datei hinzugefügt werden. Unter anderem können so ungültige ARIA-Attribute und fehlende Tastatur-Interaktionsmöglichkeiten ausfindig gemacht werden [Mohammed et al., 2019]. Tabelle 2 enthält eine vollstän-

Tabelle 2: Codelyzer-Regeln zur Überprüfung der Barrierefreiheit [Codelyzer, 2017]

Regel	Beschreibung
template-accessibility-label-for	'Input'-Elemente müssen durch ein Label beschrieben werden.
template-accessibility-alt-text	Überprüft die Verwendung von Alternativtexten für visuelle Inhalte.
template-click-events-have-key-events	'Click'-Events müssen von 'Key'-Events begleitet werden.
template-mouse-events-have-key-events	'Mouseover'- und 'Mouseout'-Events müssen von 'Focus'- und 'Blur'-Events begleitet werden.
template-accessibility-element-content	Bestimmte Elemente (z.B. Buttons) dürfen nicht leer sein, um von Screenreadern sinnvoll genutzt werden zu können.
template-accessibility-tabindex-no-positive	Das Vertauschen der Tab-Reihenfolge durch 'tabindex' sollte vermieden werden.
template-accessibility-table-scope	Überprüft die Korrektheit der Verwendung von 'scope'-Attributen bei Tabellen.
template-accessibility-valid-aria	Findet ungültige ARIA-Attribute.
template-no-autofocus	Autofocus reduziert die Barrierefreiheit der Anwendung.
template-no-distracting-elements	Visuell ablenkende Elemente wie <marquee> oder <blink> sollten vermieden werden.

dige Auflistung dieser Regeln.

3.2.3 Angular Material

Angular Material ist eine UI-Bibliothek, die wie Bootstrap eine umfangreiche Palette grafischer Komponenten bereitstellt. Bei der Entwicklung wurde verstärkt auf die Barrierefreiheit geachtet. Dies ist sowohl an der Umsetzung der Komponenten zu sehen als auch daran, dass es ein zusätzliches Paket gibt, das ausschließlich der Barrierefreiheit gewidmet wurde.

UI-Komponenten

Ähnlich wie bei den Bootstrap-Komponenten, wurde auch bei denen aus Angular Material groß Wert auf die Interaktionsmöglichkeiten mit der Tastatur gelegt. Die Arten verfügbarer Komponenten unterscheiden sich jedoch. Manche Positivbeispiele aus Kapitel 3.1.4, etwa die Breadcrumbs, gibt es in Angular Material nicht. Jedoch existieren andere Komponenten, die in Bootstrap nicht vorhanden sind. Ein Beispiel ist der sogenannte 'Stepper', der dabei hilft, umfangreiche Aktionen in mehrere Schritte zu untergliedern und somit übersichtlicher zu gestalten. Zudem existiert auch eine Komponente, mit der sich 'Autocomplete'-Funktionalitäten umsetzen lassen, inklusive der Möglichkeit, Vorschläge zu gruppieren.

Bei der Verwendung des 'Drag and Drop'-Pakets fällt hingegen auf, dass dessen Funktionalität nicht mit der Tastatur bedienbar ist.

Ein großer Vorteil der Angular-Material-Komponenten, ist die automatische Generie-

rung allgemeiner ARIA-Attribute. Für konkrete Fälle ist es jedoch notwendig, selbst Änderungen vorzunehmen. Hinweise darauf, in welchen Fällen dies nötig ist, sind umfangreich in der Dokumentation gegeben [Angular Material, o.D.a].

Anhand der 'Dialog'-Komponente, die zur Implementierung von Modal-Dialogen zur Verfügung steht, können die beschriebenen Aspekte konkretisiert dargestellt werden:

Beim Öffnen wird automatisch das erste interaktive und beim Verlassen das ursprüngliche Element fokussiert. Der Fokus wird im Dialog 'gefangen' und kann ihn somit nicht verlassen, um dahinter liegende Elemente anzuvisieren. Geschlossen werden kann der Dialog über die 'Esc'-Taste.

Die Rolle 'dialog' wird standardmäßig hinzugefügt, kann aber auch zu 'alertdialog' geändert werden. ARIA-Label sind hier abhängig vom konkreten Anwendungsfall und müssen dementsprechend selbstständig hinzugefügt werden.

A11y Package

Das 'A11y'-Paket in Angular Material stellt folgende sechs Werkzeuge zur Verfügung, die hauptsächlich dabei helfen sollen, Barrierefreiheit in selbst erstellen UI-Komponenten umzusetzen [Angular Material, o.D.b]:

- **ListKeyManager:** Um den Fokus zwischen Komponenten zu wechseln ist die 'Tab'-Taste der Standard. Doch innerhalb der jeweiligen Komponenten werden häufig Alternativen bevorzugt, zum Beispiel das Wechseln der Einträge in einem Dropdown-Menü durch die Pfeiltasten. Die 'ListKeyManager'-Klasse hilft bei der Implementierung der Tastatur-Interaktion von Listen. Dies geschieht durch die Verwendung eines Callbacks zur Auswahl des nächsten Elements, der nach bestimmten Input-Events ausgeführt wird.
- **FocusTrap:** Diese Direktive ermöglicht es, das Wechseln des Fokus auf einen bestimmten Abschnitt zu begrenzen. Dies wird hauptsächlich für die Implementierung von Overlays benötigt. Aktivieren und deaktivieren lässt sie sich mithilfe der 'enabled'-Eigenschaft. Außerdem kann mit der 'cdkTrapFocusAutoCapture'-Eigenschaft festgelegt werden, ob der Fokus beim Aktivieren automatisch in den angezielten Bereich wechselt und beim Deaktivieren wieder das ursprüngliche Element anspringen soll.
- **InteractivityChecker:** Der 'InteractivityChecker' ist ein Service, der vier Funktionen bereitstellt, die zur Überprüfung der Interaktivität übergebener Elemente verwendet werden können. Konkret lässt sich abfragen, ob sie fokussierbar, sichtbar, deaktiviert oder mit der 'Tab'-Taste anspringbar sind.
- **LiveAnnouncer:** Dieser Service bietet eine Schnittstelle zu ARIA-Live an, um Benachrichtigungen an Nutzer von Screenreadern zu übermitteln. Entsprechende Parameter erlauben es, Konfigurationen durchzuführen.
- **FocusMonitor:** Mithilfe des 'FocusMonitor'-Service ist es möglich, auf die Fokussierung von Elementen zu reagieren, abhängig davon, ob diese durch eine Maus, Tastatur, per Touchscreen oder programmatisch ausgelöst wurde. Ebenfalls abhängig von dem Eingabegerät, wird dem fokussierten Element die Klasse .cdk-[origin]-focused

zugeordnet. Somit sind für die unterschiedlichen Fälle verschiedene CSS-Styles zuordenbar.

- **Styling Utilities:** Parallel zur .sr-only-Klasse in Bootstrap, können mithilfe der .cdk-visually-hidden-Klasse in Angular Material Elemente realisiert werden, die ausschließlich von Screenreadern und vergleichbaren assistiven Technologien wahrgenommen werden können. Eine Alternative zur .sr-only-focusable-Klasse aus Bootstrap, die solche Elemente fokussierbar und im fokussierten Zustand sichtbar macht, gibt es hier jedoch nicht. Durch die Kombination mit den 'focus'- und 'focusout'-Events kann dies aber selbst umgesetzt werden.
Außerdem existiert ein 'cdk-high-contrast'-Mixin, mit dem in Stylesheets Browser mit aktiven 'High-Contrast'-Modus adressiert werden können. Unterstützt wird das vom Internet Explorer, Microsoft Edge und Firefox. Google Chrome verfügt über einen separaten Kontrast-Modus, der nicht mit diesem Mixin kompatibel ist.

3.2.4 Animationen

Angular besitzt ein Animations-System, das auf CSS aufbaut. Animationen haben den Vorteil Änderungen deutlicher zu propagieren, visuelles Nutzerfeedback bereitzustellen und die Aufmerksamkeit des Nutzers zu lenken. Doch wie bereits in vorherigen Kapiteln beschrieben, können sie auch zu Problemen bei bestimmten Erkrankungen führen. Animationen können durch das @.disabled-Binding lokal für bestimmte Komponenten oder global für die gesamte Anwendung deaktiviert werden. Somit lässt sich eine Option implementieren, die es Nutzern erlaubt, Animationen zu deaktivieren. Da auch Angular Material dieses Animations-System verwendet, ist diese Option auch mit deren Komponenten kompatibel. Es gibt jedoch Ausnahmen: So wird die Animation der 'Spinner'-Komponente nicht durch das Binding deaktiviert [Angular, o.D.c].

3.2.5 Vergleich mit einem verwandten Framework: Vue.js

Vergleichbare Frameworks verfügen über ähnliche Aspekte hinsichtlich der Barrierefreiheit. Die Modularität und das Template-System sind dabei die grundlegenden Konzepte von SPA-Frameworks.

Vue.js ist ein weiterer, beliebter Vertreter dieser Art von Framework und bietet in vielerlei Hinsicht die gleichen Chancen und Gefahren wie Angular [Vue.js, o.D.]. Werden Komponenten ausgetauscht, muss auch hier darauf geachtet werden, dass die Änderungen von allen Nutzern nachvollziehbar sind und der Fokus aktualisiert wird. Außerdem wird durch die Wiederverwendbarkeit der Komponenten die Konsistenz gesteigert. Mit 'Vuetify' ist für Vue.js ebenfalls eine Bibliothek vorhanden, die UI-Komponenten mit Material-Design und großem Fokus auf Barrierefreiheit bereitstellt. Auch hier werden ARIA-Attribute automatisch generiert und die Bedienbarkeit ohne Maus unterstützt. Das A11y-Paket in Angular Material ist allerdings recht einzigartig, denn Vue.js ist nicht die einzige Alternative, in der etwas Vergleichbares nicht zu finden ist.

3.3 Django: Ein Python-Framework

Django ist ein im Backend laufendes Python-Framework. Es bietet zwar keine expliziten Funktionen für die Umsetzung der Barrierefreiheit, doch implizit sind einige der vorhandenen trotzdem dafür von Relevanz. Darunter befinden sich das Template-System, das eingebaute Admin-Interface, die Generierung von Formularen aus Python-Klassen und die Suche in der Datenbank durch eine bereitgestellte Schnittstelle. Nachfolgend werden diese Aspekte genauer analysiert.

3.3.1 Template-System

Das Template-System in Django ist in vielerlei Hinsicht vergleichbar mit dem Komponentensystem in Angular und bietet somit auch ähnliche Vorteile. Templates können in anderen Templates inkludiert werden oder diese durch Vererbung erweitern. Somit wird auch hier die Konsistenz gefördert [Django, 2020a]. Genutzt werden kann dieses System außerdem für die Umsetzung von Breadcrumbs, indem die sich in der Template-Hierarchie weiter unten befindenden Templates die auf höheren Ebenen mit Links der aktuellen Position erweitern. In Kapitel 4 wird dies an einem Praxisbeispiel verdeutlicht.

