

# Einsatz von Smartphones, Tablets und Desktops im schulischen Sprachunterricht

Entwicklung der Lernanwendung LearnEasy

Steven Solleder

Fakultät Informatik

Hochschule für angewandte Wissenschaften Hof

Hof, Deutschland

steven.solleder@hof-university.de

Isabell Waas

Fakultät Informatik

Hochschule für angewandte Wissenschaften Hof

Hof, Deutschland

isabell.waas@hof-university.de

**Zusammenfassung**—Das Ziel dieser Arbeit ist es, eine Lernanwendung für den schulischen Sprachunterricht zu entwickeln, welche insbesondere barrierefrei und plattformunabhängig ist und keine benutzerbezogenen Daten speichert. Hierzu werden zunächst die Anforderungen auf verschiedenen Anforderungsebenen beschrieben, wobei die Anwendung schrittweise in mehreren Prototypen entwickelt werden soll. Anschließend werden die ersten beiden Prototypen im Detail konzeptioniert. Der erste Prototyp wird dann mithilfe eines RESTful Webservices im Backend und einer Progressive Web App im Frontend umgesetzt, auf verschiedene Arten getestet und mithilfe eines Abnahmeprotokolls validiert. Abschließend werden die nächsten Schritte in einem Ausblick erläutert.

**Index Terms**—Lernanwendung, Sprachunterricht, Schulunterricht, Bildung, Schüler, Lehrer, Barrierefreiheit, Datenschutz, Plattformunabhängigkeit, Progressive Web App, Vue.js, RESTful Webservice, Spring Boot

## I. EINFÜHRUNG

In der heutigen digitalisierten Welt gehören Smartphones bereits im Kindesalter zum Alltag. Der Medienpädagogische Forschungsverbund Südwest (mpfs) führt regelmäßig die sogenannte KIM-Studie (Kindheit, Internet, Medien) durch, welche sich mit dem Medienvorhalten von Kindern im Alter von 6 bis 13 Jahren befasst. Gemäß einer auf den Daten der KIM-Studie 2022 basierenden Statistik besaßen im Jahr 2022 zwar noch recht wenige der Kinder im Grundschulalter ein Smartphone. Ab der Altersgruppe 10 bis 11 Jahre steigt die Zahl der Smartphone-Besitzer jedoch auf fast 60 % und unter den 12- bis 13-Jährigen haben bereits mehr als 80 % ein solches Mobiltelefon. Bei Jugendlichen im Alter von 12 bis 19 Jahren wird der Umgang mit Medien jedes Jahr durch den mpfs im Rahmen einer Studie untersucht, die JIM (Jugend, Information, Medien) genannt wird. Auch für die JIM-Studie 2022 ist eine Statistik veröffentlicht worden, nach welcher die Zahl der Smartphone Besitzer in allen untersuchten Altersgruppen über 90 % liegt. [12]–[15]

Abgesehen von Smartphones sind auch Tablets und Desktop-Geräte heutzutage in fast allen Haushalten zu finden. Selbst wenn ein Kind oder Jugendlicher kein eigenes Smartphone besitzt, so hat er somit meist im Elternhaus Zugang zu einem

anderen digitalen Gerät. Mit der zunehmenden Verbreitung der intelligenten Endgeräte entstehen immer mehr Ideen, wie diese innovativ und sinnvoll genutzt werden können, insbesondere auch im Bereich der Bildung. Als multifunktionale Geräte ermöglichen sie nicht nur einen sofortigen Zugang zu umfangreichen Informationsquellen, sondern ebenfalls zeit- und ortsunabhängige Lernerfahrungen durch interaktive Anwendungen. Von der Nutzung der Stoppuhr bei Experimenten im Physikunterricht, über das Visualisieren von Graphen und geometrischen Formen im Mathematikunterricht, bis hin zum Abfragen und Abspielen der korrekten Aussprache von Vokabeln im Sprachunterricht—Digitale Geräte bieten in zahlreichen Schulfächern Einsatzmöglichkeiten und haben das Potenzial, wertvolle Hilfsmittel im Bildungswesen zu sein. [36]

Im Rahmen dieses Projektes soll es um eine Anwendung für den Schulunterricht gehen. Am Anfang in Kapitel II wird die gesamte Anwendung betrachtet—das heißt über alle Prototypen hinweg gehend bis zur ersten Vollversion. Am Ende dieses Kapitels in II-E wird der Umfang für die Prototypen eins und zwei definiert. Im Kapitel III wird das Konzept für die ersten beiden Prototypen in den Blick genommen. Dabei sind die Aspekte, welche nur den zweiten Prototyp betreffen, farblich hervorgehoben. In den Kapiteln IV, V und VI wird sich vollständig auf den ersten Prototyp konzentriert. Schließlich geht es im letzten Kapitel VII dann um die gesamte Anwendung inklusive aller Prototypen, der ersten Vollversion und darüber hinaus. [23]

## II. SPEZIFIKATION

Zunächst sollen die Anforderungen an die Software erläutert werden. Dies geschieht in drei Anforderungsebenen. Die erste Ebene ist der Epic. Danach folgen die User Stories und schließlich die nicht-funktionalen und funktionalen Anforderungen. Es soll hierbei erwähnt werden, dass für unsere Software später ein Adminportal erforderlich sein wird, auf welches nur die Softwarebetreiber Zugriff haben. Im Rahmen dieser Arbeit wird sich auf die Anwendung für die Schule konzentriert und daher in diesem Kapitel die Anforderungen an diese Anwendung betrachtet. [23]

## A. Epic

In diesem Kapitel wird die übergreifende und langfristige Vision dieses Projektes skizziert.

Das Ziel ist es, eine digitale Anwendung zu entwickeln, welche sowohl Schüler als auch Lehrer im Sprachunterricht unterstützt. Diese Unterstützung äußert sich insbesondere durch das Erlernen und Verfestigen von Wissen durch interaktive Übungen. Um den Einsatz der digitalen Anwendung so flexibel wie möglich zu halten, kann die Anwendung auf jedem gebräuchlichen Gerät-Smartphone, Tablet und PC genutzt werden, wobei die Daten eines Nutzers zwischen seinen verschiedenen Geräten stets auf dem gleichen Stand sind. [23]

## B. User Stories

Als nächstes sollen die User Stories aufgelistet werden. Bei diesen handelt es sich um die in Alltagssprache und aus Sicht des Benutzers formulierten Anforderungen des Hauptkunden Jürgen Heym. Die Software soll zwar generell verwendbar sein und nicht ausschließlich für den Hauptkunden entwickelt werden, weshalb in den folgenden Unterkapiteln II-C und II-D auch noch weitere Anforderungen hinzukommen. Dennoch sollen die User Stories als Basis für die Anforderungen an die Software dienen. [1], [23]

- 1) Als Nutzer möchte ich die gleiche Software auf dem Smartphone, Tablet und Desktop verwenden, um nahtlos zwischen meinen Geräten wechseln zu können und dabei eine einheitliche Benutzererfahrung zu genießen. [23]
- 2) Als Nutzer möchte ich, dass die Software barrierefrei ist, um sie auch mit gewissen Beeinträchtigungen uneingeschränkt und benutzerfreundlich verwenden zu können. [23]
- 3) Als Schulleiter möchte ich, dass die Software in der Cloud statt in der Schule läuft, damit sich in der Schule niemand um die Installation und den Betrieb der Software kümmern muss. [23]
- 4) Als Schulleiter möchte ich, dass die Software sehr einfach funktioniert, um keine aufwendigen Schulungen für Lehrer bezahlen zu müssen und dem IT-Administrator der Schule keinen großen Mehraufwand zu bereiten. [23]
- 5) Als Schulleiter möchte ich, dass in der Software keine benutzerbezogenen Daten gespeichert werden, um keine Einverständiserklärung von den Eltern einholen und anderen derartigen Sorgfaltspflichten nachkommen zu müssen. [23]
- 6) Als Schulleiter möchte ich eine zugangsgeschützte Software, damit nur tatsächliche Mitarbeiter und Schüler meiner Schule Zugang zu der Software erhalten. [23]
- 7) Als Lehrer möchte ich die gleiche Software nutzen wie meine Schüler, um die Nutzung der Software in meinem Unterricht leichter zu koordinieren und bei technischen Problemen Hilfestellung leisten zu können. [23]
- 8) Als Lehrer möchte ich, dass meine Schüler in der Software weniger Funktionen haben als ich, damit keiner der

Schüler beabsichtigt oder aus Versehen die Unterrichtsorganisation stören und den Lernprozess beeinträchtigen kann. [23]

- 9) Als Lehrer möchte ich mein Fach anlegen und meine Schüler hinzufügen können, um sicherzustellen, dass alle meiner Schüler Zugriff auf alle Lernmaterialien erhalten, jedoch niemand Außenstehendes. [23]
- 10) Als Lehrer möchte ich, dass die Software sehr einfach funktioniert, um sie auch ohne technische Vorkenntnisse oder umfassende Schulungen im Unterricht einzusetzen zu können. [23]
- 11) Als Lehrer möchte ich kein aufwändiges Passwort für den Login in der Software benötigen, um mich schnell und unkompliziert anmelden zu können ohne Zeitverlust bei dem Zurücksetzen vergessener Passwörter. [23]
- 12) Als Lehrer möchte ich das konkrete Buch für mein Fach angeben können, um dazu passende Übungen vorschlagen zu bekommen. [23]
- 13) Als Lehrer möchte ich vorgefertigte Übungen auswählen und ein paar wenige Einstellungen für diese treffen können, um den Schülern schnell und einfach digitale Übungen bereit zu stellen, die fehlerfrei erstellt sind und zu dem aktuellen Unterrichtsstoff passen. [23]
- 14) Als Lehrer möchte ich, dass nur ich die Lösungen der von mir ausgewählten Übungen sehen kann, um die Übungen besser nachvollziehen zu können, jedoch zu verhindern, dass der Lerneffekt der Schüler durch vorab bekannte Lösungen verloren geht. [23]
- 15) Als Lehrer möchte ich, dass Schüler Stempel für erfolgreich abgeschlossene Übungen erhalten, um sie durch Belohnungen zu motivieren, ihre Hausaufgaben zu machen, und so den Lernerfolg zu steigern. [23]
- 16) Als Lehrer möchte ich in der Software sehen können, wie viele Schüler eine Übung erfolgreich abgeschlossen haben, um den Lernfortschritt zu überwachen und die Motivation der Klasse, Hausaufgaben zu machen, zu überprüfen. [23]
- 17) Als Französischlehrer möchte ich Vokabelübungen auswählen und für diese ein paar wenige Einstellungen treffen können, um das digitale Erlernen und Üben des aktuellen Wortschatzes zu ermöglichen. [23]
- 18) Als Schüler möchte ich die gleiche Software nutzen wie mein Lehrer, um mit dem Lehrer ohne Missverständnisse zu kommunizieren und die in seinem Unterricht genutzten Funktionen der Software problemlos zu verstehen. [23]
- 19) Als Schüler möchte ich, dass die Software sehr einfach funktioniert, um für deren Nutzung nicht regelmäßig Hilfe von Lehrern, Eltern, Geschwistern oder Freunden zu benötigen. [23]
- 20) Als Schüler möchte ich kein aufwändiges Passwort für den Login in der Software benötigen, um mich auf das Lernen zu konzentrieren und nicht darauf, mir das Passwort merken zu müssen. [23]
- 21) Als Schüler möchte ich Stempel für erfolgreich abgeschlossene Übungen erhalten, um die Übung als Lern-

- fortschritt wahrzunehmen und mehr Spaß am Lernen zu haben. [23]
- 22) Als Schüler möchte ich die Software mit Alexa verbinden, um meine anstehenden Aufgaben in der Software abfragen und Übungen mündlich durchführen zu können. [23]

### C. Nicht-funktionale Anforderungen

Die dritte Hierarchieebene der Anforderungen an die Software stellen die nicht-funktionalen und funktionalen Anforderungen dar. Während Letztere in Unterkapitel II-D beschrieben werden, soll es im vorliegenden Unterkapitel um die nicht-funktionalen Anforderungen gehen. Diese werden häufig als Qualitätsanforderungen bezeichnet, da sie definieren, wie die Software ihre Funktionalität bereitstellen soll. Da sie bei verschiedenen Softwareanwendungen sehr ähnlich sind, sind nicht-funktionale Anforderungen in der Regel nicht kunden-spezifisch. Stattdessen werden unter anderem die in der Norm DIN-ISO-9126 aufgeführten Qualitätsmerkmale wie zum Beispiel Benutzbarkeit und Verfügbarkeit verwendet. [4], [5], [23]

*1) Plattformunabhängigkeit:* Als Erstes soll die nicht-funktionale Anforderung Plattformunabhängigkeit näher beleuchtet werden. Diese dient insbesondere zur Erfüllung der User Story 1.

Unsere Zielgruppe ist enorm vielfältig und erstreckt sich über verschiedenste Alters- und Bildungsstufen–von Gymnasiallehrern nahe der Pension über Sekretärinnen in der Mittelschule bis hin zu gerade eingeschulten Grundschülern. Jeder bevorzugt ein anderes Gerät, wobei insbesondere Laptops, Tablets und Smartphones unterstützt werden sollen, ganz unabhängig davon, ob auf ihnen macOS, Linux, Windows, iPadOS, Android, iOS oder ein anderes Betriebssystem mit grafischer Benutzeroberfläche läuft. Niemand soll wegen unserer Anwendung zur Nutzung eines bestimmten Geräts gezwungen sein.

Zusätzlich hat jede Plattform ihren eigenen Weg, Anwendungen zu beziehen. Zu diesen Wegen gehören insbesondere Paketverwaltungen und App Stores. Die Verteilung unserer Anwendung über diese vielen Kanäle schafft unnötigen Aufwand und Kosten. Auch kann es passieren, dass die Anwendung von einer dritten Instanz erst geprüft und freigegeben werden muss, was die Veröffentlichung verlangsamt. Dementsprechend soll die Anwendung nicht nur plattformunabhängig ausgeführt werden können, sondern auch plattformunabhängig verteilt werden. [37]

Diese Anforderungen werden von keiner anderen Technologie besser erfüllt, als von Progressive Web Apps (PWA). Eine PWA ist eine Webseite, wobei diese beim ersten Anfordern vom Webserver durch den Client alles Nötige für die weitere Ausführung mitbringt, wozu insbesondere alle Unterseiten und JavaScript zur Manipulation einzeln Seitenbereiche gehört. Dadurch kann eine PWA in den meisten Betriebssystemen auch anschließend so dargestellt werden, als wäre sie eine eigenständig installierte App. Indem PWAs von allen bekannten Browsern unterstützt werden und auf allen aufgezählten Betriebssystemen laufen, kann unsere PWA folglich auf den aller-

meisten üblicherweise genutzten Geräten ausgeführt werden. Dass ein Browser genutzt werden muss, stellt keine ernsthafte Hürde dar, da bei so gut wie allen Geräten sowieso schon ein Browser (vor)installiert ist. Daneben gibt es auch nur einen Distributionsort für unsere PWA, nämlich unseren Webserver, sodass unsere Anwendung auch plattformunabhängig beschafft werden kann. [23]

*2) Barrierefreiheit:* Als nächstes soll nun die nicht-funktionalen Anforderung Barrierefreiheit erläutert werden. Diese ergibt sich hauptsächlich aus den User Stories 2, 4, 10 und 19.

Die WCAG 2.1 (Web Content Accessibility Guidelines 2.1) sind ein international anerkannter Standard, der vom W3C (World Wide Web Consortium), einer Organisation für die Standardisierung von Technologien im World Wide Web, entwickelt wurde. Der Standard dient dazu, Webinhalte für Menschen mit Behinderungen zugänglicher zu machen. Dies schließt insbesondere Menschen mit visuellen, auditiven, kognitiven oder motorischen Beeinträchtigungen ein. Die im folgenden beschriebenen vier Prinzipien der Barrierefreiheit werden von den WCAG 2.1 sowie auch von den ihnen zugrunde liegenden rechtlichen Rahmenbedingungen berücksichtigt. [2], [7]

- **Wahrnehmbarkeit:** Im Rahmen des ersten Prinzips soll sichergestellt werden, dass jede Funktion und Information von jedem möglichen Nutzer zur Kenntnis genommen werden kann. Hier taucht oft der Begriff Zwei-Kanal-Prinzip auf, welcher die Wahrnehmbarkeit über zwei verschiedene Sinneskanäle meint. So sollen auditiv bemerkbare Informationen beispielsweise auch visuell dargestellt werden. [2]
- **Bedienbarkeit:** Das nächste Prinzip beinhaltet die Interaktion von Menschen mit Behinderungen mit einer Software. Diese muss beispielsweise mit der Tastatur möglich sein und einzelne Aktionen dürfen nicht zu stark zeitlich beschränkt sein. [2]
- **Verständlichkeit:** Das Ziel des Prinzips der Verständlichkeit ist es, die Software für so viele Nutzer wie möglich lesbar und plausibel zu gestalten. Dies bezieht sich nicht nur auf die Verwendung einfacher Sprache und die Bereitstellungen von Definitionen komplexer Begriffe, sondern unter anderem ebenso auf ein konsistentes und erwartbares Interface. [2]
- **Robustheit:** Bei dem letzten Prinzip wird die korrekte Nutzung von Standards wie zum Beispiel der HTML-Syntax gefordert, was eine Voraussetzung für das fehlerfreie Zusammenspiel der Software mit Webbrowsern, Screenreadern und weiteren Technologien ist. [2]

Die WCAG 2.1 enthalten basierend auf den vier vorgestellten Prinzipien 13 Richtlinien. Für jede dieser Richtlinien werden Erfolgskriterien definiert, um die Richtlinie einzuhalten. Zudem gibt es für jedes Erfolgskriterium ausreichende und zusätzlich empfohlene Techniken zu dessen Erfüllung. Hierbei soll erwähnt werden, dass die WCAG zwischen den drei Konformitätsstufen A, AA und AAA unterscheidet, wobei die

Priorität, mit der die Stufe umgesetzt werden sollte, mit jeder Stufe abnimmt. [2]

Im Rahmen dieses Projekts soll die Stufe A der WCAG 2.1 mit den ausreichenden Techniken umgesetzt werden. Im Folgenden werden nun alle Erfolgskriterien der 13 Richtlinien, die für Stufe A relevant sind, betrachtet. Hierbei wird jedes Erfolgskriterium inklusive der zu dessen Umsetzung ausreichenden Techniken kurz erläutert. Es soll angemerkt werden, dass die Techniken der Erfolgskriterien nicht ins Deutsche übersetzt wurden, da sie häufig Fachbegriffe beinhalten und ihr Sinn daher bei der Übersetzung zu leicht verloren gehen kann. [23]

a) *Für die Software nicht relevante Erfolgskriterien und Techniken:* Im Folgenden werden Erfolgskriterien und Techniken für unsere Software ausgeschlossen, weil mit diesen in Verbindung stehende Elemente der Benutzeroberfläche im Rahmen der User Stories der Software nicht geplant sind. Damit diese ausgeschlossen werden können, muss bei der Umsetzung der Benutzeroberfläche darauf geachtet werden, dass die mit diesen Erfolgskriterien und Techniken in Verbindung stehenden Elemente tatsächlich nicht in der Software vorkommen.

Die folgenden Erfolgskriterien der WCAG 2.1 Stufe A sind für unsere Software nicht relevant und werden daher nicht näher erläutert.

