FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Fachhochschule Aachen, Campus Jülich

Fachbereich 9

Medizintechnik und Technomathematik

Seminararbeit

im Studiengang Angewandte Mathematik und Informatik

Entwicklung und Evaluation eines Prototyps zur Umsetzung von Barrierefreiheitsprüfungen für Webseiten gemäß der Web Accessibility Directive (EU)

Autor:	Josha Bartsch
Matrikelnummer:	3529261
1. Prüfer:	Prof. Dr. rer. nat. Alexander Voss
2. Prüfer, Betreuer:	B. Sc. Tim Schroeder

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die Seminararbeit mit dem Thema

Entwicklung und Evaluation eines Prototyps zur Umsetzung von Barrierefreiheitsprüfungen für Webseiten gemäß der Web Accessibility Directive (EU)

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, alle Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, kenntlich gemacht sind und die Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.
Ich verpflichte mich, ein Exemplar der Seminararbeit fünf Jahre aufzubewahren und auf Verlangen dem Prüfungsamt des Fachbereiches Medizintechnik und Technomathematik auszuhändigen.

Name: Josha Bartsch

Aachen, den 15. Dezember 2023

Unterschrift

Inhaltsverzeichnis

Glossar 2
1. Einleitung 4
2. Grundlagen 5
2.1. BITV als Konsequenz der Web Accessibility Directive 5
2.2. Evaluation der Barrierefreiheit für Webseiten 6
2.3. Werkzeuge zur Bewertung der Barrierefreiheit von Webseiten 9
3. Vorstellung des Prototyps 11
3.1. Vorstellung von AXE – The Accessibility Engine 11
3.2. Konzeption und Implementierung 13
3.3. Bewertung 19
3.4. Präsentation der Anwendung, Fallbeispiele 21
3.4.1. Chat-Support-Icon 21
3.4.2. Die Navigationsleiste 24
3.4.3. Mehrfach vergebene IDs 26
4. Fazit 27
5. Ausblick 28
Bibliographie 29

Glossar

W3C World Wide Web Consortium

WCAG Web Content Accessibility Guidelines, Richtlinien zur Barrierefreiheit von Inhalten im Internet, formuliert durch das W3C 5, 10, 11, 12, 27

NGOs Non Governmental Organizations, Nicht-Regierungs-Organisationen

BITV Barrierefreie-Informationstechnik-Verordnung 4, 5, 6, 10, 27

RWTH RWTH Aachen University 4

EN Harmonisierte Normen der EU, formuliert mit dem Ziel eine einheitliche Referenz für den Eurpäische Union und alle Mitgliedsstaaten zu bilden 5

WAVE (Eigenname) WAVE Web Accessibility Evaluation Tools 14

AXE (Eigenname) Accessibility Engine 11, 14, 15, 27

User-Agent Technischer Überbegriff für Web-Browser und funktional vergleichbare Programme 3, 16, 20

SVG Scalable Vector Graphics 22, 26

IDs Eineindeutige Merkmale 16, 26

Grafana Open-Source Plattform zur Visualisierung von Daten 19, 21

DBMS Datenbank Management System 16

MongoDB (Eigenname) Ein NoSQL Datenbank Management System 16, 17, 18, 19

gerendert Interpretation von Quellcode zur (in der Regel visuellen) Aufbereitung, hier durch Browser für Webseiten 7, 9, 10, 13, 24

Screen-Reader Programme die den Bildschirminhalt interpretieren und in geeigneter Form (zum Beispiel ausgesprochen oder mittels eines Braille Geräts) wiedergeben 7, 8

Community Lose Gemeinschaft interessierter Individuen und Institutionen, im Kontext von Open-Source Projekten 11, 17

Selenium (Eigenname) Ein Open-Source Browser-Automatisierungs-Framework 13, 14, 15

WebDriver Ein plattform- und browserunabhängiges Protokoll Steuerungsprotokoll für User-Agents 13, 14, 19, 20, 28

Executor die Anweisungen ausführende Selenium- und Browser-Instanz, im Kontext des Prototyps 3, 15

Consumer Komponente die mittels eines Executor Evaluationen durchführt 15

Celery (Eigenname) Python-Framework für die asynchrone, verteilte Ausführung von Aufgaben 15

MQTT Ein Publish-Subscribe basiertes Messaging Protokoll 15

Playwright (Eigenname) Ein Open-Source Browser-Automatisierungs-Framework 19

WebCrawler Ein Programm, das systematisch Webseiten traversiert, um alle erreichbaren URLs zu erfassen 3, 16, 17, 20, 28

Scrapy (Eigenname) Ein WebCrawler 16

1. Einleitung

Die Novelle der BITV 2.0 von 2019 betrifft alle öffentlichen Stellen des Bundes. Es handelt sich dabei um die erforderliche Umsetzung der bereits 2016 beschlossenen EU-Richtlinie 2016/2102. Die Novelle führte unter anderem an, dass für alle Webseiten und mobilen Anwendungen eine barrierefreie Nutzung sichergestellt sowie der aktuelle Grad der Barrierefreiheit erfasst und dokumentiert werden muss.

Die RWTH Aachen University ist eine solche öffentliche Stelle. Besonders durch die zentrale Hochschulverwaltung werden bereits Anstrengungen unternommen, die Barrierefreiheit der von ihr zentral verwalteten Webseiten zu prüfen und entsprechende Erklärungen auszustellen.

Das IT Center der Hochschule bietet den Hochschulangehörigen eine Vielzahl an Dienstleistungen an, welche in der Regel in Form einer Webseite abrufbar sind. Als Mitarbeiter des IT Centers ist mir klar, dass eine Prüfung aller angebotenen Webseiten eine nicht unerhebliche Menge an Zeit und Ressourcen binden wird. Dies ist zum einen aufgrund der Anzahl der Webseiten zu erwarten, zum anderen weisen einige Webseiten eine hohe Dynamik an Inhalten oder komplexen, zugrundeliegenden Dienstleistungen auf.

Diese Arbeit diskutiert die für eine Barrierefreiheitsprüfung relevanten Aspekte, welche sich aus der BITV ergeben. Im Weiteren wird ein Prototyp implementiert und bewertet, der eine automatische Evaluation von Webseiten hinsichtlich ihrer Barrierefreiheit durchführen und dadurch die Ausstellung der Barrierefreiheitserklärung unterstützen soll.

Anschließend wird anhand einer Webseite des IT Centers demonstriert, wie durch den Prototyp gewonnene Informationen genutzt werden können, um einige mögliche Probleme hinsichtlich der Barrierefreiheit zu diskutieren und nachhaltig zu verbessern.

2. Grundlagen

2.1. BITV als Konsequenz der Web Accessibility Directive

Als das Europäische Parlament und der Rat 2016 die `Web Accessibility Directive` beschlossen, haben sie damit alle Mitgliedsstaaten dazu verpflichtet, die Richtlinie innerhalb von 2 Jahren in nationales Recht zu überführen. Dieser Verpflichtung entsprach Deutschland 2018 in Form von Änderungen des Behindertengleichstellungsgesetzes, sowie nachfolgend der erneuerten Barrierefreie-Informationstechnik-Verordnung 2.0 in 2019.

Die BITV hat zum Ziel, elektronisch bereitgestellte Informationen und Dienstleistungen öffentlicher Stellen für Menschen mit Behinderungen zugänglich und nutzbar zu gestalten. Zur Erfüllung der Ziele wird auf harmonisierte Normen der EU sowie „den Stand der Technik“ verwiesen, im Kontext der Barrierefreiheit von Webseiten nimmt die EN 301 549 [1] eine zentrale Rolle ein.