Dadurch, dass die Anwendung mit jedem Aufruf an das Backend neu geladen wird und die einzelnen Bestandteile nicht dynamisch ausgetauscht werden, wird der Fokus automatisch auf den Anfang zurückgesetzt. Um Nutzern, die auf den Fokus angewiesen sind die Navigation zu erleichtern, muss dieser von den Entwicklern programmatisch neu gesetzt werden.

3.3.2 Verbesserte Lesbarkeit durch Filter

Um Daten in Template-Dateien so zu formatieren, dass sie leichter zu lesen und zu verstehen sind, können vordefinierte Filter verwendet werden. So lassen sich unter anderem Datum und Zeit an unterschiedliche Situationen anpassen, auch an das System des Standorts der Nutzer. Weitere Filter, die speziell dafür vorhanden sind, den Daten eine 'menschliche Note' zu verleihen, sind im 'Humanize'-Paket zu finden [Django, 2020b]. Diese können die Anforderungen an die kognitiven Fähigkeiten der Nutzer reduzieren. Große Zahlen können damit zum Beispiel durch Punkte nach jeder dritten Ziffer oder durch alternative Text-Repräsentationen lesbarer gemacht werden. Eine vollständige Auflistung dieser Filter inklusive Beispiele kann Tabelle 3 entnommen werden.

Die Verfügbarkeit einer Anwendung in mehreren Sprachen reduziert die Barrieren für Nutzer anderer Muttersprachen. In Django lassen sich 'Translation-Strings' definieren, die je nach Präferenz in unterschiedlichen Sprachen angezeigt werden. Diese können in Python-Dateien, Templates, für URL-Patterns und auch in JavaScript-Dateien verwendet werden [Django, 2020c].

3.3.3 Admin-Interface

Ein Feature, das Django von anderen Frameworks abhebt, ist das mitgelieferte Admin-Interface, das für die Verwaltung der Datenbankinhalte genutzt werden kann [Django, 2020d]. Dieses wendet sich zwar nicht direkt an die Nutzer der Anwendung, aber auch die Administratoren können Beeinträchtigungen aufweisen, weshalb sichergestellt werden muss, dass auch diese Seite barrierefrei ist. Die Existenz dieser Funktion ist grundsätzlich hilfreich,

Tabelle 3: Filter des 'Humanize'-Pakets [Django, 2020b]

Filter	Beschreibung	Beispiel
apnumber	Schreibt die Zahlen von 1–9 aus.	„1“ wird zu „Eins“. / „10“ bleibt „10“.
intcomma	Fügt alle drei Ziffern einen Punkt (bzw. Komma) hinzu.	„1500000“ wird zu „1.500.000“.
intword	Verwendet eine Text-Alternative für große Zahlen.	„1500000“ wird zu „1,5 Millionen“.
naturalday	Verwendet ‚Gestern‘, ‚Heute‘ und ‚Morgen‘ für Daten, die maximal einen Tag entfernt liegen.	Am 20. August 2020 wird „19 Aug 2020“ zu „Gestern“.
naturaltime	Verwendet relative Angaben für das Datum.	Am 20. August 2020 um 16:01 Uhr wird „20 Aug 2020 16:00:00“ zu „Vor einer Minute“.
ordinal	Konvertiert Integer in Ordinale.	„1“ wird zu „1.“.

wenn bei Anwendenden ein geringes technologisches Verständnis vorhanden ist. Wenn es um die Barrierefreiheit geht, sind hier jedoch mehrere Schwächen vorzufinden.

Nutzer von Screenreadern werden mit der Admin-Seite auf große Schwierigkeiten treffen, denn obwohl weitestgehend semantische HTML-Elemente verwendet werden, sind jedoch an keiner Stelle ARIA-Attribute vorhanden. Problematisch ist das beispielsweise für Icons, für die es keine alternativen Beschreibungen durch ARIA-Label gibt. Auf der Startseite ist eine Auflistung der letzten Aktionen (Objekt hinzugefügt, gelöscht oder bearbeitet) vorzufinden. Das betroffene Objekt selbst ist für Screenreader verfügbar, doch die Art der Aktion ist ausschließlich visuell kodiert. Somit ist dieser Bereich für blinde Nutzer nicht effektiv nutzbar.

Einige Nutzer mit Sehschwächen können Probleme mit der Größe und dem Kontrast der Schrift haben, die sehr häufig nicht einmal die AA-Kriterien der WCAG erfüllen. Davor betroffen sind nicht nur weniger bedeutende Informationen, sondern auch relevante Überschriften. Etwas abgeschwächt wird dieses Problem durch das responsive Design, das das Heranzoomen an die Inhalte effektiv unterstützt.

Positiv zu nennen sind die Möglichkeit, bei der Erstellung neuer Objekte Felder mit Standardwerten zu belegen, die Option eine Auto vervollständigung zu verwenden, das Vorhandensein von Breadcrumbs, die deutliche Hervorhebung des aktuell fokussierten Elements und eine fast durchgehend intuitive Tab-Reihenfolge. Für Letztere gibt es beim Hinzufügen neuer Objekte eine Ausnahme, bei der die visuelle Reihenfolge der Buttons der logischen Tab-Reihenfolge widerspricht.

Bei sehr umfangreichen Datenbanken können einzelne Bereiche nicht ausgeblendet werden, was die Übersichtlichkeit beeinträchtigt. Zusätzlich gibt es keine Skip-Links, weshalb die Tastatur-Navigation in solchen Fällen ineffizient und frustrierend werden kann.

Um die genannten Probleme zu beheben, bestehen die Optionen, einzelne Komponenten der Templates zu ersetzen und Stylesheets zu überschreiben.

3.3.4 Formulare

Die Eingabefelder von Formularen können in Django durch die 'Form'-Klasse definiert oder automatisch aus einem Model abgeleitet werden [Django, 2020e]. Dabei werden Label und Feld-Typen automatisch aus den Variablenbezeichnern und Datentypen ausgewählt, können aber auch manuell überschrieben werden. Ein 'BooleanField' in Django entspricht so beispielsweise dem Typ 'Checkbox' in HTML. Konkretere Informationen über die Art der erforderlichen Eingabe können über das 'Autocomplete'-Attribut bereitgestellt werden. Dieses kann nicht von Django bestimmt werden und muss manuell hinzugefügt werden. Außerdem können Hilfstexte zu den einzelnen Feldern hinzugefügt werden. Entwickler müssen zu deren Verknüpfung mit den Input-Elementen das 'aria-describedby'-Attribut selbst hinzufügen. Gleicher gilt für Fehlermeldungen, die durch selbst geschriebene Validatoren im Backend erzeugt werden. Solche, die im Frontend durch einen Input-Typ hervorgerufen werden, sind bereits barrierefrei. Die semantische Verknüpfung von Feld und Label durch das 'for'-Attribut erfolgt automatisch.

Die Menge notwendiger Eingaben kann durch das Vorausfüllen bereits bekannter Informationen oder Standardwerte verringert werden.

Erforderliche Eingaben werden automatisch durch ein Asterisk (*) gekennzeichnet, der auch von Screenreadern vorgelesen wird. Eine bessere Implementierung ist in Angular Material vorzufinden: Dort ist das Zeichen für erforderliche Eingaben ausschließlich visuell. Screenreader können trotzdem aufgrund des 'required'-Attributs auf die Information zugreifen.

3.3.5 Suche

Häufig verfügen Web-Anwendungen über Suchfunktionen. Für deren Umsetzung bietet Django eine Schnittstelle zur Abstraktion von Datenbank-Anfragen an. Damit Nutzer mit Rechtschreibschwächen diese effektiv nutzen können, können die drei folgenden Filter verwendet werden [Django, 2020f, Django, 2020g]:

- 'Trigram Similar' stellt eine vereinfachte Fuzzy-Suche dar, bei der sowohl die Suchanfrage als auch die möglichen Ergebnisse in dreistellige Strings unterteilt werden, die anschließend miteinander verglichen werden. Überschreitet die Ähnlichkeit einen gewissen Schwellenwert, wird es als Treffer gewertet.
- 'Unaccent' ignoriert Akzente in der Suchanfrage und den möglichen Ergebnissen.
- 'Contains' akzeptiert auch Ergebnisse, in denen der Suchbegriff als Teilwort vorkommt. Soll zusätzlich die Groß- und Kleinschreibung ignoriert werden, kann 'icontains' verwendet werden.

'Trigram Similar' und 'Unaccent' sind nur mit PostgreSQL-Datenbanken kompatibel. Eine leistungsstarke Fuzzy- oder die Suche nach Synonymen lässt sich mit Django nicht direkt umsetzen. Hierfür bieten sich Suchmaschinen, wie 'ElasticSearch', an. Neben den Filtern können auch reguläre Ausdrücke zur Anwendung gebracht werden [Django, 2020g].

Des Weiteren kann die Suche für das Auswählen der Vorschläge einer Autovervollständigung verwendet werden. Diese kann Nutzer mit kognitiven oder motorischen Behinderungen unterstützen.

3.3.6 Vergleich mit einem verwandten Framework: Ruby on Rails

Auch bei Django gilt das Gleiche wie schon zuvor für Bootstrap und Angular: Verwandte Frameworks ähneln sich in der Unterstützung der Barrierefreiheit in fast jeder Hinsicht. An dieser Stelle soll ein Blick auf 'Ruby on Rails' geworfen werden. Im Gegensatz zu Django basiert dieses nicht auf Python, sondern auf Ruby [Ruby on Rails, 2020].

Formulare werden hier auf eine explizitere Art implementiert. So ist etwa die automatische Generierung der Input-Typen aus den Datentypen einer Klasse nicht möglich. Was ARIA betrifft, ist die gleiche Schwäche vorzufinden: Verknüpfungen von Fehlermeldungen und Hilfstexten durch das 'aria-describedby'-Attribut müssen selbstständig hinzugefügt werden. 'For'-Attribute für Label werden hier aber auch wie in Django automatisch generiert. Um die Probleme der Formulare automatisch beheben zu lassen, kann die Bibliothek 'WAIable' verwendet werden [Techvisionblog, 2015].

Auch die verfügbaren Filter, das Template-System und die Suche weisen einen vergleichbaren Umfang auf wie die in Django. Vorinstallierte Admin-Seiten sind generell selten vorzufinden und gibt es auch nicht in 'Ruby on Rails'. Es besteht aber die Möglichkeit, dieses Feature durch eine von mehreren dafür verfügbaren Erweiterungen hinzuzufügen. Ähnliche Probleme wie in Django sind allerdings auch bei diesen nicht auszuschließen. 'ActiveAdmin' ist ein Beispiel, bei dem im Standardlayout ebenfalls Mängel wie zu geringe Kontraste und fehlende Skip-Links zu finden sind [ActiveAdmin, o.D.].