- 1.2.1–Audio-only and Video-only (Prerecorded), 1.2.2–Captions (Prerecorded) und 1.2.3–Audio Description or Media Alternative (Prerecorded): Die drei Erfolgskriterien der Richtlinie 1.2–Time-based Media müssen für unsere Software nicht beachtet werden, da keine reinen Audio- oder Video-Inhalte geplant sind, die keine eindeutige Medienalternative für Text sind. Dies ist dadurch begründet, dass die Nutzung von Audio- und Video-Inhalten beispielsweise aufgrund der benötigten Tonausgabe für weniger technikaffine Menschen häufig schwierig ist und daher nicht mit den User Stories 4, 10 und 19 sowie der nicht-funktionalen Anforderung II-C4 zu vereinbaren wäre. [3]
- 1.4.2–Audio Control: Das zur Richtlinie 1.4–Distinguishable gehörige Erfolgskriterium trifft nicht auf unsere Software zu, da in dieser keine Audiodateien automatisch abgespielt werden, sondern erst, nachdem der Nutzer aktiv auf einen Abspiel-Button geklickt hat. Automatisch abgespielte Audiodateien würden weniger technikkundige Personen überfordern, da sie nicht verstehen, woher der Ton kommt und wie sie ihn stoppen, und würden daher ebenfalls den User Stories 4, 10 und 19 und der nicht-funktionalen Anforderung II-C4 widersprechen. [3]
- 2.1.4–Character Key Shortcuts: Das Erfolgskriterium der Richtlinie 2.1–Keyboard Accessible ist für unsere Software irrelevant, da die Zielgruppe unserer Software keine Tastenkombinationen verwendet, da diese nicht einfach zu merken und anzuwenden sind, wenn man wenig technische Vorkenntnisse hat. Auch dies gilt zur Erfüllung der User Stories 4, 10 und 19 und der nicht-funktionalen Anforderung II-C4. [3]

• 2.2.1–Timing Adjustable: Aufgrund der User Stories 4, 10 und 19 und der nicht-funktionalen Anforderung II-C4 legt kein Inhalt, insbesondere keine Übung, in unserer Software ein Zeitlimit fest, weshalb dieses Erfolgskriterium der Richtlinie 2.2–Enough Time ausgeschlossen werden kann. [3]

• 2.2.2–Pause, Stop, Hide: Auch dieses Erfolgskriterium, das Teil der Richtlinie 2.2–Enough Time ist, kann ausgeklammert werden, da sich in unserer Software keine sich automatisch bewegenden, blinkenden, scrollenden oder sich anderweitig von selbst aktualisierenden Inhalte befinden. Dies ist durch die User Stories 4, 10 und 19 und die nicht-funktionalen Anforderung II-C4 begründet, da sich derartig verhaltende Inhalte wenig technikaffine Menschen überfordern und ablenken. [3]

• 2.3.1–Three Flashes or Below Threshold: Ebenfalls ist wegen den User Stories 4, 10 und 19 und der nicht-funktionalen Anforderung II-C4 kein blinkender Inhalt in unserer Software vorgesehen, weshalb auch dieses Erfolgskriterium der Richtlinie 2.3–Seizures and Physical Reactions keine Rolle spielt. [3]

• 2.5.4–Motion Actuation: Zuletzt muss dieses Erfolgskriterium der Richtlinie 2.5–Input Modalities nicht betrachtet werden, da unsere Software weder durch Bewegung des Geräts noch des Nutzers bedient werden kann. Eine solche Bedienung wäre für mit Technik wenig erfahrene Nutzer sehr schwer zu erlernen und würde daher auch den User Stories 4, 10 und 19 und der nicht-funktionalen Anforderung II-C4 widersprechen. [3]

Darüber hinaus sind von den Erfolgskriterien, die im folgenden betrachtet werden einige Techniken nicht relevant für unsere Software. Dabei handelt es sich um die folgenden:

- PDF: Techniken bezüglich PDFs müssen in unserer Software nicht umgesetzt werden, da die Einbindung von Funktionen im Zusammenhang mit PDFs in unsere Software wegen der User Stories 4, 10 und 19 und der nicht-funktionalen Anforderung II-C4 nicht sinnvoll wäre. [3]
- CAPTCHA: Aufgrund der Tatsache, dass CAPTCHA oft nicht einfach anzuwenden sind und daher deren Verwendung den User Stories 4, 10 und 19 und der nicht-funktionalen Anforderung II-C4 entgegenstehen würde, müssen alle Techniken bezüglich CAPTCHA ebenfalls nicht beachtet werden. [3]
- Long Descriptions: In unserer Software ist kein Inhalt so komplex, dass er eine lange Beschreibung benötigen würde, da dies nicht mit den User Stories 4, 10 und 19 sowie der nicht-funktionalen Anforderung II-C4 zu vereinbaren wäre. Deshalb sind alle Techniken bezüglich langen Beschreibungen nicht relevant. [3]
- Decorative Images: Die Erfüllung der User Stories 4, 10 und 19 sowie der nicht-funktionalen Anforderung II-C4 setzt eine sehr einfache und minimalistische Benutzeroberfläche voraus, die ohne rein dekorative Bilder auskommen soll. Daher können alle Techniken bezüglich dekorativer Bilder vernachlässigt werden. [3]

- Bilder als Buttons: Auch Buttons in Form von Bildern würden sich nicht an die Vorgaben der User Stories 4, 10 und 19 sowie der nicht-funktionalen Anforderung II-C4 an die Einfachheit der Software, insbesondere der Benutzeroberfläche, halten. Daher sind alle Techniken bezüglich Bildern als Buttons irrelevant. [3]
- Ascii Art, Emoticons und Leetspeak: Kunst aus ASCII-Zeichen, Emoticons (z. B. :D, was Lachen bedeutet) und Leetspeak (z. B. H3ll0, was für Hello steht) werden in unserer Software ebenfalls ausgeschlossen, da diese für nicht technikaffine Personen nicht einfach zu verstehen sind und damit nicht die User Stories 4, 10 und 19 sowie die nicht-funktionalen Anforderung II-C4 einhalten. Alle Techniken bezüglich Ascii Art, Emoticons und Leetspeak werden somit nicht umgesetzt. [3]
- Textrichtung: In unserer Software ist Text immer wie im Deutschen üblich von links nach rechts zu lesen, da dies zur Erfüllung der User Stories 4, 10 und 19 sowie der nicht-funktionalen Anforderung II-C4 essentiell ist. Techniken in Bezug auf die Änderung der Textrichtung können daher unbeachtet bleiben. [3]
- Tabellen: Auf Tabellen wird in unserer Software verzichtet, da diese insbesondere auf kleineren Bildschirmen schwer darstellbar sind und damit der Erfüllung der User Story 1 und der nicht-funktionalen Anforderung II-C1 im Weg stehen würden. [3]
- Frames und iFrames: Auch alle Techniken, die sich auf Frames oder iFrames beziehen, spielen keine Rolle für unsere Software. Dies ist dadurch begründet, dass bei der Einbindung fremder Inhalte über Frames oder iFrames keine Barrierefreiheit sichergestellt werden kann und dies der User Story 2 sowie der aktuell betrachteten nicht-funktionalen Anforderung widersprechen würde. [3]
- HTML object-Element: Alle Techniken, die sich auf das HTML object-Element beziehen, sind irrelevant für unsere Software. Dies ist ebenfalls mit der User Story 2 und der aktuell betrachteten nicht-funktionalen Anforderung zu begründen, da mit dem HTML object-Element externe Ressourcen eingebunden werden, deren Barrierefreiheit nicht beeinflusst werden kann. [3]

*b) 1.1.1-Non-text Content:* Das erste Erfolgskriterium, das für unsere Software relevant ist, gehört zu der Richtlinie 1.1-Text Alternatives und verlangt, dass es für alle Inhalte, die kein Text sind, eine Textalternative gibt, die den gleichen Zweck erfüllt. Es werden einige Kategorien nicht-textlicher Inhalte aufgezählt und beschrieben, von denen die folgenden für unsere Software wichtig sind. [3]

- Controls, Input: Nicht-textliche Inhalte, die als Steuer-elemente fungieren oder Benutzereingaben akzeptieren, müssen einen ihren Zweck beschreibenden Namen haben. [3]
- Time-Based Media: Zu den zeitbasierten Medien gehören Audio und Video Inhalte. Nicht-textliche Inhalte, die zu den zeitbasierten Medien zählen, müssen Textalternativen enthalten, die das Medium zumindest beschreiben. [3], [8]

- Test: Wenn es sich bei einem nicht-textlichen Inhalt um einen Test oder eine Übung handelt, deren Zweck die Darstellung als Text zunichte machen würde, so muss der nicht-textliche Inhalt durch Textalternativen mit einer beschreibenden Kennzeichnung versehen werden. [3]
- Sensory: Sollen nicht-textliche Inhalte hauptsächlich ein sensorisches Erlebnis vermitteln, müssen sie Textalternativen enthalten, die den nicht-textlichen Inhalt beschreibend identifizieren. [3]
- Decoration, Formatting, Invisible: Bei nicht-textlichen Inhalten, die nur zur Dekoration oder zur visuellen Formatierung verwendet werden oder für den Benutzer unsichtbar sind, muss durch geeignete Implementierung sichergestellt werden, dass sie die Nutzung assistiver Technologien nicht stören. [3]

Nun werden die ausreichenden Techniken für das Erfolgskriterium aufgelistet und dabei nach Kategorien von Techniken zusammengefasst. Es werden für unsere Software nicht relevante Techniken weggelassen. [9]

- Generelle Techniken: [3], [9]
  - G82: Providing a text alternative that identifies the purpose of the non-text content
  - G94: Providing short text alternative for non-text content that serves the same purpose and presents the same information as the non-text content.
  - G95: Providing short text alternatives that provide a brief description of the non-text content
  - G100: Providing a short text alternative which is the accepted name or a descriptive name of the non-text content
- ARIA (Accessible Rich Internet Applications): [3], [9]
  - ARIA6: Using aria-label to provide labels for objects
  - ARIA9: Using aria-labelledby to concatenate a label from several text nodes
  - ARIA10: Using aria-labelledby to provide a text alternative for non-text content
  - ARIA15: Using aria-describedby to provide descriptions of images
- HTML (Hypertext Markup Language): [3], [9]
  - H2: Combining adjacent image and text links for the same resource
  - H24: Providing text alternatives for the area elements of image maps
  - H30: Providing link text that describes the purpose of a link for anchor elements
  - H37: Using alt attributes on img elements
  - H44: Using label elements to associate text labels with form controls
  - H65: Using the title attribute to identify form controls when the label element cannot be used
  - H67: Using null alt text and no title attribute on img elements for images that AT should ignore

*c) 1.3.1-Info and Relationships:* Bei diesem Erfolgskriterium der Richtlinie 1.3-Adaptable geht es darum, dass visuell vermittelte Informationen, Strukturen und Beziehungen

auch programmatisch bestimmt werden können oder auch in textlicher Form zugänglich sind. Es folgt wieder die Aufzählung der für unsere Software wichtigen ausreichenden Techniken. [3]

- Generelle Techniken: [3], [9]
  - G115: Using semantic elements to mark up structure
  - G117: Using text to convey information that is conveyed by variations in presentation of text
  - G138: Using semantic markup whenever color cues are used
  - G140: Separating information and structure from presentation to enable different presentations
- ARIA: [3], [9]
  - ARIA11: Using ARIA landmarks to identify regions of a page
  - ARIA12: Using role=heading to identify headings
  - ARIA13: Using aria-labelledby to name regions and landmarks
  - ARIA16: Using aria-labelledby to provide a name for user interface controls
  - ARIA20: Using the region role to identify a region of the page
  - ARIA24: Semantically identifying a font icon with role=„img“
- HTML: [3], [9]
  - H42: Using h1-h6 to identify headings
  - H44: Using label elements to associate text labels with form controls
  - H48: Using ol, ul and dl for lists or groups of links
  - H49: Using semantic markup to mark emphasized or special text
  - H65: Using the title attribute to identify form controls when the label element cannot be used
  - H71: Providing a description for groups of form controls using fieldset and legend elements
  - H85: Using OPTGROUP to group OPTION elements inside a SELECT
  - H97: Grouping related links using the nav element
- Plain-Text Techniken: [3], [9]
  - T1: Using standard text formatting conventions for paragraphs
  - T2: Using standard text formatting conventions for lists
  - T3: Using standard text formatting conventions for headings
- Client-seitige Skript-Techniken: [3], [9]
  - SCR21: Using functions of the Document Object Model (DOM) to add content to a page

d) *1.3.2-Meaningful Sequence:* Gemäß diesem Erfolgskriterium der Richtlinie 1.3–Adaptable kann durch die Programmierung eine korrekte Lesereihenfolge festgelegt werden, spielt die Reihenfolge, in der Inhalte dargeboten werden, eine Rolle für deren Bedeutung. Die für unsere Software relevanten ausreichenden Techniken sind die unten stehenden. [3]

- Generelle Techniken: [3], [9]
  - G57: Ordering the content in a meaningful sequence
- CSS (Cascading Style Sheets): [3], [9]
  - C6: Positioning content based on structural markup
  - C8: Using CSS letter-spacing to control spacing within a word
  - C27: Making the DOM order match the visual order

e) *1.3.3-Sensory Characteristics:* Dieses Erfolgskriterium der Richtlinie 1.3–Adaptable besteht darin, dass Anweisungen zum Verständnis und zur Bedienung von Inhalten nicht allein auf den sensorischen Merkmalen der Komponenten wie Form, Farbe, Größe, visuelle Position, Ausrichtung oder Klang basieren sollen. Für dessen Umsetzung wird die folgende ausreichende Technik empfohlen. [3]

- Generelle Techniken: [3], [9]
  - G96: Providing textual identification of items that otherwise rely only on sensory information to be understood

f) *1.4.1-Use of Color:* Farbe soll laut diesem Erfolgskriterium der Richtlinie 1.4–Distinguishable nicht als einziges visuelles Mittel zur Informationsvermittlung, zur Anzeige einer Aktion, zur Handlungsaufforderung oder zur Unterscheidung eines visuellen Elements herangezogen werden. Auch hierfür folgen wieder die für unsere Software relevanten ausreichenden Techniken. [3]

- Generelle Techniken: [3], [9]
  - G14: Ensuring that information conveyed by color differences is also available in text
  - G205: Including a text cue for colored form control labels
  - G182: Ensuring that additional visual cues are available when text color differences are used to convey information
  - G183: Using a contrast ratio of 3:1 with surrounding text and providing additional visual cues on hover for links or controls where color alone is used to identify them
  - G111: Using color and pattern

g) *2.1.1-Keyboard:* Das in diesem Abschnitt betrachtete Erfolgskriterium gehört zu der Richtlinie 2.1–Keyboard Accessible. Es besagt, dass mithilfe einer Tastaturschnittstelle alle Funktionen des Inhalts bedient werden können, ohne dass ein genaues Timing für einzelne Tastenanschläge erforderlich ist. Eine Ausnahme stellen Funktionen dar, die Eingaben erfordern, die vom Bewegungspfad des Nutzers und nicht nur von den Endpunkten abhängen. Verlangt nur die Eingabetechnik, z. B. Handschrift, eine pfadabhängige Eingabe, jedoch nicht die zugrundeliegende Funktion, z. B. Texteingabe, so ist dies keine solche Ausnahme. Zudem soll angemerkt werden, dass das Erfolgskriterium zwar Tastaturbedienung vorschreibt, jedoch zusätzliche Eingabemethoden nicht verbietet. Nachfolgend werden nun wieder die für unsere Software wichtigen ausreichenden Techniken aufgelistet. [3]

- Generelle Techniken: [3], [9]
  - G90: Providing keyboard-triggered event handlers
  - G202: Ensuring keyboard control for all functionality
- HTML: [3], [9]
  - H91: Using HTML form controls and links
- Client-seitige Skript-Techniken: [3], [9]
  - SCR2: Using redundant keyboard and mouse event handlers
  - SCR20: Using both keyboard and other device-specific functions
  - SCR35: Making actions keyboard accessible by using the onclick event of anchors and buttons

*h) 2.1.2-No Keyboard Trap:* Kann der Tastaturfokus über eine Tastaturschnittstelle zu einer Komponente auf der Seite verschoben werden, so soll gemäß diesem Erfolgskriterium der Richtlinie 2.1-Keyboard Accessible auch eine Tastaturschnittstelle ausreichen, um den Fokus von dieser Komponente wegzubewegen. Insofern hierzu mehr als einfache Pfeil- oder Tabulatortasten oder andere standardmäßige Beendigungsmethoden benötigt werden, muss ein Hinweis für den Benutzer erfolgen, wie er den Fokus verschieben kann. Aufgrund der Tatsache, dass eine Nacherfüllung dieses Erfolgskriteriums zu einer Unbenutzbarkeit der gesamte Seite führen kann, muss es von ausnahmslos allen Inhalten einer Webseite eingehalten werden. Es gibt nur die folgende ausreichende Technik. [3]

- Generelle Techniken: [3], [9]
  - G21: Ensuring that users are not trapped in content

*i) 2.4.1-Bypass Blocks:* Im Rahmen dieses Erfolgskriteriums der Richtlinie 2.4-Navigable soll die Umgehung von Inhaltsblöcken, die auf mehreren Webseiten wiederholt werden, möglich gemacht werden. Es folgen die ausreichenden Techniken, die für unsere Software eine Rolle spielen. [3]

- Generelle Techniken: [3], [9]
  - G1: Adding a link at the top of each page that goes directly to the main content area
  - G123: Adding a link at the beginning of a block of repeated content to go to the end of the block
  - G124: Adding links at the top of the page to each area of the content
- ARIA: [3], [9]
  - ARIA11: Using ARIA landmarks to identify regions of a page
- HTML: [3], [9]
  - H69: Providing heading elements at the beginning of each section of content
- Client-seitige Skript-Techniken: [3], [9]
  - SCR28: Using an expandable and collapsible menu to bypass block of content

*j) 2.4.2-Page Titled:* Als nächstes wird dieses Erfolgskriterium der Richtlinie 2.4-Navigable betrachtet. Es fordert, dass Webseiten einen das Thema oder den Zweck beschreibenden Titel besitzen. Folgende ausreichende Techniken sind wichtig für unsere Software. [3]

- Generelle Techniken: [3], [9]
  - G88: Providing descriptive titles for Web pages
- HTML: [3], [9]
  - H25: Providing a title using the title element
- *k) 2.4.3-Focus Order:* Wirkt sich die Reihenfolge der Navigation bei einer schrittweise navigierbaren Webseite auf die Bedeutung oder Bedienung aus, sollen bei diesem Erfolgskriterium der Richtlinie 2.4-Navigable fokussierbare Komponenten den Fokus in einer Reihenfolge erhalten, die die Bedeutung und Bedienbarkeit sicherstellt. Unten stehende ausreichende Techniken müssen in unserer Software umgesetzt werden. [3]
  - Generelle Techniken: [3], [9]
    - G59: Placing the interactive elements in an order that follows sequences and relationships within the content
  - CSS: [3], [9]
    - C27: Making the DOM order match the visual order
  - HTML: [3], [9]
    - H102: Creating modal dialogs with the HTML dialog element
  - Client-seitige Skript-Techniken: [3], [9]
    - SCR26: Inserting dynamic content into the Document Object Model immediately following its trigger element
    - SCR27: Reordering page sections using the Document Object Model
- *l) 2.4.4-Link Purpose (In Context):* Auch dieses Erfolgskriterium ist Teil der Richtlinie 2.4-Navigable. Jeder Linktext soll es allein oder zusammen mit dem programmatisch bestimmten Linkkontext ermöglichen, den Zweck des jeweiligen Links zu bestimmen. Dies gilt nicht, wenn der Zweck des Links für Nutzer im Allgemeinen mehrdeutig ist. In unserer Software sind die nachfolgenden ausreichenden Techniken relevant. [3]
  - Generelle Techniken: [3], [9]
    - G91: Providing link text that describes the purpose of a link
    - G53: Identifying the purpose of a link using link text combined with the text of the enclosing sentence
    - G189: Providing a control near the beginning of the Web page that changes the link text
  - ARIA: [3], [9]
    - ARIA7: Using aria-labelledby for link purpose
    - ARIA8: Using aria-label for link purpose
  - CSS: [3], [9]
    - C7: Using CSS to hide a portion of the link text
  - HTML: [3], [9]
    - H24: Providing text alternatives for the area elements of image maps
    - H30: Providing link text that describes the purpose of a link for anchor elements
    - H33: Supplementing link text with the title attribute

- H77: Identifying the purpose of a link using link text combined with its enclosing list item
- H78: Identifying the purpose of a link using link text combined with its enclosing paragraph
- H79: Identifying the purpose of a link in a data table using the link text combined with its enclosing table cell and associated table header cells
- H81: Identifying the purpose of a link in a nested list using link text combined with the parent list item under which the list is nested
- Client-seitige Skript-Techniken: [3], [9]
  - SCR30: Using scripts to change the link text
- m) 2.5.1–*Pointer Gestures*: Gemäß diesem Erfolgskriterium der Richtlinie 2.5–Input Modalities müssen alle Funktionen, deren Bedienung mit Mehrpunkt- oder pfadbasierten Gesten erfolgt, mit einem einzigen Zeiger ohne pfadbasierte Geste bedient werden können. Dies gilt nicht, insofern eine Mehrpunkt- oder pfadbasierte Geste unverzichtbar ist. Es gibt hierfür die beiden unten stehenden ausreichenden Techniken. [3]
  - Generelle Techniken: [3], [9]
    - G215: Providing controls to achieve the same result as path based or multipoint gestures
    - G216: Providing single point activation for a control slider
- n) 2.5.2–*Pointer Cancellation*: Das Erfolgskriterium der Richtlinie 2.5–Input Modalities sieht vor, dass mit einem einzigen Zeiger bedienbare Funktionen mindestens eine der folgenden Bedingungen erfüllen müssen. [3]
  - No Down-Event: Das Down-Event des Zeigers ist ein Ereignis, das durch Drücken beginnt. Zur Ausführung keines Teils der Funktion soll das Down-Event des Zeigers nötig sein. [3], [10]
  - Abort or Undo: Das Up-Event des Zeigers ist ein durch Loslassen ausgelöstes Ereignis. Die Fertigstellung der Funktion erfolgt mit dem Up-Event, wobei es möglich sein muss, die Funktion vor der Ausführung abzubrechen oder nach der Ausführung rückgängig zu machen. [3], [10]
  - Up Reversal: Das Up-Event hebt das Ergebnis des vorherigen Down-Events auf. [3]
  - Essential: Es ist unerlässlich, dass die Funktion mit dem Down-Event fertiggestellt wird. [3]

Nachfolgend sind die zwei ausreichenden Techniken aufgeführt.

