

communicated by:
Lehr- und Forschungsgebiet Informatik 9

Prof Dr. Ulrik Schroeder



RWTHAACHEN
UNIVERSITY

Diese Arbeit wurde vorgelegt am Lehr- und Forschungsgebiet Informatik 9

The present work was submitted to Learning Technologies Research Group

Entwicklung einer Cross-Platform Feedback-App zur Verbesserung von Unterrichtsstunden
von Lehramtsstudierenden und Referendaren

Development of a Cross-Platform Feedback App to Improve the Lessons of Teacher Students
and Trainees

Bachelorarbeit

Bachelor-Thesis

von / presented by

Möhring, Tobias

345522

Prof. Dr. Ulrik Schroeder

Prof. Dr. Jan Borchers

Aachen, 15. September 2019

Inhaltsverzeichnis

I	Einleitung und Einführung	2
1.	Motivation und Zielsetzung	3
2.	Gliederung	4
II	Konzept	5
3.	Verwandte Arbeiten	6
3.1.	Nicht-technische Lösungen	6
3.2.	Technische Lösungen	6
3.2.1.	Survey-Tools	7
4.	Entwicklungsprozess	9
4.1.	Nutzer	9
4.1.1.	Wer sind die Nutzer?	9
4.1.2.	Was wollen sie mit dem System tun?	10
4.1.3.	Was ist der Kontext?	10
4.2.	Rahmenbedingungen	11
4.3.	Entwicklung	11
5.	Nutzertest	12
5.1.	Ablauf	12
5.2.	Resultate	12
5.2.1.	Technische Störungen	12
5.2.2.	Designprobleme	13
5.2.3.	Nutzeroberfläche	14
5.2.4.	Verbesserungsvorschläge	16
6.	Applikation	17
6.1.	Startseite	17
6.2.	Umfrageseite	18
6.3.	Navigation im Nutzerbereich	18
6.4.	Nutzerverwaltung	19
6.5.	Umfrageverwaltung	21
6.6.	Nutzer-Home	23
6.7.	Visualisierung	25
6.7.1.	Kategorien	26

III Technische Grundlagen	28
7. Technologien	29
7.1. React	29
7.1.1. JSX	29
7.1.2. State in React	30
7.1.3. Redux	31
IV Implementierung	35
8. Software Ökosystem	36
8.1. Backend	36
8.1.1. Docker	37
8.1.2. Webserver	37
8.1.3. Datenbank	38
8.2. Frontend	39
9. Projektaufbau	41
9.1. Die Codebase	41
9.1.1. client	41
9.1.2. server	42
9.2. Aufsetzen der Applikation	42
V Fazit und Ausblick	43
10. Fazit	44
Anhänge	48
A. Literaturverzeichnis	48
B. Abbildungsverzeichnis	50
C. Prototypen	52
D. Screenshots	54

Zusammenfassung

Diese Arbeit setzt sich die Aufgabe eine Applikation zu implementieren, die dabei helfen soll in Unterrichtssituationen einfach und anonym Feedback von allen Teilnehmenden zu sammeln. Insbesondere wird sich mit der Frage befasst, welche Designentscheidungen bei diesem Prozess hinderlich bzw. förderlich sein können.

Diese Arbeit orientiert sich dabei an der bereits existierenden Feedback-App¹, die sich von anderen Umfrageanwendungen vor allem durch die Unterscheidung von verschiedenen Teilnehmergruppen abhebt. Diese Arbeit soll als Ablösung für die Feedback-App dienen, dementsprechend wurde diese Anfangs auf Probleme und Unzulänglichkeiten untersucht und Ideen für Verbesserungen gesammelt, etwa die Implementierung einer Entwurfshilfe für Umfragen oder das automatisierte Aufbereiten und Verfügbarmachen der Unterrichtsergebnisse.

Dann wurden die gesammelten Resultate in iterativen Entwicklungsschritten, von Storyboard-Prototypen bis hin zu einer lauffähigen Softwareversion, konkretisiert. Schließlich wurde eine Version der Arbeit in einer Unterrichtssituation auf ihre Tauglichkeit getestet. Probleme und Feedback aus dem Nutzertest, z.B. Verwirrung mit der Nutzeroberfläche zum Erstellen neuer Stunden und Performance Probleme, wurden dann anschließend behoben.

¹<http://schuelerlabor.informatik.rwth-aachen.de/fragebogenapp/>

Teil I

EINLEITUNG UND EINFÜHRUNG

Kapitel 1 Motivation und Zielsetzung

„Das Potenzial von Schüler-Rückmeldungen an die Lehrperson steht außer Frage.“ [16] Feedback ist ein zentraler Aspekt der Selbstverbesserung, auch in der Lehre.

Mit der zunehmend fortschreitenden Digitalisierung, die sich in unserer Gesellschaft abzeichnet, wird es dabei attraktiver konventionelle Methoden der Feedback-Beschaffung durch elektronische Hilfen zu ersetzen. Insbesondere das Smartphone, welches sich einer inzwischen beinahe universellen Verbreitung erfreut (77,9% der Haushalte mit im Schnitt 144,2 Smartphones auf 100 Haushalte [19]), bietet sich dabei als Medium zur Rückmeldungserfassung an. Weiterhin besteht ein gesteigerter Trieb zur Digitalisierung des Schul- und Unterrichtswesens, etwa durch bundesweite Investmentpakete wie den DigitalPakt Schule.¹

*Digitalisierung
in Deutschland*

Eine Software-basierte Feedback-Ermittlung ist natürlich nicht nur aus der durch die fortschreitende Digitalisierung bestehenden Gelegenheiten heraus interessant. Als Software-System sind die Daten bequem erreichbar und ihre Darstellung stets erweiterbar. Weiterhin lässt sich sogar argumentieren, dass ein solches System nachhaltiger ist, wenn es beispielsweise konventionelle Umfragen in Form von gedruckte Fragebögen ablöst.

*Effizienz und
Nachhaltigkeit*

Die Implementierung einer solchen Applikation ist Ziel dieser Arbeit. Dazu soll eine Multiplattform-Applikation implementiert werden, welche es Nutzern erlaubt in verschiedenen Rollen an der Bewertung von Unterrichtsstunden teilzunehmen. Ergebnisse der Stunden sollen dann jederzeit für Lehrkräfte einsehbar sein und auf geeignete Weise dargestellt werden.

Cross-Platform

Diese Arbeit zielt darauf ab Eigenschaften und Designentscheidungen herauszuarbeiten, die das Sammeln von Feedback durch eine Applikation erleichtern. Dabei wird hervorgehoben wie sich die in dieser Arbeit implementierte Applikation von anderen Methoden unterscheidet oder auf diesen aufbaut und sie verbessert. Eine qualitativer Vergleich mit anderen Methoden der Feedback-Beschaffung geht dabei leider über den Umfang dieser Arbeit hinaus.

*Verwandte
Arbeiten*

Konkret heißt das, im Rahmen dieser Arbeit wird eine Full-Stack Web-Applikation entwickelt. Es erfolgt ein anschließender Test einer lauffähigen Version der App in einer Unterrichtssituation. Dabei sollen Probleme in Design und Programmierung aufgedeckt und schließlich behoben werden.

*Ablauf der
Arbeit*

¹ <https://www.bmbf.de/de/mit-dem-digitalpakt-schulen-zukunftsaehig-machen-4272.html>

Kapitel 2 Gliederung

Im folgenden Abschnitt findet sich eine kurze Beschreibung der Gliederung und der Inhalte der unterschiedlichen Abschnitte dieser Arbeit.

Zuerst erfolgt eine nicht-technische Beschreibung der Applikation. Dazu wird ein Vergleich zu verwandten Arbeiten gezogen und erläutert wie unsere Arbeit sich von diesen absetzt. Dann werden die potenziellen Nutzer dieser Applikation charakterisiert und die Nutzungssituation untersucht. Weiterhin werden die Rahmenbedingungen dieser Arbeit erörtert und auf den Entwicklungsprozess selbst eingegangen.

Konzept

Schließlich wird der Nutzertest besprochen, der dazu diente die Leistungen dieser Arbeit in einer reellen Unterrichtssituation zu überprüfen. Die konkrete Anwendung der Ergebnisse dieses Nutzertests wird dann anhand eines Rundgangs durch die Applikation erläutert.

Darauf werden elementare technische Aspekte der Arbeit erörtert. Dies umfasst vor allem die Frameworks und Designprinzipien die einen besonders starken Einfluss auf die Entwicklung der Applikation hatten.

Technische Grundlagen

Als nächstes wird auf die technische Realisierung der Implementierung eingegangen. Das umfasst unter anderem die Applikations-Architektur und das Layout der Datenbank. Weiterhin werden Informationen vermittelt, die wichtig für die Nutzung der Anwendung und die Wartung und Erweiterung der Codebase sind.

Implementierung

Abschließend erfolgt ein kritischer Rückblick auf die Ergebnisse dieser Arbeit und die Entscheidungen die dabei getroffen wurden. Wir versuchen zu ermitteln was gut verlauten ist und wo es im Laufe dieser Arbeit zu Problemen gekommen ist. Zusätzlich werden Vorschläge erbracht, wie eine zukünftige Entwicklung der Applikation, die nicht mehr in den zeitlichen Rahmen dieser Arbeit gepasst hat, aussehen könnte.

Fazit und Ausblick

Teil II

KONZEPT

Kapitel 3 Verwandte Arbeiten

Zuerst wollen wir etablieren, warum diese Arbeit überhaupt notwendig ist. Feedback zu sammeln ist natürlich kein neuartiges Konzept. Es stehen unzählige technische und nicht-technische Lösungen für diese Problematik zur Verfügung. In diesem Abschnitt wird auf einige dieser Arbeiten eingegangen. Wir werden dabei versuchen zu argumentieren wie sich die in dieser Arbeit implementierte Methode von verwandten Arbeiten absetzt. Die Eignung der Methoden kann aber sehr abhängig sein von Präferenzen, Intention und Situation der Lehrkraft.