Die aufgeschlüsselten Ziele der BITV stellen eine direkte Referenz auf die WCAG dar: Die BITV nennt Wahrnehmbarkeit, Bedienbarkeit, Verständlichkeit und Robustheit als erforderliche Ziele (vgl. § 3 BITV 2.0 [2]), hierbei handelt es sich konkret um die durch die WCAG aufgestellten Prinzipien. Zudem führt die EN 301 549 die WCAG 2.0 und seit der Neuerung 2018 auch die WCAG 2.1 als normative Referenzen auf.

Die WCAG unterscheiden zwischen den drei Erfolgsstufen A, AA und AAA. Jede Stufe geht mit einer zusätzlichen Menge an Anforderungen einher, wobei Stufe A als das zu erreichende Minimum an Konformität betrachtet wird. Im Gegensatz dazu definiert Stufe AAA Anforderungen, die nicht als zwingende Vorgaben gelten, sondern zusätzliche Maßnahmen für eine umfassendere Barrierefreiheit darstellen. Die EN 301 549 folgt diesen Empfehlungen und benennt speziell Anforderungen der WCAG 2.1 AA als zu erreichende Konformität (Vergleich [1] Ziffer 9.0).

Die BITV fordert zudem, dass eine „Erklärung zur Barrierefreiheit“ für jede Webseite veröffentlicht werden muss. Diese führt etwaige nicht-barrierefreie Inhalte auf, sowie eine Kontaktmöglichkeit, um erkannte Probleme zu melden [3].

2.2. Evaluation der Barrierefreiheit für Webseiten

Die Barrierefreiheitserklärung spielt eine zentrale Rolle für alle Webseiten, die der BITV entsprechen müssen. Denn mit dieser kommuniziert der Anbieter, in welchem Maß die Webseite barrierefrei ist und wo Einschränkungen erkannt wurden.

Moderne Webseiten können aus einer Vielzahl unterschiedlicher Inhalte zusammengesetzt werden. Je nach Kontext haben diese jedoch unterschiedliche Stellenwerte in Bezug auf den Hauptinhalt der Webseite. So ist ein dekoratives Hintergrundbild zum Beispiel von geringer Relevanz für eine Webseite, die über aktuelle Entwicklungen informiert.

Um eine Barrierefreiheitserklärung zu formulieren, sind die folgenden Aspekte von zentraler Bedeutung.

Welche Art von Inhalten wird bereitgestellt und welche Rolle nehmen diese auf der Webseite ein?

- Texte
- Bilder
- Videos
- Dokumente wie PDFs
- Sprachbeiträge (Audio)

Die Gestaltung der Startseite ist besonders prekär. Hier muss ein besonderes Augenmerk auf der Zugänglichkeit liegen, denn ausgehend von der Startseite muss jeder Besucher in der Lage sein, mindestens bis zu der Barrierefreiheitserklärung zu navigieren.

Dort wird zusammengefasst, für welche Abschnitte oder Inhalte der Webseite Einschränkungen bekannt sind und wie neu erkannte, nicht benannte Einschränkungen gemeldet werden können.

Solche Einschränkungen können zum Beispiel vorliegen, wenn für eingebettete Video- oder Audioinhalte keine Transkripte zur Verfügung stehen oder die Steuerung derselben nicht per Tastatur zu bedienen ist.

Weitere typische Einschränkungen hängen mit Farbkontrasten zusammen: Diese müssen mindestens einen gewissen Schwellenwert erreichen, damit die Farbgestaltung als barrierefrei verstanden wird. Auch Farben, die im Zusammenhang mit Sehschwächen stehen, sollten nicht kombiniert verwendet werden. Die Prüfung aller verwendeten Farbkontraste ist mitunter sehr arbeitsaufwändig, wenn nicht direkt im Designprozess der Webseite ein Augenmerk auf diesen Aspekt gelegt wird.

Die Barrierefreiheit von Bildern ist ein Aspekt, der recht eindeutig ist. Wann immer ein Bild eingebunden wird, muss entschieden werden, ob es dekorativ oder Teil der wesentlichen Information ist. Unwesentliche Bilder sollten als solche markiert werden, indem ein leerer Alternativtext hinterlegt ist. Informative Bilder können die relevante Beschreibung ebenfalls als Alternativtext führen. Sollte jedoch bereits im begleitenden Text eine Beschreibung des Bildinhalts vorliegen, so soll das Bildelement auf den betreffenden Textabschnitt verweisen und doppelte Informationen vermeiden.

Die erläuterten Aspekte sind vor allem makroskopischer Natur in dem Sinne, dass sie alle durch Beobachtungen der gerenderten Webseite evaluiert werden können. Die Frage, ob Barrierefreiheit unter Berücksichtigung dieser Aspekte gegeben ist, kann mit etwas Zeit und einem herkömmlichen Browser beantwortet werden, ohne dass ein tieferes Verständnis für den HTML-Quellcode benötigt wird.

Die gängige Praxis an vielen öffentlichen Institutionen sieht aktuell so aus, dass anhand einer Checkliste über diese und weitere makroskopische Aspekte eine manuelle Evaluation aller Webseiten erfolgen sollte. Auf Basis einer solchen Checkliste wird dann die Barrierefreiheitserklärung ausformuliert.

Screen-Reader ermöglichen Menschen mit verschiedenen Einschränkungen Zugang zu einem Großteil der verfügbaren Inhalte. Für einen Menschen ohne die spezifischen Einschränkungen ist es allerdings schwierig, sich in die Lage von jemanden zu versetzen, der auf technische Hilfsmittel angewiesen ist, um durch Webseiten zu navigieren. Einen Einblick kann man sich verschaffen, in dem man zum Beispiel auf die Nutzung der Computermaus verzichtet. Die Navigation mittels Tastatur gibt einen Eindruck davon, wie die Seite selbst strukturiert ist.

Der nächste Schritt ist dann die Installation eines Screen-Readers und das Schließen der Augen. Wenn man nun durch die Webseite navigiert und sich die Bildschirminhalte vorlesen lässt, wird klar, wie verschiedene Elemente in natürlicher Sprache dargestellt werden: Elemente, die viele Nutzer sonst nur visuell ausmachen – wie Bilder, Knöpfe oder Hyperlinks.

Einige Elemente und Schaltflächen lassen sich durch wiederholtes Drücken der `TAB`-Taste ansteuern. Der Programmierer der Webseite kann hierbei bewusst Einfluss auf die Auswahl der Elemente und die Reihenfolge nehmen, mit der der Nutzer durch die Seite geführt wird. Alternative Eingabegeräte oder Assistenzprogramme folgend dieser Struktur ebenfalls.

2.3. Werkzeuge zur Bewertung der Barrierefreiheit von Webseiten

Neben der manuellen Evaluation gibt es Werkzeuge, die in unterschiedlichem Maße automatisch agieren. Einige Webseiten bieten zum Beispiel an, Farbkontraste zu überprüfen. Erst müssen hier die in Frage kommenden Farbwerte ermittelt und übertragen werden, bevor eine Rückmeldung über die Unbedenklichkeit des Kontrastes gegeben wird.

Eine andere Kategorie dieser Werkzeuge bilden Browser-Erweiterungen. Mit einer Browser-Erweiterung wird zum Beispiel die Prüfung von Kontrasten vereinfacht: Der Nutzer ruft lediglich die fragliche Webseite auf und aktiviert die Erweiterung. Diese ist dann in der Lage, in der gerenderten Ansicht des Nutzers alle Farbwerte zu erkennen und relevante (benachbarte) Farb-Paare auf ihren Kontrast prüfen.