  - Generelle Techniken: [3], [9]
    - G210: Ensuring that drag-and-drop actions can be cancelled
    - G212: Using native controls to ensure functionality is triggered on the up-event.
  - o) 2.5.3–*Label in Name*: Besitzt eine Komponente der Benutzeroberfläche eine Beschriftung, die Text oder Bilder von Text enthält, so beinhaltet ihr Name den visuell dargestellten Text. Die für das Erfolgskriterium der Richtlinie 2.5–Input Modalities ausreichenden Techniken sind die folgenden. [3]
    - Generelle Techniken: [3], [9]
      - G208: Including the text of the visible label as part of the accessible name
      - G211: Matching the accessible name to the visible label
    - p) 3.1.1–*Language of Page*: Mit diesem Erfolgskriterium der Richtlinie 3.1–Readable wird die Möglichkeit zur programmatischen Festlegung der Standardsprache für jede Webseite verlangt. Nur die folgende ausreichende Technik ist für unsere Software relevant. [3]
      - HTML: [3], [9]
        - H57: Using the language attribute on the HTML element
    - q) 3.2.1–*On Focus*: Im Rahmen dieses Erfolgskriteriums der Richtlinie 3.2–Predictable soll keine Komponente der Benutzeroberfläche eine Änderung des Kontexts auslösen, indem sie fokussiert wird. Hierfür gibt es eine einzige ausreichende Technik. [3]
      - Generelle Techniken: [3], [9]
        - G107: Using „activate“ rather than „focus“ as a trigger for changes of context
    - r) 3.2.2–*On Input*: Des Weiteren beinhaltet Richtlinie 3.2–Predictable dieses Erfolgskriterium. Passt man die Einstellung einer Komponente der Benutzeroberfläche an, bewirkt dies nicht automatisch eine Änderung des Kontexts. Dies gilt nicht, wenn vor der Verwendung der Komponente ein Hinweis auf dieses Verhalten erfolgt ist. Zu diesem Erfolgskriterium gehören folgende ausreichende Techniken, wobei nur für unsere Software wichtige aufgeführt sind. [3]
      - Generelle Techniken: [3], [9]
        - G13: Describing what will happen before a change to a form control that causes a change of context to occur is made
        - G80: Providing a submit button to initiate a change of context
      - HTML: [3], [9]
        - H32: Providing submit buttons
        - H84: Using a button with a select element to perform an action
      - Client-seitige Skript-Techniken: [3], [9]
        - SCR19: Using an onchange event on a select element without causing a change of context
      - s) 3.3.1–*Error Identification*: Bei einem automatisch erkannten Eingabefehler soll laut diesem Erfolgskriterium der Richtlinie 3.3–Input Assistance das fehlerhafte Element gekennzeichnet und der Fehler für den Nutzer textuell beschrieben werden. Das Erfolgskriterium beinhaltet die folgenden für unsere Software relevanten ausreichenden Techniken. [3]
        - Generelle Techniken: [3], [9]
          - G83: Providing text descriptions to identify required fields that were not completed
          - G84: Providing a text description when the user provides information that is not in the list of allowed values

- G85: Providing a text description when user input falls outside the required format or values
  - ARIA: [3], [9]
    - ARIA18: Using aria-alertdialog to Identify Errors
    - ARIA19: Using ARIA role=alert or Live Regions to Identify Errors
    - ARIA21: Using Aria-Invalid to Indicate An Error Field
  - Client-seitige Skript-Techniken: [3], [9]
    - SCR18: Providing client-side validation and alert
    - SCR32: Providing client-side validation and adding error text via the DOM
- t) 3.3.2-Labels or Instructions:* Dieses ebenfalls zur Richtlinie 3.3-Input Assistance gehörige Erfolgskriterium fordert Beschriftungen oder Anweisungen für Inhalte mit Benutzereingaben und umfasst folgende ausreichende Techniken, die unsere Software beachten muss. [3]
- Generelle Techniken: [3], [9]
    - G83: Providing text descriptions to identify required fields that were not completed
    - G89: Providing expected data format and example
    - G131: Providing descriptive labels
    - G162: Positioning labels to maximize predictability of relationships
    - G167: Using an adjacent button to label the purpose of a field
    - G184: Providing text instructions at the beginning of a form or set of fields that describes the necessary input
  - ARIA: [3], [9]
    - ARIA1: Using the aria-describedby property to provide a descriptive label for user interface controls
    - ARIA9: Using aria-labelledby to concatenate a label from several text nodes
    - ARIA17: Using grouping roles to identify related form controls
  - HTML: [3], [9]
    - H44: Using label elements to associate text labels with form controls
    - H71: Providing a description for groups of form controls using fieldset and legend elements
    - H90: Indicating required form controls using label or legend

*u) 4.1.1-Parsing:* Mit Auszeichnungssprachen erstellte Inhalte sollen vollständige Start- und End-Tags haben, Elementen ihren Spezifikationen entsprechend verschachteln, keine doppelten Attribute besitzen und über eindeutige Ids verfügen. Dies gilt nicht, wenn die Spezifikationen diese Abweichungen zulassen. Dieses Erfolgskriterium der Richtlinie 4.1-Compatible ist insofern besonders, da es immer als erfüllt angesehen werden kann, wenn ein Inhalt mit HTML oder XML verfasst ist. Aus diesem Grund werden keine ausreichenden Techniken aufgelistet. [3]

*v) 4.1.2-Name, Role, Value:* Im Rahmen des Erfolgskriteriums der Richtlinie 4.1-Compatible sollen alle Komponenten der Benutzeroberfläche die folgenden Anforderungen erfüllen, insbesondere Formularelemente, Links und durch Skripte erstellte Komponenten. [3]

- Ihr Name und ihre Rolle sind programmatisch definierbar. [3]
- Ihre durch den Nutzer einstellbaren Zustände, Eigenschaften und Werte sind programmatisch definierbar. [3]
- Bei Änderungen dieser Elemente ist es möglich, Benutzeragenten, einschließlich assistive Technologien, zu benachrichtigen. [3]

Zu beachten ist, dass vordefinierte HTML-Elemente bei vorgesehener Verwendung dieses Erfolgskriterium bereits erfüllen und es daher nur relevant ist, wenn man eigene Komponenten der Benutzeroberfläche programmiert. Das Erfolgskriterium wird mit den folgenden ausreichenden Techniken umgesetzt. [3]

- Generelle Techniken: [3], [9]
  - G10: Creating components using a technology that supports the accessibility notification of changes
  - G108: Using markup features to expose the name and role, allow user-settable properties to be directly set, and provide notification of changes
  - G135: Using the accessibility API features of a technology to expose names and notification of changes
- ARIA: [3], [9]
  - ARIA4: Using a WAI-ARIA role to expose the role of a user interface component
  - ARIA5: Using WAI-ARIA state and property attributes to expose the state of a user interface component
  - ARIA14: Using aria-label to provide an invisible label where a visible label cannot be used
  - ARIA16: Using aria-labelledby to provide a name for user interface controls
- HTML: [3], [9]
  - H44: Using label elements to associate text labels with form controls
  - H65: Using the title attribute to identify form controls when the label element cannot be used
  - H88: Using HTML according to spec
  - H91: Using HTML form controls and links

*3) Datenschutz:* Im Rahmen der folgenden nicht-funktionalen Anforderung wird näher erklärt, wann das Datenschutzrecht Anwendung findet und bewertet, ob dieses bei unserer Anwendung zu Tragen kommt. Dies dient insbesondere zur Erfüllung der User Story 5.

Die Schulen wollen keinen zusätzlichen Aufwand im Zusammenhang mit Datenschutz, der insbesondere durch die Notwendigkeit des Einholens einer Zustimmung zu einer Datenschutzerklärung durch die Eltern der Schüler entstehen würde. Zur Einhaltung dieser und weiterer datenschutzrechtlicher Verpflichtungen sind die Schule und wir als Betreiber nur gezwungen, insofern das EU-weit vereinheitlichte Datenschutzrecht, die Datenschutz-Grundverordnung (DSGVO), für

unsere Anwendung gilt. Dies wäre der Fall wenn die folgenden zwei Aspekte zutreffen.

- 1) Es werden ganz oder teilweise automatisiert personenbezogene Daten verarbeitet oder nicht automatisiert verarbeitete Daten in einem Dateisystem gespeichert (Art. 2 (1) DSGVO). Außerdem müssen die Daten von Unternehmen oder Behörden erhoben werden. Datenerhebung von Personen durch Behörden auf Basis von rein nationalem Recht, zur öffentlichen Sicherheit oder betreffend der gemeinsamen Außen- und Sicherheitspolitik der EU oder Datenerhebung von Personen durch Personen zu rein persönlichen oder familiären Zwecken sind ausgenommen (Art. 2 (2) DSGVO). Personenbezogene Daten sind alle Informationen, welche eindeutige Rückschlüsse auf eine konkrete Person geben. Dazu zählen zum einen offensichtlich zuordbare Daten wie zum Beispiel der Name der Person. Zum anderen zählen dazu auch Daten, die nicht unmittelbar einer konkreten Person zugeordnet werden können, aber aus denen dennoch die eindeutige Identität einer konkreten Person abgeleitet werden kann. Dazu zählen zum Beispiel Daten wie der Standort, der Inhalt einer Patientenakte oder Kennnummern wie das Kfz-Kennzeichen einer konkreten Person (Art. 4 (1) Nr. 1 DSGVO).
- 2) Es werden Daten von Personen, die sich in der Europäischen Union befinden, erhoben, unabhängig davon, ob das Unternehmen seinen Sitz in der Europäischen Union hat (Art. 3 DSGVO).

Da Aspekt 1 nicht gegeben ist, wie sich aus verschiedenen funktionalen Anforderungen, insbesondere II-D1, ergibt, gilt die DSGVO für unsere Anwendung nicht. Dadurch entfällt das Einhalten dieses Rechtsgebiets. Neben dem Wegfallen der zuvor genannten Verpflichtungen müssen unter anderem zusätzlich keine Funktionalitäten implementiert werden, welche die DSGVO vorschreibt. Hierzu zählen zum Beispiel die Funktionen zum Herunterladen oder Löschen der eigenen Daten. Außerdem braucht es keine Juristen, welche die Einhaltung der DSGVO speziell überprüfen. Abgesehen davon ergibt sich dadurch noch eine weitere positive Folge für die Anwendung: Es werden Zeit und Kosten gespart.

4) *Einfachheit*: In diesem Kapitel soll nun näher darauf eingegangen werden, inwiefern Einfachheit eine Rolle spielt und welche Prinzipien zur Umsetzung der Einfachheit vor allem im Vordergrund stehen. Diese nicht-funktionale Anforderung dient insbesondere zur Erfüllung der User Stories 3, 4, 7, 10, 11, 18, 19 und 20.

Unsere Anwendung wird von zahlreichen Nutzern bedient, die sich wenig oder überhaupt nicht mit digitalen Geräten auskennen. Auch diese sollen die Anwendung ohne größere Schwierigkeiten nutzen und von ihren Vorteilen Gebrauch machen können. Nur so ist garantiert, dass die Anwendung auch tatsächlich von Lehrern genutzt und von Schülern angenommen wird.

Um die Anwendung einfach zu gestalten, werden unter anderem die folgenden drei Prinzipien beachtet.

- Fokus auf das Wesentliche: Befinden sich auf einem Screen viele UI-Elemente, sind selbst erfahrene Nutzer im ersten Moment davon überfordert, welche Aktion wie möglich ist. Um dies zu verhindern, werden auf jedem Screen so wenige UI-Elemente wie möglich angezeigt. Wären zu viele UI-Elemente notwendig, wird der Screen in zwei separate Screens aufgeteilt.
- Klare Struktur: Unsere Anwendung besteht aus verschiedenen Screens, welche zueinander in Beziehungen stehen. Um die App bedienen zu können, muss der Nutzer ein Verständnis des Navigationsflusses besitzen und wissen, wo er sich in diesem Navigationsfluss aktuell befindet. Durch entsprechende UI-Elemente wird dafür Sorge getragen, dass der Nutzer stets weiß, an welchem Ort innerhalb der Anwendung er sich befindet.
- Schlichte Visualisierungen: Es gibt viele verschiedene Designstile, von Flat, über Neumorphismus bis hin zu Skeuomorphismus. Design ist jedoch kein Selbstzweck. Es soll die Inhalte hervorheben und nicht davon ablenken. Dementsprechend wird ein schlichtes Design zur Umsetzung genutzt, welches hauptsächlich auf wenige Formen, Schatten, wenige unauffällige Farben und eine unauffällige Schriftart setzt. [17]

5) *Ortsunabhängigkeit und Skalierbarkeit*: Als nächstes soll die Notwendigkeit der Ortsunabhängigkeit und Skalierbarkeit erklärt werden. Diese nicht-funktionale Anforderung dient insbesondere zur Erfüllung der User Story 3.

Die Anwendung soll ohne Installation und Betrieb vor Ort in den Schulen benutzbar sein. Außerdem sollen die Schüler und Lehrer beide auf dieselben Daten zugreifen können. Um diesen Anforderungen zu genügen, benötigt es zwingend eine Client-Server Architektur. Hierbei werden Daten und Verarbeitungslogik auf einem Rechner beziehungsweise einem Rechnerverbund, dem Server, gebündelt. Die Clients, also Geräte der Schüler und Lehrer, greifen über das Internet auf diesen Server zu. [4], [38]

Die Nutzung eines einzelnen Servers hat jedoch auch zur Folge, dass dieser alle Daten aller Schulen speichern und alle Anfragen von allen Nutzern verarbeiten muss. In der ersten Zeit nach der Veröffentlichung der App mögen der benötigte Speicher und die aufzuwendende Leistung überschaubar sein. Wenn die Anwendung und ihre Nutzerzahlen jedoch wachsen, werden der Speicher und die Leistung eines einzigen Computers nicht mehr ausreichen. Aus diesem Grund muss die Architektur der Anwendung so beschaffen sein, dass sie mit einer steigenden Anzahl von Schulen mit wachsen kann. Dazu werden alle Teile der Anwendung, welcher einer großen Last ausgesetzt sein könnten, containerisiert. Damit die Infrastruktur unabhängig von einem konkreter Hoster ist, werden ausschließlich Technologien zur Containerisierung genutzt, welche die Open Container Initiative Specification (OCI-Spec) einhalten. [4], [38], [48]

#### D. Funktionale Anforderungen

Wie zuvor erwähnt, sollen nun auch die funktionalen Anforderungen näher betrachtet werden. Wie der Name vermuten

lässt, geht es hier um die tatsächliche Funktionalität der Software, das heißt, um die konkreten Aktionen, die Nutzer mit ihr durchführen können. Daher sind funktionale Anforderungen im Gegensatz zu nicht-funktionalen Anforderungen kundenspezifisch. [4], [5]

*1) Benutzerverwaltung:* Als erstes werden die funktionalen Anforderungen aus dem Bereich Benutzerverwaltung betrachtet. Diese dienen insbesondere zur Erfüllung der User Stories 4, 5, 6, 7, 8, 11, 18 und 20.

*a) Registrierung der Schule:* Das Anlegen einer neuen Schule erfolgt durch den Softwareanbieter. Hierzu nimmt das Sekretariat der Schule per E-Mail Kontakt zu dem Softwareanbieter auf. Daraufhin legt der Softwareanbieter die Schule im System an und erstellt einen Registriercode für Lehrer. Dabei handelt es sich um einen 6-stelligen Code, welcher aus einer zufälligen Folge aus groß geschriebene Buchstaben besteht. Ein Beispiel wäre der Registriercode HZZNGA. Darüber hinaus wird zur sicheren Authentifizierung eine zweizeilige Tabelle generiert. Diese enthält als erste Zeile die Positionen 1 bis 20. In der zweiten Zeile befindet sich in jeder Zelle das zur jeweiligen Position gehörige Zeichen, das ein groß oder klein geschriebener Buchstabe oder ein Sonderzeichen sein kann. Eine beispielhafte Tabelle wäre die folgende, wobei aus Platzgründen nur die ersten 12 Positionen abgebildet sind.

1	2	3	4	5	6	7	8	9	10	11	12	...
h	m	J	\$	6	B	*	#	f	A	8	6	...

Den Registriercode für Lehrer und die Tabelle zur Authentifizierung übermittelt der Softwareanbieter anschließend dem Sekretariat der Schule. [23]

*b) Registrierung als Lehrer:* Wie in II-D1a beschrieben wurde, erhält das Sekretariat von dem Softwareanbieter einen Registriercode für Lehrer. Dieser wird ausschließlich den Lehrern der Schule zur Verfügung gestellt, indem er beispielsweise im Lehrzimmer ausgehängt wird. Möchte sich ein Lehrer in der Software registrieren, so muss er lediglich den genannten Registriercode eingeben. Daraufhin wird ein Account erstellt, welchem die Rolle Lehrer zugewiesen wird, da bei der Registrierung der Registriercode für Lehrer genutzt wurde. Zudem erhält der Lehrer automatisch einen vom System generierten und nicht änderbaren Nutzernamen. Dieser setzt sich aus einem deutschen Adjektiv, einem deutschen Tiernamen und einer vierstelligen Ziffernfolge zusammen, sodass er sich leicht gemerkt werden kann. Beispieldaten Nutzernamen sind BlauerDelfin4852 oder SchlauerFuchs3384. Insofern der Nutzer seinen Nutzernamen geheim hält, ist die Anonymität somit sichergestellt, da in der Software keine benutzerbezogenen Daten gespeichert und mit dem Account verknüpft werden.

*c) Anforderung eines neuen Lehrer-Registriercodes:* Des Weiteren soll erwähnt werden, dass Registriercodes für Lehrer eine Woche lang gültig sind, um auch Lehrern, die die Schule nur an bestimmten Wochentagen besuchen, die Möglichkeit zu geben, sich in der Software zu registrieren. Wird ein neuer Registriercode für Lehrer benötigt, da der vorherige Registriercode ungültig geworden oder in die Hände von

Schülern gelangt ist, so kann das Sekretariat einen neuen anfordern. Diese und andere Funktionen der Software, welche Lehrer und Schüler nicht ausführen können sollen, sind hinter einem separaten Pfad in der Software versteckt. Wird dieser Pfad aufgerufen, so wird zunächst verlangt, dass sich der Nutzer authentifiziert. Hierzu wird die in II-D1a beschriebene Tabelle benötigt, da eine Aufforderung in der Software erfolgt, beispielsweise die Ziffern der Positionen 5, 7, 9, 13, 16 und 18 einzugeben. Nach erfolgreicher Authentifizierung werden dem Nutzer die verschiedenen Funktionen angezeigt, darunter auch die Option, einen neuen Registriercode für Lehrer zu generieren. Mit Auswahl dieser Option wird ein neuer Registriercode für Lehrer erzeugt und in der Software angezeigt. Der vorherige Registriercode für Lehrer wird durch diese Aktion ungültig, insofern er noch gültig war.

*d) Registrierung als Schüler:* Die Registrierung von Schülern erfolgt ähnlich wie die Registrierung von Lehrern. Auch hier muss ein 6-stelliger Registriercode eingegeben werden, woraufhin ein Account mit einem automatisch generierten, unveränderbaren Nutzernamen erstellt wird, der wie in II-D1b beschrieben zum Beispiel BlauerDelfin4852 lautet. Während Lehrer ihren Registriercode von dem Sekretariat erhalten und ihrem Account die Rolle Lehrer zugewiesen wird, bekommen Schüler ihren Registriercode von einem Lehrer und ihr Account erhält die Rolle Schüler. Die Registriercodes für Schüler werden in II-D2a noch einmal genauer erläutert.

*e) Login:* Nach der Registrierung benötigt ein Nutzer stets nur seinen Nutzernamen, um sich in der Software einzuloggen. Dieser ist wie in II-D1b erläutert wurde leicht zu merken, nicht durch den Nutzer veränderbar und somit sicher anonym, insofern der Nutzer ihn geheim hält.

