BA
Dokumentation

Content Analyzer

Semester: Frühling 2025

Version: 1.0
Date: 2025-06-13 13:29:39Z
Git Version: 632dabb

Projekt Team: Ali Al-Kubaisi
Simon Amberg

Projekt Betreuer: Markus Stolze

Departement Informatik
OST – Ostschweizer Fachhochschule

Abstract

Das Ziel dieser Bachelorarbeit war die Entwicklung einer Webapplikation ‘Content Analyzer‘ zur automatisierten Analyse von Webseiteninhalten. Die Applikation soll Unternehmen dabei unterstützen, den Zustand ihrer Webseiten hinsichtlich Barrierefreiheit in Bezug auf Standards wie den Web Content Accessibility Guidelines (WCAG) und Schreibstil effizient zu bewerten und zu verbessern.

Im Zentrum der Arbeit stand die Umsetzung eines Web-Scrapers und eines modularen Systems, das open-source Analysetools über einen Plugin-Mechanismus integriert. Der entwickelte Web-Scraper nutzt moderne Technologien, um Inhalte von Webseiten automatisiert zu erfassen und in einem Repository zu speichern. Anhand dieses Repositorys wird dann die erfasste Webseitenstruktur mithilfe der eingebundenen Analysetools untersucht.

Die initiale Implementierung umfasste insbesondere die Analyse von Barrierefreiheit mithilfe des Tools Axe Core sowie ergänzende Auswertungen hinsichtlich der WCAG-Richtlinien. Zusätzlich wurde ein Schreibstil-Checker entwickelt, der regelbasierte und KI-gestützte Ansätze für grammatikalische Korrekturen und Textanalysen kombiniert. Darüber hinaus bietet die Applikation Möglichkeiten zur Verwaltung und Annotation von Inhalten und Gruppierung, sowie Verwaltung von erkannten Problemen.

Besondere Herausforderungen beim Web-Scraping ergaben sich aus der Diversität an Technologien und Architekturen, die für moderne Webseiten eingesetzt werden, sowie der baumartigen Struktur von Webseiten. Diese wurden durch detaillierte Analyse der Problemdomäne, sorgfältige Technologieauswahl, sowie einer iterative Entwicklung erfolgreich bewältigt.

Das Ergebnis der Arbeit ist ein flexibles und wartbares System, das es ermöglicht Webseiten auf grundliegende Richtlinie zu überprüfen. Über eine anschauliche Darstellung der Resultate, können erkannten Problemen verwaltet und behoben werden. In Zukunft soll das System als Grundlage dienen, auf der weitere Analysetools eingebunden werden können, um aufkommende Standards abzudecken und neue Fähigkeiten einzubinden.

Inhaltsverzeichnis

Teil I

Produkt Dokumentation

Kapitel 1

Ausgangslage

1.1 Web-Scraping

Web-Scraping ist ein Prozess, der automatisiert den Inhalt von Webseiten sammelt und diesen in einem strukturierten lokalen Index speichert. Der Prozess kann in drei Phasen unterteilt werden:

Fetching phase
In der Fetching phase wird der Inhalt der Webseite, identifiziert durch einen URL abgefragt. Dafür können üblicherweise HTTP GET Anfragen verwendet werden, dieselbe Methode, die auch Browser benutzen.

Extraction phase
Aus den erhaltenen Daten werden dann in der Extraction phase relevante Informationen extrahiert. Dafür kann Textverarbeitung, HTML Parsing oder XML Path Sprachen verwendet werden.

Transformation phase
Zuletzt werden die extrahierten Informationen in einer strukturierter Form gespeichert. [8]

Eine übliche Unterscheidung von Webscraper und Webcrawler ist, dass Webscraper dafür verantwortlich sind Informationen und Inhalte von Webseiten zu extrahieren. Webcrawler hingegen übernehmen die Rolle der Navigation durch Webseiten mittels Folgen von Links. Üblicherweise startet ein Crawler mit einem Set von URLs, genannt *seed* URLs. Zuerst werden die Startseiten abgefragt anhand der seed URLs. Auf den Seiten vorkommende URLs werden extrahiert und einer Queue hinzugefügt. Dieser Prozess wird wiederholt, bis die Queue leer ist oder ein Limit erreicht ist. [2]

1.2 Rechtslage und Standards

1.2.1 Accessibility

Viele existierende Web-Scraping Lösungen sind massgeschneidert auf Firmen, die aufgrund gesetzlichen Regulationen einen bestimmten Accessibility-Standard erreichen müssen.

Dabei ist zu beachten, dass Web-Scraping nicht zwingend im direkten Zusammenhang mit Accessibility steht, jedoch können Accessibility-Standards die Struktur von Webseiten beeinflussen und damit auch die Möglichkeiten des Web-Scraping.

Oft ist dies der Americans with Disability Act (ADA) Standard. Er legt Anforderungen für Webseiten der Regierung und öffentlichen Unternehmen fest. Für Regierungen gilt Level AA von WCAG 2.1, für Unternehmen ist dies jedoch eher eine Empfehlung um rechtliche Standards zu erfüllen. [1]

In der Schweiz existiert eine weniger strikte Rechtslage, bei der nur Behörden, staatsnahe Betriebe und Betriebe des öffentlichen Rechts, wie Schulen und Spitäler bestimmte Regulationen erfüllen müssen. Der Accessibility Standard 3.0 wurde 2020 eingeführt und setzt Konformitätsstufe AA der WCAG 2.1 voraus. Er nennt als relevante Inhalte textuelle und nicht textuelle Informationen, Dokumente zum Herunterladen und Interaktionen mit Formularen und Prozessen. Ausgenommen sind Inhalten von drittanbietern, Geografische Karten und Live-Medien. [3]

1.2.2 Legalität von Web-Scraping

Webscraper befassen sich mit einer grossen Menge an Daten. Diese können teils sensitiv, nicht-öffentlich, urheberrechtlich geschützt oder spezifisch für automatisierten Zugriff gesperrt sein. Mögliche rechtliche Bedenken sind:

- Urheberrechtlich geschützte Daten
- Zugriff auf Daten, der rechtlich illegal ist (Authentifikation und Authorisierung)
- Explizite Willensäusserung eines Verbotes für automatisierte Tools zum Beispiel durch ein robots.txt File oder andere technologische Mittel
- Persönlich identifizierbare Informationen (PII), die gesetztlich geschützt sind.
- Beeinflussung der Verfügbarkeit einer Webseite durch starke Belastung des Servers

[6]

Die gesetzliche Lage zu Web-Scraping unterscheidet sich von Land zu Land und entwickelt sich stetig. Eine genaue Überprüfung, ist in jeder Anwendung nötig, da unterschiedich mit Daten umgegangen wird und möglicherweise Genehmigungen der Webseiten Inhaber vorhanden ist.

1.3 Existierende Technologien

Da Web-Scraping für viele Industrien ein relevantes Thema ist existieren viele Hilfsmittel, die dafür eingesetzt werden können und auch diverse Komplettlösungen.

Die Virginia Polytechnic Institute and State University hat eine gute Übersicht über momentane Web Accessibility Tools basierend auf Webcrawlern publiziert. Erwähnt werden Asqatasun, axe-crawler, Pa11y Dashboard und mehr. [10]

1.3.1 Asqatasun

Asqatasun ist ein Open-source Accessibility Analyse und Search-Engine Optimization Tool. Intern benutzt es crawler4j, eine Open-Source Java Bibliothek, den Inhalt von statischen Webseiten liefert. Das Tool soll auch dynamische Webseiten bzw. dynamische HTML analysieren können, was jedoch im Source-Code nicht einfach offensichtlich ist.

1.3.2 Axe Crawler

Das opensource Tool Axe Crawler ist nicht nur ein Web Crawler, sondern führt auch gleich eine Accessibility-Analyse mit Axe Core durch. Als Web Crawling Mechanismus werden hier Html Get-Requests verwendet, die den Html Inhalt liefern. Ein offensichtlicher Nachteil daran ist es, das kein Javascript ausgeführt werden kann, wodurch der Ansatz für typische SPAs (Single Page Applications) nicht funktioniert.

1.3.3 Selenium

Selenium ist ein open-source Automations-Tool, das primär vor Testing verwendet wird. Es stellt einen virtuellen Browser zur Verfügung und ist so in der Lage das Verhalten eines echten Browsers zu simulieren. Somit können auch Webseiten, die stark auf Javascript basieren gescraped werden. Der Nachteil daran ist jedoch, dass Selenium sehr Resourcen intensiv ist und somit schlecht skaliert. Verfügbar ist Selenium in einer Vielzahl von Programmiersprachen. [8] Lizenz: Apache 2.0

1.3.4 Pupeteer

Pupeteer ist eine node.js Bibliothek, die einen Browser (Chrome oder Firefox) zur Verfügung stellt ähnlich wie Selenium. Ports auf andere Programmiersprachen wie Python existieren (Pyppeteer). Es könnte verwendet werden um dynamische Webseiten zu scrapen. Auf Webseiten kann direkt zugegriffen oder der Inhalt als HTML gesetzt werden. Pupeteer benötigt generell weniger Resourcen als Selenium. [8] Lizenz: Apache 2.0

1.3.5 Playwright

Playwright stellt wie Pupeteer und Selenium einen Browser zur Verfügung. Benutzbar ist es in vielen Programmiersprachen und unterstützt mehr Browser als Pupeteer (zusätzlich

Edge und Safari). Es besitzt gewisse Vorteile wie Cross-Browser testing, parallele Tests und eine grössere Anzahl an unterstützten Sprachen, hat dafür aber mehr Overhead und leicht in einfachen Applikationen schlechtere Performance als Pupeteer. [9] Lizenz: Apache 2.0

1.3.6 BeautifulSoup

BeautifulSoup ist eine Python Bibliothek um Daten aus HTML und XML files zu extrahieren. Vergleichbare Biblioteheken existieren auch für Javascript und Java. Verwendet werden kann es um HTML Inhalt, der über Http-Requests erhalten wurde zu parsen und zu analysieren.

1.3.7 Scrapy

Scrapy ist ein Open-Source-Framework zur Extraktion von Daten aus Websites auf eine schnelle, einfache und dennoch erweiterbare Weise. Es wurde in Python entwickelt und bietet eine asynchrone Umgebung, die es ermöglicht, mit Hilfe von sogenannten Spidern systematisch Webseiten zu crawlen und strukturierte Daten zu extrahieren. Ursprünglich für Web Scraping konzipiert, kann Scrapy aber auch zur Extraktion von Daten über APIs oder als allgemeiner Webcrawler eingesetzt werden. Durch seine modulare Architektur können Entwickler eigene Middleware, Pipelines und Erweiterungen implementieren, um Scrapy flexibel an individuelle Anforderungen anzupassen. Mithilfe von Splash kann Scrapy auch mit dynamischem HTML arbeiten.

1.3.8 Crawlee

Crawlee ist eine Open-Source-Bibliothek für Web-Scraping und Browser Automation. Sie ist in Python und Javascript verfügbar und ermöglicht das extrahieren von daten aus HTML, Pdfs und Bildern. Im Hintergrund werden Puppeteer, Playwright und ähnliche Bibliotheken verwendet.

1.4 Existierende Komplettlösungen

Die rechtliche Lage und Präsenz von Accessibility in der Gesellschaft hat zu einer Vielzahl von Hilfsmitteln für die Accessibility-Analyse von Webseiten geführt. Viele davon sind kostenpflichtig, es gibt jedoch auch gute kostenlose Optionen.

EqualWeb Crawler

Das temporär noch kostenfreie Tool EqualWeb Crawler bewirbt sich als echtzeit Prüfungs-Tool auf Basis von WCAG 2.2. Sie bewerben auch die Möglichkeit, Passwortgeschützte Seiten zu analysieren.

Als aktuelle Optionen für eine Accessibility-Analyse erwähnen sie zuerst kostenfreie Tools, die aber limitiert sind auf einzelne Webistes, wodurch man eine vielzahl

an Durchläufen durchführen muss. Die zweite Option seien externe Experten und kostenpflichtige Tools, was viel Zeit und Kosten verursacht. [4]

Als relevante Elemente für Accessibility nennen sie unter anderem Text, Bilder, Formulare (HTML-Form) und Navigation.

a11y.Radar

Das kommerzielle Tool a11y.Radar ist fokussiert auf die Verhinderung von Rechtsfällen aufgrund mangelnder Accessibility. Spezifisch werden amerikanische Firmen angesprochen aufgrund ADA Regulationen, die eingehalten werden müssen. Es wird ein Dashboard angeboten, über das momentane Probleme und der Grad der Accessibility sichtbar ist. Überprüft wird anhand WCAG 2.1.

1.5 Herausforderungen

Aus den Fähigkeiten existierenden Technologien und Komplettlösungen lassen sich verschiedene Herausforderungen herauslesen, mit denen Web-Scraping konfrontiert ist. Anhand dieser Herausforderungen können vorhandene Lösungen verglichen werden.

CSRF Tokens

Cross-Site Request Forgery (CSRF) ist eine typische Vulnerability von Webseiten. Das Ziel dabei ist es, Anfragen über den Browser des Benutzers auszulösen. Die häufigste Sicherheitsmassnahme dagegen ist die Verwendung von CSRF-Token. Dies sind versteckte Werte auf der Webseite, die nur dem Benutzer zugänglich sind und bei jeder Anfrage mitgesendet werden. [8]

Authentifikation

Der Grossteil aller Webseiten ist nicht öffentlich und hinter Authentifikation versteckt. Für die Überprüfung der Authentifikation werden häufig Cookies oder HTTP-Header verwendet, die nach durch das Login erhalten werden.

Javascript/Dynamic HTML

Heutzutage basieren viele Webseiten auf Javascript Rendering, was heisst, dass die erhaltene HTML-Webseite kaum Inhalt besitzt. Der tatsächlich dargestellte Inhalt wird erst nach der Bereitstellung des DOM über Javascript erstellt. Um den dynamischen Inhalt zu erhalten, muss also erstens Javascript ausgeführt werden können und zweitens der Zeitpunkt, ab dem dieser dargestellte wird bestimmt werden.

HTTP-Headers

Gelegentlich überprüfen Webseiten auch den User-Agent HTTP-Header, um festzustellen, ob Anfragen von automatisierten Tools oder echten Benutzer kommen. Zu umgehen ist dies über das Setzen eines anpassbaren HTTP-Headers.

Weitere relevanten Headers wären der *Accept* Header. [8]

Kapitel 2

Aufgabestellung

Zur Überprüfung von Webseiten nach Standards wie Barrierefreiheit, Layout und Schreibstil sind oftmals aufwändige manuell durchgeführte Tests mittels verschiedenen existierenden Tools notwendig, um jegliche Probleme auf der Haupt- und Unterwebseiten zu finden. Die Verwaltung einer Vielzahl von Ergebnissen aus separaten Tools ist schwierig zentral an einem Ort zu behalten. Dementsprechend fehlt auch eine einheitliche Übersicht darüber.

Das Ziel dieser Arbeit ist es ein System zu entwickeln, mit dem Webseiten über einen Web-Scraper automatisch erfasst werden können. Auf dieser Basis sollen danach über einen Plugin-Mechanismus Open-Source-Tools, wie Axe Core die Webseite auf Barrierefreiheit und weitere Standards überprüfen. Ebenfalls soll eine KI-basierte Schreibstilanalyse für Webseiten implementiert werden. Erkannte Richtlinienverletzungen und Probleme sollen in Gruppen organisiert werden können um eine Arbeitsverwaltung zu ermöglichen. Der Benutzer interagiert mit diesem System über eine ebenfalls zu entwickelnde Web-Anwendung.

Kapitel 3

Anforderungen

3.1 Funktionale Anforderungen (FR)

In den funktionalen Anforderungen unserer Anwendung agiert der Benutzer stets als Akteur, der direkt mit der Anwendung interagiert. Ziel ist es, dass der Benutzer den Prozess der automatisierten Webseitenanalyse – von der Erfassung der Inhalte über die Analyse (z.B. Barrierefreiheit, Schreibstil) bis hin zum Reporting – effizient steuern und auswerten kann.

3.1.1 Minimum Viable Product (MVP)

ID	MVP-1
Name	Website-Crawler (Inhaltsanalyse und Erfassung)
Titel	Automatisches Crawlen von Websites
Beschreibung	Die Anwendung soll es Benutzern ermöglichen, eine gegebene Website automatisch zu durchsuchen und alle untergeordneten HMTL-Seiten in einer Liste zu sehen.
Priorität	Hoch

Tabelle 3.1 – MVP-Funktionale Anforderung 1 (MVP-1)

ID	MVP-2
Name	Website-Crawler (Inhaltsanalyse und Erfassung)
Titel	Automatisches Crawlen von allen Webseiteninhalten
Beschreibung	Die Anwendung soll es Benutzern ermöglichen, eine gegebene Website auf alle Inhalte (z.B. HTML-Seiten, Bilder, PDF-Dateien) zu durchsuchen und sie in einer Liste zu sehen.
Priorität	Hoch

Tabelle 3.2 – MVP-Funktionale Anforderung 2 (MVP-2)

ID	MVP-3
Name	Content Repository
Titel	Verwaltung und Speicherung der erfassten Inhalte
Beschreibung	Die Referenzen aller durch den Crawler erfassten Inhalte werden in einem strukturierten Repository (z.B. Datenbank) gespeichert, sodass sie effizient durchsucht und weiter analysiert werden können.
Priorität	Hoch

Tabelle 3.3 – MVP-Funktionale Anforderung 3 (MVP-3)

ID	MVP-4
Name	Web-Oberfläche
Titel	Darstellung von Resultaten als Website
Beschreibung	Die Anwendung stellt eine minimalistische grafische Benutzeroberfläche bereit, über die der Crawler gestartet und die erfassten Inhalte angezeigt werden können.
Priorität	Hoch

Tabelle 3.4 – MVP-Funktionale Anforderung 4 (MVP-4)

ID	MVP-5
Name	Plugin-Mechanismus
Titel	Integration externer Analysetools
Beschreibung	Über einen Plugin-Mechanismus können externe Tools (z.B. zur Prüfung von Layout und Barrierefreiheit) in die Anwendung eingebunden werden.
Priorität	Hoch

Tabelle 3.5 – MVP-Funktionale Anforderung 5 (MVP-5)

ID	MVP-6
Name	Tool-Runs
Titel	Starten von Analyseprozessen
Beschreibung	Über die Website können einzelne Plugins/Tools zur Analyse von Inhalten gestartet werden. Die Resultate der Analyse werden in der Webapplikation angezeigt.
Priorität	Hoch

Tabelle 3.6 – MVP-Funktionale Anforderung 6 (MVP-6)

ID	MVP-7
Name	Axe Core
Titel	Grundlegende Barrierefreiheits-Analyse
Beschreibung	Die Anwendung führt eine automatisierte Analyse der gecrawlten HTML-Seiten hinsichtlich der Barrierefreiheit unter Nutzung von Axe Core als erstes Plugin durch und zeigt erkannte Probleme an.
Priorität	Hoch

Tabelle 3.7 – MVP-Funktionale Anforderung 7 (MVP-7)

ID	MVP-8
Name	Metadaten von Analyseresultate
Titel	Analyseresultate (Issues) von Website-Elementen gruppieren
Beschreibung	Metadaten sollen für Individuelle Ergebnisse zu Website-Elementen (Issues) bearbeitet werden können. Darunter Kommentare und Stati wie “ignoriert“, “abgeschlossen“
Priorität	Hoch
Notizen	Mit der Einführung von FR-15 von Inhalten zu Analyseresultate aktualisiert.