3.4 Gegenüberstellung der betrachteten Web-Frameworks

Bereits beim Betrachten der jeweiligen Dokumentationen ist deutlich zu erkennen welchen Wert die drei analysierten Frameworks auf Barrierefreiheit legen. Sowohl Bootstrap als auch Angular widmen der Barrierefreiheit in ihrer Dokumentation einen separaten Abschnitt, der auch auf allgemeine Aspekte unabhängig vom Framework eingeht und diese genauer erläutert. Auch an anderen Stellen wird immer wieder auf Besonderheiten der beschriebenen Konzepte und Systeme hingewiesen. Im Gegensatz dazu spielt das in der Dokumentation von Django keine Rolle.

Auch bei den vorhandenen Features sind bei Django größere Schwächen zu erkennen. Zwar bieten die anderen beiden Frameworks dadurch, dass sie im Frontend operieren und mehr mit der Nutzer-Interaktion zusammenhängen, auch mehr Ansatzpunkte, dennoch wären vor allem bei den Formularen und dem Admin-Interface mehr Möglichkeiten vorhanden gewesen. Dafür ist das Template-System in Django etwas mächtiger als das in Angular, da Komponenten nicht nur andere Komponenten inkludieren können, sondern auch Vererbung ermöglicht wird.

Sowohl bei Django als auch bei Angular ist das Fokus-Management bei Wechseln der aktuellen Seite von Bedeutung. Bei Angular kommt die Besonderheit hinzu, dass das auch beim dynamischen Austauschen der Inhalte relevant ist.

Beim Vergleich der UI-Komponenten aus Bootstrap und Angular Material ist bei Letzterem vor allem das automatische Generieren entsprechender ARIA-Attribute positiv hervorzuheben. Beide Frameworks verfügen zum großen Teil über sehr ähnliche UI-Komponenten. Ausnahmen dabei sind die 'Breadcrumb'-Komponente in Bootstrap, die es nicht in Angular Material, und die 'Stepper'-Komponente in Angular Material, die es so nicht in

Bootstrap gibt. Eigene barrierefreie UI-Komponenten zu erstellen, kann unter Umständen mühsam werden. Angular Material stellt zu dessen Unterstützung mehrere Werkzeuge im A11y-Paket bereit.

Mit den Template-Filtern und dem dazugehörigen 'Humanize'-Paket können Informationen in Django einfacher für Nutzer mit kognitiven Beeinträchtigungen präsentiert werden. Angular bietet die gleiche Funktion in Form sogenannter 'Pipes'. Da aber nur sehr wenige, mit keinem allzu großen Wert für die Barrierefreiheit verfügbar sind, wurden diese nicht im Angular-Kapitel genannt. Durch die Implementierung eigener Pipes ist es aber dennoch möglich, die gleiche Funktionalität wie in Django zu erhalten.

Bootstrap und Angular Material stellen beide eine Vielzahl an Icons zur Verfügung, die genutzt werden können, um das Verständnis für manche Funktionalitäten zu fördern.

Grundsätzlich haben alle Frameworks ihre Besonderheiten, die bei der Umsetzung der Barrierefreiheit helfen können. In manchen Fällen bietet sich die Kombination verschiedener Frameworks an, um die jeweiligen Vorteile miteinander zu kombinieren und Schwächen auszugleichen. So kann es etwa für Django sinnvoll sein, es zusammen mit einem CSS-Framework wie Bootstrap zu verwenden. Das wird im nächsten Kapitel noch deutlicher zu erkennen sein.

Kapitel 4

Demonstration der Umsetzungsmöglichkeiten

Nachdem in Kapitel 3 gezeigt wurde, wie die verschiedenen Arten von Web-Frameworks mit Barrierefreiheit umgehen, wird die Umsetzung nun konkret anhand einer beispielhaften Anwendung demonstriert. Zunächst werden das Thema und die Anforderungen an diese skizziert, bevor näher auf den technischen Aufbau eingegangen wird.

4.1 Thema und Anforderungen

Das gewählte Beispiel ist eine Web-Anwendung einer fiktiven Fluggesellschaft namens 'Ally Airlines'. Die Grundlage dieser Entscheidung ist, dass das Fliegen und somit auch das Buchen eines Fluges etwas ist, das jedem Menschen zugänglich sein sollte. Außerdem bietet dieses Szenario genügend Umfang, um die Funktionalitäten der Frameworks möglichst vollständig abzudecken.

Die Umsetzung erfolgte in zwei unabhängigen Projekten. Für das eine wurde Django in Kombination mit Bootstrap und im anderen Angular mit Angular Material verwendet. Das Ziel war es, eine prototypische Anwendung mit den grundlegenden Basisfunktionalitäten zu entwickeln.

4.1.1 Inhaltliche Anforderungen

Die zentrale Funktionalität der Anwendung ist das Suchen und Buchen von Flügen. Zusätzlich sollen gebuchte Flüge eingesehen und wieder storniert werden können. Die Startseite soll beim Durchstöbern der Flüge Unterstützung bieten, außerdem soll es eine eigene Unterseite für Angebote geben. Eine Anmeldefunktionalität soll nur simuliert werden. Zuletzt werden auch ein Impressum und Informationen zur Kontaktaufnahme mit dem Unternehmen benötigt.

4.1.2 Anforderungen an die Barrierefreiheit

Eine Web-Anwendung für jeden individuellen Nutzer barrierefrei zu machen, ist vermutlich niemals vollständig möglich. Deshalb wurden für den Fokus dieser Arbeit vier Personae erstellt, die besonders häufig auftretende Fälle repräsentieren, und für die die Anwendungen

optimiert werden sollen (Abbildung 6). Die Wahl der Eigenschaften der darin beschriebenen Personen basiert auf folgender Datengrundlage:

- **Persona #1** Im Jahr 2016 wurde die Anzahl der an Parkinson Erkrankten auf über sechs Millionen geschätzt. Mit mehr als doppelt so vielen Erkrankungen im Vergleich zum Jahr 1990 ist es die am schnellsten zunehmende, neurologisch bedingte Krankheit. Tendenziell sind Männer häufiger betroffen als Frauen [Dorsey et al., 2018].
- **Persona #2** Die Katarakt bildet sich häufig erst im hohen Alter, kann aber auch angeboren sein. Laut einer Umfrage aus 2005 sind etwa 7,6 % der Deutschen schon einmal daran erkrankt. Es gibt einen großen Geschlechterunterschied, demzufolge Frauen deutlich häufiger betroffen sind als Männer [RKI, 2017].
- **Persona #3** Zwar ist die Katarakt die häufigste Ursache für Erblindungen, durch die inzwischen guten Möglichkeiten zur Behandlung hat aber in Deutschland das Glaukom mit über 15 % einen deutlich größeren Anteil an den Ursachen der Neuerblindungen. Ab 60 Jahren überschreitet das Erkrankungsrisiko bei Männern 3 % [RKI, 2017].
- **Persona #4** Weltweit gibt es etwa 50 Millionen an Demenz erkrankte Menschen. Zwischen 5 und 8 % der über 60-Jährigen sind davon betroffen, was die Krankheit zu einer der Hauptursachen für Behinderungen in dieser Altersgruppe macht [WHO, 2019a].

Da die Optimierung für Nutzer mit Hörbehinderungen größtenteils durch reines HTML, unabhängig von Frameworks, erreicht wird und die Maßnahmen für Nutzer mit psychischen Krankheiten eine große Schnittmenge mit denen der kognitiven Beeinträchtigungen aufweisen, wurden für diese Arten von Behinderungen keine Personae erstellt.

4.2 Architektur der Anwendungen

Die Architekturen der beiden Anwendungen sind sich grundsätzlich sehr ähnlich. Deshalb wird an dieser Stelle hauptsächlich nur die des Angular-Projekts genauer betrachtet. Detalliertere Informationen können den ReadMe-Dateien und Dokumentationen entnommen werden.

Die Anwendung besteht aus einer Homepage und acht Unterseiten. Für jede Seite existiert eine Hauptkomponente, die wiederum in weitere Unterkomponenten unterteilt sein kann. Die Wurzelkomponente des Projekts ('AppComponent') enthält das Navigationsmenü und eine Fußzeile, sowie das 'router-outlet'. Abhängig von der aktuellen Route wird darin die entsprechende Komponente der angeforderten Seite platziert. Die Routen stimmen in beiden Projekten vollständig überein.

Abbildung 7 zeigt grafisch, wie die Angular-Komponenten in Beziehung zueinander stehen und durch welche Routen sie adressiert werden können.

Die grundlegenden Model-Klassen repräsentieren Flüge, Buchungen und Angebote. In Abbildung 8 sind alle Klassen in einem Entity-Relationship-Diagramm (ER-Diagramm) dargestellt. Diese sind identisch mit denen aus dem Django/Bootstrap-Projekt, mit der

KAPITEL 4. DEMONSTRATION DER UMSETZUNGSMÖGLICHKEITEN

Persona #1

Name	Nils Edwards
Alter	24 Jahre
Familie	Ledig, keine Kinder
Beruf	Dolmetscher



Technologie-Erfahrung
Internet:
Verwendet: Notebook, Smartphone

Art der Beeinträchtigung
Leidet an einer frühen Phase von Parkinson. Diese ist gekennzeichnet durch Bewegungsarmut und zitternde Händen in Ruhe.

Bezug zur Anwendung
Er hat mehrere Freunde im Ausland, die er hin und wieder besuchen geht. Diese wohnen teilweise an Orten, die nur mit dem Flugzeug zu erreichen sind. Aufgrund seiner Krankheit ist er darauf angewiesen, dass Web-Anwendungen nicht zu präzise Mauseingaben erfordern und bestenfalls vollständig mit der Tastatur bedienbar sind.

Persona #2

Name	Jennica Berger
Alter	31 Jahre
Familie	Verheiratet, keine Kinder
Beruf	Fotografin



Technologie-Erfahrung
Internet:
Verwendet: Notebook, Smartphone

Art der Beeinträchtigung
Leidet an Katarakt, der für eine getrübte Sicht sorgt. Zu geringe Kontraste und zu kleinen Dingen kann sie nicht, oder nur mit sehr großer Anstrengung erkennen.

Bezug zur Anwendung
Als Fotografin macht sie viele Reisen, für die sie oft Flüge buchen muss. Sie besitzt zwar ausreichende Erfahrungen mit dem Internet, trifft dort aber immer wieder auf Inhalte, deren visuelle Eigenschaften ihr Probleme bereiten.