Vergisst ein Nutzer seinen Nutzernamen, so muss er sich mit einem Registriercode für Lehrer beziehungsweise für Schüler neu registrieren und hat keine Möglichkeit mehr, auf seinen ursprünglichen Account in der Software zuzugreifen. Dies ist dadurch begründet, dass von einer realen Person keine Rückschlüsse auf ihren Nutzernamen gezogen werden können und umgekehrt. Was die in der Software für Accounts gespeicherten Daten betrifft, so ist es einem Lehrer ohne Zugriff auf seinen ursprünglichen Account nicht möglich, neue Registriercodes für seine Fächer zu erhalten (siehe II-D2b). Er muss daher seine Fächer alle neu erstellen und seinen Schülern neue Registriercodes für die Fächer geben. Ein Schüler kann hingegen bei Verlust seines Passworts seine Fächer mithilfe von neuen Registriercodes, die er von seinen Lehrern erhält, mit einem neuen Account verknüpfen und so wiederherstellen. Daten wie beispielsweise Stempel für abgeschlossene Übungen (siehe II-D2h) sind hingegen nicht wiederherstellbar, jedoch wäre ihr Verlust auch möglich, wenn sie in physischer Form vorliegen würden.

*f) Logout:* Das Abmelden aus der Software erfolgt wie bei anderen Anwendungen üblich mit einem Logout Button. Wird dieser angeklickt, so wird der Nutzername aus dem lokalen Speicher der Software gelöscht. Um erneut Zugriff auf seinen Account zu erhalten, muss der Nutzer sich mit seinem Nutzernamen erneut einloggen.

*g) Accountlöschung:* Möchte ein Nutzer seinen Account löschen, so findet er hierfür einen Button in der Software. Betätigt der Nutzer diesen, so wird sein Account unwiderstehlich gelöscht. Der Zugriff auf die Software ist erst nach erneuter Registrierung mit einem Registriercode für Lehrer beziehungsweise Schüler möglich.

Darüber hinaus soll erwähnt werden, dass am Ende jedes Schuljahres alle Accounts von Lehrern und Schülern gelöscht werden. Dadurch wird sichergestellt, dass die Accounts von Lehrern und Schülern, die die Schule verlassen, sowie von außenstehenden Personen, die unberechtigterweise an einen Registriercode gekommen sind, gelöscht werden. Zudem kann davon ausgegangen werden, dass insbesondere viele Schüler ihren Nutzernamen ohnehin über die Sommerferien vergessen würden.

*2) Fächerverwaltung:* Im Folgenden werden die zur Fächerverwaltung gehörenden funktionalen Anforderungen thematisiert. Sie setzen hauptsächlich die User Stories 5, 6, 7, 8, 9, 12, 13, 14, 16, 17 und 18 um.

*a) Facherstellung und Schüler-Registriercodes:* Fächer können nur von Nutzern erstellt werden, welche mit einem Nutzernamen eingeloggt sind, dessen zugehöriger Account die Rolle Lehrer hat. Bei der Facherstellung muss ein Fachname eingegeben sowie ein Buch ausgewählt werden. Welche Vorteile dies hat, wird in II-D2f beschrieben. Wurde das Fach erfolgreich erstellt, so wird ein neuer Registriercode für Schüler in der Software angezeigt. Dieser kann zum einen von Schülern genutzt werden, welche noch nicht registriert sind und damit noch keinen Account für die Software haben. Gibt ein solcher Schüler den Registriercode in die Software ein, so wird für ihn ein Account mit der Rolle Schüler erstellt (siehe II-D1d) und zugleich auch dem Fach zugewiesen. Zum anderen kann der Registriercode für Schüler auch von Schülern verwendet werden, welche bereits einen Account für die Software haben und mit diesem eingeloggt sind. Dabei dient der Registriercode dann nicht zur Accounterstellung, sondern nur dazu, den Account des jeweiligen Schülers mit dem Fach zu verknüpfen.

*b) Anforderung eines neuen Schüler-Registriercodes:* Auch Registriercodes für Schüler haben eine begrenzte Gültigkeitsdauer. Diese ist anders als bei Lehrern nicht eine Woche, sondern lediglich eine Stunde. Dies ist dadurch begründet, dass ein Registriercode für Schüler in der ersten Unterrichtsstunde des jeweiligen Faches zu Beginn des Schuljahres eingegeben werden soll und anschließend nicht mehr benötigt wird. Eine längere Gültigkeitsdauer würde somit wenig positiven Nutzen haben, sondern lediglich das Risiko erhöhen, dass Schüler den Registriercode unberechtigterweise an außenstehende Personen weitergeben. Selbstverständlich kann es dennoch Fälle geben, wo ein Registriercode für Schüler auch nach seiner Gültigkeit benötigt wird, beispielsweise wenn ein Schüler in der ersten Unterrichtsstunde eines Faches krank war. In diesem Fall kann der Lehrer in der Software für das Fach einen neuen Registriercode für Schüler erzeugen, welcher ebenfalls wieder eine Stunde lang gültig ist.

*c) Fächerübersicht:* Für eingeloggte Nutzer ist die Fächerübersicht die grundlegende Benutzeroberfläche der Software. In ihr hat ein Nutzer die Möglichkeit, all seine Fächer zu sehen. Die Fächerübersicht sieht für Lehrer und Schüler nahezu identisch aus.

*d) Fachdetails:* Während die Fächerübersicht (siehe II-D2c) alle Fächer eines Nutzers zeigt, so bezieht sich die Fachdetailsseite nur auf ein einziges Fach. Hier sieht der Nutzer alle Übungen und Angaben zu dem Buch dieses Faches. Auch die Fachdetailsseite sieht für Lehrer und Schüler nahezu identisch aus.

*e) Übungsvorlagenbaukasten:* Um die Erstellung von Übungen für Lehrer so einfach wie möglich zu gestalten, gibt es in der Software vorgefertigte Übungsvorlagen zu Büchern. Bevor ein Lehrer auf diese Übungen zugreifen kann (siehe II-D2f), müssen sie jedoch erst einmal erstellt werden. Diese Übungsvorlagen werden entweder von den Betreibern der Software selbst erstellt oder-und das ist der häufigere Fall- von dem Verlag des entsprechenden Buches. Zum einen muss der Vorgang der Übungsvorlagenerstellung so einfach wie möglich durchzuführen sein, damit auch weniger technikaffine Personen damit zurecht kommen. Zum anderen muss der Übungsvorlagenbaukasten selbst, aber auch die entworfenen Übungsvorlagen vollständig barrierefrei sein. Als Grundlage dafür kommt ausschließlich eine grafische Benutzeroberfläche infrage. Jegliche andere Form von Benutzeroberfläche wie z. B. die Nutzung der Befehlszeilenschnittstelle kann im Vorhinein ausgeschlossen werden, da diese viel zu kompliziert zu nutzen ist. Die grafische Benutzeroberfläche selbst erfüllt dabei alle in II-C2 genannten Anforderungen. Ebenso ist die grafische Benutzeroberfläche so gestaltet, dass derjenige, der die Übungsvorlage erstellt, dazu gezwungen ist, alle Informationen bei der Übungsvorlagenerstellung anzugeben, die zur barrierefreien Darstellung der Übung nötig sind. Beinhaltet die Übungsvorlage zum Beispiel Bilder, so forciert die grafische Benutzeroberfläche die Angabe von alternativen Texten für die Bilder.

*f) Übungsauswahl:* Wie zuvor in II-D2a erwähnt wurde, muss bei der Facherstellung ein Buch ausgewählt werden, welches der Lehrer in dem Fach verwendet. Für dieses Fach können dann speziell für dieses Buch vorgefertigte Übungsvorlagen ausgewählt werden, auf denen basierend Übungen erstellt werden können. Wer diese Übungsvorlagen erstellt und wie er dies tut, wurde im vorherigen Abschnitt II-D2e erläutert. Um eine zum Lernstoff passende Übungsvorlage zu finden, kann das Kapitel des Buches ausgewählt werden. Je nach Übungsvorlage können dann ein paar wenige Einstellungen getroffen werden. So ist es bei Vokabelübungsvorlagen beispielsweise möglich, die Wörter aus dem gewählten Kapitel, die abgefragt werden sollen, und die Anzahl der Wörter, welche für den erfolgreichen Abschluss der Übung gewusst werden müssen, festzulegen. Der Lehrer kann zudem für Übungstypen, wo dies sinnvoll ist, die vorge sehene Lösung ansehen, um die Übung nachzuvollziehen.

*g) Übungsdurchführung:* Schüler sehen in der Fachdetailsseite (siehe II-D2d) die Übungen, die der Lehrer für

dieses Fach erstellt hat, und können sie durchführen. Je nach Übungstyp sieht die Benutzeroberfläche unterschiedlich aus. So kann beispielsweise verlangt sein, Begriffe in Eingabefelder einzugeben, Kärtchen mit Begriffen mit drag and drop zu verschieben, zusammengehörige Begriffe miteinander zu verbinden oder jeweils zwei Kärtchen mit Begriffen nacheinander anzuklicken. Ebenfalls könnten abgesehen von Begriffen zum Beispiel auch Bilder genutzt werden. Bei Vokabelübungen könnte zudem die Aussprache fremdsprachiger Begriffe mithilfe von Audiodateien angehört werden. Darüber hinaus ist es auch denkbar, dass in einer einzigen Übung mehrere der genannten Methoden zur Lösungseingabe genutzt werden können, um den Schüler entscheiden zu lassen, welche Eingabemöglichkeit er präferiert.

h) *Stempelsystem*: Nach erfolgreichem Abschluss einer Übung erhält der entsprechende Schüler für diese Übung einen Stempel. Dieser zeigt dem Schüler zum einen den Lernfortschritt in der Software an, zum anderen soll er motivieren, weiter zu üben. Insofern ein Lehrer dies umsetzen möchte, wäre es zusätzlich denkbar, dass ein Schüler ein Fleißkärtchen erhält, wenn er dem Lehrer seinen Stempel für die aktuelle Übung in der Software zeigt. Hat ein Schüler beispielsweise fünf Fleißkärtchen gesammelt, könnte er mit einer guten mündlichen Note in dem Fach belohnt werden. [6]

i) *Übungsstatistiken*: In der Software wird für eine Übung weder die Anzahl der Versuche noch die Zeit gespeichert, die ein Schüler benötigt hat, um die Übung erfolgreich abzuschließen. Ebenso wenig wird die erreichte Punktzahl festgehalten. Stattdessen wird lediglich gespeichert, ob ein Schüler eine Übung mit Erfolg beendet hat. Der Lehrer kann folglich für jede Übung sehen, wie viele Schüler sie erledigt haben, jedoch keine weiteren Statistiken.

j) *Fachlöschung*: Ein Fach kann nur von dem Lehrer gelöscht werden, der es erstellt hat. Schüler können hingegen nur aus dem Fach austreten. Um wieder Zugriff auf dieses Fach zu erhalten, müssen sie erneut einen Registriercode für dieses Fach eingeben.

Gemeinsam mit den Accounts aller Lehrer und Schüler (siehe II-D1g) werden auch alle Fächer am Ende jedes Schuljahres gelöscht.

3) *Alexa*: In diesem Kapitel wird zuerst die Funktionsweise des digitalen Sprachassistenten Alexa und dessen Fähigkeiten erläutert. Anschließend wird betrachtet, wie und ob Alexa im Rahmen dieses Projektes eingesetzt werden kann. Dies dient insbesondere zur Erfüllung der User Story 22.

Alexa ist ein digitaler Sprachassistent von Amazon. Alles, was ein Gerät braucht, um Alexa auszuführen, sind ein Lautsprecher, ein Mikrofon und eine Internetverbindung. Alexa kann einerseits händisch auf Smartphones und Tablets installiert werden. Es können jedoch andererseits auch Smartspeaker gekauft werden, auf denen Alexa vorinstalliert ist. Smartspeaker sind Lautsprecher mit mehreren Mikrofonen, welche mit dem Internet verbunden werden und dadurch alles mitbringen, um Alexa nutzen zu können. Konkrete Beispiele für solche Smartspeaker, auf denen Amazons Alexa vorinstalliert ist, sind der Amazon Echo Dot oder der Amazon Echo Studio. [37]

Um Alexa zu nutzen, muss der Nutzer einen Satz in ein Gerät, auf dem Alexa installiert ist, einsprechen. Daraufhin wird die dabei entstandene Audioaufnahme an Amazons Server geschickt, ausgewertet und eine passende Aktion ausgeführt. Diese Aktion kann zum Beispiel darin bestehen, dass eine Information aus dem Internet gesucht wird und dem Nutzer als Antwort gesendet wird. Ebenso können Lichter im Smart-Home-System des Nutzers entsprechend dem Wunsch des Nutzers gesteuert werden. Es gibt jedoch auch die Möglichkeit, Aktionen, die durch Dritte entwickelt wurden, hinzuzufügen. Dies geschieht über die sogenannten Alexa Skills. So hat unter anderem die Deutsche Bahn einen Skill entwickelt, welcher es dem Nutzer ermöglicht, Fahrplanauskünfte abzurufen. Hierzu fragt er zum Beispiel „Alexa, frag Deutsche Bahn, wann am Freitag um 15 Uhr ein Zug von Frankfurt nach Berlin fährt“. [37]

Die Entwicklung eines solchen Skills würde sich im Rahmen unserer Software dafür eignen, Aufgaben bestimmter Aufgabentypen nicht am Handy, sondern mit der Stimme zu erledigen. Um Alexa nutzen zu können, benötigt man zwingend einen Amazon Account. In diesem werden zahlreiche personenbezogene Daten gespeichert. Über den gleichen Amazon Account werden dann Skills auf Alexa installiert. Dabei müsste dann auch der eigene Benutzername für unsere Software hinterlegt werden. Dadurch kommt es aber nun zu einem großen Problem. Da das Unternehmen Amazon Zugriff auf die Daten des Amazon Accounts hat, weiß dieses nun, wem welcher Account in unserer Anwendung gehört. Wie in II-C3 näher erläutert, müssen die Daten der Nutzer unserer Software jedoch vollständig anonymisiert sein, was dann nicht mehr möglich wäre. Aus diesem Grund wird auf die Entwicklung eines Alexa-Skills verzichtet und Alexa ist damit keine funktionale Anforderung. [37]

#### E. Prototyp-Planung

1) *Prototyp 1*: Der erste Prototyp stellt das sogenannte MVP (Minimum Viable Product) dar. Damit ist die einfachste Version eines Produkts gemeint, welche nur die grundlegendsten Funktionen umfasst. Dies hat zum Vorteil, dass mit Entwicklungsaufwand sehr frühes Feedback eingeholt werden kann. [11].

Bereits unser erster Prototyp soll alle nicht-funktionalen Anforderungen beachten, da diese von Anfang an berücksichtigt werden müssen, damit bei der Weiterentwicklung des Prototyps keine fundamentalen Änderungen an beispielsweise der Softwarearchitektur erforderlich sind. Von den funktionalen Anforderungen werden in dem ersten Prototyp dagegen nur die im Folgenden aufgelisteten erfüllt.

- II-D1a Registrierung der Schule: Hierfür wird eine einzige Schule mit fiktiven Daten angelegt und es existiert ein beispielhafter Registriercode und eine beispielhafte Tabelle zur Authentifizierung für diese Schule, da noch kein Kontakt mit einer realen Schule aufgenommen wird.
- II-D1b Registrierung als Lehrer
- II-D1d Registrierung als Schüler
- II-D1e Login

- II-D1f Logout
- II-D2a Facherstellung und Schüler-Registriercodes
- II-D2c Fächerübersicht
- II-D2d Fachdetails
- II-D2f Übungsauswahl: Es wird noch kein Kontakt mit realen Verlagen von Büchern aufgenommen. Aus diesem Grund gibt es eine vordefinierte Vokabelübung mit einigen Beispieldaten aus einem beispielhaften Französischbuch. [39]
- II-D2g Übungsdurchführung: Da eine vordefinierte Vokabelübung ausgewählt werden kann, kann auch nur diese durchgeführt werden.
- II-D2h Stempelsystem

2) *Prototyp 2:* Nach dem ersten Prototyp folgt die Entwicklung eines zweiten Prototyps, welcher ebenfalls alle nicht-funktionalen Anforderungen umsetzt. Darüber hinaus erweitert er den Funktionsumfang des ersten Prototyps um die folgenden funktionalen Anforderungen.

- II-D1g Accountlöschung
- II-D2b Anforderung eines neuen Schüler-Registriercodes
- II-D2i Übungsstatistiken
- II-D2j Fachlöschung

3) *Weitere Prototypen:* Nach der Fertigstellung des zweiten Prototypes sind noch weitere Prototypen geplant, bis eine erste Version der Software veröffentlicht werden kann. Auch hier sollen stets alle nicht-funktionalen Anforderungen beachtet werden. Zudem werden die folgenden funktionalen Anforderungen sowie möglicherweise noch weitere funktionale Anforderungen, die in Zukunft hinzukommen, umgesetzt.

- II-D1c Anforderung eines neuen Lehrer-Registriercodes
- II-D2e Übungsvorlagenbaukasten: Erst im Rahmen dieser funktionalen Anforderung wird es mehr vordefinierte Übungstypen als die genannte Vokabelübung geben.

### III. DESIGN / KONSTRUKTION

Nachdem die Anforderungen definiert wurden, soll es in diesem Kapitel um das Design und die Konstruktion der ersten beiden Prototypen unserer Software, welche wir LearnEasy genannt haben, da das einfache Lernen im Vordergrund steht, gehen. Dabei wird die Gesamtstruktur der Software betrachtet, jedoch auch auf das Backend, den serverseitigen Teil, und das Frontend, den clientseitigen Teil, im einzelnen eingegangen. Insbesondere sollen die Kommunikation zwischen diesen beiden Teilen sowie der Navigationsfluss und das optische Design unserer Anwendung im Vordergrund stehen.

#### A. Allgemeines

1) *Systemarchitektur:* Die Anwendung folgt der Client-Server-Architektur. Dabei stellt das Backend, der Server, seine Funktionalität über Endpunkte über das Internet bereit. Da es bei allen Funktionalitäten ausreicht, wenn ausschließlich der Client Anfragen sendet und der Server entsprechend antwortet, wird der Architekturstil Representational State Transfer (REST) nach Roy Fielding verwendet. Das Frontend, der Client, muss mit dem Internet verbunden sein und kann

dann die vom Backend zur Verfügung gestellten Endpunkte nutzen. Allgemein sieht das Zusammenspiel von Frontend und Backend bei einer Nutzerinteraktion wie folgt aus: Der Nutzer führt eine Aktion in der Benutzeroberfläche aus (z. B. den Klick auf einen Button). Daraufhin sendet das Frontend die für die Aktion erforderlichen Daten an einen Endpunkt. Der Server verarbeitet die Daten entsprechend. Anschließend schickt dieser eine passende Antwort an das Frontend, wobei dieses dann seine Benutzeroberfläche entsprechend aktualisiert. [40]

#### B. Backend

1) *Datenmodell:* Abbildung 1 zeigt das Datenmodell unseres Backends in Form eines UML (Unified Modeling Language) Klassendiagramms. Während es größtenteils selbsterklärend ist, soll auf die Nutzung abstrakter Klassen mit Vererbung bei den Übungsvorlagen und Übungen hingewiesen werden. Diese Konzeption ermöglicht es, jederzeit weitere Übungstypen hinzuzufügen, sodass es sehr einfach wäre, die Anwendung für jede Form von Unterricht zu verwenden. Des Weiteren soll erwähnt werden, dass die Attribute aller Klassen nur über Methoden ausgelesbar und manipulierbar sind. Folglich geben die Methoden wieder, was mit den Objekten der jeweiligen Klasse semantisch möglich ist. Zudem stellen sie durch eingebaute Validierung sicher, dass die Objekte jeder Klasse stets in einem validen Zustand sind. Zu beachten ist, dass die Methoden, welche erst im zweiten Prototyp implementiert werden, blau hervorgehoben wurden. [16]

2) *Schnittstellendefinition:* Das Backend unserer Software folgt dem Architekturparadigma REST. Dieses basiert auf den folgenden fünf Prinzipien.