Tabelle 3.8 – MVP-Funktionale Anforderung 8 (MVP-8)

ID	MVP-9
Name	WCAG Richtlinien Überprüfung
Titel	Erweiterte Barrierefreiheits-Analyse
Beschreibung	Die Anwendung führt eine automatisierte Analyse der gecrawlten HTML-Seiten durch hinsichtlich der WCAG Richtlinien Kriterien, die nicht von Axe Core abgedeckt sind, unter Nutzung von ergänzenden Tools und zeigt erkannte Probleme an.
Priorität	Mittel

Tabelle 3.9 – FR-Funktionale Anforderung 9 (MVP-9)

3.1.2 Zusätzliche Funktionen

ID	FR-13
Name	Schreibstil-Analyse
Titel	KI-basierte Analyse des Schreibstils
Beschreibung	Die Anwendung soll ein KI-Modul als Plugin integrieren, das den Schreibstil der erfassten Inhalte analysiert und Verbesserungsvorschläge liefert.
Priorität	Mittel

Tabelle 3.10 – Funktionale Anforderung 13 (FR-13)

ID	FR-10
Name	Content Sets und Projekte
Titel	Gruppierung von Websites und Inhalten
Beschreibung	Benutzer können mehrere Webseiten oder Inhaltselemente zu Sets bzw. Projekten zusammenfassen, um diese gemeinsam zu analysieren.
Priorität	Mittel
Notizen	Keine Anforderung mehr, wird durch FR-15 abgelöst

Tabelle 3.11 – Funktionale Anforderung 10 (FR-10)

ID	FR-11
Name	Reporting-Funktionen
Titel	Export von Analyseergebnissen
Beschreibung	Die Anwendung ermöglicht den Export der Analyseergebnisse (z.B. als CSV-Datei) zur weiteren Auswertung und Dokumentation.
Priorität	Niedrig

Tabelle 3.12 – Funktionale Anforderung 11 (FR-11)

ID	FR-12
Name	Annotation und To-Do-Listen
Titel	Kommentarfunktion und Aufgabenverwaltung
Beschreibung	Die Anwendung ermöglicht es Benutzern, Inhalte zu annotieren und To-Do-Listen zur Verwaltung von Aufgaben zu führen.
Priorität	Niedrig

Tabelle 3.13 – Funktionale Anforderung 12 (FR-12)

ID	FR-14
Name	Automatische Gruppierung
Titel	Automatische Gruppierung von Websites und Inhalten
Beschreibung	Webseiten können automatisch nach Kriterien wie Stilanalyse und Fehleranzahl oder Art Gruppiert werden.
Priorität	Niedrig
Notizen	Keine Anforderung mehr, wird durch FR-15 abgelöst

Tabelle 3.14 – Funktionale Anforderung 14 (FR-14)

ID	FR-15
Name	Gruppierung von Analyseresultate
Titel	Analyseresultate von Website-Elementen gruppieren
Beschreibung	Individuelle Ergebnisse zu Website-Elementen (Issues) sollen gruppiert werden können auf der Ansicht der Analyseresultate.
Priorität	Mittel

Tabelle 3.15 – Funktionale Anforderung 15 (FR-15)

ID	FR-16
Name	Matching von Analyseresultate
Titel	Matching von Analyseresultate zwischen Runs
Beschreibung	Von zwei aufeinanderfolgenden Ausführungen eines Plugins sollen gleiche betroffenen Elemente (Issues) erkannt werden. Dabei sollen Stati überprüft und für ignoriert übernommen werden.
Priorität	Niedrig

Tabelle 3.16 – Funktionale Anforderung 16 (FR-16)

3.1.3 Zukünftige Erweiterungen

Titel	Beschreibung	Priorität
Multi-User	Das System unterstützt mehrere Benutzer mit Zugriff auf ihre eigenen Inhalte	1 (Hoch)
Benutzerverwaltung und Zugriffskontrolle	Das System unterstützt Gruppen von Benutzer mit unterschiedlichen Zugriffsrechten (z.B. Administrator, Redakteur, Betrachter)	1 (Hoch)
Erweiterte Schreibstil-Analyse	Integration zusätzlicher KI-Modelle zur detaillierten Analyse des Schreibstils.	(Hoch)
Erweiterung des Plugin-Systems	Einbindung weiterer externer Analysetools zur Prüfung von Layout, Inhalt und weiteren Kriterien.	2 (Mittel)
Erweiterte Reporting-Funktionen	Entwicklung von Visualisierungen und PDF-Exportfunktionen für Analyseberichte.	3 (Niedrig-Mittel)
Benutzerdefinierte Dashboards	Möglichkeit für Benutzer, individuelle Übersichten und Dashboards zu erstellen.	4 (Niedrig)

Tabelle 3.17 – Zukünftige Erweiterungen

3.2 Nicht-funktionale Anforderungen (NFR)

3.2.1 MVP

Titel / ID	**NFR-1: Performance – Crawler**
Business Goals	Sicherstellen, dass das System grosse Websites (bis 1000 Seiten) innerhalb von ca. 5 bis 15 Minuten crawlen kann, um reibungslose Abläufe und Nutzerzufriedenheit zu gewährleisten.
Relevant Quality Attributes	Performance, Skalierbarkeit (ISO 25010: Performance Efficiency)
Stimulus	Ein Benutzer startet einen Crawl-Vorgang für eine Website mit bis zu 1000 Seiten.
Stimulus Source	Benutzer (Administrator/Redakteur)
Environment	Produktionsumgebung mit stabiler Internetanbindung und ausreichenden Server-Ressourcen.
Artifact	Crawler-Komponente und Datenbank (für Metadaten).
Response	Das System erfasst alle 1000 Seiten.
Response Measure	Q-Ziel: $\leq$ 5 Minuten; Akzeptabel: 15min;
Measure Method	Automatisierte Lasttests, gemessen in Minuten
Questions	Wie skaliert das System bei Überschreitung der 1000 Seiten?
Issues	Mögliche Netzwerk-Latenzen oder Ressourcenengpässe, die die Crawling-Dauer verlängern.

Tabelle 3.18 – Scenario-Refinement für NFR-1 (Performance – Kapazität)

Titel / ID	NFR-2: Usability
Business Goals	Eine intuitive Benutzeroberfläche, die neuen Benutzern das System innerhalb von kurzer Zeit verständlich macht.
Relevant Quality Attributes	Usability, Learnability (ISO 25010: Usability)
Stimulus	Ein neuer Benutzer öffnet die Webanwendung zum ersten Mal.
Stimulus Source	Endnutzer (z.B. Content Manager)
Environment	Verschiedene Browser, jedoch im MVP primär Chrome (vgl. Feedback „nur Chrome für MVP“).
Artifact	Die Web-GUI des Content Analyzers.
Response	Der Benutzer soll in der Lage sein, alle Hauptfunktionen innerhalb einer gegebenen Zeit selbstständig zu nutzen.
Response Measure	Totale Zeit - Ziel: $\leq$ 30 min; Akzeptabel: $\leq$ 1 Stunde;
Measure Method	Usability-Tests, in denen die Zeit bis zur erfolgreichen Bedienung gemessen wird, plus Feedback aus Testsessions
Questions	Sind zusätzliche Hilfestellungen wie Tooltips oder Tutorials notwendig?
Issues	Unterschiedliche Erfahrungswerte der Benutzer, unklare Navigationselemente.

Tabelle 3.19 – Scenario-Refinement für NFR-2 (Usability)

Titel / ID	NFR-7: Portabilität
Business Goals	Gewährleistung einer stabilen Nutzung der Webanwendung in den gängigsten Browsern – im MVP fokussieren wir uns primär auf Chrome.
Relevant Quality Attributes	Portability, Compatibility (ISO 25010: Portability)
Stimulus	Ein Benutzer öffnet die Anwendung in einem Browser.
Stimulus Source	Endbenutzer, die den Browser verwenden (im MVP: Chrome).
Environment	Verschiedene Browser, aber für das MVP wird vorerst nur Chrome unterstützt (mindestens aktuelle stabile Version zur Entwicklungszeit, voraussichtlich Chrome v122 oder höher).
Artifact	Frontend der Webanwendung.
Response	Die Anwendung funktioniert fehlerfrei in gewählten Browsern.
Response Measure	Alle FR können in Chrome erfüllt werden.
Measure Method	FR Tests bei Implementation.
Questions	Welche Browser sollen nach dem MVP unterstützt werden?

Tabelle 3.20 – Scenario-Refinement für NFR-7 (Portabilität)

Titel / ID	NFR-8: Zuverlässigkeit – Verfügbarkeit
Business Goals	Sicherstellung einer hohen Systemverfügbarkeit zur kontinuierlichen Nutzung des Tools.
Relevant Quality Attributes	Reliability, Availability (ISO 25010: Reliability)
Stimulus	Website wird vom Webserver angefordert.
Stimulus Source	Unerwartete Hardware-/Softwarefehler oder Netzwerkprobleme.
Environment	Produktionsumgebung mit Monitoring
Artifact	Gesamtsystem, insbesondere die Webapplikation und Datenbank.
Response	Das System liefert die erwartete Antwort.
Response Measure	Verfügbarkeit von 99 % in normalem Betrieb
Measure Method	Überwachung mittels Monitoring-Tools, Protokolle, regelmässige Auswertung.

Tabelle 3.21 – Scenario-Refinement für NFR-8 (Zuverlässigkeit – Verfügbarkeit)

Titel / ID	**NFR-9: Zuverlässigkeit – Wiederherstellbarkeit**
Business Goals	Das System muss nach einem Ausfall schnell wieder betriebsfähig sein, ohne dass Daten verloren gehen oder der Crawling-Prozess unterbrochen wird.
Relevant Quality Attributes	Reliability, Recoverability (ISO 25010: Reliability)
Stimulus	Systemabsturz oder Unterbrechung des Crawling-Prozesses.
Stimulus Source	Technische Fehler, Softwareabstürze.
Environment	Produktionsumgebung, Notfalltests.
Artifact	Crawler und Datenbank.
Response	Das System setzt nach einem Ausfall in nützlicher Zeit den Crawling-Prozess fort und stellt sicher, dass keine Daten verloren gehen.
Response Measure	Wiederherstellungszeit $\leq$ 15 Minuten
Measure Method	Notfalltests, Auswertung der Wiederherstellungszeit.
Questions	Welche Mechanismen gewährleisten den fortlaufenden Betrieb (z. B. automatische Recovery-Prozesse)? Wie wird sichergestellt, dass Datenverluste vermieden werden?
Issues	Risiko von Datenverlusten und längeren Ausfallzeiten unter extremen Bedingungen. Herausforderungen beim Recovery, besonders in Lastspitzenzeiten.

Tabelle 3.22 – Scenario-Refinement für NFR-9 (Zuverlässigkeit – Wiederherstellbarkeit)

Titel / ID	NFR-10: Leistungseffizienz – Ressourcennutzung
Business Goals	Die Webseite soll zu allen Zeiten vernünftige Ladezeiten aufweisen.
Relevant Quality Attributes	ISO 25010: Performance Efficiency - Resource Utilization
Stimulus	Anfrage der Webapplikation
Stimulus Source	Benutzeraktivität im Frontend
Environment	Ausgelastet durch Analyseprozesse
Artifact	Webserver, Analyseserver
Response	Die entsprechende Antwort wird vom Webserver geliefert
Response Measure	Antwortzeit ist nicht zu unterscheiden von nicht ausgelastetem Server
Measure Method	Auslastungstest

Tabelle 3.23 – Scenario-Refinement für NFR-10 (Leistungseffizienz – Ressourcennutzung)

Titel / ID	NFR-11: Wartbarkeit – Modularität, Modifizierbarkeit
Business Goals	Neue Plugins sollen sich mit minimalem Aufwand integrieren lassen.
Relevant Quality Attributes	ISO 25010: Maintainability - Modularity, Modifiability
Stimulus	Neue Anforderung: Neue Richtlinie muss überprüft werden
Stimulus Source	Projektmanagement
Environment	Normaler Betrieb
Artifact	Gesamtsystem
Response	Ein neues Tool wird per Plugin integriert
Response Measure	Eine Implementation soll innerhalb einer Woche machbar sein
Measure Method	In der Implementation erster Plugins evaluiert

Tabelle 3.24 – Scenario-Refinement für NFR-11 (Wartbarkeit – Modularität, Modifizierbarkeit)

3.2.2 Nicht MVP

Titel / ID	NFR-4: Sicherheit
Business Goals	Schutz sensibler Daten und Vermeidung von unautorisiertem Zugriff.
Relevant Quality Attributes	Security, Confidentiality (ISO 25010: Security)
Stimulus	Ein Angreifer versucht, sich Zugriff auf interne Daten zu verschaffen.
Stimulus Source	Externe Bedrohungen (Hacker, fehlerhafte Konfigurationen).
Environment	Produktivumgebung mit sicherheitskritischen Daten.
Artifact	Authentifizierungs- und Autorisierungsmechanismen der Webanwendung.
Response	Das System verhindert den Zugriff unautorisierter Benutzer zuverlässig.
Response Measure	Fehler-Rate 0.
Measure Method	Penetrationstests und Sicherheits-Audits.
Questions	Wie robust sind die implementierten Sicherheitsmechanismen unter realen Angriffsszenarien?
Issues	Mögliche Sicherheitslücken durch veraltete Bibliotheken oder Konfigurationsfehler.

Tabelle 3.25 – Scenario-Refinement für NFR-4 (Sicherheit)

3.3 Softwarequalität und ISO 25010-Konformität

Der ISO/IEC 25010-Standard definiert Qualitätsmerkmale für Softwareprodukte. Diese Merkmale dienen als Grundlage für die Bewertung der Softwarequalität. Die nachfolgende Analyse ordnet diese Kriterien den relevanten Anforderungen des Projekts *Content Analyzer* zu und definiert messbare Qualitätsziele.

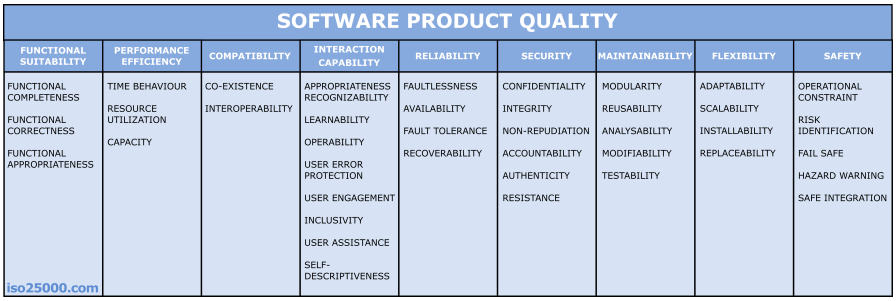


Abbildung 3.1 – Qualitätsmodell nach ISO/IEC 25010 [5].

Abbildung 3.1 zeigt die Hauptqualitätsmerkmale nach ISO/IEC 25010. Die einzelnen Kriterien und ihre Umsetzung im System werden nachfolgend erläutert.

3.3.1 Functional Suitability (Funktionale Eignung)

Functional Completeness (Vollständigkeit)

Alle funktionalen Anforderungen des MVPs werden umgesetzt. Die Erfüllung wird durch automatisierte Testprotokolle überprüft.

Functional Correctness (Korrektheit)

Die Resultate der KI-gestützten Analysen sollen sinnvoll und nachvollziehbar sein. Dies wird anhand von Benchmark-Webseiten mit bekannten Ergebnissen evaluiert. Zusätzlich erfolgt eine Verifizierung durch Nutzerfeedback.

Functional Appropriateness (Angemessenheit)

Die Analysezeit soll nicht mehr als 15 Minuten betragen. Das Ziel ist es den Analyseprozess im Vergleich zu existierenden Tools effizienter zu machen.

3.3.2 Performance Efficiency (Leistungseffizienz)

Time Behaviour (Zeitverhalten)

Die Gesamtanalysezeit wird in die Ladezeit der Webseite und die Verarbeitungszeit der Analyse-Tools unterteilt. Ziel ist eine Ladezeit von unter einer Sekunde.

Resource Utilization (Ressourcennutzung)

Die Leistung der analysierten Webseite darf durch die Analyseprozesse nicht negativ beeinflusst werden.

Capacity (Kapazität)

Das System unterstützt die parallele Ausführung mehrerer Analysetools.

3.3.3 Compatibility (Kompatibilität)

Interoperability (Interoperabilität)

Die Anwendung ist mit externen Tools zur Überprüfung der WCAG-Richtlinien kompatibel. Diese können auch unabhängig von der Anwendung gestartet werden.

3.3.4 Interaction Capability (Benutzbarkeit)

Appropriateness Recognizability (Erkennbarkeit)

Die Benutzerfreundlichkeit wird durch UX-Tests und Nutzerfeedback evaluiert. Benutzer erkennen, ob sie die Zielgruppe der Anwendung sind.

Learnability (Erlernbarkeit)

Neue Benutzer sollen innerhalb einer definierten Zeit typische Aufgaben erfolgreich ausführen können.

Operability (Bedienbarkeit)

Zentrale Aktionen, wie das Starten einer Analyse, sollen in maximal drei Sekunden zugänglich sein.

User Error Protection (Fehlervermeidung)

Der Benutzer wird davon abgehalten Fehler in der Operation der Anwendung zu begehen durch Massnahmen wie Input Validation. In der Implementation des Frontends wird dies in Reviews überprüft.

3.3.5 Reliability (Zuverlässigkeit)

Fault Tolerance (Fehlertoleranz)

Das System soll auch bei Fehlern einzelner Komponenten funktionsfähig bleiben.

Recoverability (Wiederherstellbarkeit)

Nach einem Absturz soll das System sich selbst und die vorherigen Daten wiederherstellen können.

3.3.6 Security (Sicherheit)

Da Sicherheitsaspekte nicht Teil des MVPs sind, werden sie für spätere Versionen dokumentiert.

3.3.7 Maintainability (Wartbarkeit)

Modularity (Modularität)

Neue Plugins sollen sich ohne Beeinträchtigung anderer Komponenten integrieren lassen. Eine Implementation soll innerhalb einer Woche machbar sein, was während der Implementation erster Plugins evaluiert werden soll.

Testability (Testbarkeit)

Jede Komponente soll unabhängig getestet werden können.

3.3.8 Flexibility (Übertragbarkeit)

Scalability (Skalierbarkeit)

Das System soll auch bei wachsender Nutzeranzahl performant bleiben und skalierbar sein.

3.3.9 Referenz zu nicht-funktionalen Anforderungen

Die hier beschriebenen Qualitätsmerkmale stehen in direktem Zusammenhang mit den nicht-funktionalen Anforderungen (siehe Abschnitt *2.1.4 Nicht-funktionale Anforderungen*). Insbesondere decken sich folgende Attribute mit den definierten Szenarien:

- **Performance Efficiency (Leistungseffizienz)** → NFR-1: Performance – Crawler, NFR-3: Performance – Frontend, NFR-10: Leistungseffizienz – Ressourcennutzung
- **Interaction Capability (Benutzbarkeit)** → NFR-2: Benutzerfreundlichkeit
- **Reliability (Zuverlässigkeit)** → NFR-9: Zuverlässigkeit – Wiederherstellbarkeit
- **Security (Sicherheit)** → NFR-4: Sicherheit
- **Maintainability (Wartbarkeit)** → NFR-11: Wartbarkeit - Modularität, Modifizierbarkeit

Diese Metriken werden in der abschliessenden Evaluierung des Systems überprüft.

Kapitel 4

Domain Analysis

In diesem Kapitel wird das Domain Model für das Projekt vorgestellt. Es dient dazu, die zentralen Konzepte und Beziehungen innerhalb der Problemdomäne zu verdeutlichen.

4.1 Domain Model Diagramm

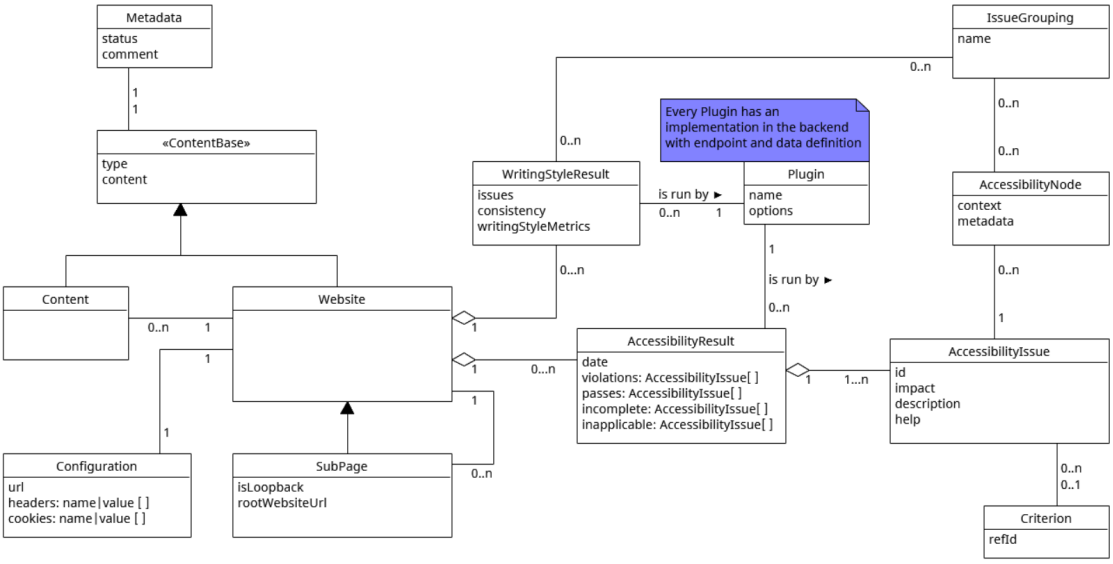


Abbildung 4.1 – Domain Model Diagramm

4.2 Beschreibung der zentralen Domänenklassen

Die folgende Übersicht erläutert die wichtigsten Domänenklassen im System. Sie dient dazu, komplexe Zusammenhänge zu klären und ergänzt das grafische Domain Model in Abbildung 4.1.