3.1. Nicht-technische Lösungen

Natürlich ist es nicht notwendig Software für das Sammeln von Schülerfeedback zu verwenden. Es gibt viele analoge Methoden, die auch Verwendung in der Ausbildung von Lehrkräften finden. Das Staatsinstitut für Schulqualität und Bildungsforschung München (ISB) unterteilt diese Methoden in zwei Kategorien: „Schriftliche und vollständig dokumentierte Methoden“, sowie „Mündliche und/oder non-verbale Methoden“. [17]

Zu ersterer Kategorie zählen u.a. altbekannte Fragebögen, aber auch Methoden die sich auf das Sammeln und Clustern von Freiform-Feedback konzentrieren (vergleiche Kartenabfrage in [17]). Ausschlaggebend ist hier, dass das Endresultat der Rückmeldungen in dokumentierter Form vorliegt und im Nachhinein ausgewertet und besprochen werden kann, wie die Bezeichnung andeutet. Vorteil dieser Methoden ist natürlich, dass kein zusätzlicher Dokumentationsaufwand anfällt und das sich diese Methoden auch für Lehrkräfte ohne Erfahrung im Sammeln von Feedback eignen. Wie wir sehen werden fallen technische Lösungen auch größtenteils in diese Kategorie, da sie sich insbesondere für den Zweck der systematischen Dokumentation und Auswertung von Ergebnissen eignen.

*Schriftliche
Methoden*

Mündliche und non-verbale Methoden hingegen werden bestenfalls teilweise dokumentiert und erfordern einige Voraussetzungen um Erfolgreich zu sein. Das Wesen der Schüler sowie die Existenz oder Abwesenheit eines vertrauensvollen Klimas in der Klasse können Einfluss auf die Ergebnisse nehmen und müssen entsprechend beachtet werden. Auch eine Auswertung mit der Klasse muss beinahe direkt folgen, da keine Dokumentation des Feedbacks vorliegt. Andererseits ist für diese Art von Methoden der Vor- und Nachbereitungsaufwand geringer als bei schriftlichen Methoden.[17]

*Mündliche/Non-
Verbale
Methoden*

3.2. Technische Lösungen

Erwartungsgemäß befasst sich diese Arbeit allerdings mit der Implementierung einer technischen Lösung für die Sammlung von Feedback. Wie bereits angedeutet fallen technische Lösungen generell unter die schriftlichen, voll dokumentierten Methoden, die exakt in die Stärken von digitalen Systemen spielen. Es lassen sich viele Argumente finden, warum digitale Umfragesysteme Potenzial für Unterrichtssituationen haben, insbesondere in der Ausbildung von Lehrkräften.

Wie analoge, schriftlich voll dokumentierte Methoden haben technische Systeme den Vorteil das Resultate in dokumentierter Form vorliegen und somit jederzeit gesichtet und ausgewertet werden können. Die Rohdaten, die als Grundlage für die Auswertung und Visualisierung der Ergebnisse dienen können einfach archiviert werden und stehen somit für beliebige Auswertungsschemas zur Verfügung, sind also auch nach der Erfassung und Auswertung noch von Interesse. Sollten neue Auswertungsmethoden wünschenswert sein können diese generell die alten Datensätze wiederverwenden, außer die Natur der erfassten Daten ändert sich.

*Volldokumen-
tiert,
auswertungs-
agnostisch*

Auch der Vorbereitungs- und in gewissem Maße der Nachbereitungsaufwand lässt sich durch technische Systeme reduzieren. Das Anfertigen und Verteilen von Fragebögen erfolgt digital und Fragebogenschemas können geteilt und wiederverwendet werden. Die Auswertung und Visualisierung kann ebenfalls automatisch übernommen werden.

*Weniger
Aufwand*

Natürlich ist die Verwendung dieser Auswertungsmethodik abhängig vom Anwendungskontext. Für erfahrenere Lehrkräfte, oder Lehrkräfte die andere Methoden bevorzugen ist der Mehrwert einer technischen Lösung nicht unbedingt gegeben. Für Nutzungssituationen wie sie beispielsweise im Nutzertest vorliegen (vergleiche Kapitel 5), in denen die Lehrkräfte mit unbekannten Schülerinnen und Schülern konfrontiert werden sollten solche Bedenken hingegen weniger problematisch sein.

*Eignung ist
situationsbe-
dingt*

3.2.1. Survey-Tools

Für uns relevante technische Lösungen fallen alle mehr oder weniger in die selbe Kategorie von Survey-Tools. Hierzu lassen sich unzählige kommerzielle und auch freie Lösungen finden und Umfragen sind so weit verbreitet das sie sich oft sogar beiläufig auf Webseiten eingebettet oder in dedizierten Terminals in Supermärkten finden.

Interessant sind vor allem die Survey-Tools die vollständige Umfragen erlauben, ähnlich analogen Fragebögen. Davon existieren diverse Varianten, manche sehr allgemein und vielzweckig, andere wiederum speziell mit der Absicht im Unterricht (oder anderen spezifischen Situationen) verwendet zu werden.

Allgemeine Survey-Tools, etwa SurveyMonkey[15] oder Google Forms[8] sind dabei bereits hoch entwickelte General Purpose Tools, die professionell Entwickelt werden. Dieser Bereich ist also eher uninteressant für uns. Andere *Special Purpose Tools* in diesem Bereich, etwa die Schülerfeedback-App Edkimo[3] fallen in eine ähnliche Nische. Es ist also notwendig sich auf andere Weise von diesen Werkzeugen abzuheben, die im Rahmen dieser Arbeit einhaltbar sind. Dazu lohnt es sich die Applikation zu betrachten, die durch diese Arbeit abgelöst werden soll.

Feedback-App

Die an der RWTH Aachen University entwickelte Feedback App [6] unterscheidet sich vor allem in einem zentralen Aspekt: die betrachteten Nutzergruppen.

Die bereits aufgeführten Applikationen erlauben eine Gruppe von Probanden für eine Umfrage, seien es Schülerinnen und Schüler, Kunden, Nutzer, oder welche Gruppe auch immer gerade im Fokus der gewünschten Umfrage steht. Wichtig ist das lediglich eine Gruppe untersucht wird. Die Feedback App sammelt hingegen Rückmeldungen von allen teilnehmenden Fraktionen, also von Schülerinnen und Schülern, Lehrkräften und Beobachtern. Alle Gruppen haben dabei unterschiedliche Rollen und Interessen, wie wir in Kapitel 4 diskutieren werden.

*Differenzie-
rungsfaktor
Nutzergruppen*

Warum also eine neue Applikation schreiben? Der momentane Stand der Feedback App ist ein zufriedenstellendes Umfragetool, leidet allerdings an Unzulänglichkeiten in der Nutzbarkeit der App und Visualisierung der Daten. Umfrageergebnisse werden zwar auf einem Server persistiert, sind aber für Nutzer nicht ohne weiteres einsehbar und müssen statt dessen erst aus der Datenbank extrahiert werden.

*Probleme der
Feedback App*

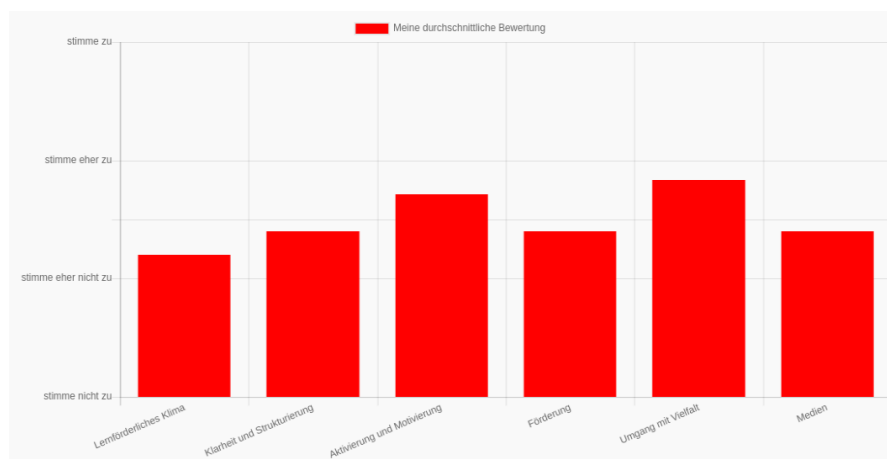


Abbildung 3.1.: Balkendiagramm durchschnittlicher Kategorie-Bewertungen.[6]

Ein weiteres Problem ist die Visualisierung der Ergebnisse (siehe Abbildung 3.1), die auf das abschließen einer Umfrage folgt. Ergebnisse werden als durchschnittliche Bewertung jeder Kategorie angezeigt. Das ist ein akzeptabler Indikator für den allgemeinen Verlauf der Stunde in den jeweiligen Kategorien, aber es werden auch eine Menge Details ausgelassen die potenziell von Bedeutung sein können. Beispielsweise die Komposition der Bewertungen geht verloren. Diese Graphik spiegelt auch nur die Bewertung eines einzelnen Nutzers wieder.



Abbildung 3.2.: Slider-Komponente, die in der Umfrage verwendet wird.[6]

Auch die Slider-Komponente (siehe Abbildung 3.2) die für den Umfrageteil Verwendung findet hat ein Problem, wenn dieses auch nicht wirklich drastisch ist. Das graphische Design des Sliders scheint eine Skala von links nach rechts zu implizieren. Dabei ist der rot markierte Wert „k.A.“, also „keine Angabe“, was aber zu einem möglichen Konflikt in der Interpretation des Elements führt. Schließlich wird durch „k.A.“ bewusst eine Wertung ausgelassen. Visuell sieht es hingegen so aus als wäre „k.A.“ eine stärkere Ablehnung als sein Nachbarwert „stimme nicht zu“, welches wie alle anderen Wertung in einer strikten Skala angeordnet ist.

Diese Arbeit macht also nun Gebrauch von der Gelegenheit auf der Feedback App aufbauend die Nutzererfahrung zu verbessern und das Featureset zu erweitern.

Konkret heißt das, dass Nutzerkonten für Lehrkräfte und Beobachtende eingeführt werden über die diese stets Zugriff auf absolvierte Stunden bekommen. Außerdem soll die Visualisierung überholt werden. Die Designentscheidungen für die Nutzeroberfläche und die Visualisierung, wie sie in der abschließenden Version der Software zu finden sind, werden in dem Applikations-Walkthrough in Kapitel 6 erläutert.

Kapitel 4 Entwicklungsprozess

Dieses Kapitel befasst sich mit dem Entwicklungsprozess, also der Analyse der Problem-
domäne und der Methodik mit der an dieses Problem herangegangen wurde. Dabei
bleibt dieser Abschnitt frei von den technischen Details der Lösung, welche im Imple-
mentationsteil dieser Arbeit besprochen werden, und befasst sich statt dessen mehr mit
den Rahmenbedingungen und dem Werdegang der Applikation.