(a) Persona 1: Nils Edwards
(b) Persona 2: Jennica Berger

Persona #3

Name	Markus Kröger
Alter	68 Jahre
Familie	Verheiratet, ein Kind
Beruf	Physiotherapeut



Technologie-Erfahrung
Internet:
Verwendet: Desktop-PC, Tablet, Smartphone

Art der Beeinträchtigung
Er ist mit 65 Jahren an Folge eines Glaukoms erblindet.

Bezug zur Anwendung
Er lernt gerne die Kulturen anderer Länder kennen. Aufgrund seiner Erblindung ist er bei der Verwendung von Web-Anwendungen auf Screen-Reader angewiesen. Häufig sind die Anwendungen, die er verwenden möchte, jedoch nicht dafür optimiert.

Persona #4

Name	Frieda Schmidt
Alter	71 Jahre
Familie	Geschieden, 3 Kinder
Beruf	Rentnerin



Technologie-Erfahrung
Internet:
Verwendet: Desktop-PC

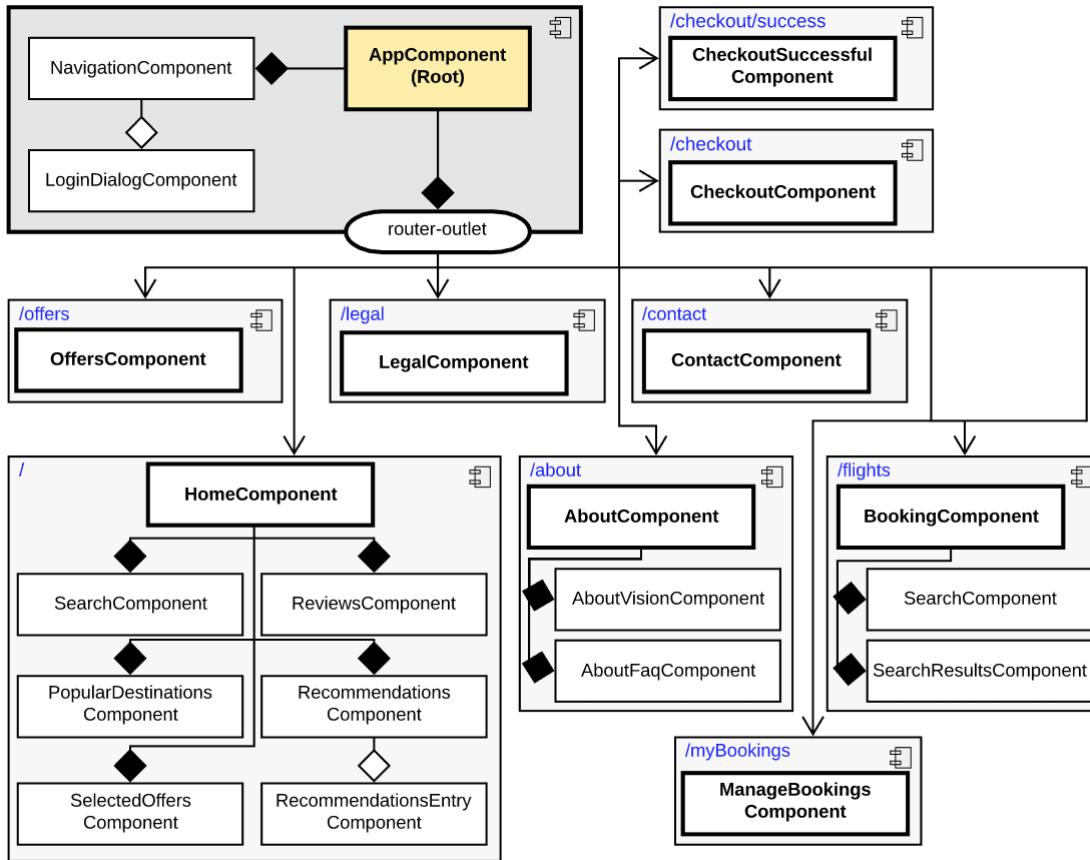
Art der Beeinträchtigung
Leidet an beginnender Demenz und hat nur geringe Erfahrungen mit modernen Technologien. Deshalb verspürt sie große Unsicherheit bei der Verwendung des Internets und hat Angst, dabei Fehler zu begehen.

Bezug zur Anwendung
Sie plant, ihre beiden ausgewanderten Kinder zu besuchen. Seit einem Jahr besitzt sie einen PC, auf dem sie nur die grundlegendsten Funktionalitäten beherrscht. Auch wegen ihrer Demenz spielen Konsistenz und Einfachheit eine große Rolle beim Design der Web-Anwendungen.

(c) Persona 3: Markus Kröger
(d) Persona 4: Frieda Schmidt

Abbildung 6: Personae

29



Hinweis:

Die 'SkipLinkComponent' ist in mehreren Komponenten enthalten und wurde der Übersichtlichkeit halber weggelassen.

Abbildung 7: Übersicht über die Angular-Komponenten und das Routing

Ausnahme, dass es dort keine 'SearchRequest'-Klasse gibt und alle eine von Django automatisch generierte ID zugewiesen bekommen.

4.3 Umsetzung der Barrierefreiheit in den Anwendungen

Als nächstes werden die Aspekte der Anwendungen beschrieben, die für die Barrierefreiheit von Relevanz sind. Dabei wird zuerst ein Bezug zu den erstellten Personae (Abbildung 6) hergestellt, bevor zum Schluss noch auf weitere Besonderheiten eingegangen wird.

4.3.1 Aspekte mit Bezug zu Persona #1: Parkinson

Aspekte, die Persona #1 betreffen, sind die Unterstützung der vollständigen Tastaturl-Bedienbarkeit, das Fokus-Management und die Reduzierung benötigter Eingaben.

In beiden Anwendungen enthält die Wurzelkomponente, in der sich die statischen Inhalten befinden ('app.component.html' in Angular und 'base.html' in Django/Bootstrap),

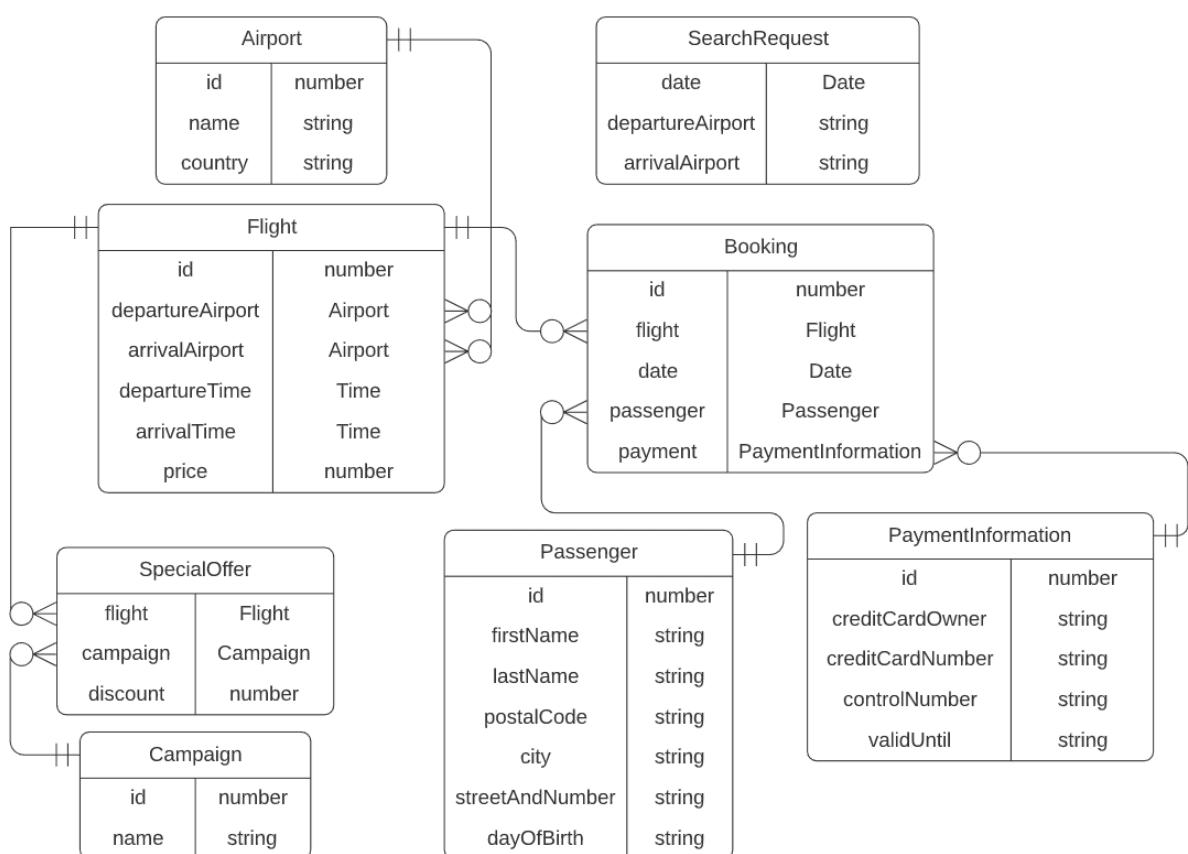


Abbildung 8: ER-Diagramm der Klassen aus dem Angular-Projekt

```
// Subscribe to the NavigationEnd-Event
this.router.events.pipe(filter(e => e instanceof NavigationEnd))
  .subscribe((ng: NavigationEnd) => {
    // Focus the main header after every navigation
    const mainHeader: HTMLElement = document.querySelector('#mainHeader');
    mainHeader.focus();
  });

```

app.component.ts

Abbildung 9: Fokus-Management mit dem NavigationEnd-Event des Angular-Routers

```
<!-- Booking Details Overlay -->
<div role="dialog" aria-labelledby="bookingInfos"
  (keydown.escape)="hideBookingDetails()"
  cdkTrapFocus cdkTrapFocusAutoCapture>
  ...
</div>
```

manage-bookings.component.html

Abbildung 10: Implementierung eines Overlays mit der Anguar 'FocusTrap'

das oberste Heading-Element mit dem Titel der aktuellen Seite. Abhängig von der Route ändert sich hier nur der Inhalt. Nach fast jeder Navigation wird diesem der Fokus zugewiesen. In Angular geschieht das Fokus-Management über das Auffangen des NavigationEnd-Events, das vom Router ausgelöst wird (Abbildung 9).

Für das Anmeldefenster wurde die Modal- bzw. die Dialog-Komponente verwendet. Bei beiden wechselt der Fokus nach dem Schließen zwar zurück zu dem Element, das ihn zuvor hatte, da der An- bzw. Abmelden-Button aber bei Erfolg der Aktion ausgetauscht wird, ist das ursprüngliche Element nicht mehr verfügbar. Deshalb mussten hierfür Änderungen vorgenommen werden.