- Identifizierbarkeit: Auf jede Ressource kann über einen eindeutigen Identifier, eine URI (Uniform Resource Identifier), zugegriffen werden. [40]
- Repräsentationen: Da Ressourcen in unterschiedlichen Formaten (z. B. application/json) vorliegen können, eignen sich Client und Server mithilfe von HTTP-Headern auf eine Repräsentation. [40]
- Selbstbeschreibung: Um möglichst selbsterklärend zu sein, werden bei der Schnittstelle standardisierte HTTP-Methoden (z. B. POST, GET) und Statuscodes (z. B. 200 – OK) genutzt. [40]
- Hypermedia: Die Antworten des Backends beinhalten nicht nur die gewünschten Informationen, sondern auch Verlinkungen zu weiteren Ressourcen. [40]
- Zustandslosigkeit: Die Client-Server-Kommunikation ist zustandslos und der Client schickt stets alle relevanten Informationen in jeder Anfrage mit. [40]

Die Schnittstellendefinition eines REST-Backends gibt eine Übersicht über die abfragbaren Informationen. Hierzu werden Endpunkte angegeben, die unter anderem eine HTTP-Methode und eine URI beinhalten. Für die Schnittstellendefinition wurde die standardisierte OpenAPI Spezifikation genutzt. Die Abbildungen 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17 und 18 zeigen unsere Endpunkte, wobei die erst im zweiten Prototyp umgesetzten mit einem blauen Punkt rechts oben markiert wurden. [22]

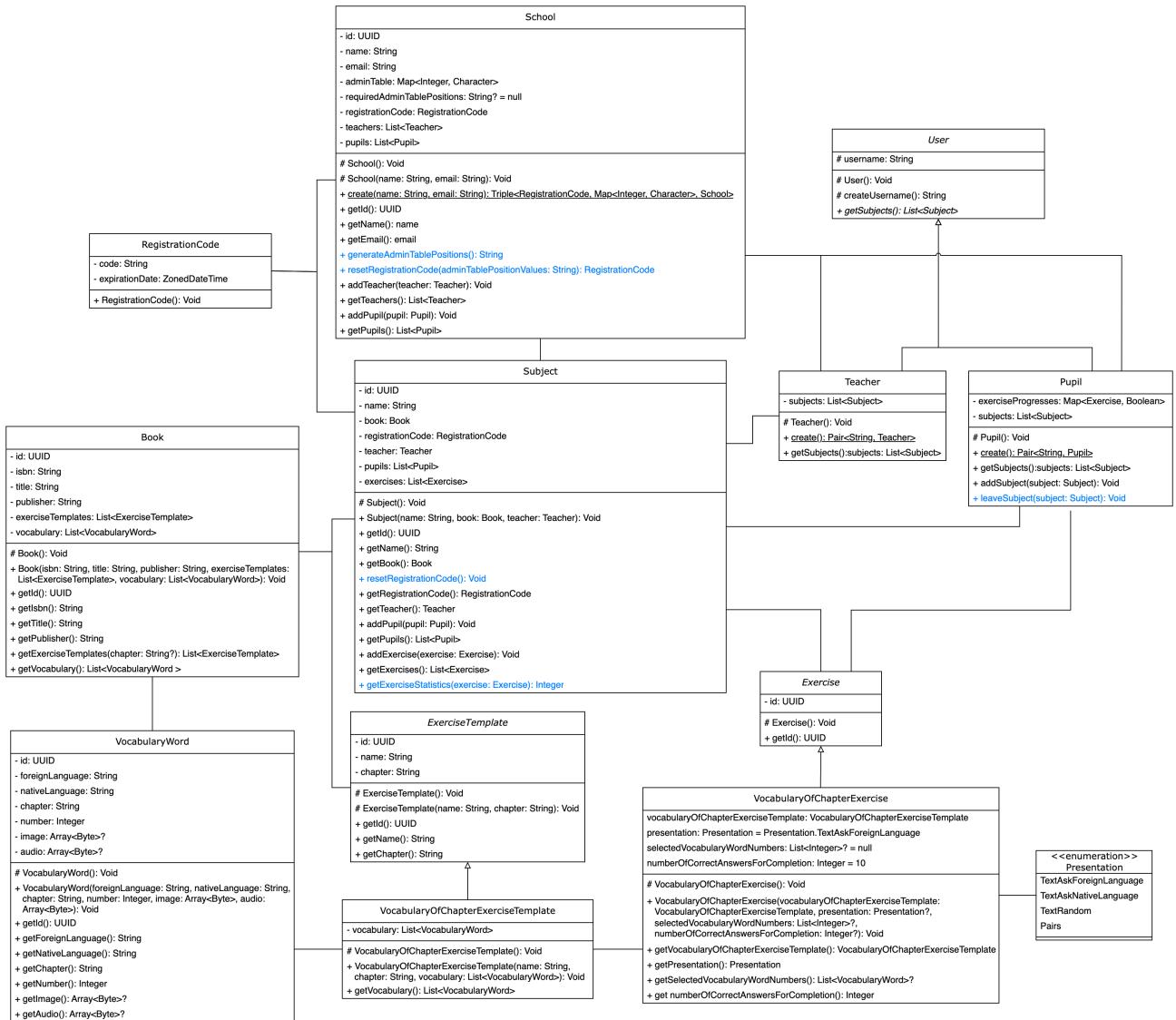


Abbildung 1. Datenmodell

GET /v1/books

Get all books.

**Parameters**

**Name** **Description**

perPage	<i>Default value : 5</i> integer(\$int32) (query) 5 maximum: 20 minimum: 5
pageNumber	pageNumber integer(\$int32) (query) minimum: 1

**Responses**

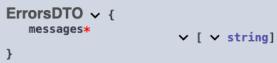
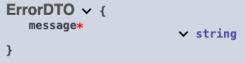
Code	Description	Links
200	OK	No links
	Media type <b>application/json</b>	
	Controls Accept header.	
	Example Value   Schema	
		
400	Bad Request	No links
	Media type <b>application/json</b>	
	Example Value   Schema	
		
500	Internal Server Error	No links
	Media type <b>application/json</b>	
	Example Value   Schema	
		

Abbildung 2. Endpunkt zum Abrufen aller Bücher

**GET /v1/books/{bookId}/exercise-templates**

Get all exercise templates of a book.

**Parameters**

Name	Description
bookId * required	bookId string(\$uuid) (path)
chapter	chapter string (query)
perPage	Default value : 50 integer(\$int32) (query)
pageNumber	pageNumber integer(\$int32) (query)

**Responses**

Code	Description	Links
200	OK	No links
	Media type <b>application/json</b>	
	Controls Accept header.	
	Example Value   Schema	
	<pre>ExerciseTemplatesPageDTO ∵ {     maximumPage*           ∵ integer(\$int32)                            minimum: 0     exerciseTemplates*     ∵ [         maxItems: 100         minItems: 10         ∵ {             oneOf -&gt;                 VocabularyOfChapterExerciseTemplatePartialDTO ∵ {                     id*                      ∵ string(\$uuid)                     name*                   ∵ string                     chapter*                ∵ string                     type*                   ∵ string                 }         }     } }</pre>	
400	Bad Request	No links
	Media type <b>application/json</b>	
	Example Value   Schema	
	<pre>ErrorsDTO ∵ {     messages*             ∵ [ ∵ string] }</pre>	
404	Not Found	No links
	Media type <b>application/json</b>	
	Example Value   Schema	
	<pre>ErrorDTO ∵ {     message*              ∵ string }</pre>	
500	Internal Server Error	No links
	Media type <b>application/json</b>	
	Example Value   Schema	
	<pre>ErrorDTO ∵ {     message*              ∵ string }</pre>	

Abbildung 3. Endpunkt zum Abrufen der Übungsvorlagen eines Buches

GET /v1/books/{bookId}/chapters

Get all chapters of a book.

**Parameters**

Name	Description
bookId * required	bookId string(\$uuid) (path)
perPage	Default value : 50 integer(\$int32) (query) 50 maximum: 100 minimum: 10
pageNumber	pageNumber integer(\$int32) (query) minimum: 1

**Responses**

Code	Description	Links
200	OK	No links
	Media type application/json	
	Controls Accept header.	
	Example Value   Schema	
	<code>ChaptersPageDTO {     maximumPage*     chapters* }</code>	
400	Bad Request	No links
	Media type application/json	
	Example Value   Schema	
	<code>ErrorsDTO {     messages* }</code>	
404	Not Found	No links
	Media type application/json	
	Example Value   Schema	
	<code>ErrorDTO {     message* }</code>	
500	Internal Server Error	No links
	Media type application/json	
	Example Value   Schema	
	<code>ErrorDTO {     message* }</code>	

Abbildung 4. Endpunkt zum Abrufen der Kapitel eines Buches

**DELETE** /v1/users/{username}

Delete a user.

**Parameters**

Name	Description
username <small>* required</small> string (path)	username

**Responses**

Code	Description	Links
204	No Content	No links
400	Bad Request	No links
	Media type <b>application/json</b>	
	Example Value   Schema	
	<code>ErrorsDTO ↴ {   messages* ↴ [ ↴ string] }</code>	
404	Not Found	No links
	Media type <b>application/json</b>	
	Example Value   Schema	
	<code>ErrorDTO ↴ {   message* ↴ string }</code>	
500	Internal Server Error	No links
	Media type <b>application/json</b>	
	Example Value   Schema	
	<code>ErrorDTO ↴ {   message* ↴ string }</code>	

Abbildung 5. Endpunkt zum Löschen eines Nutzers

**GET /v1/users/{username}**

Get details of a user.

**Parameters**

**Name** **Description**

username <small>* required</small>	string (path)	username
------------------------------------	------------------	----------

**Responses**

Code	Description	Links
200	OK	No links
	Media type	
	application/json	
	Controls Accept header.	
	Example Value   Schema	
	<pre>oneOf --&gt;     PupilDTO {         type* string         exerciseProgresses*             &lt; * &gt;             boolean     }     TeacherDTO {         type* string     } }</pre>	
400	Bad Request	No links
	Media type	
	application/json	
	Example Value   Schema	
	<pre>ErrorsDTO {     messages*         [ string ] }</pre>	
404	Not Found	No links
	Media type	
	application/json	
	Example Value   Schema	
	<pre>ErrorDTO {     message*         string }</pre>	
500	Internal Server Error	No links
	Media type	
	application/json	
	Example Value   Schema	
	<pre>ErrorDTO {     message*         string }</pre>	

Abbildung 6. Endpunkt zum Abrufen eines Nutzers

**POST** /v1/users

Create a new user.

**Parameters**

No parameters

**Request body** required

application/json

Example Value | Schema

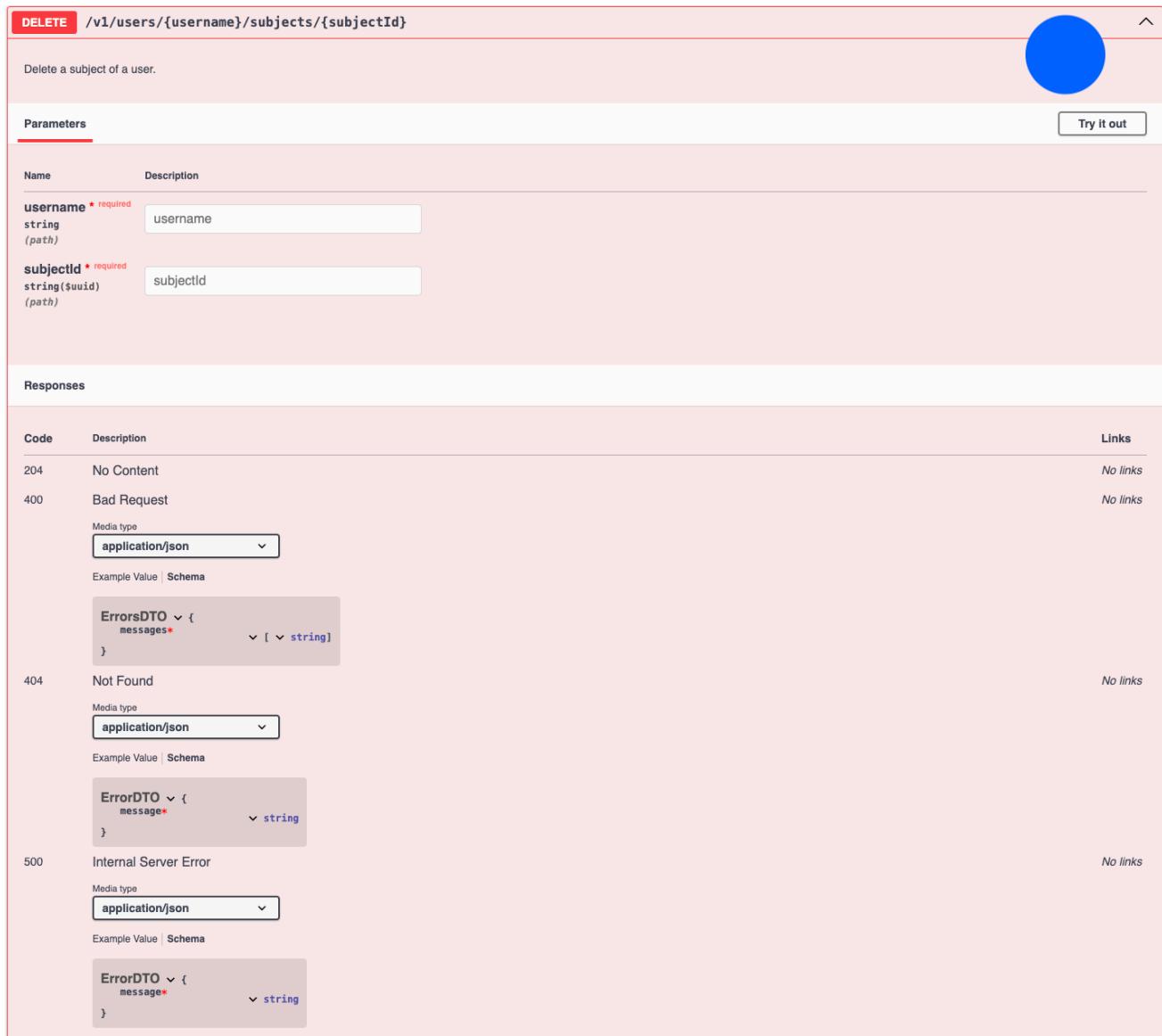
```
RegistrationCodeDTO ▶ {
  registrationCode*           ▶ string
    maxLength: 6
    minLength: 6
}
```

**Responses**

Code	Description	Links
201	Created	No links
	Media type	
	application/json ▾	
	Controls Accept header.	
	Example Value   Schema	
	UsernameDTO ▶ {     username*           ▶ string   }	
400	Bad Request	No links
	Media type	
	application/json ▾	
	Example Value   Schema	
	ErrorsDTO ▶ {     messages*           ▶ [ ▶ string]   }	
500	Internal Server Error	No links
	Media type	
	application/json ▾	
	Example Value   Schema	
	ErrorDTO ▶ {     message*           ▶ string   }	

Abbildung 7. Endpunkt zum Erstellen eines Nutzers

**DELETE** /v1/users/{username}/subjects/{subjectId}



Delete a subject of a user.

**Parameters**

Name Description

username <small>* required</small>	string (path)	username
subjectId <small>* required</small>	string(\$uuid) (path)	subjectId

**Responses**

Code	Description	Links
204	No Content	No links
400	Bad Request	No links
	Media type	
	application/json	
	Example Value   Schema	
	<code>ErrorsDTO ↴ {   messages* ↴ [ ↴ string] }</code>	
404	Not Found	No links
	Media type	
	application/json	
	Example Value   Schema	
	<code>ErrorDTO ↴ {   message* ↴ string }</code>	
500	Internal Server Error	No links
	Media type	
	application/json	
	Example Value   Schema	
	<code>ErrorDTO ↴ {   message* ↴ string }</code>	

Abbildung 8. Endpunkt zum Löschen eines Faches eines Nutzers

GET /v1/users/{username}/subjects

Get all subjects of a user.

**Parameters**

Name	Description
username* required	Username
string (path)	
perPage	Default value : 5
integer(\$int32) (query)	5
	maximum: 20 minimum: 5
pageNumber	pageNumber
integer(\$int32) (query)	minimum: 1

**Responses**

Code	Description	Links
200	OK	No links
	Media type	
	application/json	
	Controls Accept header.	
	Example Value   Schema	
	<pre>SubjectsPageDTO {     maximumPage*     subjects*         SubjectPartialDTO {             id*             name*         } }</pre>	
400	Bad Request	No links
	Media type	
	application/json	
	Example Value   Schema	
	<pre>ErrorsDTO {     messages* }</pre>	
404	Not Found	No links
	Media type	
	application/json	
	Example Value   Schema	
	<pre>ErrorDTO {     message* }</pre>	
500	Internal Server Error	No links
	Media type	
	application/json	
	Example Value   Schema	
	<pre>ErrorDTO {     message* }</pre>	

Abbildung 9. Endpunkt zum Abrufen der Fächer eines Nutzers

**GET** /v1/users/{username}/subjects/{subjectId}

Get details of a subject of a user.

**Parameters**

**Name** **Description**

**username** \* required  
string  
(path)

**subjectId** \* required  
string(\$uuid)  
(path)

**Responses**

Code	Description	Links
200	OK	No links
	Media type	
	application/json	
	Controls Accept header.	
	Example Value   Schema	
	<pre>SubjectDetailedDTO v {     id*           v string(\$uuid)     name*         v string     book*         BookDTO v {         id*           v string(\$uuid)         isbn*          v string         title*         v string         publisher*    v string     } }</pre>	
400	Bad Request	No links
	Media type	
	application/json	
	Example Value   Schema	
	<pre>ErrorsDTO v {     messages*     v [ v string] }</pre>	
404	Not Found	No links
	Media type	
	application/json	
	Example Value   Schema	
	<pre>ErrorDTO v {     message*      v string }</pre>	
500	Internal Server Error	No links
	Media type	
	application/json	
	Example Value   Schema	
	<pre>ErrorDTO v {     message*      v string }</pre>	

Abbildung 10. Endpunkt zum Abrufen eines Faches eines Nutzers

GET /v1/users/{username}/subjects/{subjectId}/registration-code

Get the registration code of a subject of a user.

Parameters

Name	Description
username * required	username string (path)
subjectId * required	subjectId string(\$uuid) (path)

Responses

Code	Description	Links
200	OK	No links
	Media type	
	<input checked="" type="button"/> application/json	
	Controls Accept header.	
	Example Value   Schema	
	<pre>RegistrationCodeDTO v {     registrationCode*   v string                         maxLength: 6                         minLength: 6 }</pre>	
400	Bad Request	No links
	Media type	
	<input checked="" type="button"/> application/json	
	Example Value   Schema	
	<pre>ErrorsDTO v {     messages*          v [ v string] }</pre>	
404	Not Found	No links
	Media type	
	<input checked="" type="button"/> application/json	
	Example Value   Schema	
	<pre>ErrorDTO v {     message*           v string }</pre>	
500	Internal Server Error	No links
	Media type	
	<input checked="" type="button"/> application/json	
	Example Value   Schema	
	<pre>ErrorDTO v {     message*           v string }</pre>	

Abbildung 11. Endpunkt zum Abrufen des Registriercodes eines Faches eines Nutzers

**POST** /v1/users/{username}/subjects

Create a new subject for a user.

**Parameters**

**Name** **Description**

**username** \* required string (path)

**Request body** required

Example Value | Schema

```
SubjectCreationDTO ▶ {
  name*           ▶ string
  bookId*        ▶ string($uuid)
}
```

**Responses**

Code	Description	Links
201	Created	No links
	Media type	
	<input type="text" value="application/json"/>	
	Controls Accept header.	
	Example Value   Schema	
	<pre>RegistrationCodeDTO ▶ {   registrationCode* ▶ string   maxLength: 6   minLength: 6 }</pre>	
400	Bad Request	No links
	Media type	
	<input type="text" value="application/json"/>	
	Example Value   Schema	
	<pre>ErrorsDTO ▶ {   messages*      ▶ [ ▶ string] }</pre>	
403	Forbidden	No links
	Media type	
	<input type="text" value="application/json"/>	
	Example Value   Schema	
	<pre>ErrorDTO ▶ {   message*       ▶ string }</pre>	
404	Not Found	No links
	Media type	
	<input type="text" value="application/json"/>	
	Example Value   Schema	
	<pre>ErrorDTO ▶ {   message*       ▶ string }</pre>	
500	Internal Server Error	No links
	Media type	
	<input type="text" value="application/json"/>	
	Example Value   Schema	
	<pre>ErrorDTO ▶ {   message*       ▶ string }</pre>	

Abbildung 12. Endpunkt zum Erstellen eines Faches für einen Nutzer

**POST** /v1/users/{username}/subjects/join

Join a subject as a user.

**Parameters**

**Name** **Description**

username <small>* required</small>	string (path)
------------------------------------	------------------

**Request body** required

Example Value | Schema

```
RegistrationCodeDTO {
    registrationCode*     string
                          maxLength: 6
                          minLength: 6
}
```

**Responses**

Code	Description	Links
204	No Content	No links
400	Bad Request	No links
	Media type	
	application/json	
	Example Value   Schema	
	ErrorsDTO {         messages*           [ string ]       }	
403	Forbidden	No links
	Media type	
	application/json	
	Example Value   Schema	
	ErrorDTO {         message*            string       }	
404	Not Found	No links
	Media type	
	application/json	
	Example Value   Schema	
	ErrorDTO {         message*            string       }	
500	Internal Server Error	No links
	Media type	
	application/json	
	Example Value   Schema	
	ErrorDTO {         message*            string       }	

Abbildung 13. Endpunkt zum Beitreten zu einem Fach als Nutzer

**GET /v1/users/{username}/subjects/{subjectId}/exercises**

Get all exercises of a subject of a user.