- **Website**: Repräsentiert eine analysierte Webseite. Enthält die zugehörigen Inhalte sowie Analyseergebnisse und ist der zentrale Einstiegspunkt für alle nachfolgenden Verarbeitungsschritte.
- **SubPage**: Modelliert einzelne Unterseiten oder dynamische Bestandteile einer Website. Wird verwendet, um verschachtelte oder logisch getrennte Inhaltsbereiche darzustellen.
- **Configuration**: Beinhaltet Konfigurationsdaten wie HTTP-Header und Cookies, die für den Zugriff auf bestimmte Webseitenbereiche erforderlich sind. Jede Analyse greift auf eine spezifische Konfiguration zurück.
- **Content**: Stellt konkrete Inhalte einer Webseite dar (z. B. HTML-Abschnitte, Text, Bilder). Inhalte sind einer Website zugeordnet und können Metadaten besitzen.
- **Metadata**: Enthält zusätzliche Informationen zu einem Inhalt, wie den Bearbeitungsstatus oder redaktionelle Kommentare durch den Benutzer.
- **Plugin**: Abstrakte Darstellung einer Analyse-Komponente. Jedes Plugin führt eine bestimmte Art von Analyse durch, wie z. B. Barrierefreiheit oder Schreibstil.
- **AccessibilityResult**: Ergebnis einer Barrierefreiheitsanalyse. Enthält verschiedene Gruppen von Befunden wie `violations`, `passes`, `incomplete`, usw.
- **AccessibilityIssue**: Einzelner Befund aus einer Accessibility-Analyse. Beinhaltet Informationen wie Beschreibung, Schweregrad, Auswirkung und ggf. Hilfetexte zur Behebung.
- **AccessibilityNode**: Repräsentiert den betroffenen DOM-Knoten bzw. den Kontext eines AccessibilityIssues innerhalb der analysierten Seite. Referenziert auch zugehörige Metadaten. Implementierte Accessibility-Plugins liefern Resultate auf dem Level einzelner DOM-Elemente, weshalb im Gegensatz zu WritingStyleResult hier eine feinere Trennung möglich ist.
- **IssueGrouping**: Dient zur Kategorisierung mehrerer AccessibilityNodes, z. B. nach Typ oder Regelwerk. Erlaubt gruppierte Darstellung im Frontend für z.B. Arbeitszuteilung.
- **Criterion**: Bezieht sich auf ein extern definiertes Kriterium (z. B. WCAG-Richtlinie), zu dem ein AccessibilityIssue gehört. Ermöglicht eine normgerechte Einordnung.
- **WritingStyleResult**: Enthält das Ergebnis der Schreibstilanalyse einer Webseite. Dazu zählen Kennzahlen wie Lesbarkeit, Konsistenz und erkannte stilistische Auffälligkeiten. Interne Klassen wie issues oder writingStyleMetrics sind im Diagram ausgelassen, da sie keine weiteren Beziehungen haben

Kapitel 5

Architektur

Die Architektur Dokumentation folgt der Arc42 Struktur. Themen, die schon an anderen Orten ausführlich behandelt wurden, werden lediglich referenziert. Hauptsächlich Anforderungen und Qualitätsziele.

5.1 Qualitätsziele und Anforderungen

Das Ziel der Anwendung soll es sein den Inhalt existierender Webseiten zu durchsuchen, in einer Webapplikation einen Index dazu darzustellen und über diesen den Inhalt nach WCAG-Richtlinien zu analysieren. Spezifischen Funktionalen Anforderungen sind in im Kapitel 3.1 aufgelistet.

Die Architektur ist so gewählt, dass sie die wichtigsten angestrebten Qualitätsziele und Anforderungen unterstützt. Die wichtigsten Qualitätsziele sind:

- NFR-11: Wartbarkeit – Modularität, Modifizierbarkeit
- NFR-10: Leistungseffizienz – Ressourcennutzung
- NFR-1: Performance – Crawler
- NFR-3: Performance – Frontend

5.2 Randbedingungen

Aufgrund der Natur des Projektes existieren gewisse Vorgaben.

- Es wird eine Webapplikation vorausgesetzt, Remote User Interface.
- Inhalte sollen persistent sein, eine Datenbank wird benötigt.
- Existierende Analysetools zur Accessibility wie Axe Core sollen unterstützt werden.

- Da die Verwendung eines Web-Scrapers eine zwingende Anforderung ist, muss dieser entweder mit der Backend-Programmiersprache kompatibel sein oder alternativ über eine z.B. Web-Schnittstelle ansprechbar sein.
- Risiko - End of life von Dependencies: Verwendete Bibliotheken sollen Open-Source sein.

Projekt-Einschränkungen

- Die maximale verfügbare Zeit für die Implementierung ist 10 Wochen.
- Das Team besteht aus 2 Mitgliedern.

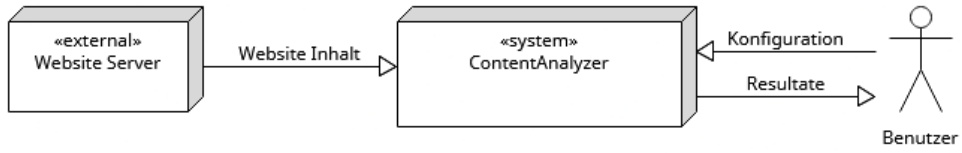
5.3 Kontextabgrenzung

Business

Nachbar	Input	Output
Benutzer	Gibt URL und notwendige Konfiguration vor	Erhält Analyseresultate

Technologie

Das System kommuniziert mit den, von Benutzern vorgegebenen Servern, um den Inhalt von Webseiten zu erhalten.



Lösungsstrategie

Die Lösungen in 5.1 wurden gewählt, um definierte Qualitätsziele zu erreichen. Die getroffenen Entscheidungen und weitere Ausführungen dazu können in Abschnitt 5.8 gefunden werden.

Technologien

Die folgenden Technologien wurden gewählt für die groben Schichten der Architektur:

- Frontend: Typescript, React.js 5.8.4

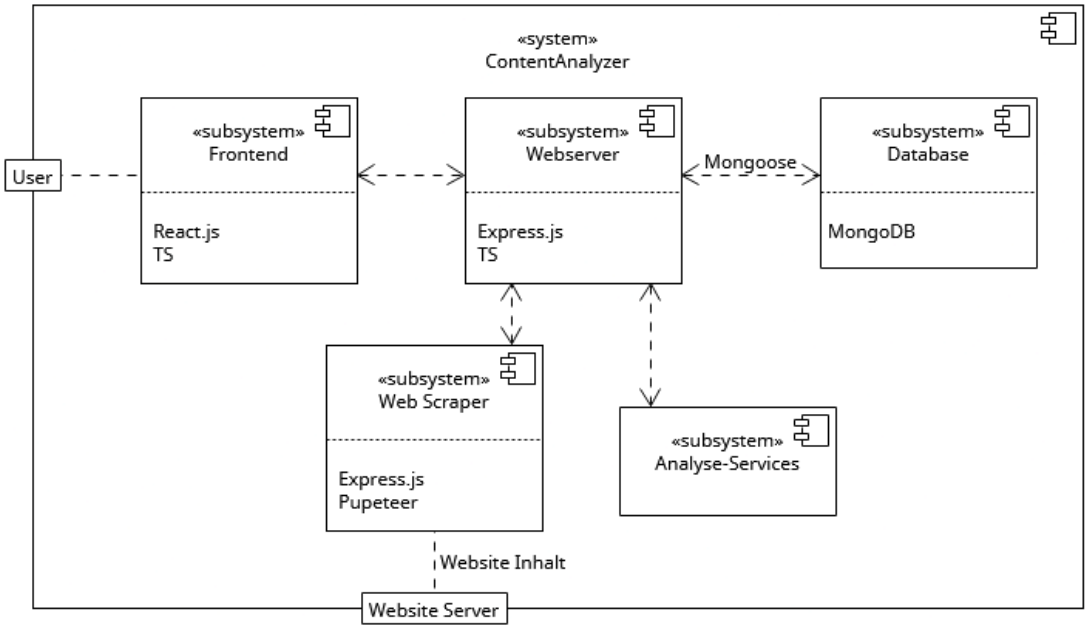
Ziel	Lösung	Verweis zu Details
NFR-11	Abstraktion zu Plugins für Modularität	MVP-5
NFR-10	Unabhängiger Server für Plugins und Web Scraper	ADR-2
NFR-1	Wahl einer genügend performanten Web Scraping Technologie	ADR-1
NFR-3	Unabhängiger Server für die Website um individuell zu skalieren	ADR-2

Tabelle 5.1 – Lösungsstrategie

- Backend: Express.js mit TypeScript. Mongoose für Datenbank-Verbindung 5.8.5
- Datenbank: MongoDB (FPR Object Storage) 5.8.6
- Web Scraper: Node.js + Puppeteer 5.8.1

5.4 Bausteinsicht

Level 1



Subsystem	Beschreibung
Frontend	Eine Typescript basierte React.js Webapplikation, die durch den Webserver ausgeliefert wird.
Webserver / Backend	Express.js RESTful API-Server für die Webapplikation. Dient als Webserver, der diese ausliefert. Koordiniert zwischen allen Backend-Komponenten und Services.
Datenbank	MongoDB Object Storage für Metadaten und Baumstrukturen von Webseiten. Unterstützung flexibler und semi-strukturierter Datenmodelle.
Web Scraper	Node.js basierter Service mit Puppeteer, der Inhalte von Webseiten von externen Servern abruft, parst und transformiert um Informationen zu dem Server zu liefern.
Analyse-Services	Microservices, die Modulare Plugins zur inhaltlichen Analyse von Webseiten bereitstellen. Dazu gehört auch die Schreibstil-Analyse, welche sprachliche Merkmale bewertet.

Level 2

Die folgenden Übersichten sollen einen Überblick über die Struktur der Subsysteme (Untersysteme) schaffen.

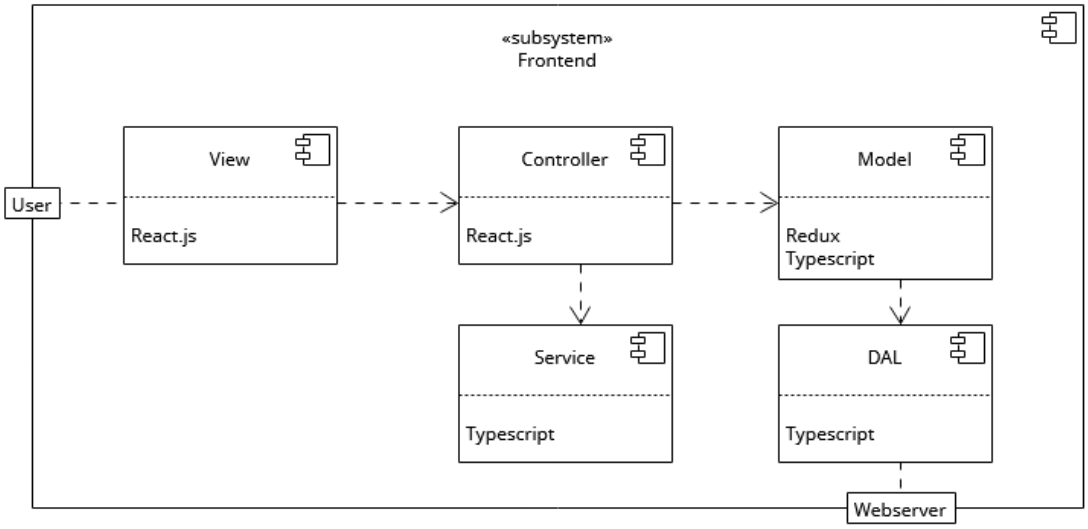
Gestrichelte Pfeile repräsentieren technische Abhängigkeiten zwischen Komponenten. Datenfluss findet in beide Richtungen statt, sofern Linien vorhanden sind, ausser falls anders spezifiziert.

Frontend

Beim Frontend handelt es sich um eine Webapplikation geschrieben in Typescript, die auf React.js basiert. Ein MVP Pattern wird angestrebt in dem Redux als Model verwendet wird, React als Controller und View Teil, wobei zusätzlich möglichst wenig Logik in den individuellen React View-Komponenten verwendet werden soll. Es wird Dependency Injection verwendet, um die Schichten voneinander zu entkoppeln.

Rolle und Verantwortlichkeiten:

- Interaktionspunkt des Endbenutzers mit dem gesamten System. Kommuniziert direkt mit dem Webserver.
- Präsentiert dem Benutzer die Informationen und Zustände des Systems.



Webserver

Der Webserver nutzt Express.js als Framework, das auf Node.js läuft. Als Sprache wird Typescript benutzt.

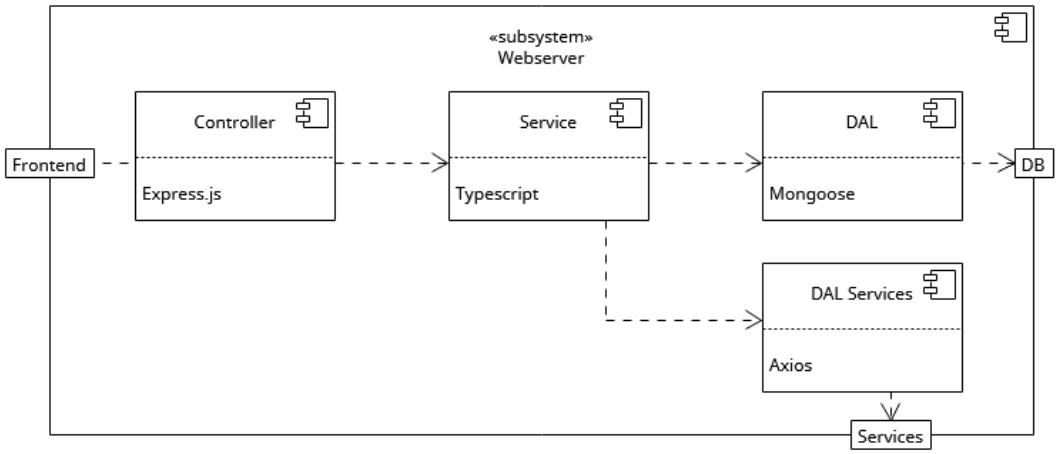
Er folgt einer Schichtenarchitektur mit Verantwortlichkeiten unterteilt in Controller, Service und Data Access Layer.

- **Controllers:** Repräsentieren API-Endpunkte und sind verantwortlich für Authentifizierung, Autorisierung, sowie das Delegieren von Anfragen an die entsprechenden Services.
- **Services:** Implementieren die Business Logik und kommunizieren mit dem Data Access Layer (DAL) oder anderen Services.
- **Data Access Layer (DAL):** Zuständig für den Erhalt und die Persistierung von Daten in Datenbanken oder anderen Services. Als ORM wird hier Mongoose für MongoDB verwendet.

Es wird Dependency Injection zwischen den Schichten verwendet.

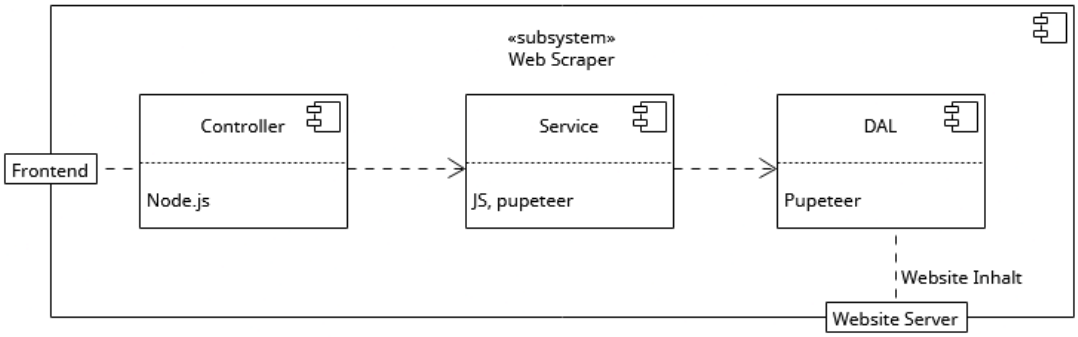
Rolle und Verantwortlichkeiten:

- Exponiert eine RESTful API, über die die Frontend-Applikation Scraping-Jobs initiiert und Ergebnisse abruft.
- Dient als einziger Eintrittspunkt aller Anfragen für das gesamte Backend System.
- Koordiniert den Ablauf der Scraping-Prozesse, indem es Aufgaben an den Web Scraper delegiert und Ergebnisse aus der Datenbank abruft.



Web Scraper

Die Web Scraper Komponente führt den kompletten Web-Scraping-Prozess durch und liefert dem Webserver strukturierte, transformierte Daten. Er nutzt Express.js als Framework und verwendet hauptsächlich Puppeteer für die Funktionalität. Der Web Scraper folgt derselben Schichtenarchitektur wie der Webserver, wobei der Data Access Layer hier wegfällt, beziehungsweise durch Pupeteer abstrahiert wird, das direkt mit externen Webseiten kommuniziert.



Analyse-Services

Die Analyse-Komponenten sind modular als Services implementiert und über REST-APIs angebunden. Sie erhalten die erfasste Webseitenstruktur oder spezifische Daten direkt vom Backend und führen daraufhin eine Analyse der jeweiligen Website durch.

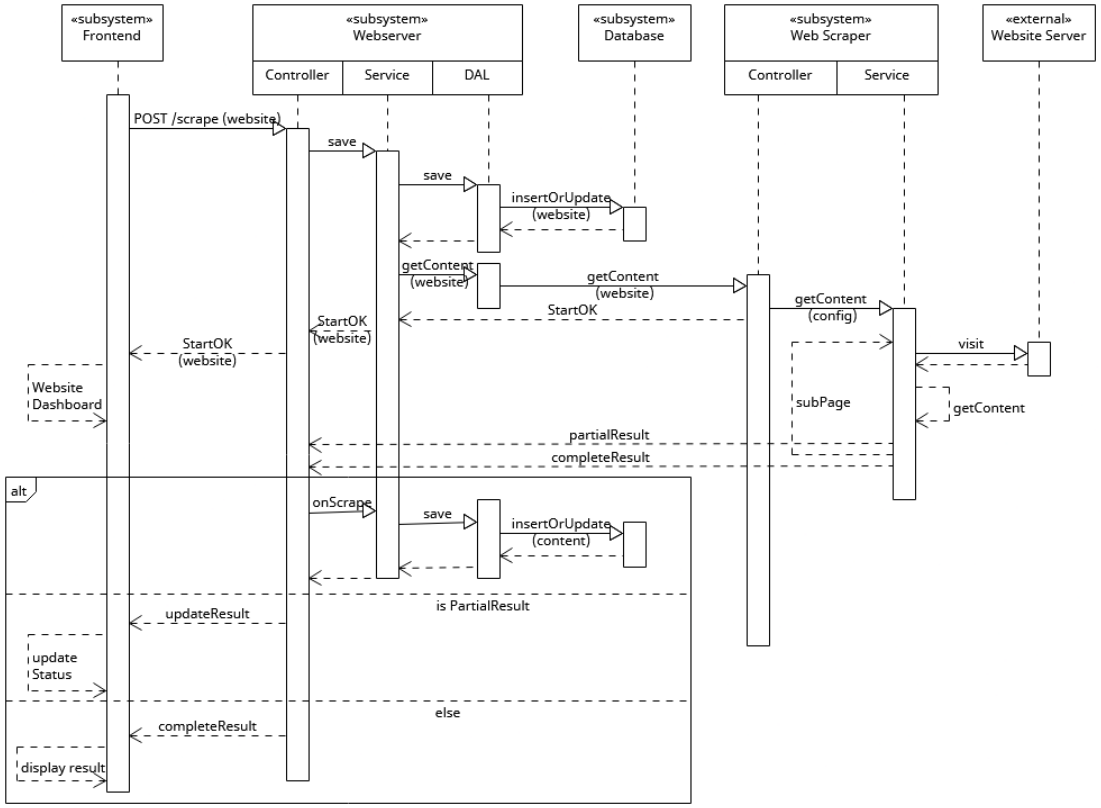
Ein Bestandteil dieser Schicht ist die **Schreibstil-Analyse**, die sprachliche Eigenschaften der Webseiteninhalte bewertet. Der Benutzer kann im Frontend zwischen verschiedenen Analysearten wählen. Die Services sind containerisiert und können unabhängig vom Hauptsystem betrieben werden.