Zur Demonstration der Funktionsweise der 'FocusTrap' in Angular Material wurde diese dort für das Overlay der Detailansicht von Buchungen eingesetzt. Die Direktive 'cdkTrapFocus' verhindert, dass der Fokus hinter das Overlay gelangt und 'cdkTrapFocusAutoCapture' setzt ihn nach dem Öffnen auf das erste fokussierbare Element im definierten Bereich. Durch das Auffangen des 'Keydown'-Events wird beim Drücken der 'Esc'-Taste das Overlay wieder geschlossen. Außerdem wurde die Rolle 'dialog' hinzugefügt (Abbildung 10). Vor allem bei Buttons ist die visuelle Hervorhebung des Fokus sowohl in Angular Material als auch in Bootstrap sehr schwach, weshalb dieser an mehreren Stellen verstärkt wurde.

Eine Vielzahl der Bootstrap und Angular Material UI-Komponenten wurden in den Anwendungen verbaut. Das Verhalten eines Karussells aus Bootstrap wurde erweitert, so dass es im fokussierten Zustand automatisch pausiert. Außerdem wurde die 'ScrollSpy'-Komponente um Skip-Links, die vom Inhalt zurück zur Navigation verweisen, ergänzt. Alle anderen Komponenten erfüllen die Anforderungen an die Interaktionsmöglichkeiten bereits von selbst.

Die Startseite der Anwendungen enthält eine Liste mit Empfehlungen für Flüge. In beiden

```
# Returns the options for the autocomplete of the search form
# as JSON-Response, depending on the input term in the GET-Parameters
def autocomplete(request):
    inputTerm = request.GET.get('search', False)
    # Filters airports, which names or countries include the search term
    fittingAirports = Airport.objects.filter(
        Q(name__icontains=inputTerm) | Q(country__icontains=inputTerm))
    return JsonResponse({'options': list(fittingAirports.values())})
```

booking/views.py

Abbildung 11: Auswahl der Optionen der Autovervollständigung mit Hilfe der Schnittstelle zur Datenbank in Django

Projekten kann durch sie mit der 'Tab'-Taste navigiert werden. An dieser Stelle wurde im Angular-Projekt der FocusKeyManager, eine Variante des ListKeyManagers, eingesetzt, damit die Liste auch mit den Pfeiltasten als Alternative verwendet werden kann (zu finden in 'recommendations.component.ts').

Das Suchformular für Flüge wurde um eine Autovervollständigung ergänzt. Dabei wurde im ersten Projekt die Angular Material 'Autocomplete'-Komponente und im zweiten das HTML <datalist>-Element in Kombination mit der Django Suche verwendet. Letzteres geschieht über asynchrone Anfragen an das Backend, das diese mit dem 'icontains'-Filter auf Flughafen- und Ländername ausgewertet (Abbildung 11).

Die Formulare in der Buchungsabwicklung werden aus Django-Klassen generiert. Um die Vorschläge der Autovervollständigung zu optimieren, wurden zusätzlich zu den automatisch hinzugefügten 'type'- konkretere 'Autocomplete'-Attribute hinzugefügt (Abbildung 13).

Außerdem werden manche Formularfelder mit bereits bekannten Informationen von Django vorausgefüllt: Die Buchungsabwicklung besteht aus drei Schritten. Beim Zurückgehen zu einem abgeschlossenen Schritt müssen bereits getätigte Angaben nicht erneut eingegeben werden.

Während mit Bootstrap Skip-Links durch die Screenreader-Klassen umgesetzt werden konnten, wurde dafür in Angular eine separate Komponente erstellt, die die .cdk-visually-hidden-Klasse verwendet. Die Beschriftung des Links hat einen Standardwert, kann aber wie das Zielement und dessen Tabindex konfiguriert werden (Abbildung 12). Die Komponente sorgt bei der Initialisierung dafür, dass Letzterer entsprechend geändert wird. Wird kein Tabindex angegeben, überprüft der 'InteractivityChecker', ob das Zielement bereits fokussierbar ist. Ist das nicht der Fall, erhält es den Tabindexwert -1, damit es zwar nicht Teil der Tab-Reihenfolge ist, aber bei Fokussierung durch Skip-Links visuell hervorgehoben wird.

4.3.2 Aspekte mit Bezug zu Persona #2: Katarakt

Für Persona #2 ist es wesentlich, dass Kontraste groß genug sind und Inhalte problemlos vergrößert werden können.

```
<app-skip-link targetElement="mainHeader"
                description="Zum Hauptinhalt springen"
                targetTabIndex="-1">
</app-skip-link>
```

Abbildung 12: Beispielanwendung der 'SkipLinkComponent' in Angular

```
class PaymentForm(forms.ModelForm):
    class Meta:
        widgets = {
            'creditCardOwner': TextInput(
                attrs={'autocomplete': 'cc-name'}),
            'creditCardNumber': TextInput(
                attrs={'aria-describedby': 'error_1_id_creditCardNumber',
                       'autocomplete': 'cc-number'})}
```

checkout/forms.py

Abbildung 13: ARIA- und 'Autocomplete'-Attribute in Django-Formularen (vereinfachter Ausschnitt)

Beide Anwendungen sind responsiv und können auch bei hoher Zoomstufe noch vollständig genutzt werden. Um das zu erreichen wurden im Angular-Projekt Media-Queries verwendet und im anderen kommt das Bootstrap Grid-System zur Anwendung. Icons die rein visuell sind, werden bei Letzterem ab einer bestimmten Größe durch die .d-none-Klasse ausgeblendet, um anderen Inhalten mehr Platz zu verschaffen.

In Bootstrap wurden einige der Farben überschrieben und in Angular Material wurde ein eigenes 'Theme' erstellt, um vollständige AAA-Konformität der Kontraste zu erreichen.

4.3.3 Aspekte mit Bezug zu Persona #3: Erblindung

Für Persona #3 spielen ebenfalls die Aspekte von Persona #1 zur Tastatur-Interaktion eine Rolle. Zusätzlich musste die Anwendung für Screenreader optimiert werden.

Die UI-Komponenten wurden mit den benötigten ARIA-Attributen ausgestattet. Für die aus Angular Material waren nur kleine Änderungen zu den bereits automatisch generierten notwendig.

Um sowohl die Formulare aus Django als auch Bootstrap zu demonstrieren, wurden beide an verschiedenen Stellen eingesetzt. Die ARIA-Attribute, die Django nicht zur Verknüpfung der Hilfstexte und Fehlermeldungen mit den Input-Elementen generiert, wurden in den Form-Klassen definiert (Abbildung 13).

Der 'LiveAnnouncer' aus dem A11y-Paket wurde für mehrere Zwecke eingesetzt: Nach erfolgreichem An- und Abmelden, nach dem Stornieren von Buchungen und nach dem Rückgängig machen von Stornierungen werden an Screenreader entsprechende Nachrichten übermittelt. Außerdem wird auch nach einer Suche die Anzahl der gefundenen Ergebnisse angekündigt.

```

<section role="search" aria-labelledby="search">
    <h2 id="search">Flug suchen</h2>
    <app-skip-link targetElement="popular">
    </app-skip-link>
    <app-search></app-search>
</section>

<section aria-labelledby="popular">
    <h2 id="popular">Beliebte Reiseziele</h2>
    <app-skip-link targetElement="selectedOffers">
    </app-skip-link>
    <app-popular-destinations></app-popular-destinations> <!-- Sub-Component -->
</section>
...

```

home.component.html

Abbildung 14: Typischer Aufbau der Hauptkomponenten von Unterseiten

nisse angekündigt.

Die Strukturierungsmöglichkeiten der Template-Systeme von Django und Angular halfen dabei, eine logische Hierarchie der Überschriften umzusetzen, Seitenregionen festzulegen und Skip-Links sinnvoll zu platzieren. Die meisten Hauptkomponenten sind nach dem Muster aus Abbildung 14 aufgebaut: Logisch zusammenhängende Inhalte wurden in Unterkomponenten ausgelagert. Vor ihnen befinden sich Skip-Links und Überschriften der gleichen Ordnung. Eingebettet sind sie in Seitenregionen, deren Label von den Überschriften bereitgestellt werden.

Das 'title'-Element wird in Django von erbenden Templates überschrieben, um den darin enthaltenen Inhalt an die aktuelle Unterseite anzupassen. In Angular geschieht das mit Hilfe des Title-Service.

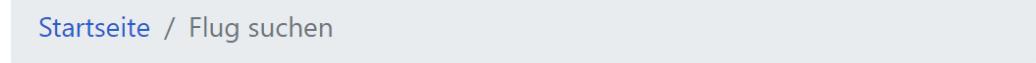
Die .sr-only-Klasse aus Bootstrap konnte neben den Skip-Links noch an weiteren Stellen Anwendung finden. Beispielsweise wird nach dem Abschluss einer Buchung ein 'Spinner' für die Repräsentation eines Ladeprozesses angezeigt. Dieser wurde mit einem Hinweis ausschließlich für Screenreader ausgestattet.

4.3.4 Aspekte mit Bezug zu Persona #4: Demenz

Um Persona #4 zu adressieren, wurde auf die Konsistenz und Einfachheit der Anwendung geachtet. Außerdem wurden mögliche Fehlerquellen der Nutzereingaben reduziert.

Konsistenz wurde erreicht durch die Wiederverwendung bestimmter Elemente, etwa dem Suchformular, und durch das Bestehenbleiben statischer Elemente wie der Navigation und der Fußzeile.

Das Vererbungssystem in Django kommt auch in Kombination mit der Breadcrumb Komponente aus Bootstrap zum Einsatz. Die Wurzelkomponente enthält das Gerüst, davon erbende Komponenten erweitern dieses mit dem Link der aktuell angezeigten Seite (Abbildung 15).



```

<!-- BREADCRUMBS SCAFFOLD -->
<nav aria-label="Breadcrumbs">
    <ol class="breadcrumb">
        {% block breadcrumb %}
        <li class="breadcrumb-item">
            <a href="/">Startseite</a>
        </li>
        {% endblock %}
    </ol>
</nav>

```

base.html


```

<!-- BREADCRUMBS EXTENSION -->
{% block breadcrumb %}
{{ block.super }}
<li class="breadcrumb-item">
    <a href="/flights">
        Flug suchen
    </a>
</li>
{% endblock %}

```

booking/booking_base.html

Abbildung 15: Template-Vererbungssystem in Django für Breadcrumbs

Die Auto vervollständigung, Hinweise und Validierung inklusive Fehlermeldungen bei den Formularen verhindern mögliche Fehler und bieten Unterstützung bei der korrekten Eingabe. Die Buchungsabwicklung wurde in mehrere Schritte unterteilt. Um den bisherigen Fortschritt und noch ausstehende Schritte zu kommunizieren, wurde in Bootstrap der Fortschrittsbalken und in Angular der 'Stepper' verwendet. Bevor die Buchung abgesendet wird, können die Angaben im letzten Schritt noch einmal eingesehen und kontrolliert werden.