4.1. Nutzer

Im Zentrum einer interaktiven Applikation wie dieser stehen selbstverständlich die Nut-
zer. Zuerst ist es notwendig drei Kernfragen über die Interaktion durch Nutzer zu be-
trachten.

4.1.1. Wer sind die Nutzer?

Hauptsächlich sind drei verschiedene Nutzergruppen im Fokus dieser Arbeit:

Schülerinnen und Schüler

Die in der Regel größte Nutzergruppe ist die der Schülerinnen und Schüler. Im Kontext
einer Umfrage stellt diese Gruppe die Zielgruppe dar, deren Rückmeldung zentral für
die Auswertung ist. Im Nutzertest wurden hierbei Schülerinnen und Schüler aus unter-
schiedlichen Jahrgängen der Sekundarstufe betrachtet, Teilnehmer waren also eher tech-
nikaffin und auch im Umgang mit unbekannter Software flexibel.

Dabei können aber auch andere demographische Gruppen unter diese Nutzerklasse fal-
len, beispielsweise im Rahmen eines universitären Kurses. Hier können auch deutlich
ältere Nutzer, etwa Seniorenstudierende, betroffen sein, die gegebenenfalls andere An-
sprüche an Software haben.

*Andere
Demographien*

Lehrkräfte

Wenn Schüler die Zielgruppe einer Umfrage sind, dann sind Lehrkräfte die Hauptinter-
essenten an den Ergebnissen. Bei Lehrkräften handelt es sich in der Regel um Referenda-
re oder Lehramtsstudierende, aber auch ausgebildete Lehrkräfte sind denkbar. Dement-
sprechend ist auch hier eine weite Variation an technischer Erfahrung und Flexibilität zu
erwarten.

Beobachtende

Eine weitere Teilnehmergruppe von Interesse sind Beobachter. Damit bezeichnen wir Teilnehmer, die nicht selbst am Unterricht teilnehmen sondern nur der Bezeichnung entsprechend die Situation beobachten. Hierbei handelt es sich nicht nur um Dozierende, die einen Lehramtsstudierenden betreuen, sondern z.B. auch um andere Lehrkräfte die nur als Zuschauer anwesend sind.

Auch diese Gruppe ist generell sehr variiert. Teilnehmer aus dieser Gruppe werden aber am meisten mit der Applikation zu tun haben, da sie i.d.R. wiederholt neue Studierende betreuen. Damit sind in dieser Gruppe am ehesten "Powernutzer" oder Experten zu erwarten, die Effizienz über Nutzbarkeit bevorzugen.

Poweruser

4.1.2. Was wollen sie mit dem System tun?

Für Schülerinnen und Schüler besteht ein eher beschränktes Interesse an dem System. Sie kommen vor allem mit dem Umfrageteil in Kontakt.

*Schülerinnen
und Schüler*

Die Lehrkräfte haben auch Interesse daran ihre Selbsteinschätzung zu individuellen Stunden abgeben zu können. Darüber besteht natürlich Interesse an den tatsächlichen Ergebnissen der Stunde. Sie sollten auch selbst Stunden anlegen können und als Lehrkraft oder Beobachter an anderen Stunden teilnehmen können.

Lehrkräfte

Ähnlich zu den Lehrkräften wollen Beobachtende an Stunden teilnehmen können, Ergebnisse einsehen und Stunden erstellen, an denen andere Beobachtende und Lehrkräfte schließlich teilnehmen können.

Beobachtende

Administration

Im Rahmen der später in dieser Arbeit angesprochenen Verbesserungen in Folge des Nutzertests wurden Administrative Rechte als Nutzereigenschaft anstatt einer eigenen Rolle implementiert. Es ist aber trotzdem relevant zu erwähnen das einige Administrative Interessen, etwa die Verwaltung von Nutzern und das Erstellen von Fragebögen, existieren.

4.1.3. Was ist der Kontext?

Der Nutzungskontext lässt sich unterscheiden zwischen Nutzung während des Unterrichts und Nutzung außerhalb.

Außerhalb des Unterrichts ist in der Regel eine ruhige Arbeitsumgebung, generell an einem Laptop oder PC zu erwarten. Dabei werden Nutzer vor allem Stunden vorbereiten oder Ergebnisse einsehen. Es ist also vor allem die Verwaltungsseite der Applikation in Gebrauch.

*Außerhalb des
Unterrichts*

Interessanter ist die Unterrichtssituation. Eine erste unmittelbare Hürde besteht in der Ausstattung des Raums. In technischen Fächern, etwa Informatik, sind voraussichtlich genügend technische Mittel verfügbar als das Schülerinnen und Schüler in einem angenehmen Zeitrahmen an einer Feedback-Session teilnehmen können. Das ist aber oftmals nicht der Fall. Glücklicherweise bietet sich ein Gerät an das sich einer zunehmend weiten Verbreitung erfreut: das Smartphone.

Im Unterricht

4.2. Rahmenbedingungen

Einige Rahmenbedingungen für die Applikation wurden durch Absprache mit dem Lehr- und Forschungsgebiet Informatik 9 der RWTH Aachen University sowie aus der bereits existierenden Feedback App der selben Institution gegeben.

Eine solche Applikation sollte plattformunabhängig sein. Dadurch ist beinahe garantiert das alle Teilnehmer der Stunde ein Gerät zur Verfügung haben auf dem sie teilnehmen können, insbesondere auch wegen der vorher angesprochenen weiten Verbreitung von Smartphones.

*Multiplatform
App*

Nutzer sollen Unterrichtsstunden erstellen und an diesen teilnehmen können. Teilnehmer sollen im Nachhinein Zugriff auf die Ergebnisse aus der Unterrichtsstunde haben, welche auch entsprechend visualisiert sein sollten. Zu diesem Zweck sollen Teilnehmer sich in dem System einloggen können und so an die mit ihnen assoziierten Daten kommen können. Der Teilnahmeprozess für individuelle Stunden sollte dabei für alle Teilnehmer schnell und unkompliziert stattfinden, um den Unterricht nicht unnötig zu stören. Dabei besteht der Anspruch das die Systemnutzung sicher und anonym ist. Eine weitere wichtige Anforderung war, das der Code lesbar und wohl dokumentiert sein soll, damit die Applikation auch nach Abschluss der Arbeit anpassbar und wartbar bleibt.

Workflow

Code Qualität

4.3. Entwicklung

Für den Entwicklungsprozess wurde ein agiler Ansatz gewählt, bei dem die Applikation in iterativen Schritten implementiert wurde. Dabei wurden jeweils zuerst Anforderungen für neue Teile der Anwendung gesammelt und mit der Betreuung dieser Arbeit, welche effektiv als Klient für das Endprodukt fungiert, abgesprochen. Für neue Aspekte der Applikation wurden dann oft als einfacher Papierprototyp umgesetzt, meistens als Skizze der unterschiedlichen Applikationszustände versehen mit Transitionen die die Interaktion verdeutlichen sollten. Beispiele für solche Prototypen finden sich im Anhang C.

Nach einer Diskussion der vorgeschlagenen Features wurde dann der entsprechende Teil der Applikation implementiert. Wurden dabei Probleme deutlich wurden sie behoben, soweit dies im Zeitrahmen möglich war. Dann wurde der Zyklus mit dem nächsten Teil der Applikation fortgesetzt. Schließlich folgte der im nächsten Abschnitt besprochene Nutzertest, der wiederum einige Probleme im Design der App verdeutlicht hat. Nach dem Nutzertest wurde die verbleibende Zeit der Arbeit damit verbracht so viele der neu ermittelten Probleme zu beheben wie möglich.

Kapitel 5 Nutzertest

Um zu testen wie sich eine bereits funktionsfähige Version der Applikation in einem echten Nutzungsfall verhält, wurden Gelegenheiten im Rahmen der Schüleruni Informatik[14] wahrgenommen. Dabei wurden die „Blick über den Tellerrand“-Einheiten, bei denen Schüler der gymnasialen Oberstufe von Lehramtsstudierenden unterrichtet wurden, genutzt.

*Nutzertest in
Schüleruni
Informatik*

Die Anwendung des Feedbacks aus dieser Session wird in der Erkundung der Applikation in Kapitel 6 gründlicher erörtert.

5.1. Ablauf

Dazu war vorgesehen, das bereits bestehende Feedbacksystem durch die in dieser Arbeit entwickelte Software zu ersetzen. Die Schülerinnen und Schüler haben jeweils zwei Einheiten in Form einer Schulstunde absolviert und anschließend mittels der Anwendung Feedback über die Unterrichtsstunde und die anwesenden Lehrkräfte abgegeben. Dabei wurde jede Stunde von einem unterschiedlichen Paar Lehrkräften angeführt.

Nach Abschluss der beiden Stunden, während die Schüler sich in die Mittagspause begeben haben, wurden die Lehrkräfte direkt von den zuständigen Beobachtern in einem Evaluationsgespräch über den Verlauf der absolvierten Stunden informiert.

Während der Feedback-Sammlung wurden Probleme und Eindrücke der Schülerinnen und Schüler gesammelt. Anschließend wurde eines der Lehrkraft-Paare über die Nutzung der Feedback-App und die Visualisierung ihrer Ergebnisse befragt. Dabei war der Fokus der Befragung, Probleme in der Nutzung der Applikation aufzudecken. Dies umfasst technische Störungen, aber auch grundlegende Designprobleme und Schwierigkeiten im Umgang mit der Nutzeroberfläche.

*Durchführung
der Nutzerbe-
fragung*

5.2. Resultate

Auch bei diesem relativ simplen Testablauf ließen sich einige wertvolle Erkenntnisse erzielen. Der folgende Abschnitt geht auf diese ein.

5.2.1. Technische Störungen

Durch die große Anzahl an Probanden und die verschiedenen Systeme auf denen dieses Feedback-System im Nutzertest ausgeführt wurde war bereits abzusehen das technische Störungen auftreten würden.