Rolle und Verantwortlichkeiten:

- Eigenständige Analyse von Webseiten basierend auf erhaltenen Daten.
- Durchführung spezifischer Prüfungen (z. B. Barrierefreiheit, Schreibstil).
- Rückgabe strukturierter Analyseergebnisse an das Backend.
- Erweiterbar über ein Plugin-System.

5.5 Laufzeitsicht

Das folgende Beispiel illustriert, wie die Komponenten zur Laufzeit miteinander interagieren:



Kommunikationsfluss

1. **Initiierung:** Der Benutzer gibt über das Frontend eine URL sowie Konfigurationen ein.
2. **Job Dispatching:** Der Backend API Server validiert die Eingabe, speichert die Konfiguration in der Datenbank und startet einen Scraping-Job.
3. **Job Start:** Der Web Scraper Service übernimmt den Job, und gibt zurück, dass dieser erfolgreich gestartet hat.

4. **Job Status Anzeige:** Das Frontend wird informiert, dass der Scraping Job gestartet wurde und zeigt eine Übersicht über die Website an.

5. **Scraping-Prozess:** Der Web Scraper ruft den Webseiteninhalt ab, extrahiert und transformiert die Daten zur Rückgabe an den Webserver. Er liefert laufend Status Updates an das Backend.

6. **Datenspeicherung:** Die strukturierten Daten der Status Updates werden vom Webserver in der MongoDB abgelegt.

7. **Statusupdate:** Die partiellen Resultate werden an das Frontend weitergeleitet, dessen Status entsprechend aktualisiert wird.

8. **Ergebnisbereitstellung:** Das Backend und dann das Frontend wird über den Abschluss des Jobs informiert. Sie werden im Frontend angezeigt.

5.6 Verteilungssicht

Frontend

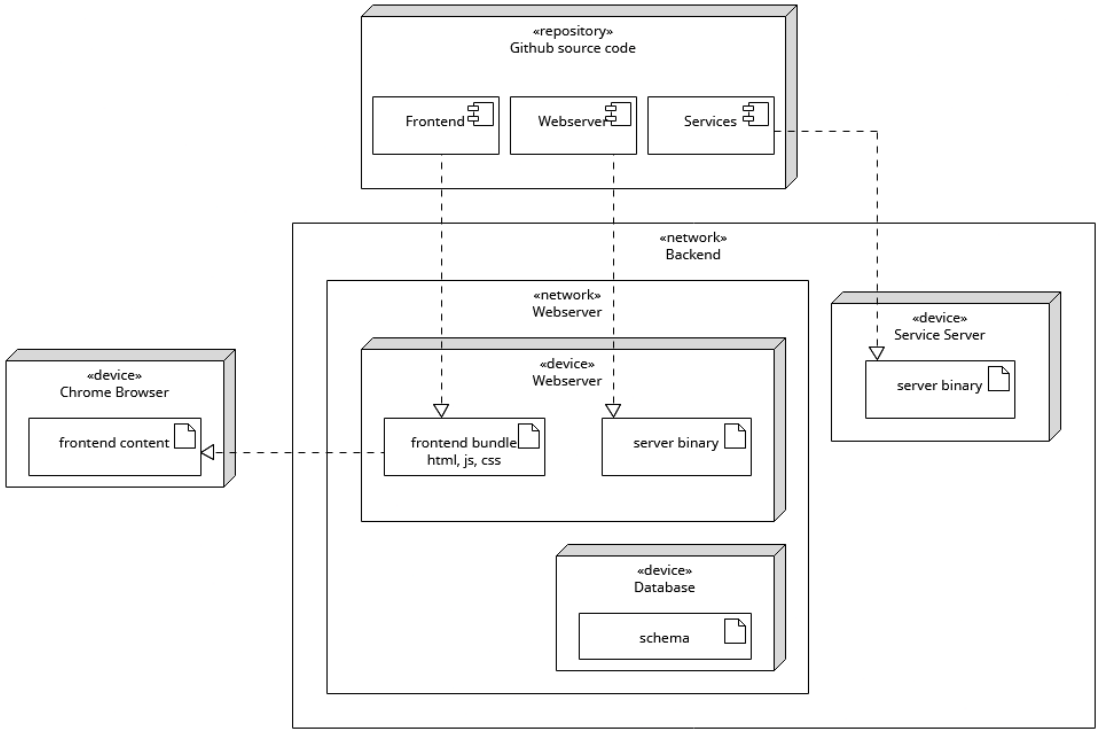
Das Frontend wird als Ordnerstruktur mit HTML, CSS und Javascript auf dem Webserver bereitgestellt. Es wird mit Webpack gebuildet und in eine einzige JS-Datei bzw. CSS-Datei gebündelt. Diese und die statischen Ressourcen wie HTML und Bilder werden den Benutzern durch den Webserver ausgeliefert.

Backend

Das Deployment erfolgt manuell über Github Actions. Die Komponenten werden kompiliert und als Binärdateien auf die jeweiligen Server verteilt.

Datenbankmigrationen werden beim Startup des Webservers durchgeführt.

Das Backend ist in zwei Netzwerkschichten unterteilt: Nur der Webserver ist öffentlich zugänglich und kommuniziert intern mit den Services sowie der Datenbank. Die Datenbank ist ausschliesslich über den Webserver erreichbar.



5.7 Konzepte

Die meisten relevanten Konzepte sind in ihren eigenen Kapiteln behandelt. Ein Überblick:

- Für Konzepte zu Frontend und Backend, siehe deren entsprechende Kapitel in Abschnitt 5.4.
- Für einen Überblick über die Domain Models siehe Abschnitt 4.1.
- UI-Skizzen für das Frontend können in Abschnitt 6 gefunden werden.

5.8 Entscheidungen

Entscheidungen werden in der Form von Architecture Decision Records festgehalten.[1]

5.8.1 ADR 1 - Wahl einer Web-Scraping-Technologie

Im Rahmen der Wahl einer Web Scraping Technologie, bei der wir auf neue Herausforderungen in Web Scraping (siehe Ausgangslage) gestossen sind, haben wir entschieden,

[1] Architecture Decision Record (ADR) Template, Y-Template von cards42.org

Puppeteer mit Node.js als Web-Scraping-Technologie zu verwenden und einen eigenen Algorithmus zu implementieren. Diese Wahl ermöglicht es, dynamisches HTML sowie HTTP-Headers effektiv zu handhaben und dabei eine gute Performance zu erzielen [8].

Dafür entschieden wir uns gegen andere Optionen, wie Pyppeteer, Selenium, Playwright und Scrapy, wie im Abschnitt beschrieben.

Vielen existierenden Bibliotheken oder Tools für Web-Scraping wie dem Axe-Crawler fehlen essenzielle Fähigkeiten, wie der Umgang mit dynamischem HTML oder Möglichkeiten Headers und Cookies zu setzen, weshalb wir uns gegen diese entschieden haben. Bei anderen Tools wie Crawlee wurde der Aufwand zu überprüfen, ob alle Anforderungen abgedeckt werden können zusammen mit dem Konfigurationsaufwand als zu hoch eingeschätzt, verglichen mit den Vorteilen der Nutzung des Tools. Der Grossteil der Logik, die einen Web-Scraper ausmacht, muss auch bei solchen Tools selbst implementiert werden und die Konfigurationsmöglichkeiten schränken die Flexibilität ein.

5.8.2 ADR 2 - Client-Server-Cut Web Scraper

Im Rahmen der Wahl von Client-Server-Cuts haben wir uns für einen Distributed Application Kernel entschieden, bei dem der Web Scraper und Analysetools getrennt vom Webserver der Webapplikation operieren. Dies verbessert die Verfügbarkeit, Performance und ermöglicht die Unabhängigkeit der Programmiersprachen, bringt jedoch erhöhte Komplexität.

5.8.3 ADR 3 - Client-Server-Cut Datenbank

Im Rahmen der Wahl von Client-Server-Cuts haben wir uns für eine Remote Database entschieden, die ausschliesslich mit dem Webserver der Webapplikation kommuniziert. Dies ermöglicht bei Bedarf eine Kommunikation mit anderen Application Kernel bzw. Services. Dafür akzeptieren wir zusätzlichen Netzwerkaufwand.

5.8.4 ADR 4 - Wahl einer Frontend-Technologie

Für die Frontend-Technologie haben wir uns für React.js mit Typescript entschieden. Da die Webapplikation eine begrenzte Komplexität besitzt, ist ein minimaleres Framework wie React gut geeignet und allen im Team bekannt. Dafür haben wir uns gegen Angular und Vue.js oder Server-Side-Rendering Lösungen entschieden.

5.8.5 ADR 5 - Wahl einer Backend-Technologie

Bei der Wahl einer Backend-Technologie haben wir uns für Express.js entschieden. Dies erlaubt es dieselbe Sprache in Front-, sowie Backend zu verwenden, womit das benötigte Wissen, um am Projekt zu arbeiten verringert wird und die Reusability von Code erhöht wird. Dafür akzeptieren wir limitierte Skalierbarkeit und leichte Performance Einbussen im Vergleich zu anderen Backend Technologien.

5.8.6 ADR 6 - Wahl einer Datenbank-Technologie

Im Rahmen der Wahl einer Datenbank-Technologie mit einem existierenden Backend in Node.js haben wir uns für MongoDB entschieden. Um einfacher Baumstrukturen von Webseiten speichern zu können, haben wir uns für eine Dokumenten orientierte Datenbank, entschieden und gegen eine relationale. Damit soll der Entwicklungsaufwand im OR Mapping minimiert werden. Dafür akzeptieren wir den initialen Lernaufwand einer neuen Technologie.

Kapitel 6

UX-Design und Benutzerfluss

Das UX-Design der Anwendung wurde initial in Figma entworfen, um eine intuitive und effiziente Benutzererfahrung zu gewährleisten. Die folgende Darstellung des Benutzerflusses basiert auf diesen Konzepten. Die finale Implementierung kann in einigen Punkten vom ursprünglichen Design abweichen.

Der vollständige Prototyp ist über folgenden Link einsehbar:

Figma-Prototyp: ContentAnalyzer-Design

In diesem Kapitel werden zusätzlich Screenshots der umgesetzten Anwendung verwendet, um die tatsächliche Benutzerführung zu illustrieren

6.1 Benutzerfluss

Die Anwendung führt den Benutzer durch alle relevanten Schritte zur Konfiguration, Analyse und Auswertung einer Website. Der typische Ablauf gestaltet sich wie folgt:

1. **Startseite**: Der Benutzer sieht eine tabellarische Übersicht aller bereits analysierten Webseiten (siehe Abbildung 6.1). Hier kann er eine bestehende Analyse öffnen, eine neue Website über *Create new Website* hinzufügen oder über das Stift-Symbol den Namen und/oder die Konfiguration einer Website bearbeiten (siehe Abbildung 6.2).

2. **Neue Website anlegen**: Nach dem Klick auf *Create new Website* kann eine neue Website-URL in das Eingabefeld eingetragen werden (siehe Abbildung 6.3). Optional lassen sich Analysekonfigurationen in Form von HTTP-Headern und Cookies hinzufügen (siehe Abbildung 6.4), z. B. für den Zugriff auf geschützte Bereiche.

3. **Inhalte sammeln**: Nach dem Start der Datenerfassung wird der Benutzer zum Dashboard weitergeleitet. Dort zeigt eine Ladeanimation den Fortschritt beim Sammeln der Webseiteninhalte an (siehe Abbildung 6.5). Inhalte werden gesammelt und verarbeitet. Nach Abschluss erscheint die Auswertung im Dashboard (siehe Abbildung 6.6).

4. **Inhaltsübersicht**: Die erfassten Inhalte werden hierarchisch in einer Baumstruktur dargestellt. Die Kachel-Ansicht erlaubt zusätzlich eine Filterung nach Inhaltstypen (siehe Abbildung 6.15).

5. **Barrierefreiheitsanalyse**: Der Benutzer kann ein Plugin aktivieren, um eine Accessibility-Prüfung durchzuführen (siehe Abbildung 6.6).

6. **Schreibstilanalyse**: Im Tab *Writing Style* kann der Benutzer die Schreibstilanalyse starten, ein Modell sowie die Sprache auswählen (siehe Abbildung 6.7).

7. **Ergebnisübersicht**: Nach abgeschlossener Analyse werden die Resultate im Dashboard angezeigt, je nach Typ (siehe Abbildungen 6.9 und 6.8).

8. **Detailansicht der Ergebnisse**: Über das Detail-Icon oder den Button *Detaillierte Analyse* können Analyseergebnisse im Detail eingesehen werden (siehe Abbildungen 6.10 und 6.11).

9. **Gruppenmanagement**: Über die Navigation gelangt man zur Verwaltung von Gruppen, um Analyseergebnisse zu organisieren (siehe Abbildung 6.12).

10. **Elemente zu Gruppen hinzufügen**: Ergebnisse lassen sich einzelnen Gruppen zuordnen. Ein entsprechender Dialog erscheint über das Gruppen-Icon an einem HTML-Element (siehe Abbildung 6.13).

11. **Metadaten bearbeiten**: Sowohl auf Website- als auch auf Inhaltsebene lassen sich Metadaten wie Status oder Kommentare anpassen (siehe Abbildung 6.14).