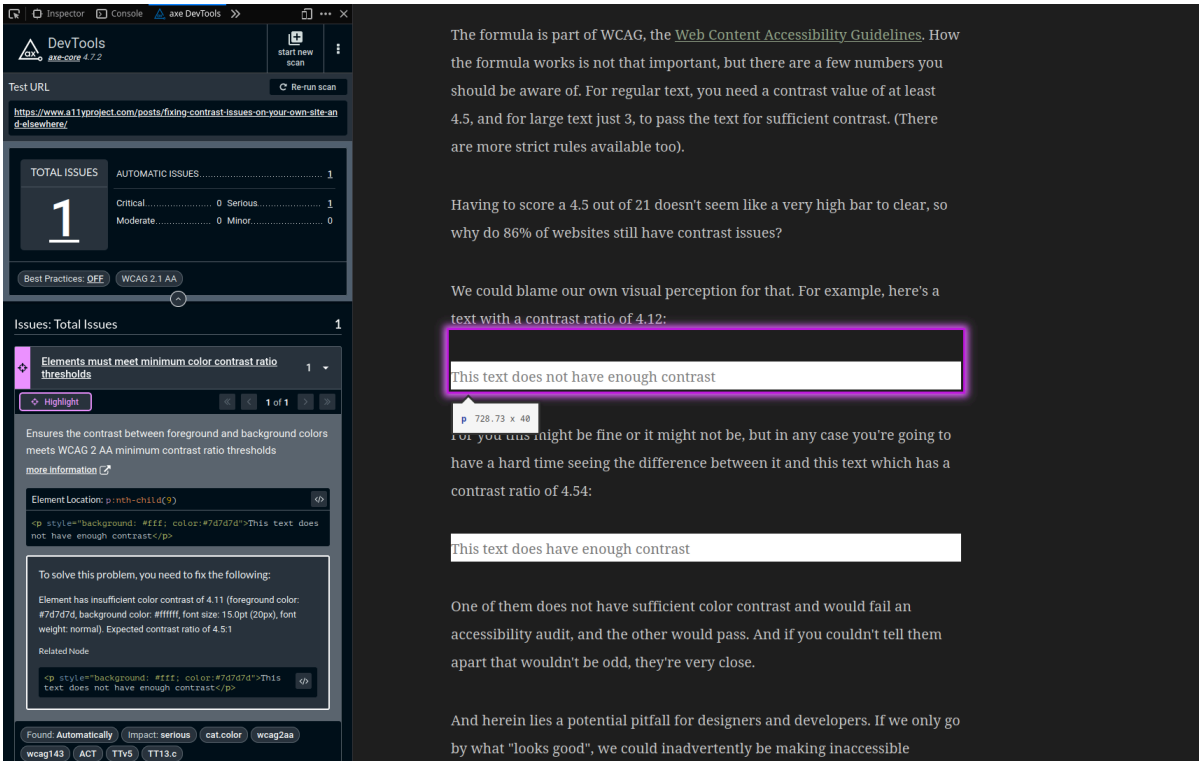

Abbildung 1: Die Browser Erweiterung AXE DevTools demonstriert die Erkennung eines problematischen Kontrastes (anhand von Webseite [4])

Die enge Integration des evaluierenden Werkzeugs in die Ansicht der testenden Person erlaubt auch die Prüfung nach komplexeren Regeln. Solche Regeln werden unter anderem auf Basis der HTML-Spezifikation oder eben der WCAG formuliert und ermöglichen eine technische Problemsuche, unabhängig vom Wissensstand des Nutzers.

In der BITV und WCAG werden mehr und weniger konkrete Forderungen an die Gestaltung von HTML-Dokumenten aufgestellt. Diese Anforderungen beziehen sich auf die in der HTML-Spezifikation beschriebenen Elemente, gehen darüber hinaus und geben vor, wie diese in unterschiedlichen Zusammenhängen zu verwenden oder zu gestalten sind.

In der HTML-Spezifikation werden bereits Aspekte der Barrierefreiheit diskutiert, insbesondere die Überführung des Dokuments in eine Struktur, die von assistierenden Programmen genutzt wird. Wenn auch nicht alle Browser die gesamte Spezifikation umsetzen, so bildet sie aber dennoch eine Zielvorstellung, an der sich die meisten Akteure orientieren.

Wenn bei der Gestaltung eines Dokuments gegen Vorgaben der Spezifikation verstoßen wird, hat dies nicht immer direkt sichtbare Auswirkungen. Browser sind weitgehend resilient gegen solche Fehler und in aller Regel rendern sie das Dokument trotz erkannter Fehler, zum beispiel indem defekte Elemente oder Details einfach ausgelassen werden. All das fällt während der Gestaltung der Webseite nicht unbedingt auf.

Für assistierende Programme hingegen sind solche Fehler und Verstöße Stolpersteine. Denn der Nutzer kann eventuell nicht visuell erkennen, worum es geht, wenn das Element fehlerhafte Angaben macht.

3. Vorstellung des Prototyps

3.1. Vorstellung von AXE – The Accessibility Engine

Aus den beschriebenen Grundlagen lässt sich schließen, dass eine manuelle Evaluation immer die Ausgangsbasis für barrierefreie Webseiten bildet. In ihrem Verlauf wird das Konzept der Webseite betrachtet, grundsätzliche Einschränkungen und notwendige organisatorische Konsequenzen werden identifiziert.

Als Beispiel sei hier Folgendes anzuführen: Eine Webseite, die Live-Video-Übertragungen anbietet, wird schnell feststellen, dass diese Übertragungen nicht ohne weiteres barrierefrei sind. In der Konsequenz sollte geprüft werden, ob Maßnahmen nach WCAG AAA wie Live-Dolmetscher für Gebärdensprache möglich sind. Stufe A wäre bereits erreicht, wenn Transkripte zu einem Zeitpunkt nach der Ausstrahlung bereitgestellt würden. Die Stufe AA der WCAG würde erfordern, dass bereits während der Live-Übertragung Untertitel verfügbar sind.

Während diese Abwägungen unbedingt notwendig sind, erfassen sie nur sehr oberflächlich die technischen Einschränkungen, die sich für Nutzer von assistierenden Programmen ergeben.

Wie zuvor umrissen, kann diese Lücke durch den Einsatz von im Browser integrierten Werkzeugen geschlossen werden.

AXE – `The Accessibility Engine` ist ein solches Werkzeug. Hierbei handelt es sich zunächst, wie der Name vermuten lässt, um eine Engine, beziehungsweise ein Framework - also ein Programm, das die relevante Basisfunktion ausführt, aber diese nur mittels Schnittstellen bereitstellt. Eine Schnittstelle zum Nutzer bildet zum Beispiel die Browser-Erweiterung `AXE DevTools`, welche vor allem die Oberfläche umsetzt, mit der Nutzer die Evaluation starten und die Ergebnisse einsehen können.

Das AXE Framework wird als Open-Source-Projekt von einer Gesellschaft (der `Deque Systems, Inc.`) gemeinsam mit einer losen Gemeinschaft interessierter Individuen und Institutionen (Community) betreut und weiterentwickelt. Durch den Open-Source-Ansatz und die gewählte Lizenz (Mozilla Public License 2.0) steht das Framework jedem zur freien Verfügung und der zugrunde liegende Regelsatz ist für Entwickler transparent, sowie jederzeit erweiterbar.

Das Framework beziehungsweise die Engine nimmt den Quellcode einer Webseite entgegen und führt systematisch Testfälle aus. Die Testfälle sind Interpretationen der WCAG und anderer Richtlinien. Während das Framework keine vollständige Abdeckung postuliert, ist klar dokumentiert, welche Testfälle bisher implementiert sind und welchen Richtlinien diese entsprechen.

Durch Beiträge unterschiedlich motivierter Entwickler sind neben den verschiedenen Richtlinien auch Testfälle für allgemeine `Best Practices` in das Framework eingeflossen. Unter diese Bezeichnung fallen Regeln, die nicht auf einzelne Richtlinien zurückzuführen sind, aber deren Grundlagen in der Praxis so weite Verbreitung finden, dass Abweichungen negativ auffallen würden.