Auch an anderen Stellen wurden die gleichzeitig angezeigten Informationen durch die UI-Komponenten verringert: Die häufig gestellten Fragen befinden sich in einem Accordion und Details zu den Buchungen werden erst auf Verlangen angezeigt. Die 'ScrollSpy'-Komponente aus Bootstrap wurde im Impressum verwendet, um einen Orientierungspunkt für die Position bereit zu stellen. Icons in beiden Anwendungen, unter anderem für die unterschiedlichen Kontaktmöglichkeiten, tragen zum erleichterten Verständnis der Informationen bei.

4.3.5 Weitere Aspekte

Abschließend werden weitere Aspekte, die keinen direkten Bezug zu den Personae aufweisen, beschrieben.

Die Startseite des Admin-Interfaces in Django wurde überarbeitet: Farben wurden so angepasst, dass Kontraste AAA-konform sind. Icons, die in der Auflistung der letzten Aktionen deren Art darstellen, wurden um Texte erweitert, um diese auch für Screenreader zugänglich zu machen. Jeder Tabelle wurde ein Skip-Link mit Bootstrap hinzugefügt. Außerdem wurde auch die Tab-Reihenfolge der Buttons, in der Ansicht zum Hinzufügen neuer Datenbankeinträge, an die visuelle Anordnung angepasst (zu finden in 'templates/admin'). Die Änderungen im Admin-Interface sind durch Translation-Strings in Deutsch und in Englisch verfügbar.

Animationen im Angular-Projekt lassen sich durch einen Schalter deaktivieren. Umgesetzt

wurde dies mit dem @.disabled-Binding in der Wurzelkomponente 'app.component.ts'. Einige minimale Animationen, zum Beispiel bei den 'Tabs' und dem 'Spinner', sind davon allerdings nicht betroffen und werden weiterhin angezeigt.

Ein Karussell auf der Startseite, dessen Inhalte automatisch rotieren, wurde über die JavaScript-Schnittstelle verlangsamt und um einen Button zum Pausieren erweitert.

Die Datumsangaben von Buchungen wurden durch den 'naturaltime'-Filter in Django formatiert. Eine ähnliche Funktion wurde in Angular mithilfe einer selbsterstellten Pipe umgesetzt ('relativetime.pipe.ts'). So sind zusätzlich zum absoluten Flugdatum relative Angaben vorhanden.

Zuletzt wurden die Codelyzer-Regeln zur Barrierefreiheit im Angular-Projekt hinzugefügt. Keiner dieser konnte Mängel ausfindig machen.

Kapitel 5

Evaluierung

Die Barrierefreiheit von Web-Anwendungen kann auf verschiedene Arten evaluiert werden. Automatisierte Tools wie der in Kapitel 3.2.2 beschriebene Codelyzer können dabei helfen, ohne großen Aufwand grobe Mängel aufzudecken und zu beheben. Da es dabei aber nicht unwahrscheinlich ist, dass nur ein Teil der Mängel gefunden wird und auch falsch-positive Ergebnisse gemeldet werden, sollte zusätzlich eine Evaluierung durch einen Menschen stattfinden [Vigo et al., 2013]. Das schließt zum einen die empirische Evaluation durch Testpersonen als auch die Überprüfung mit Hilfe von Checklisten und Guidelines ein.

Nachfolgend werden die WCAG 2.1-Kriterien zur Evaluierung der beiden Anwendungen aus Kapitel 4 verwendet.

5.1 Evaluierung mithilfe der WCAG 2.1-Kriterien

Die Tabellen 4 bis 7 zeigen die Ergebnisse der Evaluierung, geordnet nach den vier Prinzipien der WCAG. Diese enthalten neben der Information, welche Erfolgskriterien erfüllt wurden, auch solche über den Bezug zu den Frameworks. Alle Seiten mit Ausnahme der des Admin-Interfaces wurden für die Evaluierung mit einbezogen. Anmerkungen zu Letzterem sind dennoch in den Tabellen enthalten. Version 2.1 der WCAG ist bisher nur auf englisch verfügbar. Die deutschen Übersetzungen in den Tabellen entspricht zum größten Teil denen der WCAG 2.0 [W3C, 2009]. Für nicht vorhandene Übersetzungen wurden eigene erstellt.

5.2 Auswertung der Ergebnisse

Die Anwendungen sind AA-konform. Da die Lesbarkeit (Richtlinie 3.1) in der Auswertung zum Teil als nicht den Kriterien genügend eingestuft wurden und die Animationen der UI-Komponenten aus Bootstrap und Angular Material nicht vollständig deaktivierbar sind, wurde Stufe AAA nicht erreicht. Außerdem sind die Flächen der Interaktionselemente (Erfolgskriterium 2.5.5) an manchen Stellen nicht groß genug und die Beschreibungen mancher Links sind ohne den Kontext der Seitenregionen nicht eindeutig (Erfolgskriterium 2.4.9). Anhand der Tabellen ist zu erkennen, dass die betrachteten Frameworks auf eine Vielzahl der Kriterien Einfluss haben können. Ein Bereich, der weniger betroffen ist, ist der der zeitbasierten Medien (Richtlinie 1.2).

KAPITEL 5. EVALUIERUNG

Tabelle 4: WCAG Evaluierung - 'Prinzip 1: Wahrnehmbar'

Richtlinie	Erfolgskriterien	Erfüllt?	Bezug zu den Frameworks / Anmerkungen	Stufe
1.1 Text-alternativen	1.1.1 Nicht-Text-Inhalt	Ja	(AM): 'Aria-hidden' ist bei Icons standardmäßig auf 'true' gesetzt. (B): Steuerelemente des Accordions und Icons benötigten zusätzliche ARIA-Attribute.	A
1.2 Zeitbasierte Medien	1.2.1 - 1.2.9	Ja	Die Anwendung ist hiervon nicht betroffen.	A - AAA
1.3 Anpassbar	1.3.1 Info und Beziehungen	Ja	(AM): Automatisch generierte ARIA-Attribute. (AM + B): Zusätzliche ARIA-Attribute und semantische HTML-Elemente verwendet.	A
	1.3.2 Bedeutungs-tragende Reihenfolge	Ja	(A + D): Umsetzung einer logischen Hierarchie durch Template-Systeme unterstützt. (D): Visuelle Anordnung der Elemente im Admin-Interface an die des DOMs angepasst.	A
	1.3.3 Sensorische Eigenschaften	Ja	(B): Karussell-Kontrollelemente und Fortschrittsbalken wurden um ARIA-Attribute ergänzt.	A
	1.3.4 Ausrichtung	Ja	/	AA
	1.3.5 Bestimmen des Zwecks der Eingabe	Ja	(D): Eingabetypen der Formulare werden automatisch generiert. 'Autocomplete'-Attribute wurden ergänzt.	AA
	1.3.6 Bestimmen des Zwecks	Ja	(A + D): Template-Systeme haben bei der Strukturierung der Seitenregionen geholfen.	AAA
1.4 Unterscheid-bar	1.4.1 Benutzung von Farben	Ja	(AM + B): Farben enthalten keine Informationen, die nicht auch textuell vorhanden sind.	A
	1.4.2 Audio-Steuer-elemente	Ja	Die Anwendung ist hiervon nicht betroffen.	A
	1.4.3 Kontrast (Minimum)	Ja	(AM): Eigenes Theme für hohe Kontraste definiert. (B): Farbpalette wurde angepasst.	AA
	1.4.4 Textgröße ändern	Ja	(B): Grid-System half bei der Umsetzung eines responsiven Designs.	AA
	1.4.5 Bilder eines Textes	Ja	/	AA
	1.4.6 Kontrast (erhöht)	Ja	Siehe 1.4.3.	AAA
	1.4.7 Leiser oder kein Hintergrund-Audioinhalt	Ja	Die Anwendung ist hiervon nicht betroffen.	AAA
	1.4.8 Visuelle Präsentation	Ja	/	AAA
	1.4.9 Bilder eines Textes (keine Ausnahme)	Ja	/	AAA
	1.4.10 Reflow	Ja	(B): Durch das Grid-System ermöglicht.	AA
	1.4.11 Nicht-textueller Kontrast	Ja	(B): Kontrollelemente des Karussells mussten um einen dunklen Hintergrund ergänzt werden.	AA
	1.4.12 Textabstand	Ja	/	AA
	1.4.13 Inhalt bei Hover oder Fokus	Ja	Die Anwendung ist hiervon nicht betroffen.	AA

Abkürzungen: A=Angular; AM=Angular Material; B=Bootstrap; D=Django

KAPITEL 5. EVALUIERUNG

Tabelle 5: WCAG Evaluierung - 'Prinzip 2: Bedienbar'