6.2 Bildschirmansichten

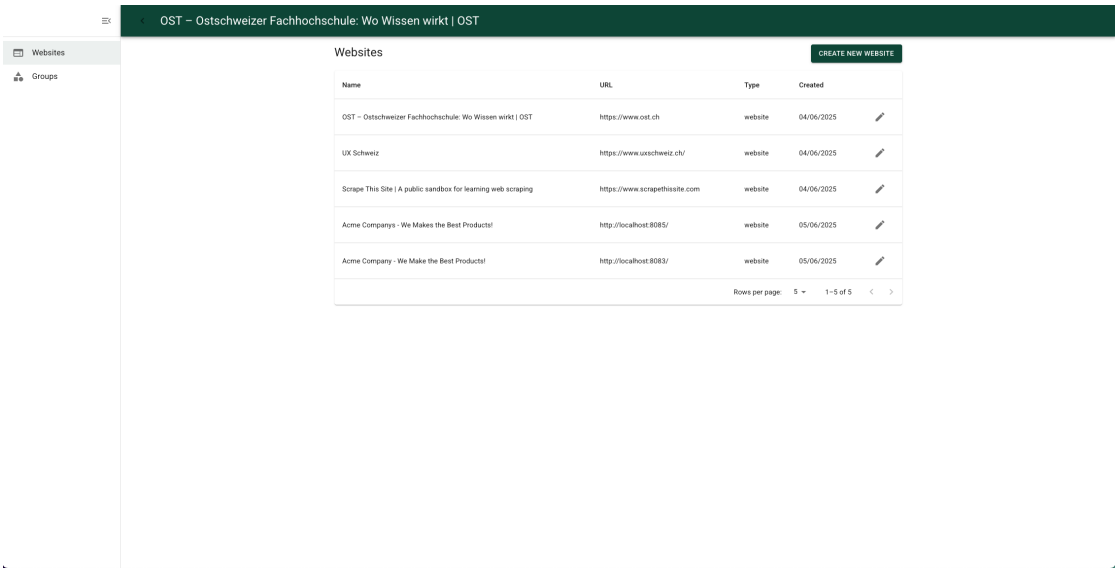


Abbildung 6.1 – Startseite der Anwendung

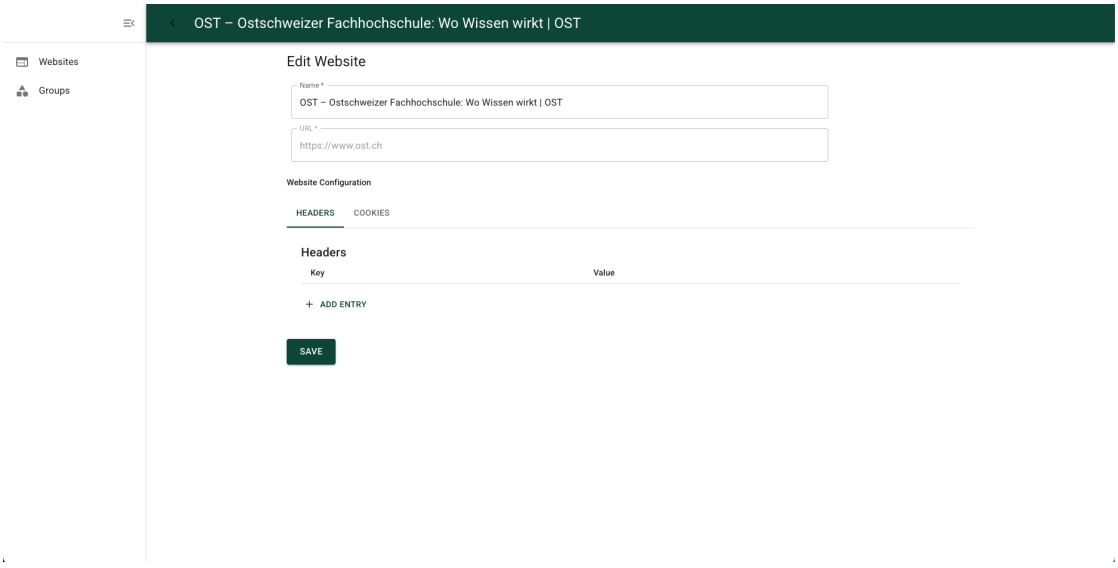


Abbildung 6.2 – Website bearbeiten

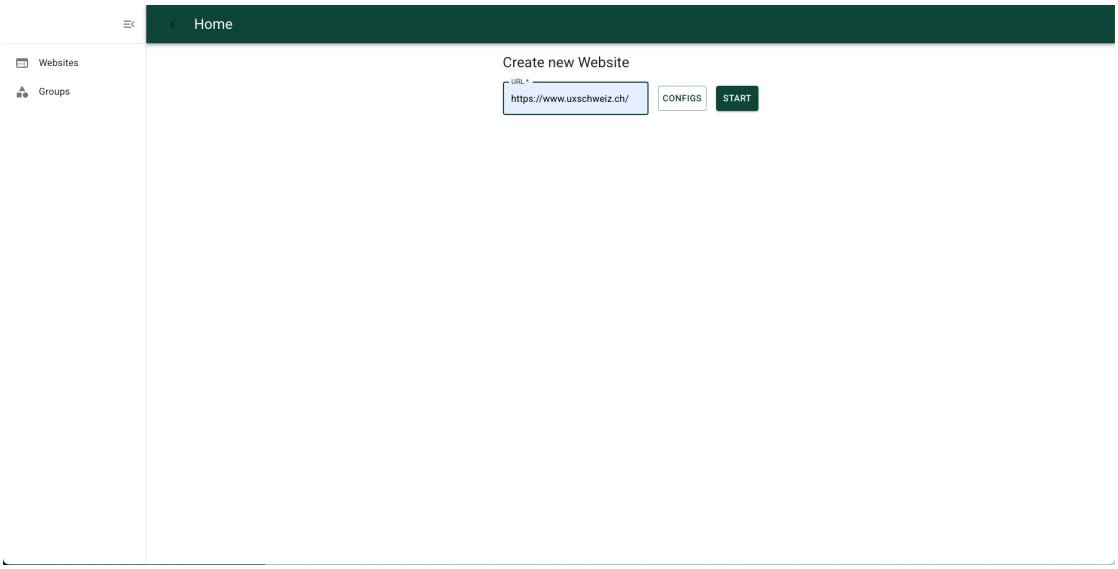


Abbildung 6.3 – Neue Website hinzufügen

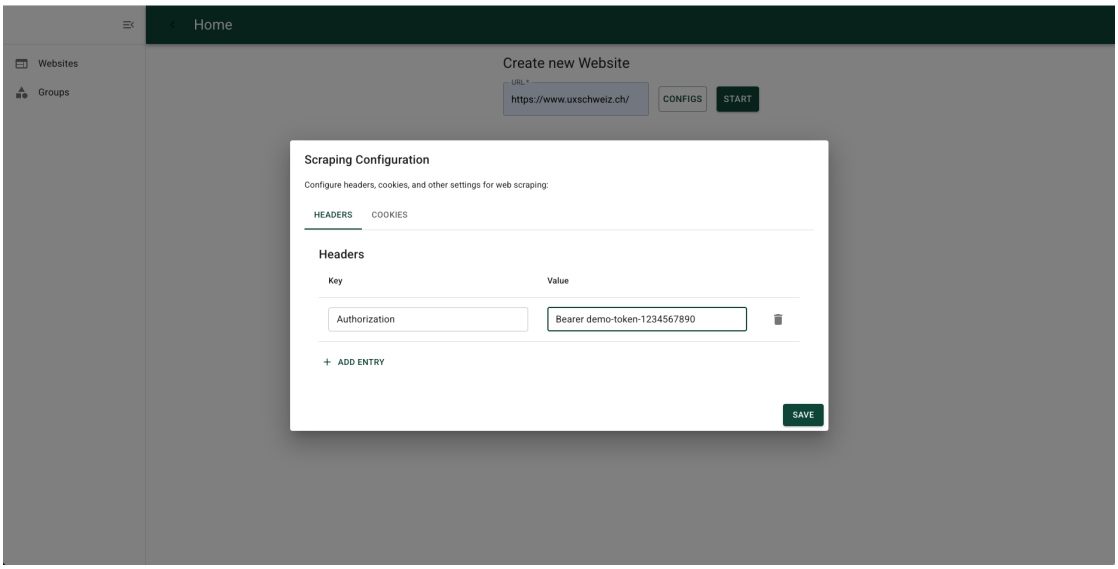


Abbildung 6.4 – Konfigurationsdialog

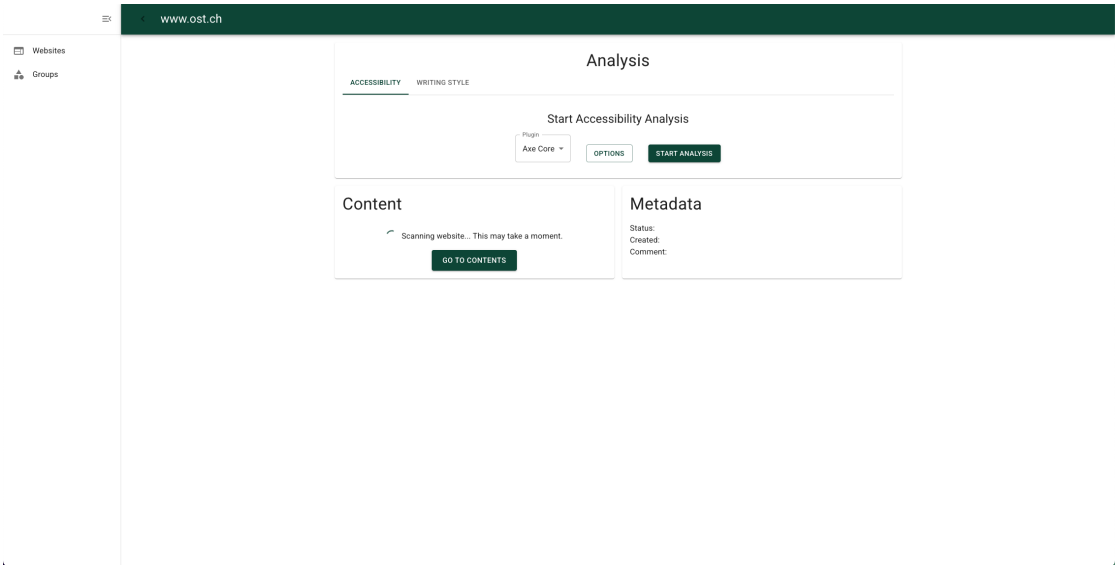


Abbildung 6.5 – Analyseprozess mit Ladeanimation

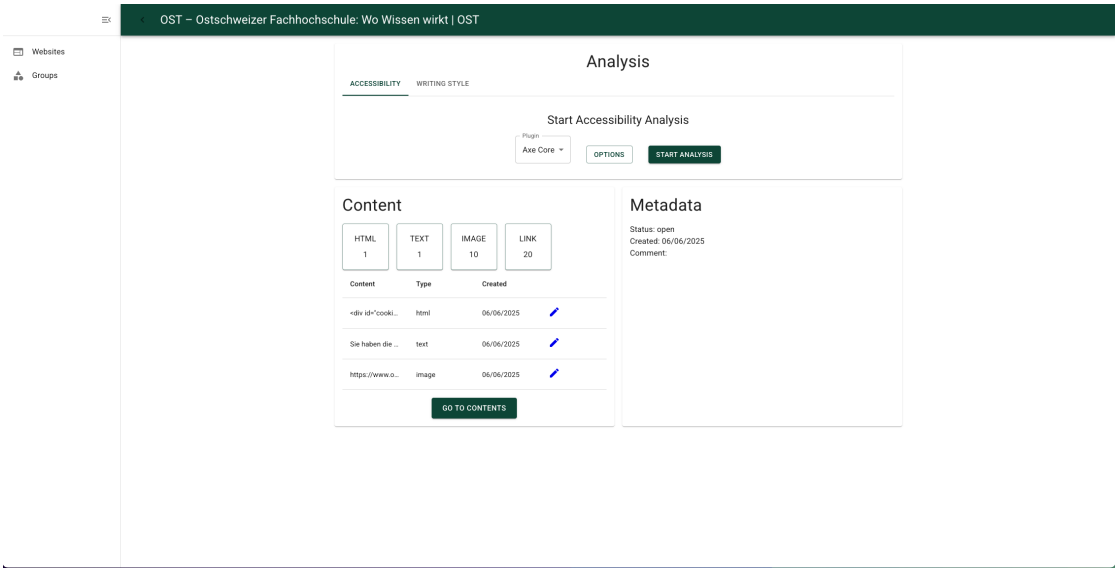


Abbildung 6.6 – Ergebnisse des Scraping-Prozesses

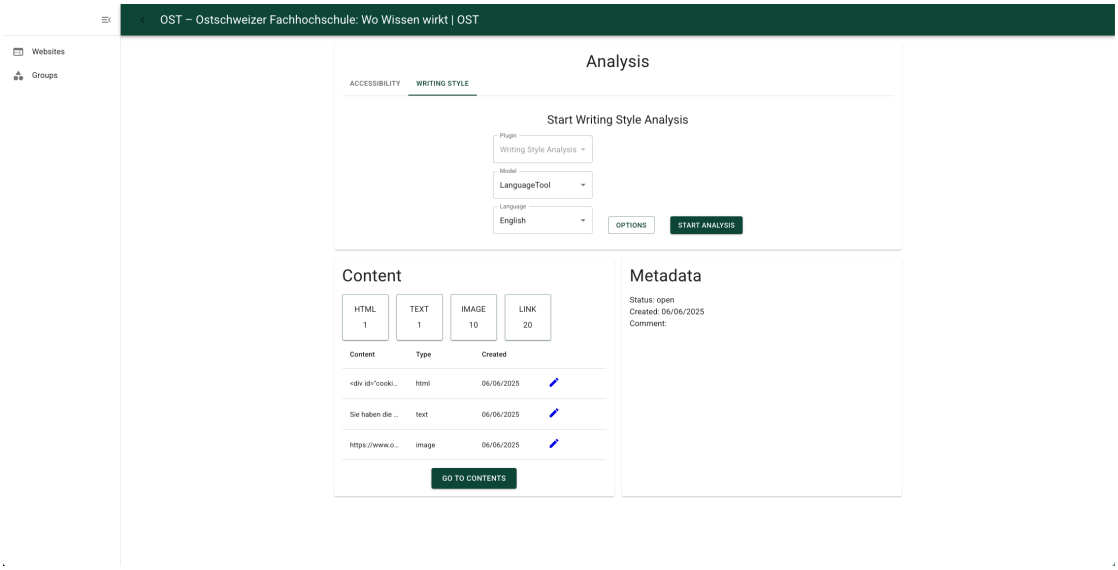


Abbildung 6.7 – Schreibstilanalyse starten

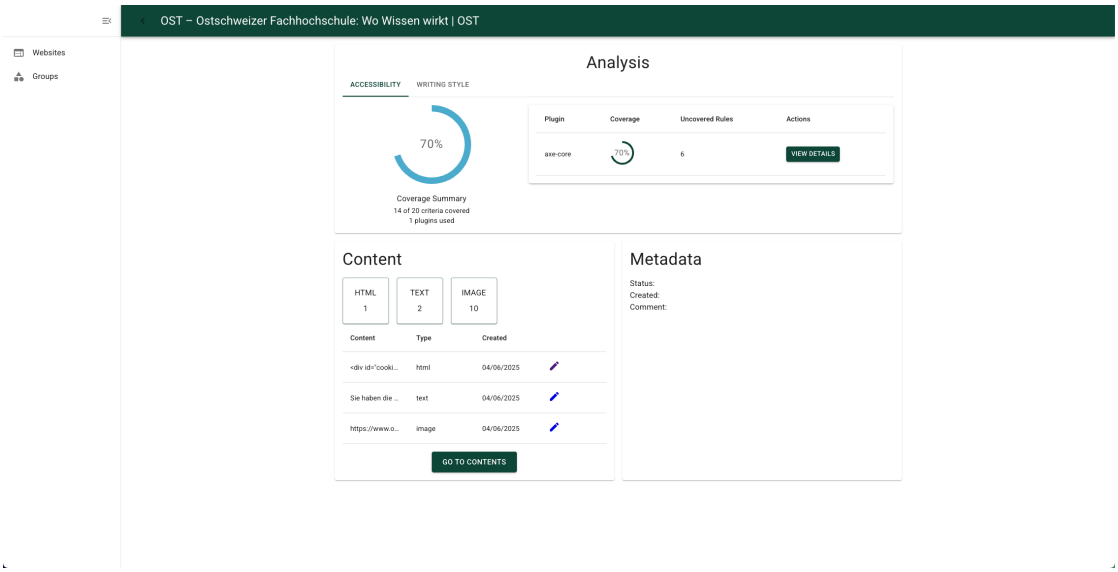


Abbildung 6.8 – Ergebnisse der Barrierefreiheitsanalyse

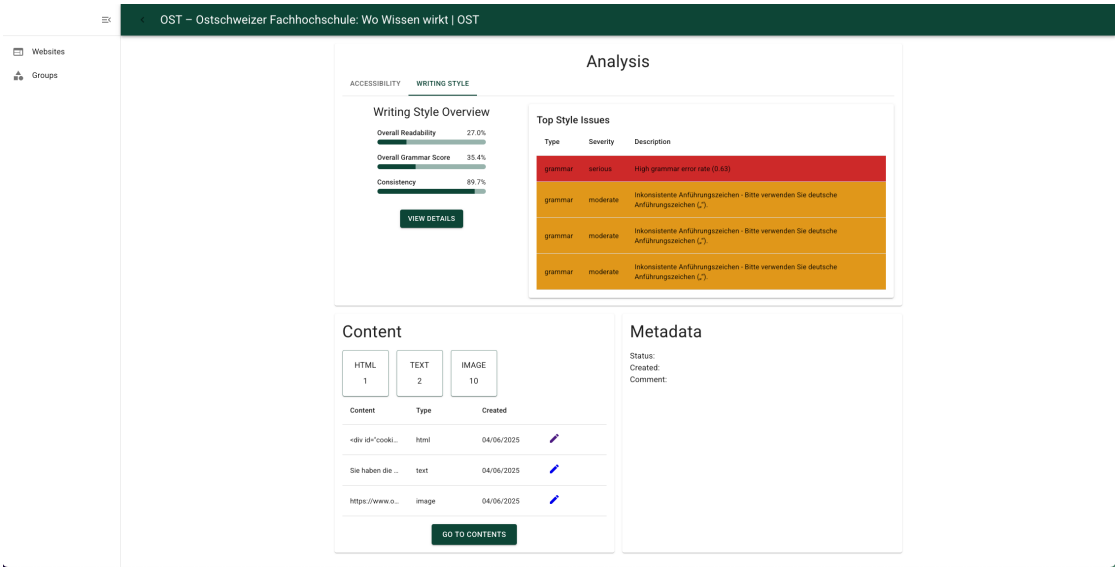


Abbildung 6.9 – Ergebnisse der Schreibstilanalyse

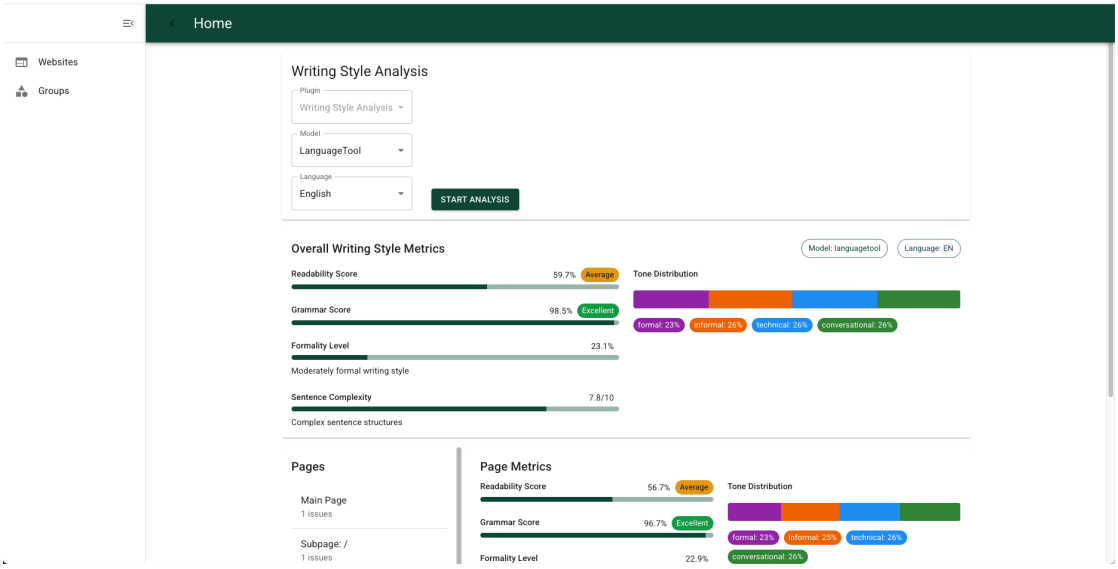


Abbildung 6.10 – Detaillierte Ergebnisse der Schreibstilanalyse

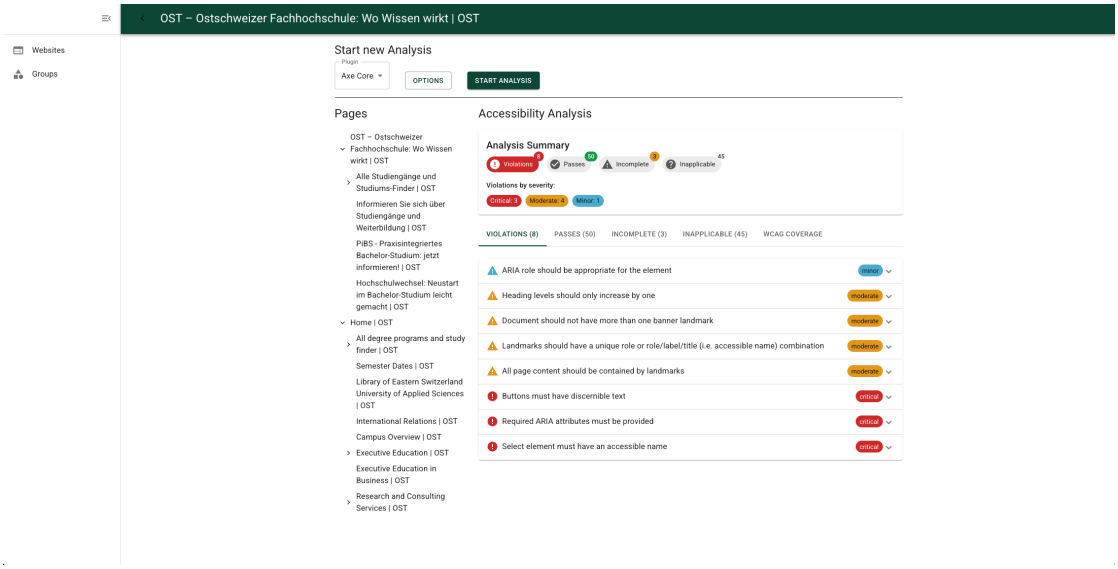


Abbildung 6.11 – Detaillierte Ergebnisse der Barrierefreiheitsanalyse

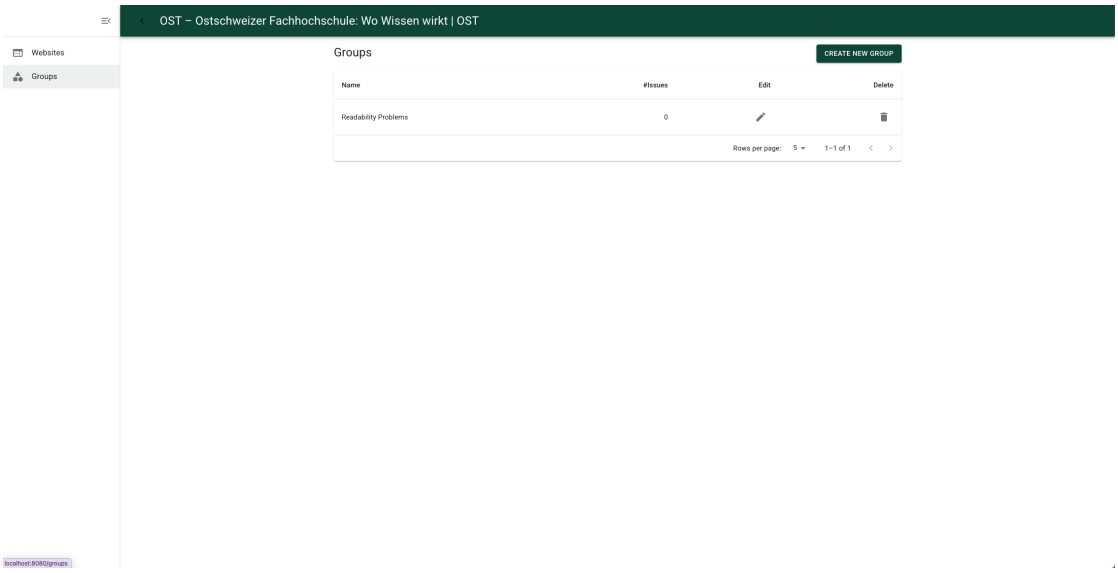


Abbildung 6.12 – Gruppenmanagement

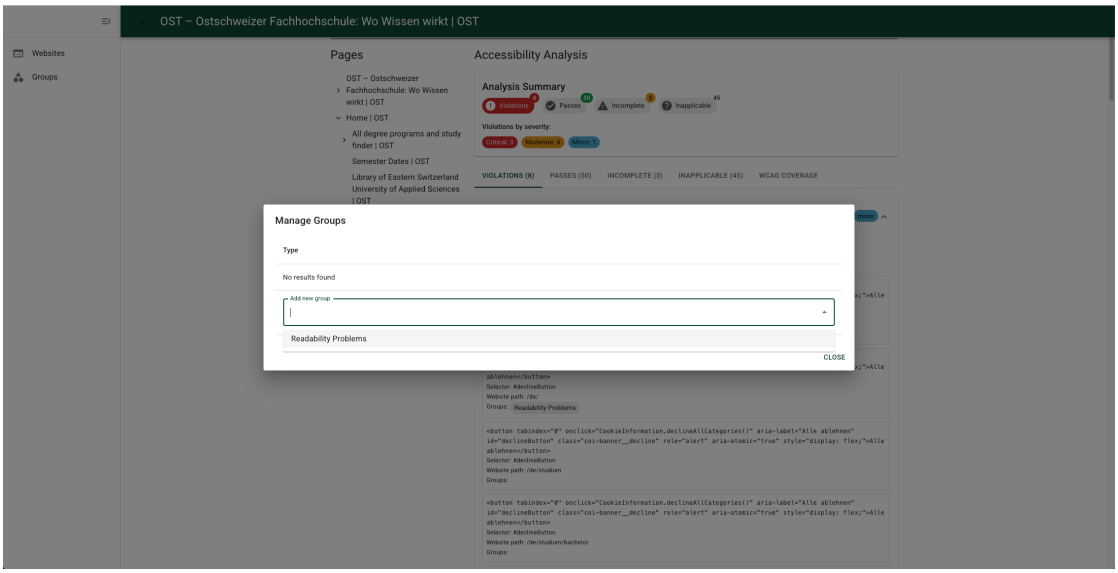


Abbildung 6.13 – Dialog zum Hinzufügen von Inhalten zu einer Gruppe

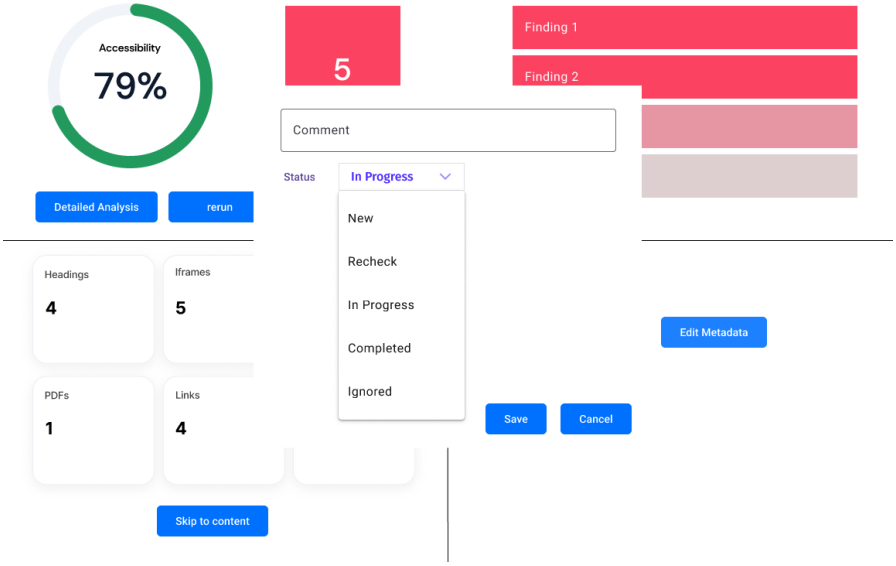


Abbildung 6.14 – Metadaten bearbeiten

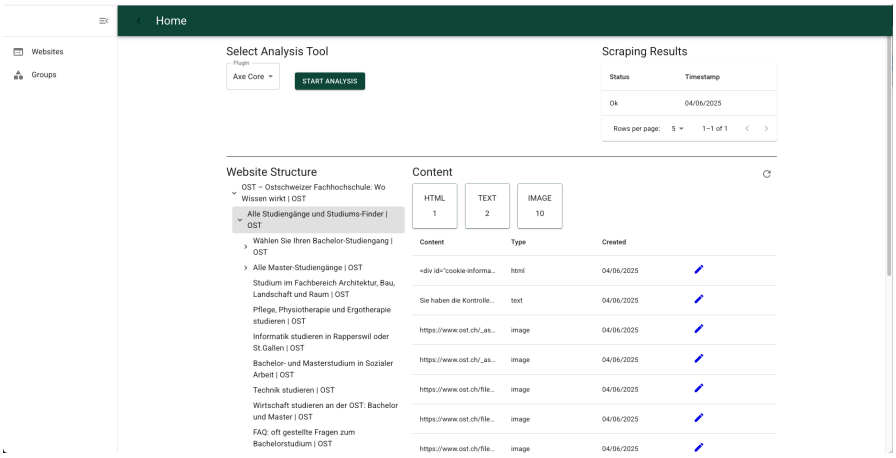


Abbildung 6.15 – Detailansicht für Inhalte

Kapitel 7

Implementation

Die Implementation des Content Analyzers erfolgte auf Basis einer modernen, modularen Architektur, die verschiedene Dienste und Technologien integriert, um eine flexible und leistungsfähige Lösung für die Analyse von Webinhalten zu schaffen. Im Folgenden wird die technische Umsetzung des Projekts detailliert beschrieben, wobei auf die einzelnen Komponenten, deren Zusammenspiel und die verwendeten Technologien eingegangen wird.

7.1 Architekturübersicht

Eine Übersicht zur gesamten Architektur ist in 5.4 zu finden.

Die Implementierung der Analyse-Services basiert auf einer Microservice-Architektur, die eine klare Trennung der Verantwortlichkeiten ermöglicht und gleichzeitig eine hohe Skalierbarkeit und Wartbarkeit gewährleistet. Die Anwendung besteht aus mehreren unabhängigen Diensten, die über definierte Schnittstellen miteinander kommunizieren und jeweils spezifische Aufgaben übernehmen. Diese Architektur erlaubt es, einzelne Komponenten unabhängig voneinander zu entwickeln, zu testen und zu skalieren, was die Wartbarkeit und Erweiterbarkeit des Systems erheblich verbessert.

Die einzelnen, implementierten Analyse-Services sind:

- Ein Accessibility-Service, der die Zugänglichkeitsanalyse der gescrapten Webseiten durchführt
- Ein Huggingface-Service, der auf maschinellem Lernen basierende Analysen ermöglicht

7.2 Technologie-Stack

Für die Implementierung des Content Analyzers wurde ein moderner Technologie-Stack verwendet, der sowohl bewährte als auch innovative Technologien kombiniert, um eine robuste und zukunftssichere Lösung zu schaffen.

7.2.1 Frontend

Die Benutzeroberfläche wurde mit React und TypeScript entwickelt, was eine typsichere und wartbare Codebasis ermöglicht. Für die Gestaltung der Benutzeroberfläche wurde das Material-UI-Framework (MUI) in Version 6 verwendet, das eine konsistente und ansprechende Benutzererfahrung bietet. Die Verwendung von Komponenten wie MUI X Date Pickers und MUI X Tree View ermöglicht eine intuitive Darstellung komplexer Daten und Interaktionen.

Für das Zustandsmanagement wurde Redux Toolkit eingesetzt, das eine effiziente und vorhersehbare Verwaltung des Anwendungszustands ermöglicht. Diese Kombination aus React, TypeScript und Redux bietet eine solide Grundlage für die Entwicklung einer reaktiven und benutzerfreundlichen Anwendung.

7.2.2 Backend

Das Backend des Content Analyzers wurde mit Node.js und Express implementiert, was eine effiziente und skalierbare Verarbeitung von HTTP-Anfragen ermöglicht. Die Verwendung von TypeScript auch im Backend sorgt für eine konsistente Entwicklungserfahrung und verbessert die Codequalität durch statische Typisierung.

Für die Kommunikation mit externen Diensten und APIs wird Axios verwendet, eine beliebte HTTP-Client-Bibliothek, die eine einfache und zuverlässige Handhabung von HTTP-Anfragen ermöglicht. Die Protokollierung erfolgt mit Winston, einer flexiblen Logging-Bibliothek, die eine detaillierte und konfigurierbare Protokollierung von Ereignissen und Fehlern ermöglicht.

7.2.3 Datenbank

Als Datenbank wird MongoDB eingesetzt, eine dokumentenorientierte NoSQL-Datenbank, die eine flexible und skalierbare Speicherung von Daten ermöglicht. Die Integration mit Node.js erfolgt über Mongoose, eine Object-Document-Mapping (ODM) Bibliothek, die eine einfache und typsichere Interaktion mit der Datenbank ermöglicht.

Die Datenbank kann in einem Docker-Container betrieben werden, was eine einfache Bereitstellung ermöglicht. Die Konfiguration erfolgt über Umgebungsvariablen, die in der Docker-Compose-Datei definiert sind.

7.2.4 Web-Scraping

Für das Web-Scraping wird Puppeteer verwendet, eine Node.js-Bibliothek, die eine programmatische Steuerung eines Headless-Chrome-Browsers ermöglicht. Dies erlaubt das Crawling und Scraping von dynamischen Webseiten, die JavaScript verwenden, was mit herkömmlichen HTTP-Clients nicht möglich wäre.

Der Web-Scraper-Dienst ist als eigenständiger Microservice implementiert, der über eine RESTful-API mit dem Backend kommuniziert. Dies ermöglicht eine klare Trennung der Verantwortlichkeiten und eine einfache Skalierung des Scraping-Prozesses.

7.2.5 Accessibility-Analyse

Die Accessibility-Analyse wird durch einen dedizierten Microservice durchgeführt, der verschiedene Tools und Bibliotheken zur Überprüfung der Barrierefreiheit von Webseiten integriert. Insbesondere werden Axe-Core für Puppeteer und Pa11y verwendet, zwei führende Tools für automatisierte Accessibility-Tests.

7.2.6 Huggingface-Service und LanguageTool-Integration

Für die Textanalyse und Sprachverarbeitung wurde ein kombinierter Ansatz implementiert, der primär LanguageTool als bevorzugte Option einsetzt und durch die Huggingface Transformers-Bibliothek ergänzt wird. Diese Kombination ermöglicht eine umfassende Analyse von Webinhalten mit besonderem Fokus auf Sprachqualität und Grammatik.

Dieser Dienst wird in einem separaten Docker-Container betrieben und kommuniziert über eine RESTful-API mit dem Backend.

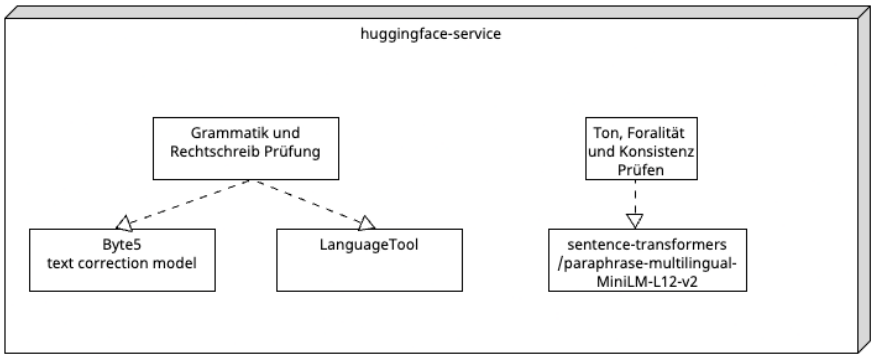


Abbildung 7.1 – Architekturdiagramm des Huggingface-Services und LanguageTool-Integration

LanguageTool als primäre Analysekomponente

LanguageTool ist ein regelbasiertes Open-Source-Tool zur Grammatik-, Stil- und Rechtschreibprüfung für über 20 Sprachen. Es verwendet keine klassischen KI-Modelle im engeren Sinne, sondern basiert primär auf linguistischen Regeln und Heuristiken, die kontinuierlich gepflegt und erweitert werden. Weitere Informationen bietet die offizielle Website unter: https://languagetool.org/de.

Im Rahmen dieses Projekts wurde LanguageTool als primäre Komponente für die Schreibstilanalyse und Grammatikprüfung ausgewählt. Ursprünglich war geplant, ausschliesslich ein ML-basiertes Transformer-Modell über Huggingface für die Analyse zu verwenden. In der Praxis zeigten sich jedoch signifikante Einschränkungen in der Analysequalität für deutschsprachige Inhalte sowie eine deutlich höhere Rechenlast. Zusätzlich erwies sich der Ressourcenverbrauch (insbesondere RAM und CPU) als deutlich höher, was lokal wie auch containerisiert zu langen Antwortzeiten und begrenzter Skalierbarkeit führte. Daher wurde LanguageTool nachträglich als bevorzugte Lösung gewählt, da