3.2. Konzeption und Implementierung

Immer mehr Webseiten setzen auf dynamische Inhalte – also zum Beispiel Teile der Webseite, die durch JavaScript manipuliert oder verzögert gerendert werden. Unter diesem Gesichtspunkt sind statische Analysen des Quellcodes der Webseiten nicht sinnvoll und für die Prüfung sollte auf etablierte Browser zurückgegriffen werden. Zwar wird eine solche Prüfung immer auf Basis des HTML-Dokuments beziehungsweise Quellcodes erfolgen, allerdings ist hier zu unterscheiden zwischen Quellcode, der durch einen Browser gerendert wurde, und solchem, der zum Beispiel durch den Webserver übermittelt wird. Im ersten Fall handelt es sich vielmehr um eine Momentaufnahme. Wenn die Webseite tatsächlich dynamisch verändert wird, werden diese Änderungen im Anschluss auch im Quellcode widergespiegelt.

Um die grundsätzlichen Abläufe der automatischen Evaluationen von Webseiten zu betrachten, wurde zunächst festgestellt, welche Voraussetzungen erfüllt sein müssen, um diese Evaluationen durchzuführen:

1. Die zu evaluierende Webseite muss bekannt vorliegen, entweder der vollständige Quellcode oder ein konventioneller Zugriff mittels HTTP.
2. Die Webseite muss gerendert, also externe Ressourcen geladen und JavaScript ausgeführt werden.

Automatisierte `End-to-End`-Tests von Webseiten sind ein vor einiger Zeit gelöstes Problem: Mit Selenium existiert ein etabliertes Framework zur Steuerung von Browsern. Durch den weithin umgesetzten WebDriver-Standard ermöglicht das Framework mittels einer vereinheitlichten Schnittstelle die Steuerung der meisten Browser, wodurch die Diversifikation der Tests vereinfacht wird.

Als Konsequenz der aufgestellten Voraussetzungen wird Selenium als Schnittstelle zu den per WebDriver steuerbaren Browsern verwendet. Mittels der Schnittstelle werden die Browser angeleitet, beliebige Webseiten aufzurufen, vordefinierte JavaScript-Ressourcen einzubinden und auszuführen sowie Ergebnisse dieser Operationen auszuleiten.

Um das Ziel automatischer, technischer Evaluationen zu erreichen, wurden zwei Werkzeuge in Form von Browser-Erweiterungen in Erwägung gezogen:

1. WAVE, eine Browser-Erweiterung, die unter anderem von der Hochschulverwaltung im Rahmen von Barrierefreiheitsprüfungen genutzt wird. Die zugrundeliegende Engine ist jedoch nicht frei verfügbar
2. AXE DevTools (wie in der Vorstellung von AXE beschrieben)

Die Browser-Interaktionen, die mittels des WebDriver Protokolls abrufbar sind, beschränken sich allerdings allein auf die abgerufenen Webseiten. Somit sind sämtliche Versuche, eine reguläre Browser-Erweiterung zu kontrollieren, um so die Barrierefreiheitsprüfung automatisch umzusetzen, fehlgeschlagen:

- Tasten-Kombinationen, die per WebDriver ausgeführt werden, werden allein im Kontext der dargestellten Webseite evaluiert, während Erweiterungen nur in dem Kontext der Browser-Oberfläche reagieren
- Das Kontext-Menü (erreichbar durch einen Rechtsklick) lässt sich zwar öffnen, seine Schaltflächen befinden sich aber wiederum nicht in dem zulässigen Kontext.

Somit wurde der Ansatz, einfach die Bedienung etablierter Browser-Erweiterungen zu automatisieren, verworfen und der Fokus auf die zugrundeliegende Funktionsweise der Erweiterungen selbst gelegt.

Die Limitierungen des WebDriver-Protokolls führt dazu, dass generell eine Verwendung von Browser-Erweiterungen im Kontext dieser Automatisierung nicht möglich ist. Um den Prototyp dennoch umzusetzen, wurde entschieden, diesen auf Basis des AXE Frameworks umzusetzen.

In ersten Experimenten mit Selenium wurde klar, dass diese Ansteuerung von Browsern einiges an zeitlichem Mehraufwand erzeugt. Zwischen dem Start der Selenium-Komponente und dem Ausführen der ersten Anweisung lagen mehr als zehn Sekunden.

Mit dem Ziel höherer Flexibilität und möglicher Skalierung setzt der Prototyp deswegen auf Selenium Hub, eine Lösung, die Abhängigkeit zwischen dem Executor und dem befehlenden Consumer zu entspannen. An dieser Lösung sind drei Klassen von Akteuren beteiligt:

- Der Hub nimmt eine delegierende Rolle ein
- Executor-Nodes verbinden sich über ein gemeinsames Netzwerk mit dem Hub
- Consumer verbinden sich ebenso mit dem Hub und können dort eine Sitzung mit einem Executor-Node anfragen

Der Großteil der benötigten Zeit und Ressourcen eines Durchlaufs fallen allein auf den Vorlauf des Executors: Verbindungsaufbau zum Hub, Zuteilung einer verfügbaren Sitzung, Verbindungsaufbau zur Sitzung und warten bis der Browser bereit für Anweisungen ist. Die Ausführung der Evaluation hingegen benötigte nur etwa zwei Sekunden.

Mit diesem Hintergrund wurde für den Prototyp entschieden, eine jobbasierte Ausführung der einzelnen Evaluationen umzusetzen. Die Architektur sieht hier einen Prozess vor, der nach und nach Eingaben konsumiert und jeweils mit diesen eine Routine durchläuft. Abhängig von den verfügbaren Ressourcen kann dieser Prozess beliebig vervielfältigt und parallelisiert werden. Jede Ausführung ist komplett unabhängig, sobald sie ihre Zieldaten erhalten hat und die betreffende Webseite erreichen kann. Der Prototyp wurde auf Basis von Celery implementiert – ein Python Framework, welches genau den oben beschriebenen Ansatz verfolgt. In diesem Kontext sieht die Abfolge der Routine wie folgt aus:

1. Beanspruche eine Sitzung mit einem Executor-Node
2. Rufe durch den Executor die übergebene URL auf und warte, bis die Webseite geladen wurde
3. Lade das AXE-Framework in die Browser-Instanz
4. Starte eine Evaluation durch das AXE-Framework und erwarte die Vollendung
5. Speichere das Ergebnis der Evaluation ab
6. Gebe die Sitzung wieder frei

Celery bezieht die Parameter für die einzelnen Ausführungen mittels MQTT von einem Message Broker. Dieser stellt in erster Linie einen Kanal bereit, in den autorisierte Akteure entweder Nachrichten hinein schreiben (veröffentlichen) oder aus dem sie Nach-

richten abholen (konsumieren). Kanäle funktionieren dabei wie eine Warteschlange (Queue). Die erste eingelieferte Nachricht wird auch mit dem ersten Lesevorgang wieder ausgegeben. Es spielt bei dieser Art von Kommunikation in der Regel keine Rolle, wie viele Akteure veröffentlichen oder konsumieren. Der Message Broker entkoppelt die beteiligten Akteure, während er ein einheitliches Kommunikationsschema definiert.

Da zunächst nicht bekannt war, welche Browser-Erweiterung oder welches Framework für die Evaluationen verwendet werden würde, welche Daten von diesem erfasst werden und welcher Anteil dieser Daten wiederum interessant für diese Arbeit ist, wurde als Datenspeicher das NoSQL Datenbank Management System (DBMS) MongoDB ausgewählt. Diese Klasse DBMS erlaubt die Speicherung nahezu beliebiger, strukturierter Objekte im JSON-Format ohne ein bekanntes Schema. So wurden im Lauf der Entwicklung Felder hinzugefügt oder entfernt, ohne dass zugleich weitere Arbeiten an der Datenbank nötig wurden.