**Parameters**

Name	Description
username <small>* required</small>	username
subjectId <small>* required</small>	subjectId
perPage	<small>Default value : 50</small>
integer(\$int32)	50
	<small>maximum: 100</small>
	<small>minimum: 10</small>
pageNumber	<small>Default value : 1</small>
integer(\$int32)	pageNumber
	<small>minimum: 1</small>

**Responses**

Code	Description	Links
200	OK	No links
	Media type	
	application/json	
	Controls Accept header.	
	Example Value   Schema	
	<pre>ExercisesPageDTO {     maximumPage*         &lt;-- integer(\$int32)         minimum: 0     exercises         &lt;-- [             maxItems: 100             minItems: 10             &lt;-- {                 oneOf -&gt;                     VocabularyOfChapterExercisePartialDTO {                         id*                             &lt;-- string(\$uuid)                         type*                             &lt;-- string                         vocabularyOfChapterExerciseTemplate*                             &lt;-- VocabularyOfChapterExerciseTemplatePartialDTO {                                 id*                                     &gt; [...]                                 name*                                     &gt; [...]                                 chapter*                                     &gt; [...]                                 type*                                     &gt; [...]                             }                     }             ]         } }</pre>	
400	Bad Request	No links
	Media type	
	application/json	
	Example Value   Schema	
	<pre>ErrorsDTO {     messages*         &lt;-- [ &lt;-- string] }</pre>	
404	Not Found	No links
	Media type	
	application/json	
	Example Value   Schema	
	<pre>ErrorDTO {     message*         &lt;-- string }</pre>	
500	Internal Server Error	No links
	Media type	
	application/json	
	Example Value   Schema	
	<pre>ErrorDTO {     message*         &lt;-- string }</pre>	

Abbildung 14. Endpunkt zum Abrufen aller Übungen eines Faches eines Nutzers

**GET /v1/users/{username}/subjects/{subjectId}/exercises/{exerciseId}**

Get details of an exercise of a subject of a user.

**Parameters**

Name	Description
username * required	username
subjectId * required	subjectId
exerciseId * required	exerciseId

**Responses**

Code	Description	Links
200	OK	No links
	Media type application/json	
	Controls Accept header.	
	Example Value   Schema	
	<pre>v {   oneOf -&gt;     VocabularyOfChapterExerciseDetailedDTO v {       id*           v string(\$uuid)       type*         v string       vocabularyOfChapterExerciseTemplate* v VocabularyOfChapterExerciseTemplateDetailedDTO v {         id*           v string(\$uuid)         name*         v string         chapter*     v string         vocabulary*   v [VocabularyWordDTO v {           id*           v string           foreignLanguage* v string           nativeLanguage* v string           chapter*       v string           number*        v integer(\$int32)           image          v string           audio          v string         }]         presentation* v string         Enum:         selectedVocabularyWordNumbers v [ v integer(\$int32)]         numberOfWorksForCompletion* v integer(\$int32)       }     }   }</pre>	
400	Bad Request	No links
	Media type application/json	
	Example Value   Schema	
	<pre>ErrorsDTO v {   messages*   v [ v string] }</pre>	
404	Not Found	No links
	Media type application/json	
	Example Value   Schema	
	<pre>ErrorDTO v {   message*    v string }</pre>	
500	Internal Server Error	No links
	Media type application/json	
	Example Value   Schema	
	<pre>ErrorDTO v {   message*    v string }</pre>	

Abbildung 15. Endpunkt zum Abrufen einer Übung eines Faches eines Nutzers

**GET** /v1/users/{username}/subjects/{subjectId}/exercises/{exerciseId}/statistics



Get statistics of an exercise of a subject of a user.

**Parameters**

**Name** **Description**

<b>username</b> * required string (path)	username
<b>subjectId</b> * required string(\$uuid) (path)	subjectId
<b>exerciseId</b> * required string(\$uuid) (path)	exerciseId

**Responses**

<b>Code</b>	<b>Description</b>	<b>Links</b>
200	OK	No links
	Media type <b>application/json</b> ▾ Controls Accept header.	
	Example Value   Schema	
	<pre>ExerciseStatisticsDTO ▶ {     successfulCompletions* ▶ integer(\$int32)     minimum: 0 }</pre>	
400	Bad Request	No links
	Media type <b>application/json</b> ▾	
	Example Value   Schema	
	<pre>ErrorsDTO ▶ {     messages* ▶ [ ▶ string] }</pre>	
404	Not Found	No links
	Media type <b>application/json</b> ▾	
	Example Value   Schema	
	<pre>ErrorDTO ▶ {     message* ▶ string }</pre>	
500	Internal Server Error	No links
	Media type <b>application/json</b> ▾	
	Example Value   Schema	
	<pre>ErrorDTO ▶ {     message* ▶ string }</pre>	

Abbildung 16. Endpunkt zum Abrufen der Statistiken zu einer Übung eines Faches eines Nutzers

**POST /v1/users/{username}/subjects/{subjectId}/exercises**

Create a new exercise for a subject of a user.

**Parameters**

**Request body required** application/json

```

{
  "oneOf": [
    {
      "VocabularyOfChapterExerciseCreationDTO": {
        "exerciseTemplateId": "$uuid",
        "type": "string",
        "presentation": "string",
        "Enum": "string",
        "selectedVocabularyWordNumbers": [
          "integer"
        ],
        "number0fCorrectAnswersForCompletion": "integer"
      }
    }
  ]
}
  
```

**Responses**

Code	Description	Links
201	Created	No links
400	Bad Request	No links
403	Forbidden	No links
404	Not Found	No links
500	Internal Server Error	No links

Abbildung 17. Endpunkt zum Erstellen einer Übung für ein Fach für einen Nutzer

**POST** /v1/users/{username}/subjects/{subjectId}/exercises/{exerciseId}/success

Mark an exercise of a subject of a user as successful.

**Parameters**

Name	Description
username <small>* required</small>	username
subjectId <small>* required</small>	subjectId
exerciseId <small>* required</small>	exerciseId

**Responses**

Code	Description	Links
204	No Content	No links
400	Bad Request	No links
	Media type	
	application/json	
	Example Value   Schema	
	<code>ErrorsDTO ↴ {   messages* ↴ [ ↴ string] }</code>	
403	Forbidden	No links
	Media type	
	application/json	
	Example Value   Schema	
	<code>ErrorDTO ↴ {   message* ↴ string }</code>	
404	Not Found	No links
	Media type	
	application/json	
	Example Value   Schema	
	<code>ErrorDTO ↴ {   message* ↴ string }</code>	
500	Internal Server Error	No links
	Media type	
	application/json	
	Example Value   Schema	
	<code>ErrorDTO ↴ {   message* ↴ string }</code>	

Abbildung 18. Endpunkt zum Markieren einer Übung eines Faches für einen Nutzer als erfolgreich

### C. Frontend

1) *Wireframes*: Wireframes dienen dazu, den grundlegenden Navigationsfluss, das Layout und die Funktionalität einer Anwendung zu skizzieren, wobei auf detailliertere Designelemente wie beispielsweise Farben und Grafiken verzichtet wird. Die Wireframes für unsere Software sind in Abbildung 19 zu sehen und zeigen die mobile Ansicht, welche sich jedoch nahezu nicht von der Darstellung auf größeren Bildschirmen unterscheidet. Bei der Gestaltung der Benutzeroberfläche wurde besonders auf einen einfachen Aufbau mit erkennbarem Lesefluss geachtet und Barrierefreiheit berücksichtigt. Wo es notwendig ist, wird zudem bei dem Navigationsfluss zwischen verschiedenen Nutzerrollen differenziert. Auch wird zwischen zwei Darstellungsarten bei der Übungsdurchführung unterschieden: Text und Paare. Während die erste Darstellungsart die textliche Eingabe der Übersetzung der gefragten Vokabel verlangt, so werden bei der zweiten immer sechs Karten angezeigt. Von diesen Karten beinhalten drei die fremde Sprache und drei die Muttersprache von jeweils drei Vokabeln und es müssen immer die zusammengehörigen Paare gefunden werden. Die Karten mit den Vokabeln in Muttersprachen beinhalten zudem ein Bild, insofern sich dies für die Vokabel anbietet. Die Karten mit den Vokabeln in der fremden Sprache ermöglichen es dagegen, eine Audiodatei abzuspielen, bei der die Vokabel vorgelesen wird. Das Abspielen der Audiodatei für die Vokabeln in fremder Sprache ist auch bei der Darstellungsart Text bei der Übungsdurchführung möglich. Wie zuvor im Datenmodell wurden Elemente und Navigationsflüsse, welche erst im zweiten Prototyp umgesetzt werden, in den Wireframes blau markiert. [17]

2) *Screendesign*: Ein Screendesign ist im Vergleich zu Wireframes detaillierter, indem es insbesondere Farben, Bilder, Icons, Animationen und Typografie beinhaltet. Somit wird das endgültige Aussehen der Anwendung veranschaulicht. Die Abbildungen 20, 21, 22 und 23 zeigen das Screendesign unserer Anwendung an vier ausgewählten Screens, die bereits als Wireframes skizziert wurden. Dabei fallen unsere Primärfarbe Orange und unsere Sekundärfarbe Grau auf. Diese wurden passend zu dem Logo (siehe Abbildung 24) unserer Software LearnEasy gewählt, welches einen Fuchs zeigt, da das Tier in Redewendungen und Fabeln häufig als schlau bezeichnet wird und dies für eine Lernanwendung sehr passend schien. Des weiteren ist bei den Eingabefeldern zu sehen, dass die Nutzereingaben validiert werden und dem Nutzer angezeigt wird, ob die Eingabe valide ist oder nicht (siehe Abbildung 20). Bei der Abbildung 21, welche den Screen Fachdetails aus der Sicht eines Schülers zeigt, ist zudem ein Fuchs-Stempel zu sehen. Dieser signalisiert, dass der in der Software angemeldete Schüler eine Übung erfolgreich abgeschlossen hat. Wie bereits in den Wireframes wird bei der Übungsdurchführung zwischen den zwei Darstellungsarten Text und Paare unterschieden (siehe Abbildungen 22 und 23). Hierbei ist nun neben den Vokabeln in fremder Sprache stets ein Megaphon-Icon zu finden, mit welchem die Audiodatei abgespielt werden kann. Zudem beinhalten die Karten in der

Muttersprache bei der Darstellungsart Paare teilweise Bilder. Es ist außerdem zu sehen, was passiert, wenn die Karten bei dieser Darstellungsart angeklickt und zugeordnet werden. Die momentan ausgewählte Karte ist hier stets in der Primärfarbe Orange hinterlegt. Um zu signalisieren, dass bei der Auswahl einer Karte mit einer Vokabel in Muttersprache bzw. einer Vokabel in der fremden Sprache keine Karten mit gleichsprachigen Vokabeln ausgewählt werden können, werden die Karten mit gleichsprachigen Vokabeln hellgrau hinterlegt. Korrekt zugeordnete Karten werden hingegen grün hinterlegt. Es soll erwähnt werden, dass alle Icons und Bilder in unserer Software Flaticons sind mit Ausnahme des Megaphon-Icons, welches ein Bootstrap Icon ist. [17]–[20]

## IV. IMPLEMENTATION

Das vorliegende Kapitel beschäftigt sich mit der konkreten Umsetzung des ersten Prototyps von LearnEasy. Hierbei sind vor allem die genutzten Technologien relevant, das bedeutet Programmiersprachen, Frameworks und Bibliotheken. Ebenfalls soll die grundlegende Codestruktur im Backend und im Frontend betrachtet werden. Zuletzt liegt im Frontend auch ein besonderer Fokus auf der Sicherstellung von Barrierefreiheit.

### A. Backend

1) *Technologien*: Als Programmiersprache für das Backend wird Java verwendet. Dies hat insbesondere die folgenden fünf Gründe.

- Automatisches Speichermanagement: Die Sprache muss ihren gesamten Speicher auf irgendeine Weise vollständig selbstständig verwalten. Dadurch werden bereits im Vorhinein Memory Leaks, welche die häufigste Sicherheitslücke darstellen, ausgeschlossen. [41]
- Kompiliert: Die Sprache wird in Bytecode kompiliert. Dadurch wird die Fehlererkennung zur Kompilierzeit, also zum frühestmöglichen Zeitpunkt, ermöglicht. Außerdem optimiert der Kompiler entstehenden Bytecode wodurch die Laufzeitleistung steigt. [41]
- Statische Typisierung: Die Sprache ist statisch typisiert. Dadurch dass das Programm überhaupt nur erstellt, also kompiliert, werden kann, wenn dieses, was die Typen angeht, in sich logisch ist, werden bereits im Vorhinein einige Fehler ausgeschlossen. [41]
- Bibliothekenanzahl: Es gibt viele Bibliotheken für die Sprache. Dadurch müssen die meisten Funktionalitäten nicht von Grund auf neu entwickelt werden, was die Entwicklungszeit und die Anzahl der auftretenden Fehler stark verringert. [42]
- Nutzung durch Teammitglieder: Alle Teammitglieder beherrschen Java. Darüber hinaus ist die Sprache sehr bekannt. Dadurch können alle Teammitglieder am Backend mitarbeiten. Daneben können weitere Teammitglieder leichter gefunden werden.

Aufbauend auf Java wird das Framework Spring Boot und einige seiner Starter-Packages verwendet, welches insgesamt

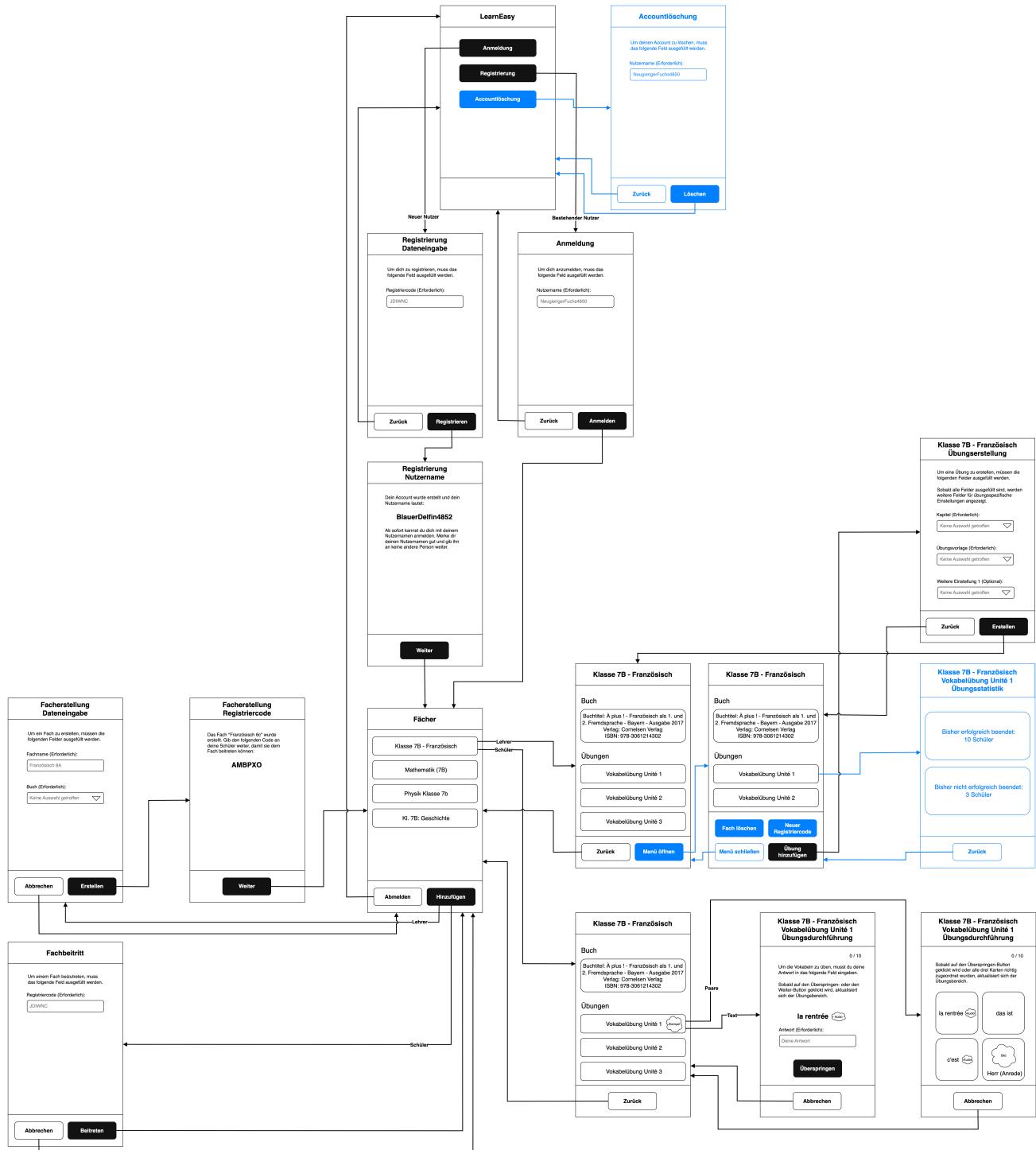


Abbildung 19. Wireframes

**Facherstellung  
Dateneingabe**

Um ein Fach zu erstellen, müssen die folgenden Felder ausgefüllt werden.

Fachname (Erforderlich):  
 ✓

Buch (Erforderlich):  
 ✓ ▾

**Abbrechen** **Erstellen**

Abbildung 20. Screendesign Facherstellung

**Französisch 6c  
Vokabelübung Unité 1  
Übungsdurchführung**

1 / 10

Um die Vokabeln zu üben, musst du deine Antwort in das folgende Feld eingeben

Sobald auf den Überspringen- oder den Weiter-Button geklickt wird, aktualisiert sich der Übungsbereich.

**vous êtes**

Antwort (Erforderlich):  
 ✓

**Weiter**

**Abbrechen**

Abbildung 22. Screendesign Übungsdurchführung mit Darstellungsart Text

**Französisch 6c**

**Buch**

Buchtitel: À plus ! - Französisch als 1. und 2. Fremdsprache - Bayern - Ausgabe 2017  
Verlag: Cornelsen Verlag  
ISBN: 978-3061214302

**Übungen**

Vokabelübung Unité 2

Vokabelübung Unité 1

**Zurück**

Abbildung 21. Screendesign Fachdetails als Schüler

**Französisch 6c  
Vokabelübung Unité 2  
Übungsdurchführung**

1 / 10

Sobald auf den Überspringen-Button geklickt wird oder alle drei Karten richtig zugeordnet wurden, aktualisiert sich der Übungsbereich.


**Überspringen**

**Abbrechen**

Abbildung 23. Screendesign Übungsdurchführung mit Darstellungsart Paare



Abbildung 24. Logo von LearnEasy

zahlreiche Funktionen wie ein ORM und Dependency Injection mit sich bringt. Die Entscheidung ergibt sich aus einer Kombination der folgenden Gründe.

- Erfahrung der Teammitglieder: Jedes Teammitglied hat schon in mindestens zwei Projekten mit dem Framework gearbeitet, wodurch die Entwicklung des Backends wesentlich schneller vorstatten geht.
- Ökosystem: Im Gegensatz zu anderen gleichartigen Frameworks für Java gibt es für Spring Boot die mit Abstand meisten direkt kompatiblen Erweiterungen. [42]
- Performance: Das Framework kann eine angemessene Zahl an Anfragen gleichzeitig und recht effizient abarbeiten. Dadurch werden die zur Verfügung stehenden Ressourcen kosteneffizient genutzt. [43]

2) *Struktur:* Das Backend besteht aus den Packages Model, Repositories, Controller und Data Transfer Objects.

Das Model ist das Datenmodell. Auf dieses wurde bereits in Kapitel III-B1 näher eingegangen. Zusätzlich muss noch erwähnt werden dass die Model-Klassen und ihre Bestandteile mit Annotationen wie z. B. „@Id“ versehen sind. Diese müssen angegeben werden, damit die Repositories die auf Basis der Model-Klassen erstellten Objekte korrekt in der Datenbank speichern können. [44]

Die Repositories sind vordefinierte Interfaces, welche durch ihre Methoden einen abstrahierten Zugriff auf die Datenbank ermöglichen, sodass keine Anfragen in SQL oder anderen derartigen Sprachen händisch geschrieben werden müssen und dadurch die Datenbank jederzeit ausgetauscht werden kann. Alleine die Signaturen der Methoden eines Repositories bestimmen, was in der Datenbank geändert beziehungsweise aus der Datenbank abgefragt wird. Es ist auch möglich, eigene Interfaces zu erstellen, welche von einem bestehenden Repository erben, um Methoden Signaturen für eigene Aktionen zu definieren. Das Framework erstellt zur Laufzeit automatisch ein Objekt welches das entsprechende Interface implementiert. Dieses Objekt kann dann via Dependency Injection genutzt werden. [44]

Die Controller sind Klassen, die Methoden beinhalten, wobei jede Methode einen Endpunkt darstellt. Jede solche Klasse und Methode ist mit Annotationen bestückt, welche dem Framework Informationen liefern, damit dieses die Methoden der Klasse als Endpunkte verwenden kann. [45]

Data Transfer Objects sind die Klassen, welche das Format der Request Bodys und Response Bodys der Endpunkte definieren.