es sich als wesentlich robuster, ressourcenschonender und für Deutsch deutlich besser geeignet erwiesen hat.

Die Entscheidung für LanguageTool erfolgte aufgrund folgender Vorteile:

- **Höhere Verarbeitungsgeschwindigkeit:** LanguageTool analysiert Texte sehr effizient, was sich besonders bei grösseren Textmengen bemerkbar macht.
- **Präzise Regelwerke:** Die umfangreichen, sprachspezifischen Regeln ermöglichen eine präzise Erkennung von Grammatik- und Stilproblemen.
- **Niedriger Ressourcenverbrauch:** Im Vergleich zu komplexen NLP-Transformer-Modellen benötigt LanguageTool deutlich weniger Speicher und Rechenleistung.
- **Anpassbarkeit:** Benutzerdefinierte Regeln ermöglichen eine flexible Anpassung an projektspezifische Anforderungen.
- **Einfache Integration:** LanguageTool bietet eine HTTP-basierte API, die eine schnelle Einbindung in bestehende Systeme erlaubt.

Ein zentrales Feature der Implementierung ist die Sprachauswahl: Benutzer müssen zwischen Deutsch und Englisch als Analysesprache wählen, da LanguageTool sprachspezifische Regeln anwendet. Diese Auswahl erfolgt über die Benutzeroberfläche und wird als Parameter an den Service übermittelt. Die Sprachauswahl beeinflusst nicht nur die Grammatik- und Stilprüfung, sondern auch die Erkennung von kulturellen und regionalen Sprachvarianten.

Huggingface Transformers als ergänzende Komponente

Der Python-basierte Huggingface-Service ergänzt LanguageTool durch fortgeschrittene NLP-Funktionen, die auf der Transformers-Bibliothek basieren.

Verwendete Modelle Im Rahmen des Projekts wurden zwei spezialisierte Transformer-Modelle aus der Huggingface-Plattform integriert:

- **Grammatikmodell:** `sdadas/byt5-text-correction` – Dieses Modell wurde zur automatischen Korrektur von Grammatikfehlern verwendet und diente als ML-basierte Alternative zu LanguageTool. Mehr Informationen unter: huggingface.co/sdadas/byt5-text-correction
- **Stilanalysemodell:** `paragraph-multilingual-MiniLM-L12-v2` – Ein semantisches Embedding-Modell zur Bewertung des sprachlichen Stils. Es ermöglicht stilistische Ähnlichkeitsvergleiche und wurde insbesondere für mehrsprachige Texte eingesetzt. Modell-Link: huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2

Der Service bietet verschiedene Endpunkte für komplexere Analysetypen, die über die Möglichkeiten von LanguageTool hinausgehen:

- Sentimentanalyse zur Bestimmung der emotionalen Tonalität von Texten
- Textklassifikation für die thematische Einordnung von Inhalten
- Entitätserkennung zur Identifikation von Personen, Organisationen und Orten
- Automatische Textzusammenfassung für lange Inhalte

Die Kombination aus LanguageTool und Huggingface Transformers ermöglicht eine umfassende Analyse von Webinhalten, die sowohl grammatikalische und stilistische Aspekte als auch semantische und kontextuelle Dimensionen abdeckt. Diese hybride Lösung bietet eine optimale Balance zwischen Verarbeitungsgeschwindigkeit, Analysetiefe und Ressourceneffizienz.

7.3 Containerisierung und Orchestrierung

Die gesamte Anwendung ist für ein Deployment containerisiert und wird mit Docker und Docker Compose orchestriert. Dies vereinfacht die Bereitstellung der Anwendung und ermöglicht einfacheres Sicherstellen von Verfügbarkeit.

Die Docker-Compose-Konfiguration definiert die verschiedenen Dienste und deren Abhängigkeiten, einschliesslich der MongoDB-Datenbank und des Huggingface-Services.

7.4 Entwicklungsprozess und Werkzeuge

Der Entwicklungsprozess des Content Analyzers wurde durch verschiedene Werkzeuge und Praktiken unterstützt, die eine effiziente und qualitativ hochwertige Implementierung ermöglichen.

7.4.1 TypeScript und Typsicherheit

Die Verwendung von TypeScript sowohl im Frontend als auch im Backend sorgt für eine konsistente Entwicklungserfahrung und verbessert die Codequalität durch statische Typisierung. Die Projektkonfiguration umfasst mehrere TSConfig-Dateien, die verschiedene Aspekte der TypeScript-Kompilierung steuern:

- `tsconfig.base.json`: Grundlegende TypeScript-Konfiguration für das gesamte Projekt
- `tsconfig.packages.json`: Spezifische Konfiguration für die verschiedenen Pakete und Dienste
- `tsconfig.shared.json`: Gemeinsame Konfiguration für geteilte Module und Bibliotheken

7.4.2 Testen

Für das Testen werden verschiedene Frameworks und Bibliotheken verwendet, darunter Chai für Assertions und NYC für die Codeabdeckung. Die Tests sind in separate Dateien organisiert und decken verschiedene Aspekte der Anwendung ab, einschliesslich der Website-Services, Accessibility-Services und Utility-Funktionen.

7.4.3 Logging und Fehlerbehandlung

Für das Logging wird Winston verwendet, eine flexible Logging-Bibliothek, die eine detaillierte und konfigurierbare Protokollierung von Ereignissen und Fehlern ermöglicht. Dies erleichtert die Fehlersuche und -behebung und verbessert die Wartbarkeit der Anwendung.

7.5 Implementierungsdetails

7.5.1 Web-Scraper

Der Web-Scraper ist als eigenständiger Microservice implementiert, der Puppeteer verwendet, um Webseiten zu crawlen und zu scrapen. Er bietet eine RESTful-API, über die das Backend Scraping-Aufträge einreichen und die Ergebnisse abrufen kann.

Der Scraping-Prozess umfasst mehrere Schritte:

1. Initialisierung eines Headless-Chrome-Browsers mit Puppeteer
2. Navigation zur Ziel-URL
3. Warten auf das Laden der Seite und die Ausführung von JavaScript
4. Extraktion des HTML-Inhalts und anderer relevanter Informationen
5. Identifikation und Verfolgung von Links für das Crawling
6. Rekursives Crawling mit den vorherigen Schritten
7. Rückgabe der gescrapten Daten an das Backend

Dabei berücksichtigt der Web-Scraper auch die robots.txt-Dateien der Webseiten und respektiert die darin definierten Crawling-Richtlinien, um eine ethische und rechtlich konforme Nutzung zu gewährleisten.

7.5.2 Accessibility-Service

Um automatisierte Accessibility-Tests durchzuführen, nutzt der Accessibility-Service Axe-Core für Puppeteer und Pa11y als Plugins. Der Dienst empfängt die gescrapten

Webseiten vom Web-Scraper und führt verschiedene Tests mithilfe der externen Analysetools durch, um Probleme mit der Barrierefreiheit zu identifizieren.

Je nach implementiertem Analysetools muss dabei rekursiv durch die Webseitenstruktur iteriert werden und das Tool pro Unter-Webseite ausgeführt werden.

Die Ergebnisse der Tests werden strukturiert aufbereitet und an das Backend zurückgesendet, wo sie in der Datenbank gespeichert und für die Benutzeroberfläche aufbereitet werden.

Kapitel 8

Qualitätsmassnahmen

8.1 Arbeitsfluss

Änderungen am Code müssen zuerst durch ein Review (Pull-Request) des Projektpartners akzeptiert werden. Dadurch sollen die Code Qualität konstant hoch gehalten werden und Flüchtigkeitsfehler gefunden werden.

Zusätzlich wird bei jedem Pull-Request die CI/CD pipline auf github ausgeführt. Diese führt für jede Komponente die existierenden Tests durch. So sollen Regressionen durch Code-Änderungen verhindert werden.

8.2 Testing

Jede Typescript Komponente verfügt über ein Test-Setup, das durch die CI/CD pipeline ausgeführt werden kann. Wichtige Funktionalitäten werden mit Tests abgedeckt, mit dem Ziel eine hohe aber vernünftige Testabdeckung zu erreichen. Zusätzlich werden Regressionstests geschrieben sobald an einem Ort ein Fehler gefunden wird.

8.3 Funktionales Test Case Protokoll

Die nachfolgende Tabelle 8.1 dokumentiert die Testresultate für alle funktionalen Anforderungen (FR). Jede Anforderung wurde systematisch getestet, wobei die Vorbedingungen, erwarteten Ergebnisse und tatsächlich eingetroffenen Resultate festgehalten wurden. Die Tests wurden primär in Chrome und Firefox durchgeführt, um die Browser-Kompatibilität zu gewährleisten.