Einen weiteren Teil des Prototyps bildet ein WebCrawler. Um den Prototyp in allen Entwicklungsschritten zu testen und zu beobachten, war es sinnvoll, eine Menge zu testender Webseiten vorzuhalten. Um eine größere Bandbreite an potenziellen Problemen zu betrachten, wurde mittels Scrapy, einer Python-Bibliothek für genau diesen Zweck, ein erster WebCrawler implementiert, der ausgehend von einer initialen URL versucht, alle erreichbaren Seiten rekursiv zu erkunden.

Bei der Erkundung dieser Problemklasse sind zwei Probleme besonders aufgefallen:

- Scrapy sieht bei der Zwischenspeicherung von URLs bereits vor, diese zu filtern, sodass jede einzigartige URL im besten Fall auch nur einmal aufgerufen wird. Jedoch war diese Funktion alleine nicht besonders zuverlässig. In der Praxis wurden einige Fälle beobachtet, bei denen eine Vielzahl von URLs immer dasselbe Ziel hatten. Sie unterschieden sich lediglich durch ein Pfad-Segment, welches jeweils eine einzigartige ID aufwies.
- Seiten, die erst durch den User-Agent mittels JavaScript aufgebaut werden, lassen sich nicht ohne Weiteres erkunden. Während die Evaluation dieser Seiten genau dieses Problem überwunden hat, ist der WebCrawler zunächst sehr leichtgewichtig und sieht zunächst keine Notwendigkeit darin, JavaScript auszuführen.

Um einige Webseiten aus dem Umfeld des Prototyps auszukundschaften, wird also einiges an Expertenwissen vorausgesetzt. Somit wurde der WebCrawler zunächst abstrahiert:

- Die Funktion, entdeckte, erstmalig gesehene URLs an den Message Broker zu senden, ist für alle betrachteten Webseiten dieselbe.
- Im Weiteren werden URLs mit einer Domäne, die von der Domäne der aktuell betrachteten Webseite abweicht, ignoriert, denn sie sind sehr wahrscheinlich nicht relevant für die laufende Betrachtung.

Spezialisierungen sehen vor allem angepasste Regelsätze vor, um individuelle Eigenheiten wie die Pfadsegmente zu mitigieren. Diese Anpassungen sind einfach umzusetzen, erfordern jedoch Expertenwissen über den jeweiligen Aufbau der Webseiten.

Die letzte Komponente für den Prototyp ist eine Visualisierung der Ergebnisse. In der Entwicklung wurde zunächst auf eine Webansicht der Datenbank zurückgegriffen, nachdem allerdings eine gewisse Menge an Evaluationen vorlag, wurde diese recht starre Ansicht unübersichtlich und weniger hilfreich.

Motiviert aus persönlicher Vorerfahrung fiel die Wahl für die Visualisierungsplattform auf Grafana. Diese ermöglicht zu einem gewissen Maß explorative Visualisierungen der Datenbasis. Die Integration der Datenbasis war jedoch eine weitere Herausforderung, denn in der verwendeten Version von Grafana ist die Integration vom Typ MongoDB nicht ohne Zusatzkosten enthalten. Durch einen Proxy und eine Erweiterung, die von der Community entwickelt wurden, wurde diese Einschränkung zunächst umgangen. Fraglich ist jedoch, welchen Einfluss dieser Umweg auf die Stabilität und Performanz des gesamten Systems hat.

In diesem Fall – in Verbindung mit MongoDB – wurde schnell klar, dass die Performanz der Datenquelle zu Schwierigkeiten führt. Oft brach Grafana eine Visualisierung ab, bevor die Datenabfrage überhaupt vollendet wurde. Um die Performanz und Abfragezeiten zu verbessern, wurden bereits Suchindizes in MongoDB erprobt, diese verkürzten die Dauer spezifischer Abfragen erheblich. Zudem wurde der Standard-Timeout von 50 auf 90 Sekunden erhöht, seitdem laden alle Visualisierungen verlässlich.

Ein einzelner Eintrag in der Datenbank repräsentiert aktuell eine Evaluation, enthält also potenziell eine Menge von erkannten Problemen. Die Abfragesprache für MongoDB bietet mit Aggregation-Pipelines einen intuitiven Umgang mit den Datenmengen an.

Die Visualisierungen bieten aktuell (vor allem Entwicklern) konkrete Anhaltspunkte, welche Element auf ihrer Webseite problematisch sind und welchen Schweregrad diese Probleme tragen. Im Weiteren lassen sich die Probleme nach Fehlernamen und betroffener Richtlinie filtern und es werden Verletzungen der `Best Practices` gesondert dargestellt.

3.3. Bewertung

Der Prototyp ist in der Lage, eine größere Menge an Webseiten zu erfassen und – mit genug Zeit – diese auf mögliche Probleme hinsichtlich der Barrierefreiheit zu evaluieren. Die Erkenntnisse, die in diesem Prozess gewonnen werden, verschaffen Beteiligten einen Überblick über mögliche Nachbesserungen bezüglich der Barrierefreiheit sowie den Schweregrad des zugehörigen Problems und ermöglichen Entwicklern eine Sensibilisierung für Details bei der Gestaltung von Nutzeroberflächen.

Einige Problemgruppen wie Kontraste und Farben sind mit diesem Ansatz eindeutig zu erkennen, ebenso ihre Abwesenheit. Dies gilt allerdings nicht so für alle Aspekte der Barrierefreiheit. Somit bleibt zu bedenken, dass mit Hinblick auf Barrierefreiheitsprüfungen immer auch eine qualifizierte Auseinandersetzung mit den Inhalten, die veröffentlicht werden, erfolgen muss.

Durch die vollständig automatische Evaluation von zusammenhängenden Webseiten wird ermöglicht, auf längere Zeit kontinuierlich auf neue Probleme zu achten. Das ist gerade bei nutzergenerierten Inhalten, wie zum Beispiel Wissensplattformen oder E-Learning-Bereichen relevant, denn dort ist davon auszugehen, dass Artikel oder andere Inhalte regelmäßigen Anpassungen unterliegen.

Das Ausführen der einzelnen Evaluationen ist mit etwa 20 Sekunden sehr langsam, vermutlich ist ein Großteil dieser Zeit auf den Aufbau der WebDriver-Sessions zurückzuführen. Offen bleibt zunächst, ob diese Verzögerung verringert werden kann und ob ein alternativer Ansatz mit einem anderen Browser-Automatisierungs-Framework wie etwa Playwright vergleichbare Ergebnisse mit geringerem Overhead liefern kann.

Die Datenmenge (36.000 Evaluationen), die alleine während der Entwicklung und Evaluation des Prototyps angefallen ist, sorgt in der Entwicklungsumgebung für Probleme. Abfragen gegen diesen Datensatz sind langsam, auch hier bleibt offen, wie die Zugriffszeiten verbessert werden können. Zuletzt bleibt festzuhalten, dass die gewonnenen Daten in einer MongoDB liegen, die von der gewählten Visualisierungsplattorm Grafana nicht unterstützt wird. Dieses Problem wurde im Rahmen des Prototyps mitigiert, eine unterstützte Datenbank ist aber langfristig wünschenswert.

Es ist wichtig anzuerkennen, dass neben den beobachteten Performanz-Problemen einzelner Anwendungen auch die Ressourcen der Entwicklungsumgebung regelmäßig voll ausgelastet waren. Während durchzuführender Evaluationen lag die Auslastung der beiden CPU-Kerne und der acht Gigabyte verfügbaren Arbeitsspeicher bei 100 %. Neben dem zuvor vermuteten Overhead wird auch die Virtualisierung in Form von Docker-Containern einen nennenswerten Teil der genutzten Ressourcen beansprucht haben. An diesen Stellen liegt sicher einiges an Einsparpotenzial an Ressourcen, welche unter anderem den Betrieb der Datenbank positiv beeinflussen könnten.