#### B. Frontend

1) *Technologien:* Bei der Auswahl der Technologien für das Frontend unserer Software waren vor allem drei Kriterien wichtig. Als Erstes musste eine plattformunabhängige Entwicklung möglich sein, das heißt, dass ein einziger Code geschrieben wird und dieser auf Smartphone, Tablet und PC unabhängig von dem Betriebssystem ausgeführt werden kann. Das zweite Kriterium war, dass die Technologien es möglichst einfach und direkt ermöglichen müssen, Barrierefreiheit nach den WCAG 2.1 umzusetzen. Als Drittes sollten bevorzugt Technologien genutzt werden, mit welchen die beteiligten Entwickler bereits gearbeitet haben. Alle drei genannten Kriterien verkürzen die Entwicklungszeit und sorgen für eine schnelle Entwicklung eines ersten Prototyps.

Es gibt verschiedene Möglichkeiten zur Entwicklung von Cross-Plattform-Anwendungen, darunter Flutter, Compose Multiplattform und PWAs. Bereits in der nicht-funktionalen Anforderung Plattformunabhängigkeit (siehe II-C1) wurde entschieden, eine PWA zu implementieren. Da diese in HTML (Hypertext Markup Language), CSS (Cascading Style Sheets) und JavaScript geschrieben wird, können die in der nicht-funktionalen Anforderung Barrierefreiheit (siehe II-C2) aufgezählten Techniken direkt umgesetzt werden. Zudem haben beide beteiligten Entwickler bereits Vorkenntnisse mit den genannten drei Sprachen. [23]

Um eine optimale Benutzererfahrung mit schnellen Ladezeiten, interaktiven Funktionen und dynamischen Inhalten zu ermöglichen soll zudem eine Single-Page-Applikation (SPA) statt einer klassischen Webanwendung programmiert werden. Da die Logik sich bei SPAs zu großen Teilen auf die Clientseite verlagert, gibt es verschiedene Web-Frameworks, welche die Grundstruktur der Software sowie einige weitere Funktionen bereits zur Verfügung stellen. Ebenfalls unterstützen sie die Einhaltung des Designprinzips „Separation of Concerns“, indem sie sich um die Einheit zur Visualisierung von Informationen kümmern und die benötigten Daten durch Abfragen an die hierfür zuständige Einheit, das Backend, abrufen. Von den drei beliebtesten, allesamt komponentenbasierten Open-Source-Frameworks Angular, React und Vue.js wurde für LearnEasy Letzteres ausgewählt. Dies ist nicht nur durch die Vorkenntnisse der Entwickler begründet, sondern auch dadurch, dass der Einstieg in Vue, wie Vue.js auch oft genannt wird, als einfacher gilt und das Framework leichtgewichtig und dadurch sehr performant ist. [21], [24]

Gemeinsam mit Vue kommt die den Entwicklern bereits bekannte Programmiersprache TypeScript zum Einsatz. Diese erweitert JavaScript um statische Typisierung und moderne Features, wodurch Fehler im Code früher erkannt werden und die Wartbarkeit insgesamt erhöht wird. TypeScript wird zu JavaScript transpiliert, weshalb die Sprache problemlos mit jedem JavaScript-Framework verwendet werden kann. [25] Die letzte, bereits an dieser Stelle zu erwähnende Technologie

ist Bootstrap. Dieses wird genutzt, um unsere Software auf einfache Weise optisch ansprechend zu gestalten. [28]

2) *Struktur:* Wie viele andere moderne Frameworks arbeitet Vue nach dem Architekturmuster Model View ViewModel (MVVM). Hierbei besteht das Model aus verschiedenen Klassen, welche die Struktur der in der Anwendung genutzten Daten festlegen. Die View bilden die Kernelemente einer Vue-Anwendung: Wiederverwendbare Komponenten. Diese können einzeln genutzt oder ineinander verschachtelt werden. In Vue werden Komponenten in der Regel als Single-File Components (SFC) verfasst, welche sich jeweils aus den folgenden drei Teilen zusammensetzen. [26], [27]

- **Template:** Durch das Template wird das Layout einer Komponente definiert. Dies geschieht in einer deklarativen, HTML-ähnlichen Syntax, in welcher sowohl HTML-Elemente als auch Vue-Komponenten genutzt werden können. [27]
- **Script:** Die Logik einer Komponente ist im Script Abschnitt zu finden, welcher in JavaScript oder, im Falle von LearnEasy, in TypeScript verfasst ist. Hier werden beispielsweise Variablen und Methoden definiert, auf die innerhalb des Templates durch Binding zugegriffen wird. Vue stellt dabei sicher, dass Zustandsänderungen im JavaScript beziehungsweise TypeScript Code erkannt werden und zu einer Aktualisierung im Document Object Model (DOM) führen. [27], [29]
- **Style:** Der Style Abschnitt ermöglicht es, das Aussehen der Elemente einer Komponente gekapselt von den restlichen Komponenten zu verändern. Er wird mit CSS geschrieben. [27]

In jeder Vue-Applikation gibt es eine Hauptkomponente, welche den Einstiegspunkt in die Anwendung darstellt und alle weiteren Komponenten beinhaltet. In unserer Anwendung ist dies die Komponente App, deren Aufbau Abbildung 25 veranschaulicht. Bevor dieser näher erläutert wird, soll auf das ViewModel eingegangen werden. In unserer Software gehört zu diesem der Store, welcher mit der Bibliothek Pinia umgesetzt ist. Er speichert den Hauptzustand der Anwendung, der in mehreren Komponenten benötigt wird, in einem globalen Objekt, das die „Single Source of Truth“ darstellt. In LearnEasy beinhaltet der Store beispielsweise unter anderem den Nutzernamen des aktuell angemeldeten Nutzers oder das momentan ausgewählte Fach. Der Store erhält und sendet seine Daten über eine Klasse, welche mit dem Backend kommuniziert und bei unserer Anwendung ServerAccessor genannt wird. Der ServerAccessor sendet HTTP-Anfragen mithilfe der Bibliothek Axios. Die Struktur der Antworten wird dabei stets mit der Bibliothek Zod für TypeScript validiert. Darüber hinaus ist auch der Router Teil des ViewModels, welcher mit der Bibliothek Vue Router hinzugefügt wird. An dieser Stelle soll genauer auf die Abbildung 25 eingegangen werden. Diese zeigt, dass die Hauptkomponente unserer Software oben und unten ein Leiste hat. Diese Leisten sind stets zu sehen, auch wenn sich ihr Inhalt im Laufe des Navigationsflusses leicht verändert wie die Wireframes zeigen (siehe III-C1). Der

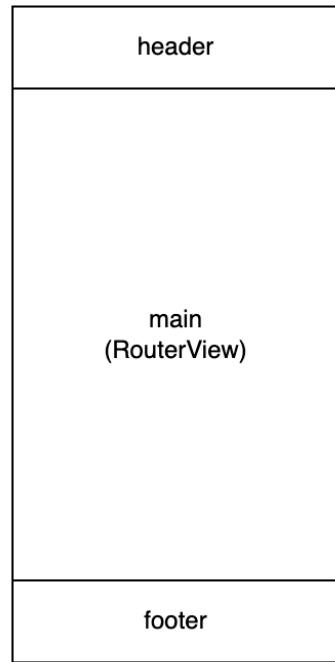


Abbildung 25. Aufbau der Hauptkomponente App

Bereich zwischen den beiden Leisten wird dagegen dynamisch vollständig ausgetauscht und kann im Laufe des Navigationsflusses von statischem Text, über Textfelder bis hin zu Listen oder sogar einer gesamten Übung verschiedenste Vue-Komponenten und HTML-Elemente beinhalten. Um dies zu ermöglichen beinhaltet er eine Komponente namens RouterView, welche von der Bibliothek Vue Router bereitgestellt wird. Je nach aktueller URI in der Adressleiste des Browsers wird in der RouterView eine andere Komponente gerendert. Welche Komponente zu welcher URI gehört, wird in einer Konfigurationsdatei des Routers, die bei LearnEasy router.ts heißt, angegeben. Die Navigation zwischen den verschiedenen Routen erfolgt in unserer Software meist mit dem Klick auf einen Button, was ebenfalls in den Wireframes visualisiert ist. Es soll zuletzt noch auf den Unterschied zwischen den beiden Packages container und components in dem Frontend von LearnEasy eingegangen werden. Dieser besteht darin, dass im Package container nur Komponenten enthalten sind, welche aus semantischen Gründen einmalig verwendbar sind und in der Datei router.ts mit den URIs verknüpft sind. Sie stellen die verschiedenen Seiten im Wireframe dar. Das Package components enthält hingegen Komponenten, welche mehrfach verwendet werden können, beispielsweise TextField oder GeneralButton. [30]–[33], [37]

3) *Barrierefreiheit:* Nun soll erläutert werden, wie die Barrierefreiheit nach WCAG 2.1 Level A im Frontend sicher gestellt wurde. Hierzu wurden die in der nicht-funktionalen Anforderung Barrierefreiheit (siehe II-C2) aufgelisteten Techniken für alle Erfolgskriterien umgesetzt, wobei die Techniken und Erfolgskriterien ausgenommen sind, welche in II-C2a

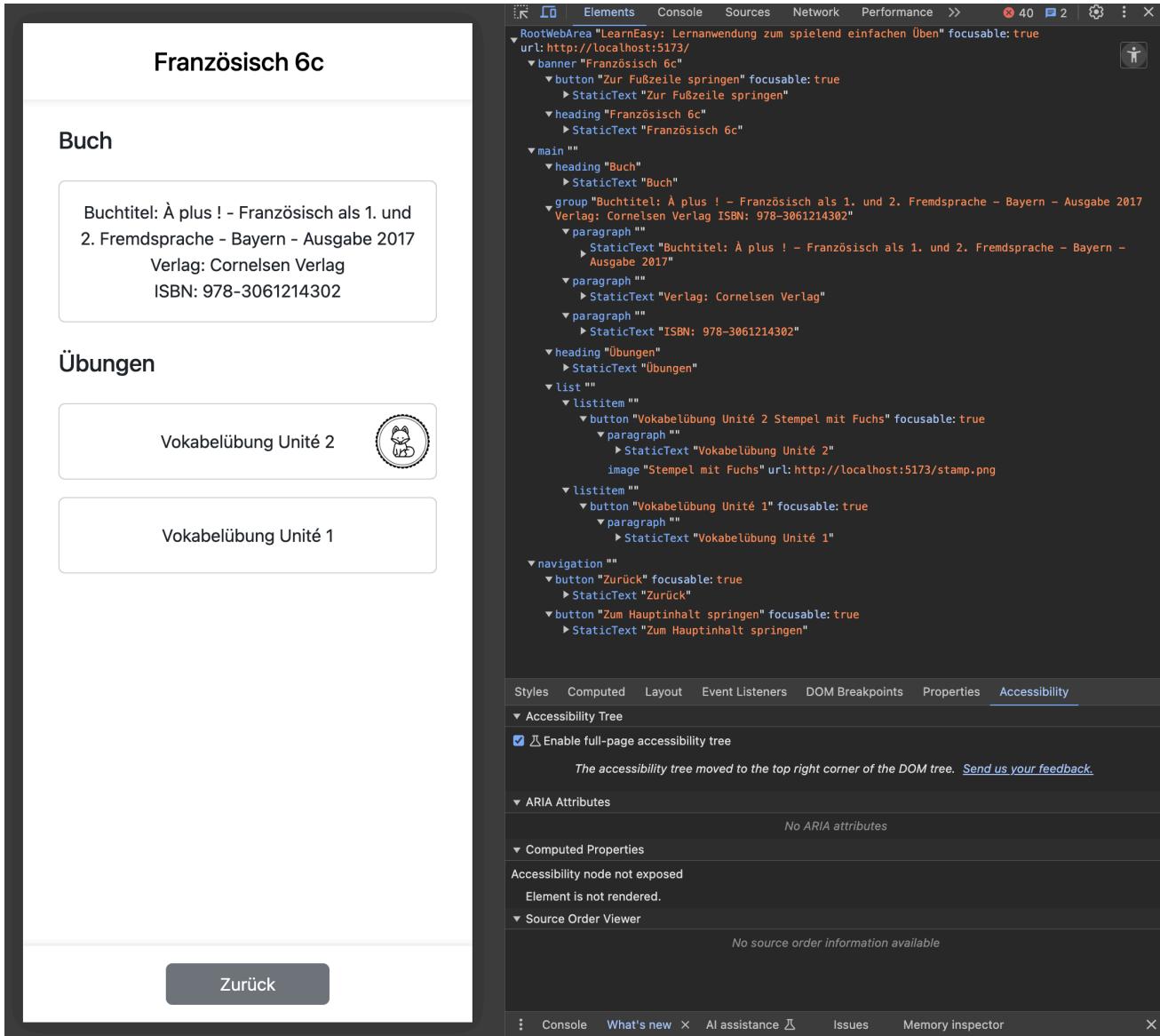


Abbildung 26. Bildschirmfoto der Entwicklertools von Chrome

genannt werden. Anschließend wurde zum einen manuell überprüft, ob unsere Software barrierefrei ist, indem unter anderem das DOM betrachtet und auf Angaben wie ARIA-labels oder alt-Attributen geachtet wurde. Dies war mit den Entwicklertools des Browsers Chrome möglich, welche auch einen sogenannten Accessibility Tree anzeigen und in Abbildung 26 zu sehen sind. Ebenfalls wurde der in das Betriebssystem Mac OS von Apple integrierte Screenreader VoiceOver im Zusammenhang mit LearnEasy ausprobiert. Zum anderen wurde die Barrierefreiheit mithilfe des Chrome Plugins Equal-Web Accessibility Checker analysiert, welches Webseiten auf Konformität mit den WCAG 2.1 prüft und dabei detaillierte Auskünfte gibt. Beispielsweise wird genau angezeigt, welches Element einer Webseite nicht mit welchem Erfolgskriterium auf welcher Konformitätsstufe übereinstimmt und wie dies behoben werden kann. Abbildung 27 zeigt ein Bildschirmfoto

der Analyse des Plugins. Dabei muss angemerkt werden, dass entsprechend der Spezifikation (siehe II) Erfolgskriterien mit Level AA und darüber nicht beachtet wurden. Ebenso werden trotz der auf manchen Screens sichtbaren Warnung des Plugins zu dem Erfolgskriterium „1.3.1–Info and Relationships“ keine „optgroup“ zum Gruppieren von „option“-Elementen in einem „select“-Element genutzt, da jedes „select“-Element eine einzige „optgroup“ darstellt. Darüber hinaus zeigt das Plugin auf manchen Screens eine Warnung zu dem Erfolgskriterium „4.1.2–Name, Role, Value“ an, welche darauf hindeutet, dass Elemente nicht über die Bedienungshilfe erreichbar sind. Diese Warnung wird jedoch fälschlicherweise angezeigt, wie der Accessibility Tree von Chrome beweist. [34], [35], [37]

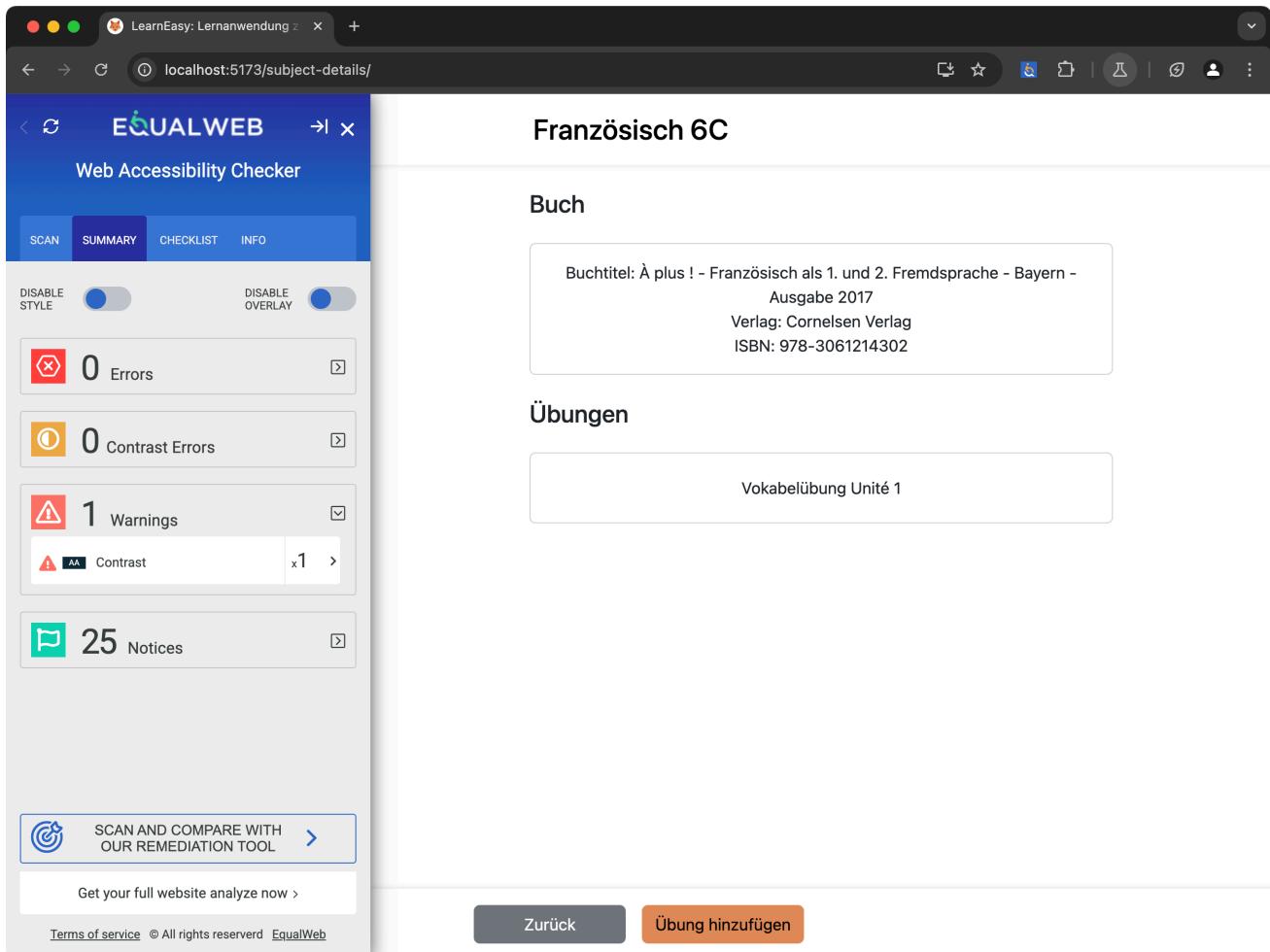


Abbildung 27. Bildschirmfoto des Plugins EqualWeb Accessibility Checker

## V. TEST UND VERIFIKATION

In diesem Kapitel soll es nun darum gehen, wie unser erster Prototyp durch die Entwickler getestet wird. Hierfür wurden folgende Testfälle definiert, welche alle grundlegenden Funktionen des ersten Prototyps abdecken.

- 1) Ein Lehrer registriert sich.
- 2) Ein Lehrer erstellt ein Fach.
- 3) Ein Lehrer erstellt eine Vokabelübung für ein Fach.
- 4) Ein Schüler registriert sich und tritt so einem Fach bei.
- 5) Ein Schüler tritt einem Fach bei.
- 6) Ein Schüler macht erfolgreich eine Vokabelübung.
- 7) Ein Schüler oder Lehrer meldet sich an.
- 8) Ein Schüler oder Lehrer meldet sich ab.