Performance

Ein Bereich der beinahe erwartungsgemäß Probleme bereitet hat war die Performance der Anwendung. Dabei ist nicht die Leistung der App selbst gemeint, diese war dem Anschein nach in einem akzeptablen Rahmen und nicht merklich störend. Viel mehr ist die ursprüngliche Ladezeit der App problematisch. Da es sich bei dieser Feedback App

um einen Single-Page Applikation handelt muss die gesamte Anwendung essenziell auf einen Schlag geladen werden. Dies umfasst nicht nur den der App zugehörigen Code sondern auch alle verwendeten Pakete, auf denen die App aufbaut.

Hierzu sollten idealerweise so viele Abhängigkeiten wie möglich aus geteilten Quellen die nicht vom Webserver selbst sondern von öffentlichen Repositories geladen werden, z.B. dem Content Delivery Network (CDN). Das entlastet unseren Server und eröffnet die Chance das Pakete bereits im Cache des verwendeten Gerätes liegen.

Edge Browser

Überraschenderweise sind nicht viele Probleme durch die unterschiedlichen Browser die von den Nutzern verwendet wurden aufgetreten. Einzig Microsofts *Edge*-Browser hat Probleme verursacht. Nutzer konnten nicht über die Eingabe eines Session-Codes an der Umfrage teilnehmen und mussten zu einem anderen Browser wechseln. Dieser Fehler scheint mit einem Autofill-Plugin zusammen zu hängen, der im Edge-Browser integriert ist.

Löschen der Nutzerdatenbank

Das erste Lehrkräftepaar konnte weder eine Selbsteinschätzung abgeben noch ihre Ergebnisse einsehen da zwischen den Vorbereitungen am Vortag und der Feedback-Session die Nutzerdatenbank überraschend gelöscht wurde. Die tatsächliche Ursache dafür ist noch immer nicht eindeutig, der Fehler wurde aber vermutlich durch Probleme die beim Erstellen eines Standardnutzers beim Start des Servers entstanden sind ausgelöst.

Für alle weiteren Gruppen wurde provisorisch neue Nutzeraccounts und neue Stunden angelegt. Für die erste Gruppe hingegen musste eine Änderung in der App vorgenommen werden sodass sie mit einem neuen Account Einblick in ihre Ergebnisse bekommen konnten.

Dieser Bug lies sich seitdem nicht mehr replizieren.

5.2.2. Designprobleme

Die problematischste Erkenntnis der Nutzerstudie war, dass einige Aspekte der Nutzerinteraktion falsch oder zumindest nicht flexibel genug eingeschätzt wurden. Dabei haben sich vor allem zwei zentrale Aspekte herauskristallisiert.

Nutzerrollen

Ursprünglich waren Nutzerrollen viel zu eng mit spezifischen Funktionalitäten verbunden. Die Intention war, dass nur Lehrkräfte über ihren Account neue Stunden anlegen konnten und Beobachter dann an diesen teilnehmen. Im Kontext des Nutzertests erwies sich aber das es durchaus sinnvoll ist, wenn auch Beobachter oder sogar an der Stunde unbeteiligte Dritte eine Stunde eröffnen könnten. Indiz hierzu ist, das für den Nutzertest nicht die Lehrkräfte selbst ihre Stunde angelegt haben. Statt dessen wurden Accounts für die Lehrkräfte angelegt und für diese Accounts ohne Zutun der Lehrkraft jeweils eine Stunde angelegt.

*Nicht nur
Lehrkräfte
sollen Stunden
erstellen*

Unmittelbar wurde auch offensichtlich das eine Unterrichtssituation mehrere Lehrkräfte involvieren kann, was zum Zeitpunkt des Nutzertests ebenfalls nicht unterstützt war. Um dies zu kompensieren wurden vorläufig Nutzeraccounts für jedes Lehrerpaar erstellt, welche von den Probanden geteilt wurden.

*Mehrere
Lehrkräfte*

Schließlich zeigte sich das manche der vorher für die Adminrolle reservierten Funktio-

*Flexiblere
Adminrolle*

nalitäten potenziell von Interesse für andere Nutzer, hauptsächlich Beobachtende, sein können. Diese mussten zuvor einen weiteren Account mit Admin-Privilegien erstellen, was umständlich in der Nutzung ist. Schließlich wäre es besser wenn Nutzer nicht zwischen zwei Konten wechseln müssten. Dies betrifft vor allem das Bearbeiten von Umfragen und Erstellen von Nutzern.

Eine Lösung für dieses Problem erfordert ein Umdenken der Nutzerrollen. Die Unterrichtssituation kann von Fall zu Fall stark variieren zwischen betreuten Stunden wie sie in diesem Nutzertest vorlag oder Stunden in denen Lehrkräfte nur ihre eigenen Leistungen überprüfen wollen, ohne das fachliche Betreuer anwesend sind. Auch die Auffassung darüber, wer auf die erweiterten Funktionalitäten der Applikation zugriff haben sollte war zu eng gefasst. Zugriff auf eine spezifische Rolle zu beschränken anstatt auf eine Nutzereigenschaft, könnte die Nutzungssituation mancher Nutzer unvorteilhaft einschränken.

Unterrichtssituation variiert von Fall zu Fall

Auch das die Person die eine Stunde erstellt automatisch die zuständige Lehrkraft sein muss war eine Fehleinschätzung und benötigt eine Revision. Für zukünftige Arbeit an der Applikation ließe sich vermutlich auch argumentieren das es vorteilhaft wäre Nutzerrechte weiter zu diversifizieren, aber dies ließ sich nicht im verbleibenden zeitlichen Rahmen der Arbeit durchführen.

Nicht nur Lehrkräfte öffnen Stunden

Zusammengehörige Fragen

Ebenfalls aufgefallen ist das eine Änderung an der Visualisierung der Ergebnisse zu einzelnen Fragen sinnvoll sein könnte. Viele der Kategorien in den Fragebögen werden von den Nutzergruppen geteilt und viele der Fragen in geteilten Kategorien sind nur Formulierungen der Frage die spezifisch für eine der Nutzergruppen verfasst wurde. Dies war Nutzern allerdings nicht unmittelbar bewusst und die Nutzeroberfläche der Visualisierung unterstützte auch keinen direkten Vergleich von Ergebnissen der unterschiedlichen Gruppen für jede der Fragen. Für Kategorien wurde dieser Designfehler bereits im Vorfeld des Nutzertests behoben, aber für Fragen fehlte weiter jegliche Assoziation zwischen verwandten Elementen.

Keine Assoziation von verwandten Fragen

Hierbei war ein Neudesign der Umfragerepräsentation in der App notwendig. Frageformulierungen sind jetzt in einem Frageobjekt vereinigt, was zu jeweils einer Kategorie gehört. Für jede Frage existieren jetzt also Datensätze die mit spezifischen Nutzerrollen assoziiert sind.

5.2.3. Nutzeroberfläche

Im Design der Nutzeroberfläche sind ebenfalls einige Probleme offensichtlich geworden. Dabei lassen sich mehrere Gruppen von Problemen unterscheiden, die im Folgenden beschrieben werden.

Undeutliche Interaktionen (Discoverability)

Eine Klasse von UI Problemen wird ausgelöst durch Elemente die nicht deutlich genug machen das sie existieren und notwendig sind.

In Abbildung 5.1 ist beispielsweise ein Button dargestellt der ein Pop-Up zur Auswahl eines Fragebogen für die Stunde öffnet. Nutzer haben diesen Button nicht als solchen erkannt oder übersehen und waren lediglich verwirrt warum der "Neue Stunde anlegen"-Button nicht aktiviert war. Das Problem war, das die Aktionsfolge nicht offensichtlich

Interaktionsmöglichkeit bleibt verborgen

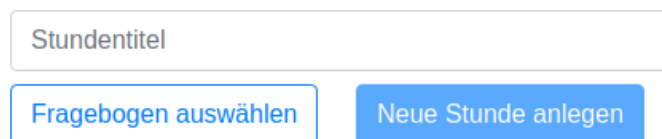


Abbildung 5.1.: Hinter dem “Fragebogen auswählen”-Button verbirgt sich ein Pop-Up.

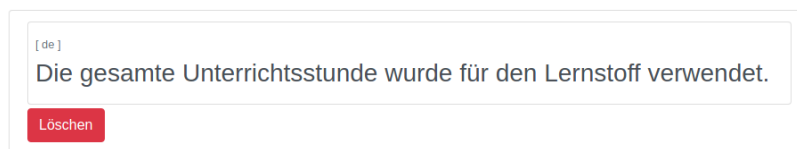


Abbildung 5.2.: Nur der innere Kasten kann geklickt werden.

war und Nutzer dementsprechend nicht wussten das sie einen Fragebogen auswählen konnten bzw. mussten.

Dieses Problem kann auf mehrere Wege gelöst werden. Dem Nutzer muss dazu verdeutlicht werden, dass das entsprechende Element eine Auswahl repräsentiert und das dieses Auswahl ein kritischer Schritt für das Fortschreiten in der aktuellen Aktionsfolge ist. Dazu können konventionellere Auswahlelemente wie Comboboxen¹ verwendet werden, die Nutzer eher mit dem Akt des Auswählens verbinden, oder alternativ kann die Aktion auch erzwungen werden. Der “Neue Stunde anlegen”-Button hätte beispielsweise statt dessen das entsprechende Pop-Up mit der Fragebogenselektion öffnen können. Auch problematisch ist, wenn Klick-Interaktionen nicht explizit als Button stilisiert werden, oder zusammen mit expliziten Buttons auftreten. Kein solcher Fall ist während des Nutzertests unmittelbar aufgefallen, der Umfrageeditor hat allerdings Listenelemente welche genau unter diesem Problem leiden (Abbildung 5.2).

*Combobox oder
erzwungene
Reihenfolge*

Dabei wird ein Element dieser Liste hervorgehoben wenn sich die Maus auf ihm befindet. Dies suggeriert allerdings das das Element geklickt werden kann. Auf einen Klick reagiert aber nur der “Löschen”-Button und die mittlere Box, die nicht mit hervorgehoben wird, die dafür aber eine vorher unsichtbare Textbox ausklappt.

*Problematisches
Listenelement*

Dieses Element löst natürlich Verwirrung dadurch aus, das nicht eindeutig ist wie mit ihm interagiert werden soll. Um dies zu vermeiden werden Interaktionen durch Buttons verdeutlicht oder das gesamte Element klickbar gemacht wenn es als Teil einer Liste auswählbarer Elemente auftaucht.