Der entwickelte WebCrawler hat bisher nur auf öffentliche Webseiten zugegriffen. Zukünftige Anwendungsgebiete – wie die E-Learning-Plattform – setzen voraus, dass der Crawler einen autorisierten Zugang erhält oder eine separate Testinstanz verfügbar gemacht wird. Nur so können auch die hochdynamischen nutzererstellten Inhalte berücksichtigt werden.

Eine mobile Anwendung, die `RWTHapp` bietet zwar einen wie zuvor beschriebenen Zugang an, dieser setzt jedoch zu jedem Zeitpunkt einen User-Agent voraus, der JavaScript ausführen kann. Damit der WebCrawler diesen Zugang nutzen kann, muss er seine Aufrufe ebenfalls mit einer WebDriver-Implementation umsetzen.

3.4. Präsentation der Anwendung, Fallbeispiele

Die Auswertung aller gesammelten Probleme für einen spezifischen Host, gruppiert nach der Fehlerart und -quelle, geben einen recht guten Überblick über das Ausmaß wiederkehrender, spezifischer Probleme. Als Fallbeispiel wird hier eine spezifische Ansicht der Webseite `ITC Help` angeführt.

Wir nehmen die ansichtsübergreifende Statistik als Ausgangspunkt, hier werden einzigartige Probleme über Ansichten hinweg erfasst. Für das Ziel `help.itc.rwth-aachen.de` werden die drei prominentesten Problemfelder betrachtet, denn diese wurden jeweils in mehr als 100 Ansichten erfasst.

Violations by Site

Count ↓	Element	Description	Location	Violation
1070	<a href="" class="chat-support-header">	Ensures links have discernible text	https://help.itc.rwth-aachen.de/en/service/1jefzdccuvuch/article/26abb...	link-name
1004	<ul data-v-6317687a="" id="nav-lang" class="navbar-nav ms-auto">	Ensures that lists are structured correctly	https://help.itc.rwth-aachen.de/en/service/8d9eb2f36eea4fcaa9abd0e1...	list
1002	<ul data-v-6317687a="" id="nav-search" class="navbar-nav ms-auto nav...	Ensures that lists are structured correctly	https://help.itc.rwth-aachen.de/en/service/8d9eb2f36eea4fcaa9abd0e1...	list
622	<a href="/en/" class="nuxt-link-active">	Ensures links have discernible text	https://help.itc.rwth-aachen.de/en/service/1jefzdccuvuch/article/26abb...	link-name
609	<li id="langToggle" role="button" aria-label="Toggle language" aria-disabl...	Ensures interactive controls are not nested as they are not always announ...	https://help.itc.rwth-aachen.de/en/service/3fb56fd931ed44568c2b210b...	nested-interactive
446	<a href="/" class="nuxt-link-active">	Ensures links have discernible text	https://help.itc.rwth-aachen.de/service/1ieuyxzfvgdpd/	link-name
419	<li id="langToggle" role="button" aria-label="Sprache ändern" aria-disabl...	Ensures interactive controls are not nested as they are not always announ...	https://help.itc.rwth-aachen.de/service/1jeqhtat4k0o3/article/25d6d1e7...	nested-interactive
155	<a data-v-1b4987bc="" rel="license" href="https://www.gnu.org/licenses/...	Ensure links are distinguished from surrounding text in a way that does no...	https://help.itc.rwth-aachen.de/en/service/8d9eb2f36eea4fcaa9abd0e1...	link-in-text-block
139	<a data-v-1b4987bc="" rel="license" href="https://www.gnu.org/licenses/...	Ensure links are distinguished from surrounding text in a way that does no...	https://help.itc.rwth-aachen.de/en/service/8d9eb2f36eea4fcaa9abd0e1ca0...	link-in-text-block
66	<ul id="nav-lang" class="navbar-nav ms-auto" data-v-6317687a="">	Ensures that lists are structured correctly	https://help.itc.rwth-aachen.de/en/service/cfff1f34d6814a05ae8570d9c...	list
66	<ul id="nav-search" class="navbar-nav ms-auto nav-search-container se...	Ensures that lists are structured correctly	https://help.itc.rwth-aachen.de/en/service/cfff1f34d6814a05ae8570d9c...	list
64	<div data-v-a85dd6e0="" role="list" class="list-group service-mostreleva...	Ensures elements with an ARIA role that require child roles contain them	https://help.itc.rwth-aachen.de/service/vbf6fx0gom76/	aria-required-children
34	<a href="https://www.itc.rwth-aachen.de/umfrage" target="_blank">https...	Ensures the contrast between foreground and background colors meets ...	https://help.itc.rwth-aachen.de/service/rhr4fjjutttf/article/70199df94112...	color-contrast
27	<a rel="license" href="https://www.gnu.org/licenses/gpl-3.0.de.html" targ...	Ensure links are distinguished from surrounding text in a way that does no...	https://help.itc.rwth-aachen.de/en/service/8d9eb2f36eea4fcaa9abd0e1ca0...	link-in-text-block
23	<a rel="license" href="https://www.gnu.org/licenses/gpl-3.0.en.html" targ...	Ensure links are distinguished from surrounding text in a way that does no...	https://help.itc.rwth-aachen.de/en/service/8d9eb2f36eea4fcaa9abd0e1...	link-in-text-block
15	<pre style="overflow: auto;">	Ensure elements that have scrollable content are accessible by keyboard	https://help.itc.rwth-aachen.de/en/service/rhr4fjjutttf/article/70199df94...	scrollable-region-focusab...
7	<img src="https://api.app.rwth-aachen.de/v3/sabio/app/image?path=096...	Ensures <img> elements have alternate text or a role of none or presentat...	https://help.itc.rwth-aachen.de/en/service/u7okrwv9h0s2/article/d22d4...	image-alt
7	<img src="https://api.app.rwth-aachen.de/v3/sabio/app/image?path=096...	Ensures <img> elements have alternate text or a role of none or presentat...	https://help.itc.rwth-aachen.de/en/service/u7okrwv9h0s2/article/d22d4...	image-alt

‹ 1 2 3 4 5 6 7 ··· 66 › 1 - 18 of 1188 rows

Abbildung 2: Eine Übersicht der gefundenen Probleme auf allen erfassten Ansichten, gruppiert nach Art des Problems und dem Element, das als Urheber des Problems identifiziert wurde (eigens mittels Grafana erzeugte Ansicht).

3.4.1. Chat-Support-Icon

Am häufigsten wird ein Anker-Element bemängelt, welches einen leeren Hyperlink enthält, aber keine Erläuterung über dessen Ziel oder Zweck. Dieses Element wird genauer im Webbrowser untersucht, um nachzuvollziehen, wie es verwendet wird, welchen Zweck es erfüllt und wie die Interaktion damit aussehen könnte.

Es handelt sich um ein Bildelement, welches in der unteren, rechten Ecke des Viewports fixiert ist. Ein Klick auf dieses Element ändert zunächst nichts, denn das `href`-Attribut ist leer. Durch eingebundenes JavaScript wird jedoch ein Eventlistener gesetzt, der bei Interaktion ein neues Browserfenster öffnet. In diesem Fenster wird ein Chat angeboten, der nicht Teil dieser Betrachtung sein soll.