Richtlinie	Erfolgskriterien	Erfüllt?	Bezug zu den Frameworks / Anmerkungen	Stufe
2.1 Per Tastatur zugänglich	2.1.1 Tastatur	Ja	(AM + B): Komponenten sind Tastaturbedienbar, kleine Optimierungen wurden vorgenommen. (z.B. Karussell pausiert bei Fokussierung)	A
	2.1.2 Keine Tastaturfalle	Ja	(AM + B): Komponenten mit Fokusfallen ('Modal', 'Dialog') können mit der Tastatur wieder verlassen werden. (AM): Selbst implementiertes Overlay mithilfe der 'FocusTrap' kann mit der Tastatur verlassen werden.	A
	2.1.3 Tastatur (keine Ausnahme)	Ja	Siehe 2.1.1.	AAA
	2.1.4 Tastatur-Kurzbefehle	Ja	/	A
2.2 Ausreichend Zeit	2.2.1 Zeiteinteilung anpassbar	Ja	(AM + B): 'Toasts' und 'Snackbars' werden so lange angezeigt, bis die Informationen veraltet sind.	A
	2.2.2 Pausieren, beenden, ausblenden	Ja	(B) Karussell wurde um eine Pause-Funktion erweitert.	A
	2.2.3 Keine Zeiteinteilung	Ja	Siehe 2.2.1.	AAA
	2.2.4 - 2.2.6	Ja	Die Anwendung ist hiervon nicht betroffen.	AAA
2.3 Anfälle	2.3.1 Drei Blitze oder weniger	Ja	/	A
	2.3.2 Drei Blitze	Ja	/	AAA
	2.3.3 Durch Interaktion ausgelöste Animation	Nein	(A): Animationen können grundsätzlich deaktiviert werden. Label der Formulare und Tabs enthalten trotzdem noch minimale Animationen. (B): Animationen können über das Betriebssystem deaktiviert werden (durch Media-Query 'prefers-reduced-motion'). Der 'Spinner' ist davon aber nicht betroffen.	AAA
2.4 Navigierbar	2.4.1 Blöcke umgehen	Ja	(AM): Skip-Links wurden mithilfe der Funktionen des A11y-Pakets implementiert. (B): Skip-Links wurden mithilfe der dafür verfügbaren Klassen implementiert. (D): Die Admin-Seite wurde um Skip-Links ergänzt.	A
	2.4.2 Seite mit Titel versehen	Ja	(A): Titel wird mit dem 'Title'-Service aktualisiert. (D): Titel wird über Vererbung im Template-System geändert.	A
	2.4.3 Fokus-Reihenfolge	Ja	(D): Tab-Reihenfolge im Admin-Interface wurde angepasst.	A
	2.4.4 Linkzweck (im Kontext)	Ja	/	A
	2.4.5 Verschiedene Methoden	Ja	(A + D): Die Navigation befindet sich in der Wurzelkomponente und ist auf jeder Unterseite verfügbar. (Außerdem existieren Verlinkungen zu verwandten Seiten).	AA
	2.4.6 Überschriften und Beschriftungen	Ja	(A + D): Vor semantisch abgeschlossenen Unterkomponenten befinden sich Überschriften, die diese beschreiben. (D): Label in Formularen werden automatisch generiert.	AA
	2.4.7 Fokus sichtbar	Ja	(AM + B): Fokus wird in allen UI-Komponenten visuell hervorgehoben. Deutlichkeit wurde erhöht.	AA
	2.4.8 Position	Ja	Allgemein: Aktuelle Seite wird in Navigation hervorgehoben. (B): 'Breadcrumbs'-Komponente verwendet.	AAA
	2.4.9 Linkzweck (reiner Link)	Nein	Der Zweck einiger Links ist nur mit Kontext eindeutig.	AAA
	2.4.10 Abschnittsüberschriften	Ja	Siehe 2.4.6.	AAA
2.5 Eingabe Modalitäten	2.5.1 Zeigergesten	Ja	Die Anwendung ist hiervon nicht betroffen.	A
	2.5.2 Zeiger Abbruch	Ja	/	A
	2.5.3 Beschriftung mit Namen	Ja	/	A
	2.5.4 Bewegungssteuerung	Ja	Die Anwendung ist hiervon nicht betroffen.	A
	2.5.5 Zielgröße	Nein	Wird zu großen Teilen erreicht, aber nicht überall.	AAA
	2.5.6 Gleichzeitige Eingabemechanismen	Ja	/	AAA

Abkürzungen: A=Angular; AM=Angular Material; B=Bootstrap; D=Django

KAPITEL 5. EVALUIERUNG

Tabelle 6: WCAG Evaluierung - 'Prinzip 3: Verständlich'

Richtlinie	Erfolgskriterien	Erfüllt?	Bezug zu den Frameworks / Anmerkungen	Stufe
3.1 Lesbar	3.1.1 Sprache der Seite	Ja	(D): Admin-Interface erhält 'lang'-Attribut automatisch.	A
	3.1.2 Sprache von Teilen	Ja	/	AA
	3.1.3 Ungewöhnliche Wörter	Nein	Die 'Über Uns'-Seite enthält Begriffe, für die sich ergänzende Erläuterungen (z.B. für 'assistive Technologien') oder leichter verständliche Synonyme (z.B. für 'Vision') anbieten. Ähnliches gilt für das Impressum. (A): Pipes für Formatierungen erstellt und verwendet. (D): 'Naturaltime'-Filter verwendet.	AAA
	3.1.4 Abkürzungen		Die exakten Bezeichner der Flughäfen sind nur als Abkürzungen vorhanden.	
	3.1.5 Leseniveau		Siehe 3.1.3.	
	3.1.6 Aussprache			
3.2 Vorher-sehbar	3.2.1 Bei Fokus	Ja	/	A
	3.2.2 Bei Eingabe	Ja	/	A
	3.2.3 Konsistente Navigation	Ja	(A): Dynamisches Austauschen der Komponenten. (A + D): Durch die Template-Systeme und die Wiederverwendbarkeit der Komponenten unterstützt.	AA
	3.2.4 Konsistente Erkennung	Ja	Siehe 3.2.3	AA
	3.2.5 Änderung der Anfrage	Ja	/	AAA
3.3 Hilfe-stellung bei der Eingabe	3.3.1 Fehlererkennung	Ja	(A + D): Durch Validatoren erkannte Fehler in den Formularen werden um Fehlermeldungen ergänzt. (AM + B): 'Form'-Komponenten unterstützen die Anzeige von Fehlermeldungen.	A
	3.3.2 Beschriftungen oder Anweisungen	Ja	(AM + B): 'Form'-Komponenten unterstützen die Anzeige von Hinweistexten. (D): Label werden für Formulare automatisch erzeugt.	A
	3.3.3 Fehlerüberprüfung	Ja	(AM + D): Autovervollständigung hilft bei der korrekten Eingabe. Weiteres: Siehe 3.3.1 und 3.3.2.	AA
	3.3.4 Fehlervermeidung (rechtliche, finanzielle, Daten)	Ja	(A + D): Validatoren prüfen Eingaben. (AM + B): 'Stepper' bzw. 'Progress'-Komponente enthält in der Buchungsabwicklung einen Schritt zur Überprüfung und Bestätigung der Angaben.	AA
	3.3.5 Hilfe	Ja	(AM + D): Autovervollständigung für Formulare, Hilfertexte für Eingabefelder und Validierung der Formate.	AAA
	3.3.6 Fehlervermeidung (alle)	Ja	Siehe 3.3.4.	AAA

Abkürzungen: A=Angular; AM=Angular Material; B=Bootstrap; D=Django

KAPITEL 5. EVALUIERUNG

Tabelle 7: WCAG Evaluierung - 'Prinzip 4: Robust'

Richtlinie	Erfolgskriterien	Erfüllt?	Bezug zu den Frameworks / Anmerkungen	Stufe
4.1 Kompatibel	4.1.1 Syntax-analyse	Ja	/	A
	4.1.2 Name, Rolle, Wert	Ja	(AM): ARIA-Attribute werden teilweise automatisch generiert. (AM + B): Zusätzliche ARIA-Attribute wurden manuell hinzugefügt.	A
	4.1.3 Status-meldungen	Ja	(AM): 'Dialog'- und 'Snackbar'-Komponente erhalten automatisch die korrekte Rolle. (B): 'Modal' erhält Rolle von Bootstrap. Die für 'Alerts' wurden manuell hinzugefügt.	AA

Abkürzungen: A=Angular; AM=Angular Material; B=Bootstrap; D=Django

Kapitel 6

Ausblick und Fazit

Menschen mit Behinderungen treffen regelmäßig auf Barrieren, die ihnen teilweise die uneingeschränkte Teilnahme am alltäglichen Leben in der Gesellschaft verwehren. Digitale Technologien, wie das Internet, haben das Potenzial mehr Inklusion zu ermöglichen, sind jedoch oft selbst Teil des Problems.

Trotzdem lassen sich in jüngster Vergangenheit in vielen Bereichen Bemühungen feststellen die versuchen das zu verhindern. Ein Beispiel hierfür Abseits vom Web sind Videospiele, bei denen zunehmend mehr Einstellungen für die Barrierefreiheit vorzufinden sind. Das im Juni 2020 erschienene „The Last of Us Part 2“ weist einen Umfang an Optionen für die Barrierefreiheit auf, der zuvor in einem Videospiel dieser Größe noch nicht vorzufinden war [Nathoo, 2020].

HTML bietet zahlreiche, durchaus weitreichende Möglichkeiten Barrierefreiheit zu unterstützen. Im Rahmen dieser Arbeit lag der Fokus auf modernen Web-Frameworks. Insbesondere Bootstrap, Django und Angular wurden genauer betrachtet. Wie die Analyse zeigte, ersetzen diese Technologien nicht die Funktionen in HTML, sondern können vielmehr dabei helfen, diese umzusetzen. Vor allem in Bootstrap und Angular sind gute Ansätze zu erkennen, auch wenn noch an manchen Stellen ungenutztes Potenzial vorhanden ist. Da es sich bei Django um ein Backend-Framework handelt und dieses somit weniger mit der Interaktion des Nutzers zu tun hat, hat es zwar weniger Möglichkeiten, dennoch ist dort noch mehr erreichbar.

Generell werden Web-Frameworks vermutlich niemals vollständig Barrierefreiheit garantieren können. Zu vieles hängt vom konkreten Anwendungsfall ab und liegt somit in der Verantwortung der Entwickler.

Weitere Technologien, die sich für zukünftige Forschung anbieten, sind Content-Management- (CMS) und Lernmanagement-Systeme (LMS). Letztere haben mit ihrem Einsatz bei Schulen und Universitäten einen eindeutigen Grund, Barrierefreiheit nicht zu ignorieren. Universitäten sind dazu verpflichtet, ihre Webauftritte barrierefrei zugänglich zu machen. Die Otto-Friedrich-Universität Bamberg macht in einer Erklärung deutlich, dass manche technischen Unzulänglichkeiten durch die Hersteller der zugrunde liegenden Software-Produkte bedingt sind [Universität Bamberg, 2019]. Dennoch stellte Armin Odenkirchen bereits 2011 in seiner Bachelorarbeit fest, dass die Webseite der Universität anscheinend frei von gravierenden Barrieren ist. Größere Mängel konnten jedoch beim

KAPITEL 6. AUSBLICK UND FAZIT

Virtuellen Campus festgestellt werden. Eine Analyse zum aktuellen Stand könnte von Interesse sein.

Spannend zu beobachten sein wird auch, welchen Einfluss aktuelle Trends wie das 'Internet-of-Things', 'Virtual Reality' und Sprachassistenten wie 'Amazon Echo' in Zukunft auf den Bereich der Barrierefreiheit haben werden. Es werden mit Sicherheit neue Möglichkeiten geschaffen, Barrieren zu überwinden, aber auch neue Herausforderungen entstehen.

Schon zu Beginn der Entwicklung neuer Produkte ist es wesentlich, ein grundlegendes Verständnis über diejenigen zu haben, die es letztendlich verwenden werden. Denn was für die eine Person nur störend ist, sorgt bei der anderen vielleicht dafür, dass sie das Produkt nicht benutzen kann.