Feedback App

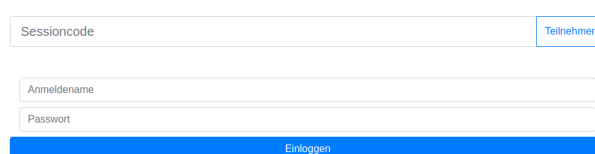


Abbildung 5.3.: Ursprüngliche Homepage.

Ein letztes nennenswertes Problem ist Resultat von mangelndem Abstand zwischen Login-Formular und Teilnahmeelement auf der Hauptseite (siehe 5.3). Insbesondere die Schülerinnen und Schüler waren hierbei durch das Login-Formular verwirrt, da sie dachten dies gehöre zu den nötigen Angaben für die Teilnahme an der Stunde. Grund dafür

¹ <https://de.wikipedia.org/wiki/Combobox>

ist vermutlich die mangelnde Distanz zwischen den Elementen und der hohen Auffälligkeit des "Einloggen"-Buttons im Vergleich zum "Teilnahme"-Button.

Um dies zu korrigieren sollte das Login-Formular visuell in den Hintergrund treten, sodass die Teilnahme-Textbox das zentrale Element der Seite ist und keine Mehrdeutigkeit mehr besteht.

5.2.4. Verbesserungsvorschläge

Zusätzlich sind noch einige Wünsche und direkte Verbesserungsvorschläge von Nutzern geäußert worden.

Umfrageteil

Für den Umfrageteil der Applikation wurde vorgeschlagen beim Abschluss einer Kategorie direkt zur nächsten zu springen. Im Nutzertest mussten Nutzer Kategorien aus einer Übersicht wählen auf die sie jedes Mal nach Abschluss der Kategorie zurück geleitet werden. Eine direkte Weiterleitung soll diesen Vorgang etwas weniger umständlich machen.

Weiterhin wurde darum gebeten das Freitext-Feedback in zwei Felder, jeweils für positive und negative Aspekte, aufzuteilen. Dadurch lässt sich ein klarerer Fokus der Feedback-Texte erwarten, da Nutzer dazu neigten sich auf positives oder negatives Feedback zu beschränken. Die Trennung der beiden Kategorien hilft Nutzern auch damit ihr eigenes Feedback besser logisch zu gliedern.

Visualisierung



Abbildung 5.4.: Anteilmäßige Visualisierung der Ergebnisse für eine Frage.

Bei der Visualisierung befasst sich die Rückmeldung beinahe ausschließlich mit dem in Abbildung 5.4 dargestellten Element. Dieses Balkendiagramm stellt die Anteile der jeweiligen Bewertungen für eine Frage proportional zur Anzahl der gesamt abgegebenen Werte. Dabei stehen die Farben von links nach rechts für *stimme nicht zu*, *stimme eher nicht zu*, *stimme eher zu* und *stimme zu*.

Eine Anmerkung hierzu war, dass die Farben nicht ganz eindeutig sind. Nutzer waren mit der Wahl eine Blautons für "stimme zu" verwirrt. Es bietet sich hier an statt dessen für die positiven Auswahlmöglichkeiten Farbtöne zu wählen, die mehr zu dem erwarteten Farbverlauf von gelb zu grün passen.

Weiterhin wurde darum gebeten die Label der einzelnen Balken in Prozent anzugeben. Die ursprüngliche Angabe in Anzahl der Teilnehmer die diese Bewertung gewählt haben ist ausgewählt worden um zu indizieren wie viele Stimmen tatsächlich in die einzelnen Balken eingeflossen sind. Hier ist es vermutlich am besten beide Möglichkeiten zu unterstützen und zwischen diesen wechseln zu können.

Kapitel 6 Applikation

Bevor auf die technischen Details dieser Arbeit eingegangen wird, verbleibt es einen Blick auf das Endresultat selbst zu werfen. Dieser Abschnitt stellt einen Rundgang durch die wichtigsten Aspekte dieser Feedback App dar. Zur Illustration werden hier Fragmente der Nutzeroberfläche in diversen Abbildungen verwendet. In Anhang D finden sich hingegen vollständige Screenshots der Applikation, die die Komposition der hier beschriebenen Elemente besser verdeutlichen sollen.

6.1. Startseite

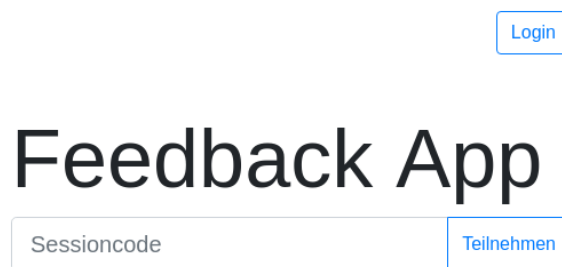


Abbildung 6.1.: Homescreen der Feedback App

Nutzer werden Anfangs¹ durch eine Startseite begrüßt, die sowohl den Zugang zum Nutzerbereich als auch zu dem Umfrageteil für SchülerInnen beinhaltet. Dabei (siehe Abbildung 6.1) handelt es sich um eine simple Seite die nur den App-Namen und zwei weitere Schaltflächen anzeigt: Einen Login-Button und eine Textbox die Schüler an Umfragen teilnehmen lässt.

Die mit „Sessioncode“beschriebene Textbox ist Fokuspunkt der zentralen Interaktion für die Schülerinnen und Schüler. Hier wird ein vierstelliger Code eingegeben, den Lehrkräfte aus ihrem Nutzerbereich entnehmen und der Klasse mitteilen können, damit diese Zugriff auf die korrespondierende Stunde bekommt. Anschließend werden die Details der Stunde in einem Pop-Up Fenster dargestellt, in dem Nutzer dann entscheiden können ob sie an der richtigen Stunde teilnehmen.

Für Lehrkräfte hingegen ist vor Allem der Login-Button relevant, der sie mit dem Nutzerbereich verbindet. Ein Klick auf den Button öffnet ein anderes Pop-Up Fenster mit dem Login Formular, wo sie schließlich ihre Accountdaten eingeben können. Bei korrekter Eingabe werden sie in den Nutzerbereich weitergeleitet.

Dieses Auslagern des Login Formulars ist eine direkte Antwort auf die Verwirrungen die

*Teilnahme per
Code-Eingabe*

*Verstecktes
Login
Formular*

¹ z.B. via <https://http://feedback.elearn.rwth-aachen.de>

im Nutzertest über das in Abbildung 5.3 sichtbare ehemalige Design der Frontseite entstanden ist. Das neue Design versteckt das Login Formular und zieht den Fokus auf das Teilnahme-Feld.

6.2. Umfrageseite

Der Teil der Applikation der am ehesten noch der Vorgängerapplikation ähnelt ist der tatsächliche Umfrageteil. Umfragen sind aufgegliedert in verschiedene Kategorien, die jeweils einige Fragen umfassen. Kategorien und Fragen können sich zwischen den Nutzergruppen unterscheiden.

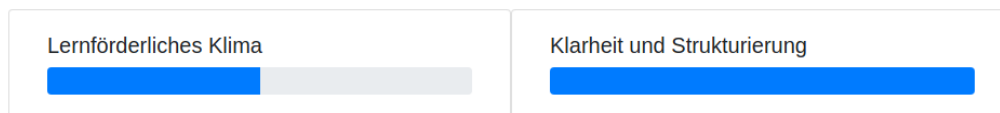


Abbildung 6.2.: Zwei Kategorien mit unterschiedlichem Fortschritt.

Die Umfrageseite besteht aus zwei primären Elementen. Zuerst findet sich eine Liste der Kategorien, wie exemplarisch in Abbildung 6.2 zu sehen ist, aus denen Nutzer dann eine auswählen können um sie zu bearbeiten. Für jede Kategorie wird dabei der momentane Fortschritt angezeigt, definiert als Anteil der Fragen die noch nicht bearbeitet wurden.

Ich war freundlich zu den Schüler-innen.

k.A.	stimme nicht zu	stimme eher nicht zu	stimme eher zu	stimme zu
------	-----------------	----------------------	----------------	-----------

Abbildung 6.3.: Eine Frage mit zugehöriger Antwortauswahl.

Nach Klick auf eine Kategorie werden den Nutzern die der Kategorie zugeordneten Fragen angezeigt. Jede Frage ist wie in Abbildung 6.3 sichtbar mit einem Auswahlelement gekuppelt. Das Auswahlelement soll dabei nicht nach Inhalt der Frage diskriminieren, also keine Wertung für die Auswahl der Antwort implizieren. Es existiert ebenfalls eine vertikal angeordnete Variante die sich besser für Mobilgeräte eignet.

Ein weiterer Unterschied zur Vorgängeranwendung ist, dass Nutzer separate Freitextfelder für positives sowie negatives Feedback haben. Inzwischen wurden auch am unteren Ende der Kategorieseite Knöpfe eingefügt die es erlauben schnell zur nächsten oder vorherigen Kategorie zu springen oder zurück zur Übersicht zu gelangen. Für die letzte Kategorie in der Auflistung springt die Auswahl für die nächste Kategorie automatisch zur Übersicht.

Zum Abschluss der Umfrage werden Nutzer auf die Visualisierung der Umfrageergebnisse weitergeleitet. Diese unterscheidet sich nur durch das auslassen der Informationen von angemeldeten Nutzern, also Lehrkräften und Beobachtenden, von der Visualisierung die eben diese Nutzergruppen zu sehen bekommen.

6.3. Navigation im Nutzerbereich

Die Navigation im Nutzerbereich erfolgt anhand eines "Hamburger"-Menüs, welches eine Liste von für die Nutzer verfügbaren Seiten sowie einen Logout-Button anzeigt.