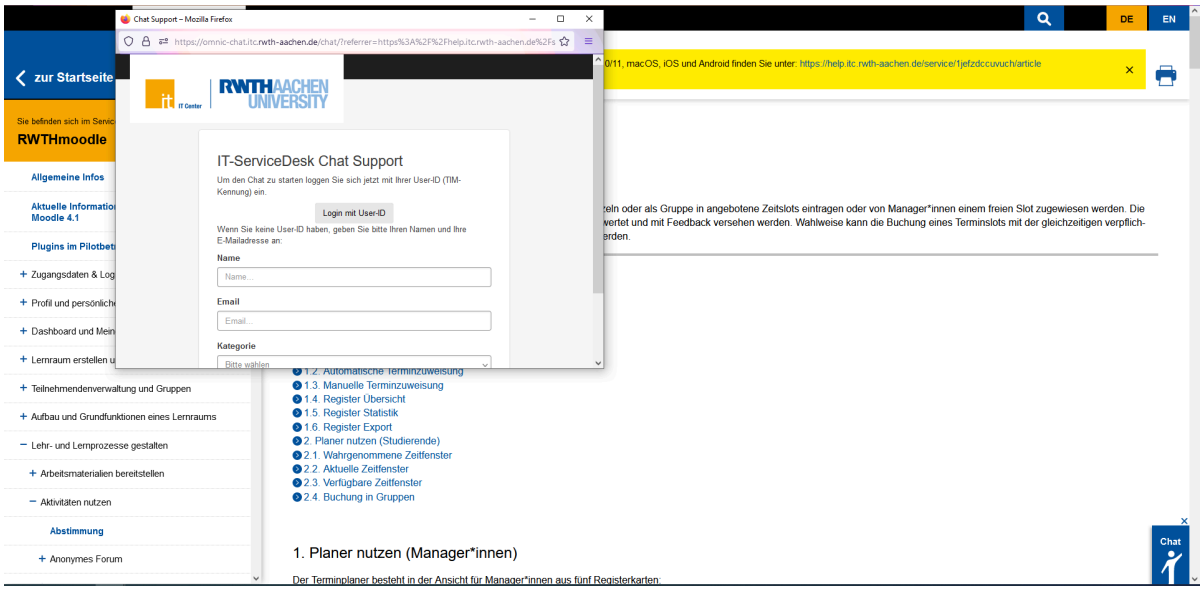

Abbildung 3: Das geöffnete Chat-Fenster [5]

Hierbei fallen zwei Punkte auf:

- Das Element selbst bietet keinerlei Information über seine Funktion. Erkennbar ist, dass es sich um einen Verweis ohne Beschreibung und mit einem leeren Ziel (`href=""`) handelt. Browser interpretieren ein leeres Ziel als Selbstverweis, wenn sie also dem Verweis folgen, landen sie auf derselben Seite, auf der sie den Verweis aufgefunden haben.
- In das Element wird ein SVG-Element eingebettet, ein Vektorgrafik-Format, das kompatibel zu HTML ist und unter anderem auch die Einbettung von Text vorsieht. Während Text, der direkt dem Anker untergeordnet ist, als Beschreibung des Ankers interpretiert werden kann, gilt dies nicht für Text, der wiederum dem eingebetteten SVG-Element untergeordnet ist (das wäre hier der `Chat` Schriftzug).

Der leere Hyperlink wird in der Regel als Selbstverweis interpretiert, der Browser würde dieselbe Seite an derselben Position aufrufen. Dieses Verhalten wird erst durch JavaScript manipuliert, im Detail wird das Neuladen der Seite unterbunden und stattdessen der Link auf die externe Chatseite in einem neuen Fenster geöffnet.

Hierbei handelt es sich um eine bewusste Design-Entscheidung: In gängigen Browsern wird hierdurch sichergestellt, dass die neue Seite in einem neuen Fenster geöffnet wird und die ursprüngliche Seite im Hintergrund geöffnet bleibt.

Problematisch bei dieser Umsetzung ist in erster Linie, dass das Ziel des Elements nicht beschrieben wird und der Verweis selbst leer ist; somit haben assistive Programme keinen Anhaltspunkt, wie sie das Element dem Nutzer erklären könnten. Sinnvoll ist in jedem Fall die Ergänzung eines Label-Attributs, welches die Funktion erklärt [6].

3.4.2. Die Navigationsleiste

Gleich mehrere der Probleme haben direkten Bezug zu der Navigationsleiste und der Sprachauswahl.

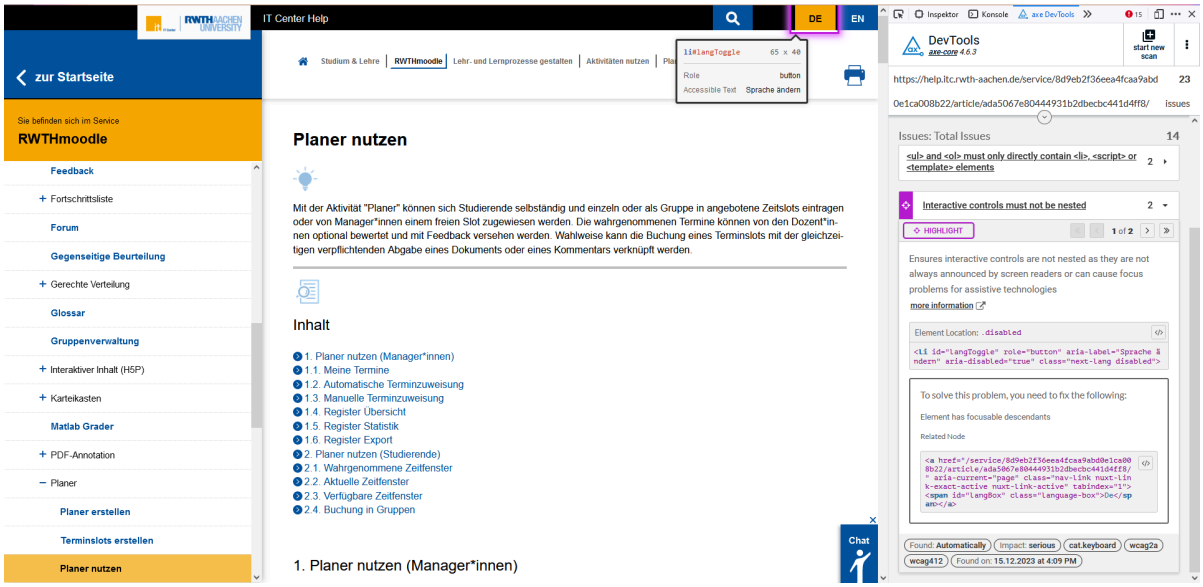

Abbildung 4: `AXE DevTools` ausgeführt für [5]

Die oben stehende Abbildung zeigt eine Webseite mit der zusätzlich gerenderten Browser-Erweiterung `AXE DevTools`. Diese markiert einen Verstoß gegen die HTML-Spezifikation: `Interaktive Elemente dürfen nicht vernestet werden`.

Die Navigationsleiste selbst ist als Liste strukturiert. Die HTML-Spezifikation schränkt die zulässigen Klassen untergeordneter Elemente stark ein: neben `<li>`-Tags - Listeneinträgen sind ausschließlich `<script>`- und `<template>`-Tags erlaubt. Die betrachtete Webseite verstößt gegen diese Vorgabe drei Mal, indem Listeneinträge mittels dem Rollen-Attribut als Knöpfe deklariert werden. Das betrifft die Suche sowie die zwei Sprachumschalter.

Das verwendete Rollen-Attribut verändert, wie Browser das Element interpretieren und in diesen Fällen kommt eine nicht standardkonforme Komposition von Elementen zustande. Wenn die Vorgaben an erlaubte Kompositionen so explizit sind, kann die Interpretation durch assistive Programme oder die Nutzernavigation eingeschränkt werden.