Nr	Anforderung	Vorbedingung	Erwartung	Eingetroffen	nicht-/bestanden	Tester, Zeit
1.1	MVP-1	Die Applikation läuft	Erfasste Unter-Webseiten werden zurückgegeben und angezeigt	erfasst und angezeigt	erfüllt	Simon, 02.06
1.2	MVP-2	Scraping in 1.1 ist gelaufen	Inhalte der Webseite sind nach Scraping ersichtlich	Html, Text, Bilder, Links angezeigt	erfüllt	Simon, 02.06
1.3	MVP-3	Scraping in 1.1 ist gelaufen	Erfasste Inhalte durch Scraping sind in DB gespeichert	erfüllt, sind persistent	erfüllt	Simon, 02.06
1.4	MVP-4	Scraping in 1.1 ist gelaufen	Das Scraping wird über ein GUI gestartet und Resultate angezeigt	über WebUI bedienbar	erfüllt	Simon, 02.06
1.5	MVP-5	Webseite wurde durch Web-Scraper erfasst	Es können externe Tools zur Analyse gestartet werden	über WebUI startbar	erfüllt	Simon, 02.06
1.6	MVP-6	Webseite wurde durch Web-Scraper erfasst	Es können Plugins zur Analyse gestartet werden über die Webapplikation gestartet werden	erfüllt	erfüllt	Simon, 02.06
1.7	MVP-7	Webseite wurde durch Web-Scraper erfasst	Es wird automatisiert eine Analyse mit Axe Core durchgeführt	Die Analyse kann über die Webapplikation gestartet werden	teilweise erfüllt	Simon, 02.06
1.8	MVP-8	Webseite wurde durch Web-Scraper erfasst und durch Axe Core analysiert	Metadaten für Gefundene Probleme sollen auf Ebene der betroffenen Elemente bearbeitet werden können	Nur Gruppierung möglich, Stati noch nicht	nicht erfüllt	Simon, 02.06
1.9	MVP-9	Webseite wurde durch Web-Scraper erfasst und durch Axe Core analysiert	Nicht abgedeckte Richtlinien sind ersichtlich und können analysiert werden soweit es Pluins gibt dafür	Abdeckung ist ersichtlich pro Plugin	erfüllt	Simon, 02.06
1.10	FR-13	Webseite wurde durch Web-Scraper erfasst	Der Schreibstil der Webseite kann analysiert werden über ein Plugin	Möglich über writing style analysis	erfüllt	Simon, 02.06
1.11	FR-15	Webseite wurde durch Web-Scraper erfasst und durch Axe Core analysiert	Individuelle Issues zu Elementen können gruppiert werden	Möglich bei der Ansicht der Resultate	erfüllt	Simon, 02.06
2.1	MVP-8	Webseite wurde durch Web-Scraper erfasst und durch Axe Core analysiert	Metadaten für Gefundene Probleme sollen auf Ebene der betroffenen Elemente bearbeitet werden können	Stati und Kommentare können bearbeitet werden	erfüllt	Simon, 13.06
2.1	MVP-8	Webseite wurde durch Web-Scraper erfasst und durch Axe Core analysiert	Metadaten für Gefundene Probleme sollen auf Ebene der betroffenen Elemente bearbeitet werden können	Stati und Kommentare können bearbeitet werden	erfüllt	Simon, 13.06
2.2	MVP-9	Webseite wurde durch	Nicht abgedeckte Richtlinien sind ersichtlich	Abdeckung ist ersichtlich pro Plugin und de-	erfüllt	Simon, 02.06

Die Testergebnisse zeigen, dass 10 von 11 funktionalen Anforderungen vollständig erfüllt wurden. Anforderung MVP-8 (Metadaten-Bearbeitung) ist nur teilweise implementiert, da die Status-Funktionalität noch nicht vollständig umgesetzt wurde.

8.4 Nicht Funktionales Test Case Protokoll

Die folgende Tabelle 8.2 fasst die Testresultate für die nicht-funktionalen Anforderungen (NFR) zusammen.

Nr	**Anforderung**	**Vorbedingung**	**Erwartung**	**Eingetroffen**	**nicht-/ bestanden**	**Tester, Zeit**
1.1	NFR-7	Die Applikation läuft	Alle FR können in Chrome umgesetzt werden	Alle FR tests durchgeführt in Firefox und Chrome, siehe 8.1 1.1 - 1.11	erfüllt	Simon, 02.06
1.2	NFR-8	Die Applikation läuft auf einem Deployment	Verfügbarkeit von 99% in normalem Betrieb	Aus Zeitgründen wurde die Applikation nicht remote deployed	-	Simon, 02.06
1.3	NFR-9	Die Applikation läuft auf einem Deployment	Das System muss nach einem Ausfall schnell wieder betriebsfähig sein und Crawler läuft weiter	Aus Zeitgründen wurde die Applikation nicht remote deployed, es wären zusätzliche Anpassungen am Crawler notwendig	-	Simon, 02.06
1.4	NFR-10	Die Applikation läuft auf einem Deployment	Die Webseite soll zu allen Zeiten vernünftige Ladezeitenaufweisen.	Aus Zeitgründen wurde die Applikation nicht remote deployed	-	Simon, 02.06
1.5	NFR-11	bestehende Plugin Abstraktionen im Code	Neue Plugins sollen sich mit minimalem Aufwand integrieren lassen	pa11y wurde in weniger als einer Woche implementiert	erfüllt	Simon, 02.06

Tabelle 8.2 – NFR Resultate

Von den 5 getesteten nicht-funktionalen Anforderungen konnten 2 vollständig validiert werden. Die Anforderungen NFR-8 bis NFR-10 bezüglich Deployment und Verfügbarkeit konnten aufgrund zeitlicher Beschränkungen nicht getestet werden, da kein Remote-Deployment durchgeführt wurde.

Kapitel 9

Nutzer und Rollen

9.1 Zielgruppe

Das Content Analyse Tool richtet sich an Unternehmen, die ihre eigenen Webseiten hinsichtlich Barrierefreiheit, Schreibstil und technischer Qualität überprüfen möchten. Innerhalb eines Unternehmens gibt es unterschiedliche Nutzergruppen mit spezifischen Anforderungen.

9.2 Applikationsbenutzer und Use-Cases

Die Hauptnutzer der Anwendung lassen sich in folgende Personas unterteilen:

- **Web-Entwickler** – Technische Experten, die die Struktur und den Code der Website optimieren. Sie nutzen das Tool, um HTML, CSS und Accessibility-Probleme zu identifizieren.
- **Content-Manager** – Verantwortlich für die Pflege von Inhalten. Sie verwenden das Tool, um den Schreibstil und die Verständlichkeit von Texten zu analysieren.
- **UX-Experten** – Überprüfen, ob die Website benutzerfreundlich und gut strukturiert ist. Sie achten auf Ladezeiten, Link-Fehler und allgemeine UX-Optimierungen.
- **Projektmanager** – Möchten einen Überblick über den Status der Analyse haben und priorisieren, welche Findings behoben werden sollen.

9.3 Rollen und Aufgabenbereiche

Um eine geregelte Nutzung der Plattform zu gewährleisten, gibt es verschiedene Rollen mit spezifischen Aufgabenbereiche:

Rolle	Aufgabenbereiche
Entwickler und UX-Experten	• Webseiten analysieren • Usability-Checks durchführen • Accessibility-Probleme analysieren
Content-Manager	• Schreibstil-Analyse durchführen • Texte bearbeiten und optimieren
Projektmanager	• Ergebnisse überwachen • Prioritäten für Fixes setzen

Tabelle 9.1 – Nutzerrollen und deren Aufgabenbereiche

Teil II

Projekt Dokumentation

Kapitel 10

Projekt Plan

10.1 Projektplan

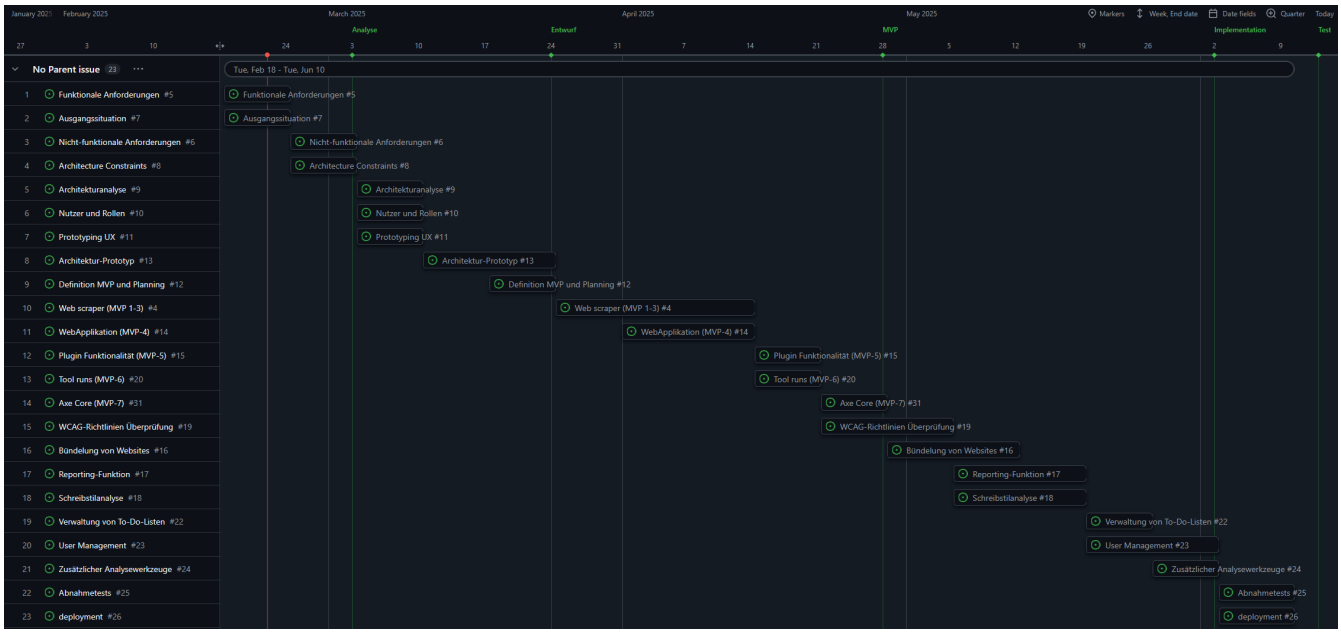


Abbildung 10.1 – Ursprünglicher Projektplan

Meilensteine sind definiert als Projektphasen Analyse, Entwurf, Implementation und Test. Sie beinhalten mehrere einwöchige Iterationen. Anforderungen, Risiken und Architekur werden laufen durch neue Erkenntnisse genauer definiert. Auf einen formalen agilen Iterationsablauf wird aufgrund der Teamgrösse verzichtet und dafür auf direkte tägliche Zusammenarbeit gesetzt.

Analyse [2w]

Die Vision des Projektes ist definiert und aus der Aufgabenstellung werden funktionale, sowie nicht-funktionale Anforderungen hergeleitet. Ein Bericht über die Ausgangssituation des Projektes liegt vor und informiert über mögliche Technologien und gängige Tools zur Analyse von Usability. Notwendige Hilfsmittel und Pipelines für die Entwicklung sind aufgesetzt.

Entwurf [3w]

Die Architektur wird definiert anhand erarbeiteten Informationen. Zusammen mit UX-Prototyping dient dies als Basis zur Implementierung eines ersten Prototypes, der die Architektur repräsentiert.

Implementation [10w]

Die geplanten funktionalen Anforderungen werden der Priorität nach implementiert. Somit wird zuerst ein MVP erreicht und danach darauf aufbauend weitere Funktionalitäten umgesetzt. Zusätzliche Anforderungen werden im Design der Applikation so weit berücksichtigt, dass eine spätere Implementation möglichst einfach machbar ist.

Test [1w]

Abschliessende Tests werden durchgeführt und die Applikation in einen betriebsfähigen Zustand gebracht. Ein Deployment in die Betriebsumgebung wird vorgenommen und deren weiterer Betrieb wird geregelt.

10.1.1 Aktueller Projektplan

Der ursprüngliche Projektplan wurde im Laufe des Projektes angepasst um den neuen Schätzungen zu Features und durchschnittlichem Arbeitsfortschritt zu entsprechen. Zusätzliche Anforderungen bis auf Schreibstilanalyse mussten komplett weggelassen werden, sowie jegliche zukünftige Erweiterungen wie User Management. Dafür wurde die Nützlichkeit der Applikation in den Fokus gestellt mit einer komplexeren aber nützlicheren Gruppierung von Issues, siehe MVP-8, FR-15, FR-16.

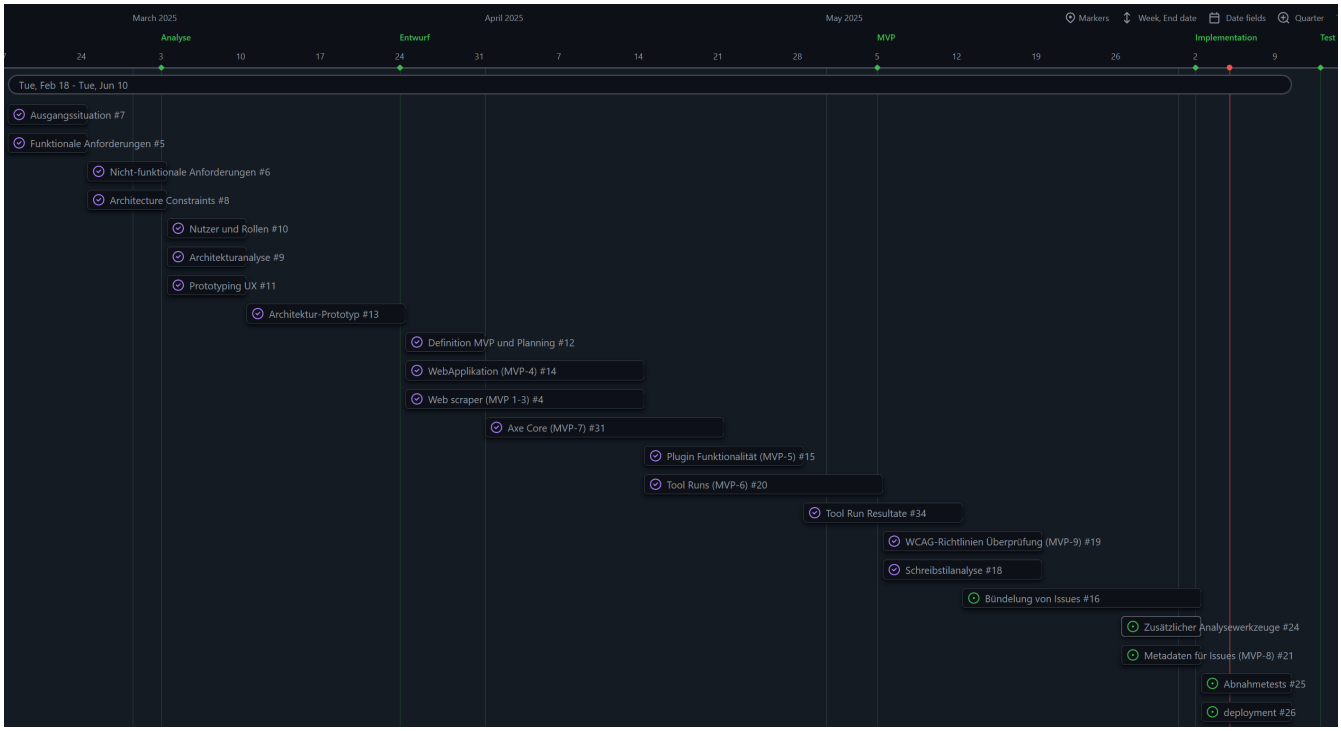


Abbildung 10.2 – Aktueller Projektplan

10.2 Prozesse

Gefolgt wird einem agilen RUP Prozess. Dabei wird für die einwöchigen Iterationen ein Kanban-Ansatz verfolgt.

Meetings

Tägliches Stand-Up

Zu Beginn eines Arbeitstages wird der Fortschritt abgeglichen und der weitere Verlauf festgelegt.

Iterations-Planung and Review

Zu Beginn jeder Woche werden Anforderungen, Architektur und Risiken evaluiert und aktualisiert. Individuelle Arbeitspakete werden aufgeteilt in der Planung für die neue Iteration.

Wöchentliches Betreuer-Meeting

Grundsätzlich wird jede Woche eine Besprechung mit dem zuständigen Betreuer gehalten. Der Fortschritt des Teams wird diskutiert und verglichen mit Risiken und dem

Projektplan. Wichtige Entscheidungen sind im Protokoll festgehalten.

10.2.1 Verwendete Hilfsmittel und Tools

Für die Entwicklung und Dokumentation kamen verschiedene moderne Tools zum Einsatz, die den Workflow unterstützt und beschleunigt haben. Die Auswahl erfolgte unter Berücksichtigung von Effizienz, Team-Kompetenz und Interoperabilität.

Entwicklung

- **Visual Studio Code (VSCode) und Webstorm von Jetbrains** – Zentrale Entwicklungsumgebung.
- **Git & GitHub** – Quellcodeverwaltung und Issue-Tracking.
- **Docker & Docker Compose** – Containerisierung von Services.
- **GitHub Copilot und ChatGPT** – KI-unterstütztes Coding

Dokumentation

- **LaTeX (LaTeX Workshop)** – Mit dem LaTeX Workshop Plugin in VSCode wurde die Dokumentation erstellt.
- **ChatGPT und ClaudeAI** – Unterstützung bei der Textstrukturierung, Umformulierung und Fehlererkennung.

Projektmanagement

- **GitHub Projects** – Planung, Aufgabenmanagement und Fortschrittskontrolle.
- **Jira** – Erweiterte Projektplanung und Aufgabenverfolgung.
- **Clockify** – Zeiterfassung pro Issue via GitHub Integration.

Auf komplexere Tools, die ansonsten in der agilen Entwicklung eigesetzt werden, wird aufgrund der Teamgrösse und zur Vermeidung von überverhältnissmässigem Aufwand verzichtet.

Die Verantwortung für Inhalt, Struktur und Korrektheit der Arbeit liegt trotz punktueller KI-Unterstützung vollständig beim Projektteam.

10.3 Riskomanagement

Über die Zeitspanne des Projektes spielt Risikomanagement eine essentielle Rolle in der Priorisierung von Arbeitsschritten und beeinflusst Architektur und Technologieentscheidungen.

Mögliche Risiken werden regelmässig aktualisiert und Massnahmen zur Risikominderung geplant.