Literaturverzeichnis

- [3PlayMedia, 2014] 3PlayMedia: *Case Study: This American Life*. 2014. <https://www.3playmedia.com/customers/case-studies/this-american-life/>, Abruf: 14.05.2020
- [ActiveAdmin, o.D.] ActiveAdmin. (o.D). <https://activeadmin.info/>, Abruf: 07.08.2020
- [Angular, o.D.a] Angular. (o.D). <https://angular.io/>, Abruf: 02.08.2020
- [Angular, o.D.b] Angular: *Guide: Accessibility in Angular*. (o.D). <https://angular.io/guide/accessibility>, Abruf: 02.08.2020
- [Angular, o.D.c] Angular: *Documentation: @angular/animations trigger-Function*. (o.D). <https://angular.io/api/animations/trigger>, Abruf: 02.08.2020
- [Angular Material, o.D.a] Angular Material. (o.D). <https://material.angular.io/>, Abruf: 02.08.2020
- [Angular Material, o.D.b] Angular Material: *API reference for Angular CDK a11y*. (o.D). <https://material.angular.io/cdk/a11y/api>, Abruf: 02.08.2020
- [Bootstrap, 2020a] Bootstrap: *Documentation: About*. 2020. <https://getbootstrap.com/docs/4.5/about/overview/>, Abruf: 29.07.2020
- [Bootstrap, 2020b] Bootstrap: *Documentation: Accessibility*. 2020. <https://getbootstrap.com/docs/4.5/getting-started/accessibility/>, Abruf: 29.07.2020
- [Bootstrap, 2020c] Bootstrap: *Documentation: Layout*. 2020. <https://getbootstrap.com/docs/4.5/layout/>, Abruf: 29.07.2020
- [Bootstrap, 2020d] Bootstrap: *Documentation: Colors*. 2020. <https://getbootstrap.com/docs/4.5/utilities/colors/>, Abruf: 29.07.2020
- [Bootstrap, 2020e] Bootstrap: *Documentation: Breadcrumbs*. 2020. <https://getbootstrap.com/docs/4.5/components/breadcrumb/>, Abruf: 29.07.2020
- [Bootstrap Blog, 2020] Bootstrap Blog: *Bootstrap 5 alpha*. 2020. <https://blog.getbootstrap.com/2020/06/16/bootstrap-5-alpha/>, Abruf: 10.08.2020
- [Codelyzer, 2017] Codelyzer: *Codelyzer Core Rules*. 2017. <http://codelyzer.com/rules/>, Abruf: 29.07.2020
- [Cooper, 2016] Cooper, M.: *WAI-ARIA Overview*. 2016. <https://www.w3.org/WAI/standards-guidelines/aria/>, Abruf: 24.05.2020

LITERATURVERZEICHNIS

- [Django, 2020a] Django Software Foundation: *Documentation: Templates*. 2020. <https://docs.djangoproject.com/en/3.0/topics/templates/>, Abruf: 05.08.2020
- [Django, 2020b] Django Software Foundation: *Documentation: django.contrib.humanize*. 2020. <https://docs.djangoproject.com/en/3.0/ref/contrib/humanize/>, Abruf: 05.08.2020
- [Django, 2020c] Django Software Foundation: *Documentation: Internationalization and localization*. 2020. <https://docs.djangoproject.com/en/3.0/topics/i18n/>, Abruf: 05.08.2020
- [Django, 2020d] Django Software Foundation: *Documentation: The Django admin site*. 2020. <https://docs.djangoproject.com/en/3.0/ref/contrib/admin/>, Abruf: 05.08.2020
- [Django, 2020e] Django Software Foundation: *Documentation: Working with forms*. 2020. <https://docs.djangoproject.com/en/3.0/topics/forms/>, Abruf: 05.08.2020
- [Django, 2020f] Django Software Foundation: *Documentation: PostgreSQL specific lookups*. 2020. <https://docs.djangoproject.com/en/3.1/ref/contrib/postgres/lookups/>, Abruf: 05.08.2020
- [Django, 2020g] Django Software Foundation: *Documentation: Making queries*. 2020. <https://docs.djangoproject.com/en/3.0/topics/db/queries/>, Abruf: 05.08.2020
- [Dorsey et al., 2018] Dorsey, E. Ray et al.: *Global, regional, and national burden of Parkinson's disease, 1990–2016: a systematic analysis for the Global Burden of Disease Study 2016*. 2018. The Lancet Neurology, Volume 17, Issue 11, 939 - 953. DOI:[https://doi.org/10.1016/S1474-4422\(18\)30295-3](https://doi.org/10.1016/S1474-4422(18)30295-3)
- [Firth, 2019] Firth, A.: *Practical Web Inclusion and Accessibility: A Comprehensive Guide to Access Needs*. 2019. ISBN:978-1-4842-5451-6
- [Foundation, o.D.] Foundation: *Foundation: Accessibility*. (o.D.). <https://get.foundation/sites/docs/accessibility.html>, Abruf: 05.08.2020
- [JHM, 2019] Johns Hopkins Medicine: *Mental Health Disorder Statistics*. 2019. <https://www.hopkinsmedicine.org/health/wellness-and-prevention/mental-health-disorder-statistics>, Abruf: 15.05.2020
- [Kearney et al., 2019] Kearney M., Gash D., Dodson R.: *Web Fundamentals: Using tabindex*. 2019. <https://developers.google.com/web/fundamentals/accessibility/focus/using-tabindex>, Abruf: 24.05.2020
- [Locke, 2020] Locke, H.: *Why ‘dark mode’ causes more accessibility issues than it solves*. 2020. <https://levelup.gitconnected.com/why-dark-mode-causes-more-accessibility-issues-than-it-solves-d2f8359bb46a>, Abruf: 27.06.2020
- [Mohammed et al., 2019] Mohammed Z. K., Gechev M.: *Audit your Angular app’s accessibility with codelyzer*. 2019. <https://web.dev/accessible-angular-with-codelyzer/>, Abruf: 29.07.2020

LITERATURVERZEICHNIS

- [Nathoo, 2020] Nathoo, Z.: *Last of Us 2 has 'changed the game' for accessibility.* 2020. <https://www.cbc.ca/news/entertainment/video-game-last-of-us-2-1.5613572>, Abruf: 06.07.2020
- [PUXL, o.D.] PUXL Framework. <https://get.foundation/sites/docs/accessibility.html>, Abruf: 29.07.2020
- [PLAIN, 2011] Plain Language Action and Information Network: *Federal Plain Language Guidelines.* 2011.
- [RKI, 2017] Robert Koch Institut: *GBE-Themenheft: Blindheit und Sehbehinderung.* 2017. ISBN: 978-3-89606-233-8. DOI: 10.17886
- [Ruby on Rails, 2020] Ruby on Rails API. <https://api.rubyonrails.org/>. 2020, Abruf: 07.08.2020
- [UN-Vollversammlung, 1948] UN-Vollversammlung: *Allgemeine Erklärung der Menschenrechte.* (217 [III] A). 1948. Paris.
- [UNHRC, 2016] United Nations Human Rights Council: *Thirty-second session: The promotion, protection and enjoyment of human rights on the Internet.* (A/HRC/32/L.20). 2016. Genf.
- [Universität Bamberg, 2019] Otto-Friedrich-Universität Bamberg: *Allgemeine Erklärung zur Barrierefreiheit.* 2019. <https://www.uni-bamberg.de/rz/verfahrensweisen/barrierefreiheit-erklaerung/>, Abruf: 06.07.2020
- [W3C, 1998] World Wide Web Consortium (W3C): *Level 1 Document Object Model Specification.* 1998. <https://www.w3.org/TR/WD-DOM/cover.html>, Abruf: 16.08.2020
- [W3C, 2009] World Wide Web Consortium (W3C): *Web Content Accessibility Guidelines 2.0.* 2009. <https://www.w3.org/Translations/WCAG20-de/>, Abruf: 15.08.2020
- [W3C, 2010] World Wide Web Consortium (W3C): *Web Style Sheets CSS tips & tricks.* 2010. <https://www.w3.org/Style/Examples/007/units.en tmpl>, Abruf: 16.08.2020
- [W3C, 2015] World Wide Web Consortium (W3C): *Multiple Ways.* 2015. <https://www.w3.org/WAI/tutorials/menus/multiple-ways/>, Abruf: 05.07.2020
- [W3C, 2018] World Wide Web Consortium (W3C): *Web Content Accessibility Guidelines 2.1.* 2018. <https://www.w3.org/TR/2018/REC-WCAG21-20180605/>, Abruf: 25.07.2020
- [W3C, 2019] World Wide Web Consortium (W3C): *WAI-ARIA Authoring Practices 1.1.* 2019. <https://www.w3.org/TR/wai-aria-practices/>, Abruf: 05.08.2020
- [WebAIM, 2013] WebAIM: *Visual Disabilities: Low Vision.* 2013. <https://webaim.org/articles/visual/lowvision>, Abruf: 14.05.2020
- [WebAIM, 2014] WebAIM: *Motor Disabilities.* 2014. <https://webaim.org/articles/motor/>, Abruf: 14.05.2020
- [WebAIM, 2017a] WebAIM: *Screen Reader User Survey #7 Results.* 2017. <https://webaim.org/projects/screenreadersurvey7/#impacts>, Abruf: 14.05.2020

LITERATURVERZEICHNIS

- [WebAIM, 2017b] WebAIM: *Designing for Screen Reader Compatibility*. 2017. <https://webaim.org/techniques/screenreader/>, Abruf: 22.07.2020
- [WebAIM, 2017c] WebAIM: *Seizure Disorders*. 2017. <https://webaim.org/articles/seizure/>, Abruf: 05.08.2020
- [WebAIM, 2018b] WebAIM: *Cognitive Disabilities: Introduction*. 2018. <https://webaim.org/articles/cognitive/>, Abruf: 24.05.2020
- [WebAIM, 2020] WebAIM: *The WebAIM Million*. 2020. <https://webaim.org/projects/million/>, Abruf: 13.05.2020
- [Vigo et al., 2013] Vigo M., Brown J., Conway V.: *Benchmarking Web Accessibility Evaluation Tools: Measuring the Harm of Sole Reliance on Automated Tests*. 2013. Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility. DOI: 10.1145/2461121.2461124
- [Vue.js, o.D.] Vue.js. (o.D.). <https://vuejs.org/>, Abruf: 02.08.2020
- [Techvisionblog, 2015] Techvisionblog: *WAIable 1.0.0 released! Now you can make your Rails forms accessible*. 2015. <https://accessiblerails.wordpress.com/>, Abruf: 07.08.2020
- [WHO, 2011] World Health Organisation: *World Report on Disability*. 2011. ISBN:978-92-4-156418-2
- [WHO, 2019a] World Health Organisation: *Dementia*. 2019. <https://www.who.int/news-room/fact-sheets/detail/dementia>, Abruf: 24.04.2020
- [WHO, 2019b] World Health Organisation: *World Report on vision*. 2019. ISBN:978-92-4-151657-0
- [WHO, 2020] World Health Organisation: *Deafness and hearing loss*. 2020. [who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss](https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss), Abruf: 24.05.2020
- [Yesilada et al., 2019] Yesilada Y., Harper S.: *Web Accessibility: A Foundation for Research (Second Edition)*. 2019. ISBN:978-1-4471-7440-0

Erklärung

Ich erkläre hiermit gemäß § 17 Abs. 2 APO, dass ich die vorstehende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

19.08.2020

Datum

Leonard Siegmayer

Unterschrift