Diese Listenelemente wurden als Schaltflächen erkannt. Um der Designentscheidung treu zu bleiben, wäre die einfachste Option, dem Listenelement selbst die Rolle zu entziehen und stattdessen ein weiteres Element einzufügen, welches die Rolle des Knopfes einnimmt.

Gleiches gilt für die Sprachauswahl. Hier kommt jedoch ein weiteres Problem zum Tragen: Die Knöpfe der Sprachauswahl sind eigentlich keine Knöpfe, denn das innerste Element ist ein Hyperlink auf die entsprechend lokalisierte Version der aktuellen Ansicht. Auch hier liegt ein Bruch mit der HTML-Spezifikation vor [7], denn diese sieht eine Verschachtelung interaktiver Elemente nicht vor. Insgesamt wäre hier eine Abwägung angebracht, ob die Rolle der Knöpfe nicht einfach entfallen sollte.

3.4.3. Mehrfach vergebene IDs

Ebenfalls als Verstoß gegen die HTML-Spezifikation [8] wird bemängelt, dass für die meisten Ansichten einige Element-IDs mehrfach vergeben werden. Der Sinn von IDs ist offensichtlich und es ist wichtig, dass IDs auch tatsächlich pro Ansicht einmalig vergeben werden, um eine vorhersehbare Navigation in dem Dokument zu erreichen. IDs werden vor allem genutzt, um gezielt einzelne Elemente in einem Dokument anzusprechen, im Gegensatz zu Klassen, welche zum Ziel haben, mehrere Elemente einer Kategorie gleichzeitig anzusprechen.

Eine wichtige Rolle spielen hierbei Label-Elemente, diese nehmen über die ID Bezug auf das Element, das sie beschreiben. Ist dieser Bezug nicht eindeutig, führt dies zu undefiniertem Verhalten des Browsers und anderer interpretierender Programme.

Die vorliegenden Fälle sind allerdings anderer Natur, hier geht es vor allem um SVG-Elemente. Diese enthalten immer dieselben IDs an denselben Stellen in ihrer Struktur (zum Beispiel ID `Ebene_1` für das Wurzelelement).

Das Auftreten der immer selben IDs ist sehr wahrscheinlich auf das Programm zurückzuführen, mit dem die SVGs erstellt wurden. Bei SVGs handelt es sich nicht selten um freie Ressourcen, die in Form großer Sammlungen der Allgemeinheit bereitgestellt werden. Einfluss auf diese Sammlungen zu nehmen, ist sehr schwierig bis aussichtslos. Vielmehr sollten Entwickler darauf achten, dass fremde Elemente eine Art Sanitisierungsverfahren durchlaufen. Wenn SVGs auf eine Art in die Webseite eingebunden werden, dass die vergebenen IDs transparent im Quellcode der Webseite auftauchen, so dürfte genau dieser Prozess die Gelegenheit bieten, bei Bedarf IDs zu entfernen oder systematisch abzuändern.

4. Fazit

Der vorliegende Prototyp demonstriert, wie das AXE Accessibility Evaluation Framework genutzt werden kann, um automatische Überprüfungen von Webseiten vorzunehmen. Auf Basis der erzeugten Daten wurden erste Visualisierungen entworfen, die Entwickler dabei unterstützen, Barrieren strukturiert zu erkennen und zu verstehen sowie Handlungsempfehlungen abzuleiten.

Wie in der Bewertung des Prototyps diskutiert wurde, kann diese automatische Analyse nur den Prozess einer Barrierefreiheitsprüfung unterstützen. Um der BITV zu entsprechen ist es notwendig, dass zunächst in dem Konzept der Webseite Verstöße gegen die WCAG gefunden werden. Hierfür ist jedoch ein Verständnis für die betreffenden WCAG-Kriterien erforderlich. Auf dieser Basis müssen in der Barrierefreiheitserklärung erkannte strukturelle Probleme dokumentiert werden.

Zusätzlich dazu bieten die durch den Prototyp erzeugten Evaluationen einen Überblick, ob zusätzliche technische Barrieren vorliegen, die bis zur Behebung ebenfalls dokumentiert werden sollen.

5. Ausblick

Es wurden zwei konkrete Ziele für die Weiterentwicklung des vorliegenden Prototyps identifiziert:

1. Wie bereits in der Bewertung angeführt, sind starke Verbesserungspotentiale in Bezug auf die Datenspeicherung zu erwarten.
2. Sollte der Vorlauf für Evaluationen verkürzt werden können, so ist auch dort ein sehr starkes Potenzial zu verorten.

Zudem sind in Bezug auf die Nützlichkeit des Prototyps zwei weitere Ziele naheliegend:

3. Der WebCrawler ist aktuell eingeschränkt in den Zielen, die er erreichen kann. Sollte eine Integration mit einem per WebDriver steuerbaren Browser gelingen, die keine allzu große Verzögerung mit sich bringt, so wird der Nutzen der Anwendung erheblich gesteigert.
4. Für mindestens eine zu evaluierende Webseite fehlt aktuell ein nutzbarer Zugang. Durch die hochdynamische Gestalt einer E-Learning-Plattform, die sich einer Vielzahl an Erweiterungen bedient und die von den Nutzern sehr stark angepasst werden kann, wäre auch in Betracht zu ziehen, ob nicht eine Evaluation der erstellten Lehrinhalte sinnvoll wäre.

Zuletzt, auch mit Bezug auf Punkt 4 ist es wichtig, in den direkten Austausch mit den Beteiligten der zu prüfenden Webseiten zu treten. Mit diesen sollten zunächst die drängendsten Probleme durchgesprochen werden, hierbei kann die Darstellung der Ergebnisse besprochen und verbessert werden. Besonders für die verschiedenen beteiligten Gruppen – Autoren und Entwickler – wäre es sinnvoll, zeitnah eine entsprechende Trennung der Ergebnisse zu erreichen.

Bibliographie

[1] CEN, CENELEC, und ETSI, Zugegriffen: 10. März 2021. [Online]. Verfügbar unter: https://web.archive.org/web/20231111005436/https://www.etsi.org/deliver/etsi_en/301500_301599/301549/03.02.01_60/en_301549v030201p.pdf

[2] Bundesministerium der Justiz und Bundesamt für Justiz, Zugegriffen: 12. September 2023. [Online]. Verfügbar unter: https://web.archive.org/web/20231111010313/https://www.gesetze-im-internet.de/bitv_2_0/BJNR184300011.html

[3] *Gesetz zur Gleichstellung von Menschen mit Behinderungen § 12b Erklärung zur Barrierefreiheit*. [Online]. Verfügbar unter: https://web.archive.org/web/20231113161906/https://www.gesetze-im-internet.de/bgg/___12b.html

[4] K. Valkhof, „Fixing contrast issues, on your own site and elsewhere“. Zugegriffen: 11. Dezember 2023. [Online]. Verfügbar unter: https://www.a11yproject.com/posts/fixing-contrast-issues-on-your-own-site-and-elsewhere

[5] „Planer nutzen (RWTHmoodle) - IT Center Help“. Zugegriffen: 15. Dezember 2023. [Online]. Verfügbar unter: https://help.itc.rwth-aachen.de/service/8d9eb2f36eea4fcaa9abd0e1ca008b22/article/ada5067e80444931b2dbecbc441d4ff8/

[6] „<a>: The Anchor element“. Zugegriffen: 4. Dezember 2023. [Online]. Verfügbar unter: https://web.archive.org/web/20231204084534/https://developer.mozilla.org/en-US/docs/Web/HTML/Element/a#external_links_and_linking_to_non-html_resources

[7] I. Hickson, Hrsg., „4.10.8 The button element“. in HTML5 - A vocabulary and associated APIs for HTML and XHTML.

[8] I. Hickson, Hrsg., „3.2.6 Global attributes“. in HTML5 - A vocabulary and associated APIs for HTML and XHTML.