10.3.1 Risiken

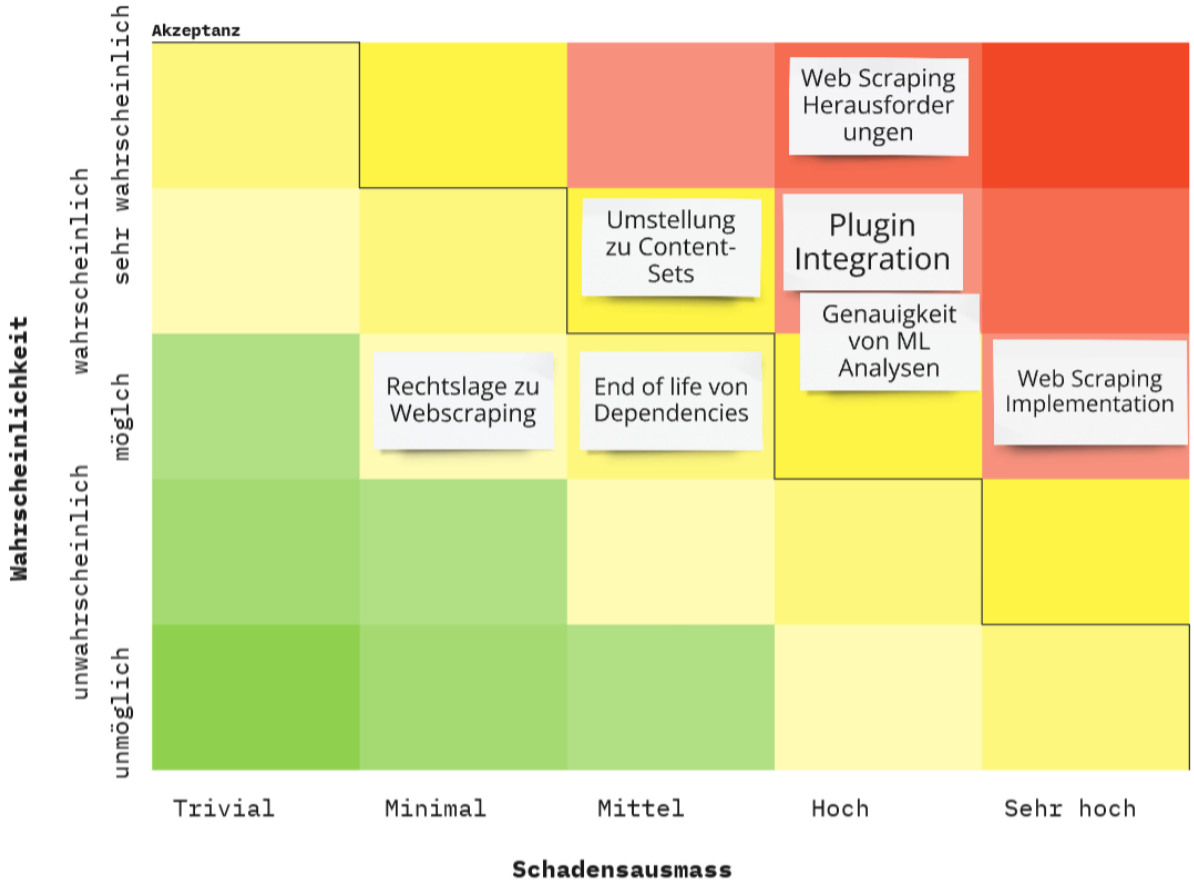


Abbildung 10.3 – Ursprüngliche Risikomatrix

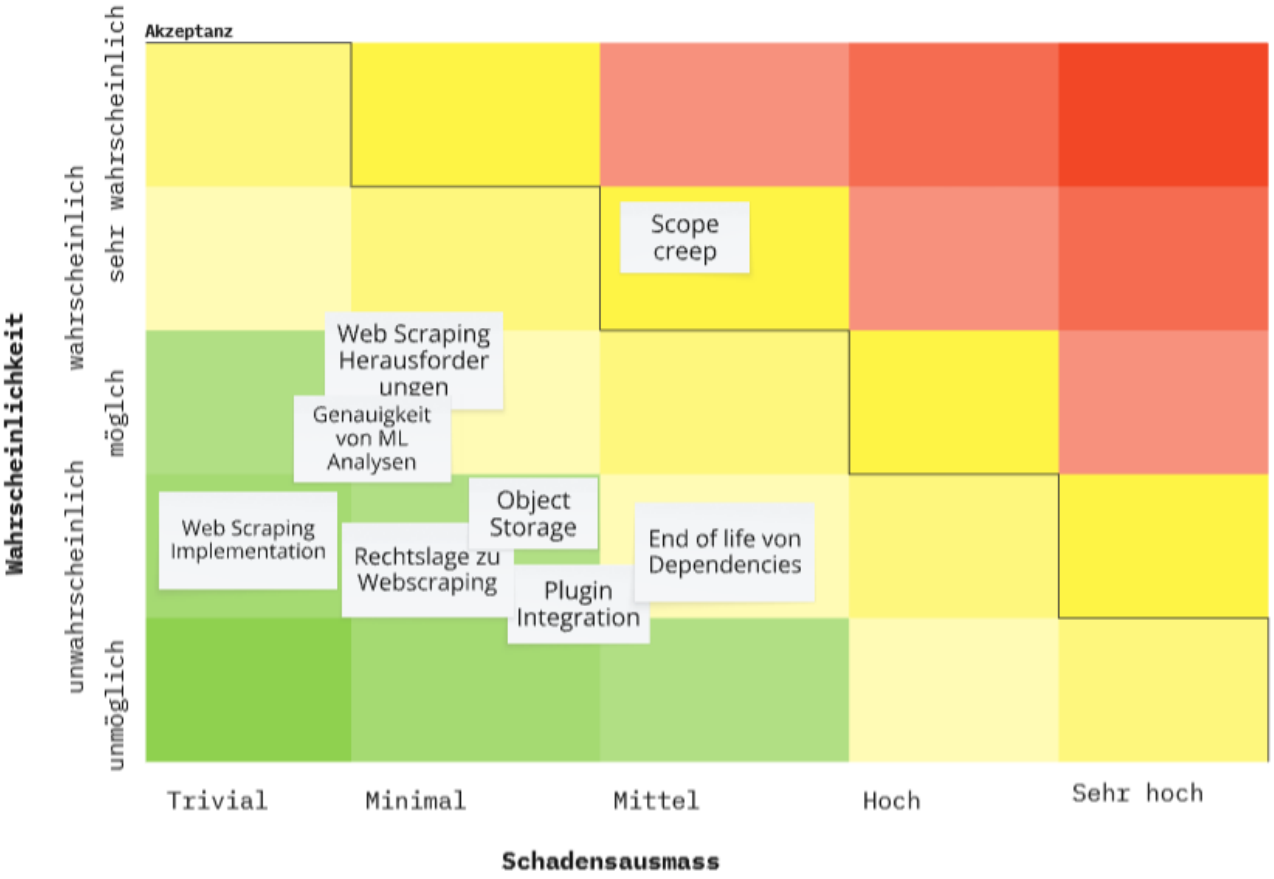


Abbildung 10.4 – Aktuelle Risikomatrix

Name	Beschreibung
Web Scraping Implementation	Dem Team fehlt Erfahrung mit Web-Scraping. Dadurch sind existierende Technologien unbekannt und Aufwand sowie Machbarkeit schwierig einzuschätzen.
Web Scraping Herausforderungen	Durch Recherche haben sich Authentifizierung und dynamische HTML als aktuelle Herausforderungen von Web-Scraping ergeben, wofür Lösungen gefunden werden müssen.
Umstellung zu Content-Sets	Content-Sets (FR-10) müssen für Tool Runs und Metadaten gleich behandelt werden wie Content-Items. Die Umstellung könnte komplexer ausfallen als notwendig Kein geplantes FR mehr. Wird durch (FR-15) abgelöst.
Plugin Integration	(MVP-5) Die Einbindung von externen/internen Analyse tools könnte sich als komplexer herausstellen als erwartet. Spezifisch Input und Output Format.
Rechtslage zu Web-Scraping	Web-Scraping könnte rechtlich problematisch sein. Die rechtskonformität des Projektes muss geprüft werden. Was passiert wenn Benutzer nicht ihre eigenen Webseiten analysieren. Was ist die Lage wenn es ihre eigenen sind.
Genauigkeit von ML Analysen	Genutzte ML-Tools könnten qualitätsmässige Mängel haben. Das würde sich auf die Korrektheit der Analysen auswirken.
End of life von Dependencies	Analysetools oder auch anderen verwendeten Bibliotheken könnten nicht mehr verfügbar oder end-of-life werden.
Object Storage	Die Benutung und der Umgang mit Object Storage NoSQL Datenbanken ist dem Team neu, dadurch könnte Mehraufwand entstehen
Scope Creep	Während der Entwicklung werden durch neue Erkenntnisse weitere Funktionale Anforderungen gefunden. Siehe FR-15, FR-16. Andere Anforderungen müssen dafür neu priorisiert werden.

10.3.2 Risikominderung

Name	Massnahmen
Web Scraping Implementation	Die Implementierung eines Web-Scrapers erhält oberste Priorität. In der Phase Analyse werden existierende Technologien und Standards recherchiert. In der Entwurf-Phase werden Technologien getestet und ein Prototyp implementiert.
Web Scraping Herausforderungen	In der Analysephase werden existierende Technologien und Methoden gesucht, die gegebene Herausforderungen bewältigen können. Im ersten Prototyp soll klar sein ob Lösungen gefunden werden können oder Alternativen gesucht werden müssen.
Umstellung zu Content-Sets	Eine Abstraktion um die Modelle gleich zu behandeln soll vor der Implementierung von Tool Runs und Metadaten implementiert werden.
Plugin Integration	Fehlende Richtlinien analysieren, Tools zur Überprüfung finden und deren Integration testen früh in der Entwicklung.
Rechtslage zu Web-Scraping	In der Analysephase wird die Rechtslage abgeklärt und darauf basierend Massnahmen geplant.
Genauigkeit von ML Analysen	Genauigkeit und Qualität muss bei Recherche zu möglichen Modellen berücksichtigt werden. Recherche zeitgleich mit der Suche nach Tools zur Richtlinienprüfung.
End of life von Dependencies	Es sollen Open source Dependencies gewählt werden. Idealerweise mit grosser Community um langfristigen Support zu gewährleisten
Object Storage	Mehraufwand wird akzeptiert, es wird mit Sorgfalt nach Dokumentation und best-practices vorgegangen
Scope Creep	Siehe FR-15, FR-16. Neue Anforderungen ersetzen existierende innerhalb des Projektes, anstatt zusätzliche Anforderungen zu werden. Dadurch bleibt das Scope des Projektes gleich, das Projekt verliert aber an ursprünglich angedachten Funktionalitäten.

10.3.3 Risikoverlauf

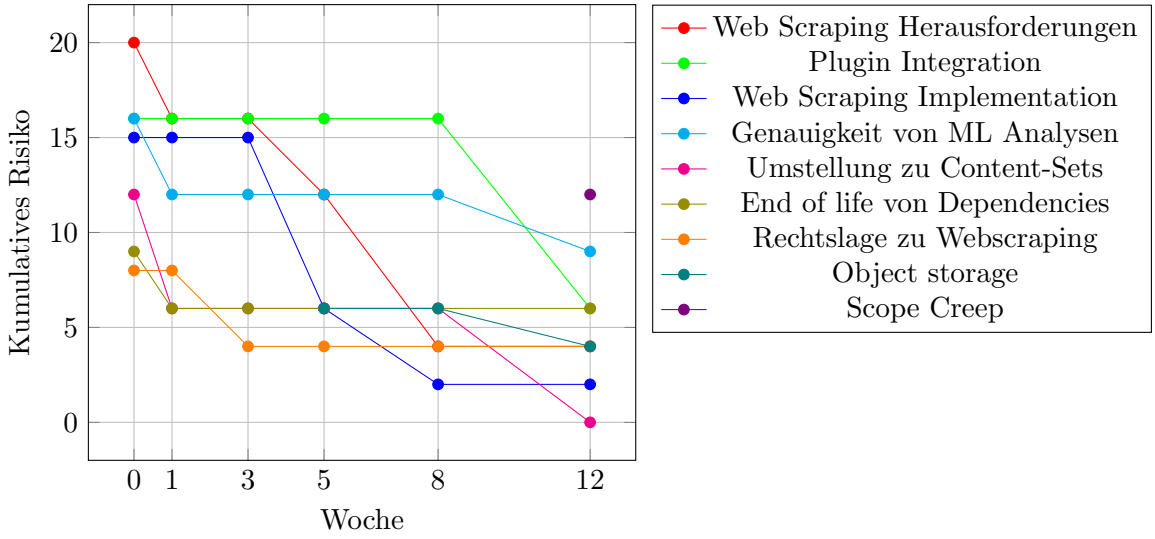


Abbildung 10.5 – Kumulatives Risiko über Zeit

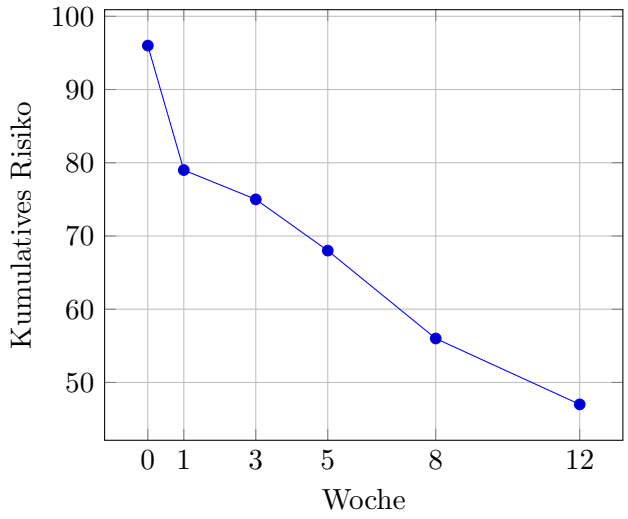


Abbildung 10.6 – Summiertes kumulatives Risiko über Zeit

Das kumulative Risiko wurde anhand der Risikomatrix berechnet. Wobei für die Einträge das Schadensausmass mit der Wahrscheinlichkeit multipliziert wurde, startend mit $1 * 1$ für trivial/unmöglich bis $5 * 5$ für sehr hoch/sehr wahrscheinlich.

Name	Massnahmen
Woche 1	Risiken werden das erste mal evaluiert und entsprechende Massnahmen geplant
Woche 3	Die Rechtslage wurde abgeklärt, was das damit verbundene Risiko minimiert
Woche 5	Ein Prototyp ist implementiert, der schon grosse Teile der Web Scraping Herausforderungen besteht
Woche 8	Die Implemenation der Webapplikation und des Webscrapers sind stabil. Der Webscraper bewältigt JavaScript und Authentifizierung.
Woche 12	Es besteht Erfahrung und Routine bei der Arbeit mit MongoDB. Die Implementation einer ersten ML Analyse zeigt, dass verlässliche Tools, bzw. Modelle gibt. Axe-Core wurde als erstes Plugin implementiert und ist stabil. Nach Diskussionen über die Gruppierung und die Arbeit mit der Webapplikation, stellen sich neue funktionale Anforderungen heraus.
Woche 15	Als Alternative zu reinen ML Modellen wurde LanguageTool als Tool auf Algorithmus-Basis eingebunden. Plugin Integration funktioniert gut: Es wurde ein zweites Plugin innerhalb einer Woche implementiert.

Teil III

Resultate

Kapitel 11

Ergebnisse

Im Laufe des Projektes wurde nach den erfassten Funktionellen und Nicht-funktionellen Anforderungen und einem dementsprechenden Projektplan vorgegangen. Diese wurden mit sich ändernden Anforderungen und neuen Erkenntnissen angepasst und neu priorisiert. Zwingende Anforderungen wurden als MVP klassifiziert (siehe Abschnitt 3.1.1). Weitere Anforderungen wurden aufgenommen und zu Beginn des Projektes ebenfalls optimistisch eingeplant.

Während des Projektes machten sich dass sich neue Anforderungen klar und wurden vor anderen priorisiert. Spezifisch wurde (FR-15) neu aufgenommen. Es sollten neu betroffene Elemente von Analyseresultaten gruppiert werden, anstelle von Webseiten. Dies führte zu grossem Mehraufwand, spezifisch einem Refactoring im Backend, aufgrund des schon existierenden Domänenmodell und Codes, bei dem individuelle Elemente nicht als adressierbar angedacht waren. Die Implementierung eines Web-Scrapers (MVP-1) dauerte ebenfalls länger als eingeschätzt. Aufgrund dieser Verzögerungen musste sich gegen Ende des Projektes entschieden werden zwischen einem angedachten Deployment und den damit verbundenen NFR zu Verfügbarkeit und Performance: (NFR-8), (NFR-9), (NFR-10), (NFR-10) oder der Umsetzung letzter Funktionalen Anforderungen, die für die Nützlichkeit der Webseite essentiell waren:

- (MVP-8): Verwaltung von Metadaten zu betroffenen Elementen von Richtlinienverletzungen
- (MVP-9): Erfüllung von Richtlinien gemäss WCAG pro Plugin und ausgeführter Analyse
- (FR-15): Die Gruppierung betroffenen Elementen von Richtlinienverletzungen
- (FR-13): Implementierung einer KI-basierten Schreibstilanalyse

Es wurde sich für die Umsetzung der FR entschieden und auf ein aktives Deployment verzichtet.

Dies führte dazu, dass am Ende des Projektes wurden alle als MVP klassifizierten Funktionellen Anforderungen umgesetzt werden konnten. Einige zusätzlichen Anforderungen wurde ebenfalls umgesetzt anhand deren Priorisierung. Spezifisch:

- (FR-15): Die Gruppierung betroffenen Elementen von Richtlinienverletzungen
- (FR-13): Implementierung einer KI-basierten Schreibstilanalyse

Von den zusätzlichen Anforderungen nicht umgesetzt werden konnten:

- (FR-11): Reporting-Funktionen
- (FR-12): Annotation und To-Do-Listen, dies wurde mit der Umsetzung der Gruppierung und Metadatenverwaltung als weniger wichtig betrachtet
- (FR-16): Matching von Analyseresultate zwischen Ausführungen eines Plugins

Zukünftigen Erweiterungen (3.17) hingegen konnten keine umgesetzt werden. Angeplant war hier ursprünglich eine Benutzerverwaltung zu implementieren.

Das Resultat der Arbeit ist ein System, das als Grundlage für zukünftige Erweiterungen mit weiteren Plugins und Funktionalitäten dienen kann, aber auch schon grundlegende notwendige Funktionalitäten erfüllt, um Webseiten auf Barrierefreiheit zu analysieren und Resultate übersichtlich zu visualisieren. Es ist möglich das System einfach lokal zu starten und Webseiten zu analysieren mittels der Plugins Axe-Core und pa11y, sowie einer Schreibstilanalyse. Resultate können zur Arbeitsorganisation gruppiert werden und mit Stati und Kommentaren verwaltet werden.

Kapitel 12

Zukünftige Erweiterungen

Um das System produktiv zu betreiben, müsste eine Benutzerverwaltung implementiert werden. Das deployment und eine Automation davon mittels CI/CD müsste ebenfalls noch orchestriert werden. Eine Containerisierung, die dafür genutzt werden könnte existiert schon. Mit weiteren aufkommenden Tools und Standards ist angedacht, dass diese mit weiteren Plugins eingebunden werden können und das System als langfristige Platform benutzt werden kann.

Glossar

Axe Core Tool zur Analyse von Webseiten bezüglich Accessibility. i, 4, 11, 12, 27

LanguageTool LanguageTool is an algorithm based style and grammar checker that also interfaces with NLP transformers. 70

Plugin Als Plugin wird ein externes Tool bezeichnet, dass über den Plugin-Mechanismus in die Applikation eingebunden wird um Analysen von Webseiten durchzuführen. i, 10

Tool Runs Als Tool run wird die Ausführung eines Tools, bzw. Plugins auf eine Webseite bezeichnet. Effektiv ist dies der Analyseprozess.. 67, 68

Web-Scraping Web-Scraping ist ein Prozess, der automatisiert den Inhalt von Webseiten sammelt und diesen in einem strukturierten lokalen Index speichert. i–iii, 2–4, 6, 32, 35, 36, 49, 67, 68

Abkürzungen

ADA Americans with Disability Act. 3, 6

CSRF Cross-Site Request Forgery. 6

FR Funktionale Anforderungen. ii, 9, 72

MVP Minimum Viable Product. ii, 9, 72

NFR Nicht-funktionale Anforderungen. ii, 15, 72

PII Persönlich identifizierbare Informationen. 3

WCAG Web Content Accessibility Guidelines. i, 3, 5, 6, 12, 22, 27

Literatur

[1] ADA.gov. *Guidance on Web Accessibility and the ADA*. https://www.ada.gov/resources/web-guidance/. (Besucht am 23. 02. 2025).

[2] J. Cho und H. Garcia-Molina. *The Evolution of the Web and Implications for an Incremental Crawler*. Technical Report 1999-22. Stanford InfoLab, 1999. URL: http://ilpubs.stanford.edu:8090/376/.

[3] eCH. *eCH-0059 Accessibility Standard*. https://ech.ch/de/ech/ech-0059/3.0. (Besucht am 23. 02. 2025).

[4] Equalweb.ltd. *Equalweb*. https://www.equalweb.com/11595/11528/web_accessibility_crawler. (Besucht am 23. 02. 2025).

[5] iso25000. *ISO/IEC 25010*. https://iso25000.com/index.php/en/iso-25000-standards/iso-25010. (Besucht am 23. 02. 2025).

[6] Moaiad Ahmad Khder. „Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application“. In: *International Journal of Advances in Soft Computing and its Applications* (2021). URL: https://api.semanticscholar.org/CorpusID:245584401.

[7] Juanan Pereira und Juan-Miguel López-Gil. *Turning manual web accessibility success criteria into automatic: an LLM-based approach*. Long Paper López-Gil2024. 2024. URL: https://doi.org/10.1007/s10209-024-01108-z.

[8] Emil Persson. „Evaluating tools and techniques for web scraping“. Magisterarb. KTH, School of Electrical Engineering und Computer Science (EECS), 2019, S. 91.

[9] Satyam Tripathi. *Playwright vs. Puppeteer in 2024 - Which Should You Choose?* https://scrapingant.com/blog/playwright-vs-puppeteer. (Besucht am 08. 03. 2025).

[10] VirginiaTech. *Site Crawling Accessibility Testing Tools*. https://www.assist.vt.edu/web-accessibility/testing-tools/site-crawling-accessibility-testing-tools.html. (Besucht am 23. 02. 2025).