Dieser Bereich ist nur verfügbar für Lehrkräfte und Beobachtende. SchülerInnen können lediglich anonym an Stunden teilnehmen. In der Nutzertest-Version war dieses Menü noch Tab-basiert. Die Änderung ist nicht Resultat von konkreten Problemen aus dem Nutzertest sondern vielmehr eine Konsequenz der verschiedenen Designänderungen. Momentan ist die Anzahl der Navigationselemente zwar noch übersichtlich, aber ein ausklappbares Menü stellt sicher das auch mögliche Updates die Übersichtlichkeit der

*Tab zu
unübersichtlich*

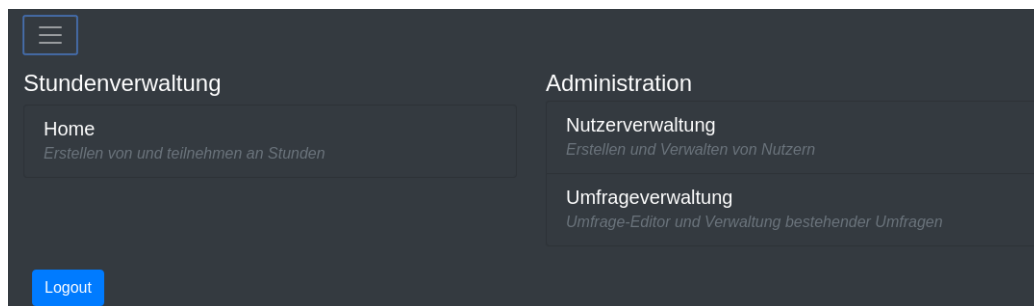


Abbildung 6.4.: Ausgeklapptes Navigationsmenü

Navigation nicht kompromittieren.

Navigationselemente können jetzt in verschiedene Untersektionen organisiert werden, die für schmale Bildschirme (etwa Mobilgeräte) auch zu einer einzigen Spalte zusammenfließen können.

*Responsive
Kategorien*

6.4. Nutzerverwaltung

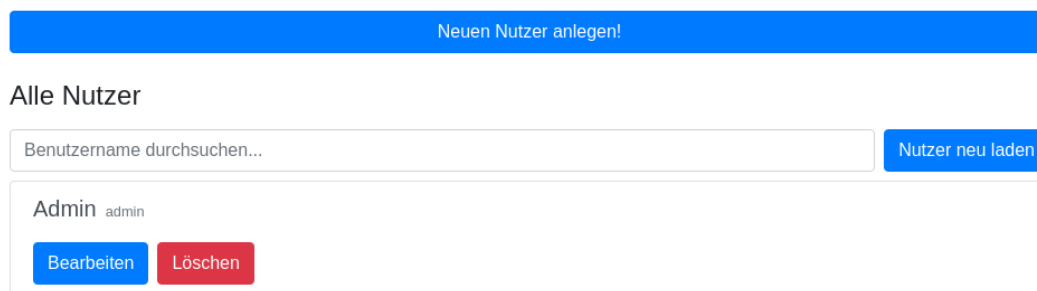


Abbildung 6.5.: Hauptansicht der Nutzerverwaltung

Ein wichtiges administratives Werkzeug ist natürlich die Nutzerverwaltung. Nutzer müssen angelegt werden und sollten jederzeit bei Bedarf löschar sein. Standardmäßig wird ein Admin-Account angelegt, sollte er nicht bereits existieren, dessen Daten in der `.env`-Datei auf dem Webserver hinterlegt sind. Abbildung 6.5 zeigt die Elemente der Nutzeradministrationsseite. Ein Klick auf den mit „Neuen Nutzer anlegen!“ beschriebenen Button öffnet ein Pop-Up Fenster, in welchem die notwendigen Daten für das erstellen eines Nutzers eingetragen werden können (siehe Abbildung 6.6). Das selbe Fenster wird wiederverwendet zum Bearbeiten bereits bestehender Nutzer.

*Erstellen neuer
Nutzer*

Das „Bevorzugte Rolle“-Feld ersetzt dabei die vorher feste Nutzerrolle. Diese Entscheidung ist nach der Erkenntnis gefallen, dass die Einsatzsituation sich sehr wohl von der Erwartung unterscheiden kann. Es ist also nicht auszuschließen das Nutzer in einer anderen Rolle an einer Stunde teilnehmen könnten. Dementsprechend kann sich ihre Rolle nun von Stunde zu Stunde unterscheiden und es wird lediglich eine Präferenz als Annehmlichkeit gespeichert.

*Änderungen in
Folge der
Nutzerstudie*

Die Admin-Rolle war ebenfalls vorher mit einem dedizierten Account verbunden, ist aber in eine Nutzereigenschaft ausgelagert worden, da insbesondere Beobachtende oft beide Rollen einnehmen. Darunter wird eine Liste aller erstellter Nutzer angezeigt, die durch eine Textbox gefiltert werden kann. Durch Klicken auf die „Bearbeiten“-Schaltfläche können Nutzer in dem selben Dialog in dem sie auch erstellt werden, bearbeitet werden. Dabei wird in allen Feldern (mit Ausnahme des Password-Feldes) ein Platzhalter mit den vorherigen Daten eingetragen. Wird ein Feld leer gelassen, bleibt es

*Bearbeiten von
Nutzern*

Neuen Nutzer anlegen

Anzeigename

Der Anzeigename hilft anderen Nutzern Sie zu identifizieren, wenn diese dazu befugt sind!

Benutzername

Wird nur als Anmeldename beim Login verwendet.

Passwort

Ein Passwort muss mindestens 8 Zeichen enthalten!

Kein Admin

Admins können Umfragen und Nutzer erstellen und verwalten.

Bevorzugte Rolle

Lehrkraft BeobachterIn

Abbrechen Änderungen speichern

Abbildung 6.6.: Pop-Up Fenster zum Bearbeiten von Nutzern

beim Speichern der Ergebnisse unverändert.

Ein Klick auf den "Löschen"-Button entfernt einen Nutzer permanent aus der Datenbank. Um versehentliches Löschen zu verhindern wird sich auf eine *forcing function* verlassen, also ein Element der Nutzeroberfläche, das Nutzer daran hindern soll gängige Fehler zu begehen. Dazu wird das tatsächliche Löschen hinter einem auffälligen Dialog versteckt (siehe Abbildung [?]), der einen letzten Ausweg vor dem Löschen darstellt.

Diese Lösung ist nicht ideal, viel eleganter und nutzerfreundlicher wäre die Möglichkeit das Löschen eines Nutzers umkehrbar zu machen. Dies ist aber in der momentanen Implementation nicht ohne Weiteres möglich, da nicht nur der Nutzer selbst sondern auch alle mit ihm verwandten Daten aus der Datenbank entfernt werden, was Wiederherstellung verkompliziert.

Fehlervermeidung mittels forcing function

Warum nicht Undo?

Der Dialog verwendet dabei bewusst kontextspezifische Buttons (anstatt „Ja“ und „Nein“) und keine Farbkodierung, um Nutzer dazu zu bringen über die Situation nachzudenken. Dies verhindert auch potenzielle Interpretationsprobleme, z.B. ob ein roter Button mit dem Löschen des Nutzers assoziiert ist, wie in der Liste, oder mit dem Abbrechen der Aktion.

Dieser Dialog wird für essenziell alle anderen Löschaktionen in der App wiederverwendet.

Maßnahmen gegen Fehler

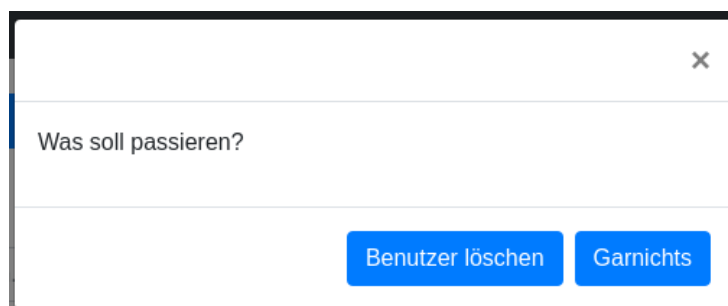


Abbildung 6.7.: Dialog für das Löschen von Nutzer-Accounts

Ein letztes nennenswertes Feature dieser Seite ist die Suchleiste, die es erlaubt nach spezifischen Nutzern anhand ihres Anzeigenamen zu suchen. Es handelt sich dabei momentan um eine simple Suche nach gleichen Textausschnitten im Namen, aber andere Suchmethoden, etwa mittels regulären Ausdrücken, sind bei Bedarf einfach nachrüstbar.

6.5. Umfrageverwaltung

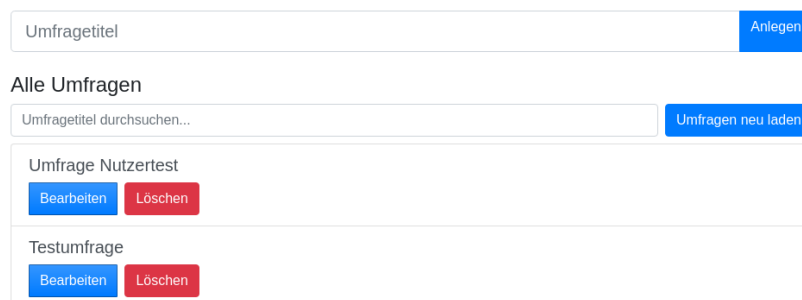


Abbildung 6.8.: Auflistung aller existierenden Umfragen

Eine der größten Erweiterungen des Konzepts der Vorgänger-Anwendung, die dieser Arbeit zugrunde liegt, ist die Ergänzung eines graphischen Umfrage-Editors. Dieser erlaubt Nutzern, momentan beschränkt auf diejenigen mit einer administrativen Rolle, neue Umfragen anzulegen und bestehende Umfragen zu bearbeiten oder zu löschen. Wie in Abbildung 6.8 zu sehen, unterscheidet sich diese Ansicht kaum von der Nutzerverwaltung. Lediglich der Button zum erstellen neuer Nutzer ist ausgetauscht durch eine Textbox, die automatisch eine leere Umfrage mit dem eingegebenen Titel erzeugt, sollte ein Nutzer auf den „Anlegen“-Button klicken.

Ähnlich zur Nutzerverwaltung

Das Bearbeiten von Umfragen erfolgt nicht in einem Pop-Up Fenster wie bei Nutzern, sondern leitet auf eine weitere Unterseite um, die eine Auflistung von Kategorien enthält welche mit der Umfrage assoziiert sind. Eine Umfrage kann eine beliebige Anzahl von Kategorien beinhalten.

In der Version aus dem Nutzertest wurden Kategorien noch in drei Untergruppen, eine für jede Nutzergruppe, eingeteilt. Diese Gruppierung ist jetzt in die Definition der Fragen gewandert. Die Absicht hinter dieser Änderung war, das viele Fragen von mehreren Nutzergruppen geteilt werden, wenn auch mit einer jeweils gruppenspezifischen Formulierung.