Diese Testfälle sollen durch verschiedene Arten von Tests überprüft werden, um unsere Anwendung möglichst umfassend zu testen und sicherzustellen, dass sie tatsächlich stabil funktioniert. Es soll jedoch darauf hingewiesen werden, dass im Rahmen dieser Studienarbeit nicht alle Testfälle mit allen Testarten umgesetzt wurden. [5]

### A. Manuelle Tests

Im Gegensatz zu automatisierten Tests erfordern manuelle Tests menschliche Interaktion statt Automatisierungstools. Die Person, die den Test durchführt, klickt sich durch die Anwendung und prüft, ob alles korrekt funktioniert. Dabei können Fehler erkannt werden, die von automatisierten Tests womöglich übersehen werden. Dennoch ist nicht auszuschließen, dass bei der Testdurchführung Benutzerfehler wie beispielsweise fehlerhafte Eingaben passieren. Zudem sind manuelle Tests zeitaufwendig und wenig effizient, insbesondere wenn sie mehrfach wiederholt werden sollen. Daher sollten stets auch automatisierte Tests durchgeführt werden. [5]

#### 1) Ein Lehrer registriert sich.: Grundvoraussetzung:

- Der Lehrer ist in der Webanwendung nicht angemeldet.
- Der Registriercode der Beispielschule ist bekannt.
- Startseite des Tests ist die Startseite unserer Webanwendung.

Durchführung:

- 1) Lehrer: Auf den Button „Registrierung“ klicken.
- 2) Lehrer: In das Textfeld „Registriercode (Erforderlich)“ den Registriercode der Beispielschule eingeben.

- 3) Lehrer: Auf den Button „Registrieren“ klicken.
- 4) Lehrer: Sich optional den angezeigten Nutzernamen notieren oder merken. Auf den Button „Weiter“ klicken.
- 5) Lehrer: Prüfen, ob nun die Fächerübersicht mit einer leeren Fächerliste zu sehen ist.

*2) Ein Lehrer erstellt ein Fach.: Grundvoraussetzung:*

- Der Lehrer ist in der Webanwendung mit seinem Nutzernamen angemeldet.
- Startseite des Tests ist die Fächerübersicht in unserer Webanwendung.

Durchführung:

- 1) Lehrer: Auf den Button „Hinzufügen“ klicken.
- 2) Lehrer: In das Textfeld „Fachname (Erforderlich)“ den Fachnamen „Französisch 6A“ eingeben.
- 3) Lehrer: In dem Dropdownfeld „Buch (Erforderlich)“ das Buch „À plus ! - Französisch als 1. und 2. Fremdsprache - Bayern - Ausgabe 2017“ auswählen.
- 4) Lehrer: Auf den Button „Erstellen“ klicken.
- 5) Lehrer: Sich optional den angezeigten Registriercode für das Fach notieren oder merken. Auf den Button „Weiter“ klicken.
- 6) Lehrer: Prüfen, ob nun die Fächerübersicht zu sehen ist und sich das erstellte Fach in der Fächerliste befindet.

*3) Ein Lehrer erstellt eine Vokabelübung für ein Fach.: Grundvoraussetzung:*

- Der Lehrer ist in der Webanwendung mit seinem Nutzernamen angemeldet.
- Der Lehrer hat mindestens ein Fach in seiner Fächerübersicht.
- Startseite des Tests ist die Fächerübersicht in unserer Webanwendung.

Durchführung:

- 1) Lehrer: Auf ein Fach in der Fächerliste klicken.
- 2) Lehrer: Auf den Button „Übung hinzufügen“ klicken.
- 3) Lehrer: In dem Dropdownfeld „Kapitel (Erforderlich)“ das Kapitel „Unité 1 La rentrée“ auswählen.
- 4) Lehrer: In dem Dropdownfeld „Übungsvorlage (Erforderlich)“ die Übungsvorlage „Vokabelübung Unité 1“ auswählen.
- 5) Lehrer: In dem Dropdownfeld „Anzeigeart (Optional)“ eine beliebige Anzeigeart auswählen.
- 6) Lehrer: Auf den Button „Erstellen“ klicken.
- 7) Lehrer: Prüfen, ob nun die Fachdetailsseite zu sehen ist und sich die erstellte Übung in der Übungsliste befindet.

*4) Ein Schüler registriert sich und tritt so einem Fach bei.: Grundvoraussetzung:*

- Der Schüler ist in der Webanwendung nicht angemeldet.
- Der Registriercode eines Faches ist bekannt.
- Startseite des Tests ist die Startseite unserer Webanwendung.

Durchführung:

- 1) Schüler: Auf den Button „Registrierung“ klicken.
- 2) Schüler: In das Textfeld „Registriercode (Erforderlich)“ den Registriercode eines Faches eingeben.
- 3) Schüler: Auf den Button „Registrieren“ klicken.
- 4) Schüler: Sich optional den angezeigten Nutzernamen notieren oder merken. Auf den Button „Weiter“ klicken.
- 5) Schüler: Prüfen, ob nun die Fächerübersicht zu sehen ist und sich das Fach, dessen Registriercode eingegeben wurde, in der Fächerliste befindet.

*5) Ein Schüler tritt einem Fach bei.: Grundvoraussetzung:*

- Der Schüler ist in der Webanwendung mit seinem Nutzernamen angemeldet.
- Der Registriercode eines Faches, dem der Schüler noch nicht beigetreten ist, ist bekannt.
- Startseite des Tests ist die Fächerübersicht in unserer Webanwendung.

Durchführung:

- 1) Schüler: Auf den Button „Hinzufügen“ klicken.
- 2) Schüler: In das Textfeld „Registriercode (Erforderlich)“ den Registriercode eines Faches, dem der Schüler noch nicht beigetreten ist, eingeben.
- 3) Schüler: Auf den Button „Beitreten“ klicken.
- 4) Schüler: Prüfen, ob nun die Fächerübersicht zu sehen ist und sich das Fach, dessen Registriercode eingegeben wurde, in der Fächerliste befindet.

*6) Ein Schüler macht erfolgreich eine Vokabelübung.: Grundvoraussetzung:*

- Der Schüler ist in der Webanwendung mit seinem Nutzernamen angemeldet.
- Der Schüler ist mindestens einem Fach beigetreten, welches mindestens eine Übung beinhaltet.
- Startseite des Tests ist die Fächerübersicht in unserer Webanwendung.

Durchführung:

- 1) Schüler: Auf ein Fach klicken, welches mindestens eine Übung beinhaltet.
- 2) Schüler: Auf eine Vokabelübung in der Übungsliste klicken.
- 3) Schüler: Die entsprechende Übung durchführen.
- 4) Schüler: Auf den Button „OK“ in dem Dialogfenster klicken.
- 5) Schüler: Prüfen, ob nun die Fachdetailsseite zu sehen ist und sich bei der erfolgreich abgeschlossenen Übung in der Übungsliste ein Stempel befindet.

*7) Ein Schüler oder Lehrer meldet sich an.: Grundvoraussetzung:*

- Der Schüler oder Lehrer ist in der Webanwendung nicht angemeldet.
- Der Nutzername des Schülers oder Lehrers ist bekannt.
- Startseite des Tests ist die Startseite unserer Webanwendung.

#### Durchführung:

- 1) Schüler oder Lehrer: Auf den Button „Anmeldung“ klicken.
- 2) Schüler oder Lehrer: In das Textfeld „Username (Erforderlich)“ den eigenen Nutzernamen eingeben.
- 3) Schüler oder Lehrer: Auf den Button „Anmelden“ klicken.
- 4) Schüler oder Lehrer: Prüfen, ob nun die Fächerübersicht mit den eigenen Fächern, falls vorhanden, zu sehen ist.

#### 8) Ein Schüler oder Lehrer meldet sich ab.: Grundvoraussetzung:

- Der Schüler oder Lehrer ist in der Webanwendung mit seinem Nutzernamen angemeldet.
- Startseite des Tests ist die Fächerübersicht in unserer Webanwendung.

#### Durchführung:

- 1) Schüler oder Lehrer: Auf den Button „Abmelden“ klicken.
- 2) Schüler oder Lehrer: Prüfen, ob nun die Startseite unserer Webanwendung zu sehen ist.

### B. Integrations Tests

Mit den Integrations Tests werden die einzelnen Endpunkte des Backends für sich und im Zusammenspiel getestet. Würde das Backend ausschließlich über die Benutzeroberfläche getestet werden und nicht direkt, könnte nicht sicher gesagt werden, ob ein Fehler im Backend oder im Frontend seine Ursache findet. Um den Test so realistisch wie möglich zu gestalten, wird die Anwendung auf genau gleichem Wege gestartet, wie sie auch in der Produktivumgebung gestartet werden würde, mit dem Unterschied, dass die von der Anwendung gespeicherten Daten an einem anderen Ort gespeichert werden. Während die Anwendung läuft, werden dann mit einem anderen Test-Programm die Tests ausgeführt. Dieses Test-Programm ist in TypeScript mithilfe des Test-Frameworks Mocha geschrieben und wird in node.js ausgeführt. Es führt in jedem seiner Tests eine Folge von HTTP-Requests aus, wobei an entsprechenden Stellen geprüft wird, ob die Response der Erwartung entspricht. Im Rahmen der Studienarbeit wurden die Testfälle 1, 2, 3, 4 und 6 umgesetzt. [5], [46]

### C. UI-Tests

UI-Tests machen die Benutzeroberfläche selbst zum Ziel des Testens. Dabei werden verschiedene, insbesondere häufige, Abläufe in der Benutzeroberfläche getestet. Damit sichergestellt werden kann, dass auftretende Fehler in der Benutzeroberfläche zu verorten sind, muss ein schon vollständig getestetes Backend verwendet werden. Da die UI-Tests genau wie die anderen Tests auf jeder Platform ausführbar und automatisiert sein sollen, wird das Selenium-Framework genutzt. Dieses ermöglicht es, verschiedene Browser wie z. B. Firefox oder Chrome auszuführen, ohne dass es dazu einen Window-Manager benötigt. Zusätzlich kann Selenium durch in verschiedenen bekannten Programmiersprachen geschriebenen Code detailliert gesteuert werden, wodurch es

möglich ist, Abläufe zu definieren, welche dann automatisiert die Benutzeroberfläche testen. Im Rahmen der Studienarbeit wurde auf UI-Tests verzichtet, da die Benutzeroberfläche des ersten Prototyps mit den anderen beiden Testarten ausreichend geprüft wurde. [5], [47]

### VI. VALIDIERUNG

Damit die Anwendung nicht nur formell getestet wurde, sondern auch die Kunden zufrieden sind, wird in diesem Kapitel ein Abnahmeprotokoll für den ersten Prototyp definiert. Dieses soll vom Kunden selbst geprüft werden.

- 1) Bedienbarkeit: Die Bedienung der Anwendung ist intuitiv sowohl für Lehrer und Schüler ohne lange Einarbeitungszeit möglich.
- 2) Barrierefreiheit: Die Anwendung ist vollständig mit der Tastatur bedienbar und es wurde ein Screenreader ausprobiert.
- 3) Fehlermeldungen und Feedback: Es gibt klare Hinweise bei falschen Nutzereingaben und anderen Fehlern.
- 4) Inhaltliche Struktur: Der Navigationsfluss in der Anwendung ist verständlich und sinnvoll gegliedert.
- 5) Funktionalität: Die Anwendung startet fehlerfrei und alle versprochenen Funktionen sind nutzbar.
- 6) Interaktivität: Die Übungen in der Anwendung funktionieren wie erwartet und fehlerfrei.
- 7) Plattformunabhängigkeit: Die Anwendung läuft stabil auf Smartphones, Tablets und PCs mit verschiedenen Betriebssystemen.
- 8) Performance: Die Ladezeiten der Anwendung sind angemessen und es treten keine nennenswerten Verzögerungen auf.
- 9) Datenschutz: Es werden keine benutzerbezogenen Daten erhoben, die Rückschlüsse auf die Identität des Nutzers ermöglichen.
- 10) Weiterentwicklung: Verbesserungsvorschläge und andere Bemerkungen wurden schriftlich festgehalten.

### VII. AUSBLICK / ZUSAMMENFASSUNG

Der entwickelte erste Prototyp von LearnEasy legt den Grundstein für eine innovative Lernanwendung für den schulischen Sprachunterricht. Diese soll in Zukunft iterativ in mehreren Prototypen weiterentwickelt werden. Kapitel II-E definiert hierzu bereits den Funktionsumfang eines zweiten Prototyps. Zudem werden bereits zwei funktionale Anforderungen für spätere Prototypen genannt, die um weitere Ideen ergänzt werden können. Daraüber hinaus bietet sich, wie in der Einführung dieser Arbeit bereits erläutert wurde, die Möglichkeit an, das Konzept von LearnEasy auf weitere Schulfächer auszuweiten. Als nächster wichtiger Schritt nach der Fertigstellung des zweiten Prototyps sollte jedoch die Erprobung der Software in realen Klassenzimmern sein. Das Einholen von Feedback von Lehrkräften und Schülern verschiedener Jahrgangsstufen und Schulen ist wertvoll, um Optimierungen vorzunehmen und die Anwendung weiter auf die Bedürfnisse der Zielgruppe zuzuschneiden. Langfristig soll so eine umfassend und einfach verwendbare, barrierefreie Lernanwendung für Smartphones,

Tables und Desktops entstehen, die den Schulunterricht bereichert und einen nachhaltigen Beitrag zur Verbesserung des Bildungswesens leistet.

## LITERATUR

- [1] "Agile User Stories". Miro. <https://miro.com/de/agile/was-ist-eine-user-story/> (abgerufen 04.01.2025).
- [2] "Web Content Accessibility Guidelines 2.1 (WCAG 2.1)". Bundesministerium des Innern und für Heimat. <https://www.barrierefreiheit-dienstekonsolidierung.bund.de/Webs/PB/DE/gesetze-und-richtlinien/wcag/wcag-artikel.html> (abgerufen 07.01.2025).
- [3] "How to Meet WCAG (Quick Reference)". W3C. <https://www.w3.org/WAI/WCAG22/quickref/?versions=2.1> (abgerufen 07.01.2025).
- [4] Vorlesung "Moderne Softwarearchitektur" von René Peinl im Master Informatik
- [5] Vorlesung "Praktikum" von René Peinl im Master Informatik
- [6] Private Kommunikation mit Daniel Drescher und Marc Ruddigkeit
- [7] "Lexikon". Bundesministerium des Innern und für Heimat. <https://www.barrierefreiheit-dienstekonsolidierung.bund.de/Webs/PB/DE/service/lexikon/functions/bmilexikon.html> (abgerufen 07.01.2025).
- [8] "Glossar". Unfallkasse Sachsen-Anhalt Landesfachstelle für Barrierefreiheit. <https://www.lf-barrierefreiheit-st.de/glossar> (abgerufen 11.01.2025).
- [9] "Techniques for WCAG 2.1". W3C. <https://www.w3.org/WAI/WCAG21/Techniques/> (abgerufen 11.01.2025).
- [10] "Glossar". Jan Hellbusch. <https://www.barrierefreies-webdesign.de/glossar/> (abgerufen 12.01.2025).
- [11] K. Kuenen. "Minimum Viable Product (MVP)". Springer Fachmedien Wiesbaden GmbH. <https://wirtschaftslexikon.gabler.de/definition/minimum-viable-product-mvp-119157> (abgerufen 14.01.2025).
- [12] "KIM-Studie. Kindheit, Internet, Medien". mpfs. <https://mpfs.de/studien/kim/> (abgerufen 24.01.2025).
- [13] F. Tenzer. "Smartphone-Besitz von Kindern in Deutschland im Jahr 2022 nach Altersgruppen". statista. <https://de.statista.com/statistik/daten/studie/1104/umfrage/smartphonebesitz-von-kindern-nach-altersgruppen/> (abgerufen 24.01.2025).
- [14] "JIM-Studie. Jugend, Information, Medien". mpfs. <https://mpfs.de/studien/jim-studie/> (abgerufen 24.01.2025).
- [15] F. Tenzer. "Anteil der Jugendlichen in Deutschland, die ein Handy/Smartphone besitzen, nach Altersgruppe im Jahr 2022". statista. <https://de.statista.com/statistik/daten/studie/589577/umfrage/smartphonebesitz-von-jugendlichen-in-deutschlandnach-altersgruppe/> (abgerufen 24.01.2025).
- [16] "Welcome to UML Web Site". Object Management Group, Inc. <https://www.uml.org/index.htm> (abgerufen 02.02.2025).
- [17] Vorlesung "Interface- und Interaction-Design" von Ina Günther im Bachelor Medieninformatik
- [18] "Ist der Fuchs wirklich so schlau?". Konradin Medien GmbH. <https://www.wissenschaft.de/erde-umwelt/ist-der-fuchs-wirklich-so-schlau/> (abgerufen 02.02.2025).
- [19] "Access 17.9M+ vector icons & stickers". Freepik Company. <https://www.flaticon.com/> (abgerufen 02.02.2025).
- [20] "Bootstrap Icons". Bootstrap. <https://icons.getbootstrap.com/> (abgerufen 02.02.2025).
- [21] P. Ackermann, "Webentwicklung. Das Handbuch für Fullstack-Entwickler". Bonn: Rheinwerk Verlag, 2021.
- [22] "OpenAPI Specification". SmartBear Software. <https://swagger.io/specification/> (abgerufen 02.02.2025).
- [23] Vorlesung "Hybride Apps" von Jürgen Heym im Master Informatik
- [24] F. Deitelhoff, "Vue.js. Von Grundlagen bis Best Practice". Heidelberg: dpunkt, 2022.
- [25] "TypeScript is JavaScript with syntax for types.". Microsoft. <https://www.typescriptlang.org/> (abgerufen 03.02.2025).
- [26] "Model View ViewModel (MVVM)". Microsoft. <https://learn.microsoft.com/de-de/dotnet/architecture/maui/mvvm> (abgerufen 03.02.2025).
- [27] "Single-File Components". Evan You. <https://vuejs.org/guide/scaling-up/sfc.html> (abgerufen 03.02.2025).
- [28] "Build fast, responsive sites with Bootstrap". Bootstrap. <https://getbootstrap.com/> (abgerufen 03.02.2025).
- [29] "Document Object Model (DOM)". mozilla.org. [https://developer.mozilla.org/en-US/docs/Web/API/Document\\_Object\\_Model](https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model) (abgerufen 03.02.2025).
- [30] "Pinia. The intuitive store for Vue.js". Eduardo San Martin Morote. <https://pinia.vuejs.org/> (abgerufen 03.02.2025).
- [31] "Vue Router. The official Router for Vue.js". Evan You, Eduardo San Martin Morote. <https://router.vuejs.org/> (abgerufen 03.02.2025).
- [32] "Getting Started". Axios. <https://axios-http.com/docs/intro> (abgerufen 03.02.2025).
- [33] "Zod". Colin McDonnell. <https://zod.dev/?id=basic-usage> (abgerufen 03.02.2025).
- [34] "Kapitel 1. Einführung in VoiceOver". Apple. [https://www.apple.com/de/voiceover/info/guide/\\_1121.html](https://www.apple.com/de/voiceover/info/guide/_1121.html) (abgerufen 03.02.2025).
- [35] "EqualWeb Accessibility Checker". Google. <https://chromewebstore.google.com/detail/equalweb-accessibility-ch/imemciokfejbnonkkinhedfigdilellg> (abgerufen 03.02.2025).
- [36] "Homeschooling: Digitale Ausstattung in Familien hängt stark vom Einkommen ab". Statistisches Bundesamt (Destatis). [https://www.destatis.de/DE/Presse/Pressemitteilungen/2020/07/PD20\\_N042\\_639.html](https://www.destatis.de/DE/Presse/Pressemitteilungen/2020/07/PD20_N042_639.html) (abgerufen 06.02.2025).
- [37] Autoren
- [38] Vorlesung "Rechnernetze" von René Peinl im Bachelor Medieninformatik
- [39] C. Mann-Grabowski, C. Jorissen, G. Gregor, O. Blume, "À plus ! - Französisch als 1. und 2. Fremdsprache - Bayern - Ausgabe 2017". Berlin: Cornelsen Verlag, 2018.
- [40] Vorlesung "RESTful Webservices" von Andrej Bachmann im Bachelor Medieninformatik
- [41] Vorlesung "Objektorientierte Programmierung I" von Barbara Ashauer im Bachelor Medieninformatik
- [42] "All Results". sonatype. <https://central.sonatype.com/search> (abgerufen 06.02.2025)
- [43] "Web Framework Benchmarks". TechEmpower. <https://www.techempower.com/benchmarks/#hwph&testfortune&section=data-r22> (abgerufen 06.02.2025)
- [44] "JPA". Broadcom. <https://docs.spring.io/spring-data/jpa/reference/jpa.html> (abgerufen 06.02.2025)
- [45] "Spring Framework Documentation". Broadcom. <https://docs.spring.io/spring-framework/reference/index.html> (abgerufen 06.02.2025)
- [46] "Mocha. simple, flexible, fun". OpenJS Foundation. <https://mochajs.org/> (abgerufen 06.02.2025)
- [47] "Selenium automates browsers. That's it! What you do with that power is entirely up to you.". Software Freedom Conservancy. <https://www.selenium.dev/> (abgerufen 06.02.2025)
- [48] "Open Container Initiative". The Linux Foundation. <https://opencontainers.org/> (abgerufen 06.02.2025)