Einteilung von Nutzergruppen

Die momentane Implementation ermöglicht und verdeutlicht diese Assoziation der Formulierungen.

The screenshot shows a web interface for managing surveys. At the top left is a blue button labeled 'Zurück zur Übersicht'. Below it, the title 'Umfrage Nutzertest' is displayed. To the right of the title is a button labeled 'Titel ändern'. Below the title is a text input field labeled 'Kategorietitel' with a button labeled 'Anlegen' to its right. Below this is a list of categories, each with a title, a count of questions, and two buttons: 'Bearbeiten' (blue) and 'Löschen' (red). The categories are: 'Klassenmanagement' (5 Fragen), 'Lernförderliches Klima' (5 Fragen), and 'Klarheit und Strukturierung' (0 Fragen).

Abbildung 6.9.: Ansicht einer ausgewählten Umfrage mit abgeschnittener Kategorienliste.

Die Detailansicht der Umfrage, die durch einen Klick auf Bearbeiten erreicht wird, ist sehr ähnlich zu der vorigen Umfrageübersicht aufgebaut (siehe Abbildung 6.9). Durch einen Klick auf den „Titel ändern“-Button öffnet sich ein Dialog in welchem der Titel geändert werden kann und ein Klick auf den „Anlegen“-Button legt eine neue Kategorie mit dem in der benachbarten Textbox angegebenen Titel.

Unterschiede zur Umfrageübersicht

Die Textbox selbst hat dabei die zusätzliche Funktion die Kategorien nach ihrem Titel zu filtern, was dabei helfen soll duplizierte oder ähnlich benannte Kategorien zu finden. Sie kann aber auch einfach als Suchhilfe genutzt werden.

Ab diesem Punkt im Editor werden alle Änderungen an der Umfrage, ihren Kategorien und Fragen automatisch mit dem Webserver synchronisiert. Dadurch gehen Änderungen nicht mehr verloren, was sich als Problem in einer vorigen Version herauskristallisiert hat. Darum wird auch hier der Bestätigungsdialog aus der Nutzerverwaltung verwendet.

Änderungen werden automatisch gespeichert

Dabei ist zu erwähnen, das hier zwar auch keine „Undo“-Funktionalität zur Verfügung steht, es aber durchaus denkbarer ist diese an einem späteren Punkt zu ergänzen, als bei den Nutzerkonten.

The screenshot shows the survey editor interface. It contains two question elements. The first element is titled 'Lehrkräfte' and has the text 'Ich war freundlich zu den Schüler-innen.' Below it is a section titled 'Beobachtende' with the text 'Die Lehrkraft war freundlich zu den Schüler-innen.' Below this section are two buttons: 'Bearbeiten' (blue) and 'Löschen' (red). The second element is titled 'SchülerInnen' and has the text 'Wenn die Lehrkraft eine Frage oder Aufgabe gestellt hat, hatte ich ausreichend Zeit zum Überlegen.' Below it is a section titled 'Lehrkräfte' with the text 'Wenn ich Fragen oder Aufgaben gestellt habe, habe ich den Schüler-innen ausreichend Zeit zum Überlegen gelassen.' Below this section is a section titled 'Beobachtende' with the text 'Wenn die Lehrkraft eine Frage oder Aufgabe gestellt hat, hatten die Schüler-innen ausreichend Zeit zum Überlegen.' Below this section are two buttons: 'Bearbeiten' (blue) and 'Löschen' (red).

Abbildung 6.10.: Zwei Frage-Elemente mit unterschiedlichen Anzahlen an Formulierungen.

Schließlich leitet der „Bearbeiten“-Button wieder zu einer weiteren Unterseite um, die alle Fragen in der ausgewählten Kategorie auflistet. Diese Seite ist wieder ähnlich aufgebaut zu den vorigen, dieses Mal mit einer Liste von Frageformulierungen als Hauptelement.

In Abbildung 6.10 zu sehen ist auch, das nicht verwendete Formulierungen in der Frageübersicht ausgeblendet werden. Eine Frage kann dabei bis zu drei Formulierungen, eine für jede Nutzergruppe, speichern. Ein weiterer Klick auf den „Bearbeiten“-Button

Bearbeitungsansicht für Fragen

Abbildung 6.11.: Bearbeitungsansicht einer Frage.

öffnet schließlich den in Abbildung 6.11 zu sehenden Dialog, in dem Nutzer die verschiedenen Frageformulierungen eingeben und anpassen können. Für diese Ansicht wurden mehrzeilige Textfelder gewählt, da Fragen sonst oft nicht auf einem Blick lesbar sind.

6.6. Nutzer-Home

Abbildung 6.12.: Nutzer können über dieses Element an beliebigen Stunden teilnehmen.

Die Home-Seite ist der zentrale Arbeitsbereich der Nutzer. Hier lassen sich alle laufenden Stunden, und alle Stunden an denen die jeweiligen Nutzer bereits teilgenommen haben, einsehen.

Das erste Element auf der Seite ist eine Begrüßung des Nutzers gefolgt von der selben Teilnahme-Textbox die wir zuvor bereits auf der Startseite gesehen haben. In dem darüberliegenden Element können Nutzer zusätzlich festlegen ob sie einer Stunde als Beobachter beitreten wollen oder eine Selbsteinschätzung abgeben wollen, also selbst als Lehrkraft an einer Stunde beteiligt sind. Welche Aktion standardmäßig ausgewählt ist wird durch die in der Nutzerverwaltung angegebene Rollenpräferenz festgelegt. Mehrfache Teilnahme über dieses Element überschreibt lediglich vorherige Ergebnisse.

*Verschiedene
Nutzerrollen
wählbar*

Das in Abbildung 6.13 abgebildete Element ist eine Liste der momentan laufenden Stun-

Abbildung 6.13.: Exemplarisches Element für die Liste laufender Stunden, inklusive Label der Liste.

den der eingeloggtten Nutzer. Einträge in dieser Liste sind nur Stunden, die von dem Nutzer selbst über den "Neue Stunde anlegen"-Button erzeugt wurden. Das weiter unten beschriebene "Meine Teilnahmen"-Element übernimmt das anzeigen aller Stunden

*Unterschied
zwischen
Teilnahmen
und laufenden
Stunden*

an denen ein Nutzer auch teilgenommen hat, Nutzer müssen nicht zwangsläufig an einer Stunde die sie erstellt haben teilnehmen oder eine Stunde erstellt haben um an dieser teilzunehmen.

Laufende Stunden haben einen Sessioncode, der von Teilnehmern der Stunde verwendet werden kann um an der Umfrage teilzunehmen. Dazu muss dieser lediglich allen Teilnehmern mitgeteilt werden, z.B. durch anschreiben an eine Tafel. Der Code ist von erstellen der Stunde an bis zu ihrem Abschluss mit ihr assoziiert.

Weiterhin werden einige generische Informationen zu der Stunde angegeben, die sich laufende Stunden und Teilnahmen teilen. Diese sind allgemeine Informationen über die tatsächliche Unterrichtsstunde, etwa ein deskriptiver Titel, das Fach und das Datum an dem die Stunde statt finden soll. Diese Informationen werden auch den Teilnehmern vor der tatsächlichen Teilnahme angezeigt.

Zwei weitere Aktionen für jede Stunde werden über Buttons getätigt. Eine Stunde zu schließen löscht das angezeigte Element und verhindert das weitere Nutzer an der Stunde teilnehmen können. Dies sollte geschehen wenn keine weiteren Teilnehmer mehr zu erwarten sind. Nur Nutzer die an der Stunde teilgenommen haben können die Ergebnisse der Stunde unter ihren Teilnahmen einsehen.

Der weitere Button ist eine Abkürzung zur Teilnahme für den Eigner der Stunde. Das Label unterscheidet sich dabei jeweils nach der Auswahl die in dem Element aus Abbildung 6.12 getroffen wurde. Es besteht kein Unterschied dazu statt dessen über den Sessioncode teilzunehmen.

*Stundenbeitritt
via Sessioncode*

*Abgeschlossene
Stunden
erlauben keine
weiteren
Teilnehmer*

Abbildung 6.14.: Beide Seiten des “Neue Stunde anlegen”-Modals.

Ein Klick auf den “Neue Stunde anlegen”-Button öffnet ein Pop-Up Fenster (siehe Abbildung 6.14), das es Nutzern erlaubt eine neue Stunde anzulegen. Das Fenster hat zwei Seiten, eines zur Angabe generischer Informationen über die Stunde und eines zur Auswahl des Fragebogens der verwendet werden soll.

Durch dieses Element wird Nutzern verdeutlicht das sie eine Umfrage auswählen können und müssen, was zuvor in der Nutzerfrage als unklar aufgefallen ist.

Ein letztes nennenswertes Element der Nutzerseite ist die Stundenliste, die alle Stunden aufführt an denen ein Nutzer selbst teilgenommen hat.

Für jede Stunde wird eine anteilmäßige Darstellung der Gesamtergebnisse angezeigt, die Nutzern dabei helfen soll unmittelbar zu identifizieren wie eine Stunde verlaufen ist.

*Resultatebal-
ken und
Farbwahl*

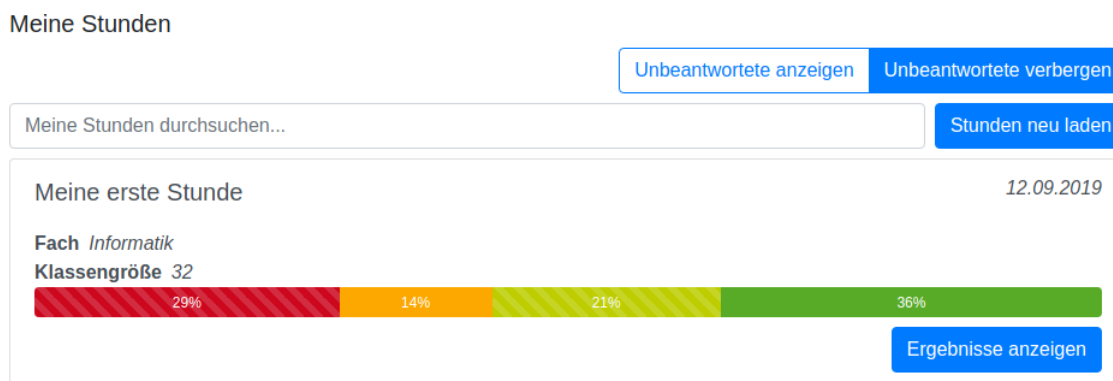


Abbildung 6.15.: Liste aller Stunden, an denen Nutzer selbst teilgenommen haben.

Die Farbwahl dieses Balkens wurde dabei nach negativem Feedback während der Nutzerstudie abgeändert und verwendet nun einen deutlicheren Farbverlauf. Die neuen Farben sind der Palette des Corporate Designs der RWTH Aachen University entnommen. [5] Weiterhin wurde auf Nutzerwünsche reagiert, die Beschriftung der Balkensegmente in Prozent anzugeben.

Oberhalb der Liste befindet sich ein Suchbalken, wie er bereits in der Nutzerverwaltung zu finden war, sowie eine Auswahl ob unbeantwortete Fragen in den Resultaten mitgezählt oder ignoriert werden sollen.

Schließlich können Nutzer über die einzelnen Stunden auf deren detailliertere Ergebnisansicht gelangen, der im Abschnitt folgenden Abschnitt genauer betrachtet wird.

6.7. Visualisierung

Die Visualisierung erfolgt wie im Nutzertest bereits angeschnitten in einer proportionalen Präsentationsweise. Dadurch lässt sich auf einen Blick sehen wie die jeweiligen Kategorien der Umfrage abgelaufen sind, ohne zusätzliche Informationen über die exakte Zusammensetzung dieser Ergebnisse zu vernachlässigen. Dabei wird hier zwischen zwei Gruppen von Kategorien in den Umfragen unterschieden, was sich auch in der Präsentation widerspiegelt: Kategorien des Schülerfeedbacks und sonstige Kategorien. Kategorien des Schülerfeedbacks sind der Kern der Umfrage, schließlich ist das Schülerfeedback der wichtigste Teil des gesammelten Feedbacks. Hier werden alle Kategorien bezeichnet, die zumindest Teilweise von Schülerinnen und Schülern beantwortet wurden.

*Anteilmäßige
Präsentation
der Ergebnisse*

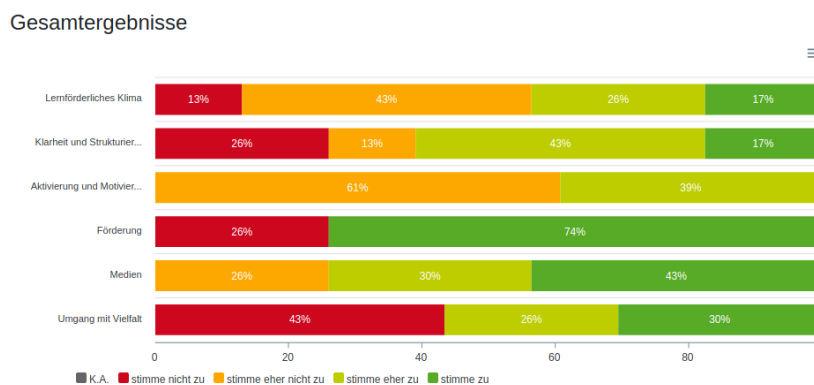


Abbildung 6.16.: Übersicht der Schülerfeedback-Kategorien.



Abbildung 6.17.: Exemplarisches Aussehen einer Schülerfeedback-Kategorie.

Zu Beginn der Visualisierungsseite werden Nutzer dabei von der in Abbildung 6.16 dargestellten Gesamtübersicht der Schülerfeedback-Kategorien begrüßt. Jede Kategorie wird hier von einem Balken dargestellt, der in der Einzelaufstellung der Kategorien, bestehend aus Elementen wie dem in Abbildung 6.17 noch einmal wiederholt wird. Die Balken repräsentieren dabei ausschließlich Schülerfeedback. Ein Klick auf eine der Kategorien führt zu einer Detailansicht der Kategorie, in der die Ergebnisse nach Fragen aufgegliedert werden.

*Balken nur für
Schülerfeed-
back*

Unter den Schülerfeedback-Kategorien werden sonstige Kategorien gelistet, die nicht über Balken verfügen aber ansonsten identisch zu den Schülerkategorien sind. Diese Kategorien sind eher relevant für Evaluationsgespräche zwischen Lehrkräften und Beobachtenden.



Abbildung 6.18.: Schriftliches Feedback, unterteilt nach positivem und negativem Feedback.

Schließlich findet sich noch das Text-Feedback auf dieser Seite, gesammelt in zwei Spalten. Die linke Spalte enthält positives und die rechte negatives Feedback. Zusätzlich wird Feedback von angemeldeten Nutzern mit ihrem Namen versehen. Den Texten wird auch eine Farbe entsprechend der Nutzerrolle zugewiesen, weiß für Schüler, rot für Lehrkräfte und blau für Beobachtende. Durch die Assoziation mit Namen soll es einfacher werden in Evaluationsgesprächen auf die im Text adressierten Punkte einzugehen, da die Autoren der Kommentare direkt konsultiert werden.

Dabei ist wichtig zu erwähnen, dass das Feedback von angemeldeten Nutzern nur anderen angemeldeten Nutzern, die an der Stunde beteiligt waren, zur Verfügung steht. Schülerinnen und Schüler können nur ihr eigenes Feedback und das ihrer Klassenangehörigen sehen.

6.7.1. Kategorien

Schließlich bleibt die Kategorienansicht die per Klick auf eine der gelisteten Kategorien zu erreichen ist. Diese Ansicht ist identisch für beide Arten von Kategorie und besteht aus einer Auflistung der einzelnen der Kategorie zugehörigen Fragen.

Dabei verfügt jede Frage über bis zu drei Sektionen, eine pro Nutzergruppe, welche fehlen können wenn die jeweilige Frage für die fehlende Nutzergruppe nicht verfügbar ist. Ergebnisse von Schülerinnen und Schüler werden wie gewohnt durch ein proportionales Balkendiagramm veranschaulicht, für die anderen Nutzergruppen wird jedoch davon

*Visualisierung
abhängig von
Nutzergruppe*

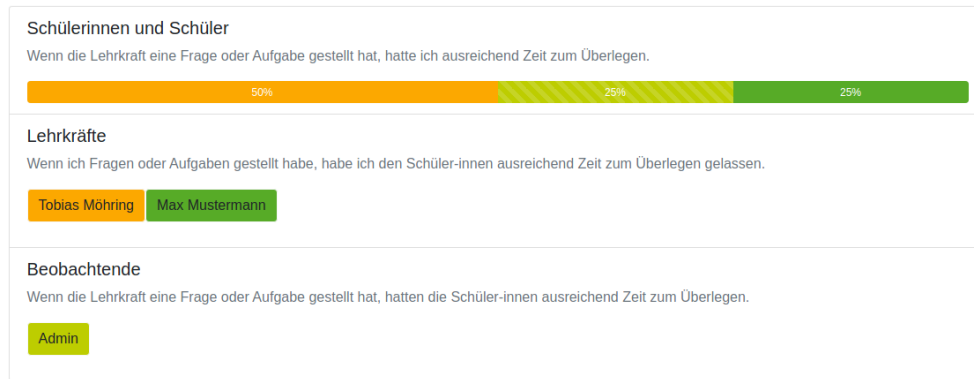


Abbildung 6.19.: Darstellung der Resultate für eine Frage.

abgesehen. Statt dessen werden hier die Nutzer gelistet, die an der Umfrage teilgenommen haben. Ihre Auswahl für die entsprechenden Fragen wird dabei farblich dargestellt und die Namen werden von „stimmt nicht zu“ zu „stimmt zu“, also von rot zu grün sortiert. Dadurch lässt sich noch immer auf einen Blick abschätzen wie der Antworttrend zu einer spezifischen Frage aussieht, es ist aber auch ohne weiteres möglich sich z.B. über Einschätzungen zu erkundigen die man gegebenenfalls nicht erwartet hat.

Teil III

TECHNISCHE GRUNDLAGEN

Kapitel 7 Technologien

7.1. React

Eine zentrale Rolle für das Frontend dieser Applikation spielt React (oder React.js)¹, eine von Facebook Inc. entwickelte Library.

Im Zusammenspiel mit anderen Technologien eignet sich React hervorragend für die Domäne des Designs von Nutzeroberflächen für Single-Page Applikationen. Es ist also sinnvoll die involvierten Technologien zu betrachten, die dafür verantwortlich sind. Zuerst lohnt es sich über JSX² zu reden.

*React als
Interface
Designsprache*

Ohne JSX, was JavaScript um eine XML-esque Syntax ergänzen soll, sieht ein simples "Hello World" Programm beispielsweise wie folgt aus:

```
1  import React from 'react'
   class HelloWorld extends React.Component {
       render() {
           return React.createElement('p', null, 'Hello, ' + this.props.who);
5      }
   }

   import ReactDOM from 'react-dom'
   ReactDOM.render(
10      React.createElement(HelloWorld, { who: World }, null),
      document.getElementById('root')
   );
```

Dabei erhält `React.createElement` drei Argumente

1. ein Element, zuerst ein HTML p-Tag, dann die von uns definierte React Komponente,
2. ein `props` Object, das in der `HelloWorld`-Klasse zugänglich ist,
3. ein Kindobjekt, das in der definierten Komponente eingenistet wird

Diese Schreibweise ist natürlich sehr umständlich im Vergleich zu ähnlichen Programmen, beispielsweise in jQuery oder unerweitertem Javascript. Selbst in reinem HTML ist dieser Code letztlich lediglich äquivalent zu `<p>Hello World</p>`. In Verbindung mit einigen weiteren Methoden, etwa `React.cloneElement`, lassen sich so Meta-Komponenten und andere interessante Designmuster schaffen, aber für die meisten Zwecke wäre eine angenehmere Schreibweise vorzuziehen.

*Verbose
Schreibweise
mit Nischenan-
wendungen*

7.1.1. JSX

Wie bereits angedeutet sind wir nicht auf die Syntax des vorigen Beispiels angewiesen.

*Transpiling als
gängige Praxis*

¹ <https://reactjs.org/>

² <https://facebook.github.io/jsx/>

Da es inzwischen eine gängige Praxis ist, den Frontendcode einen automatisierten Vorverarbeitungsschritt durchlaufen zu lassen, z.B. um moderne ES6 Features im Code auf für alle gängigen Browser verträglichen JavaScript Code übersetzen zu lassen.

JSX macht genau von dieser Phase Gebrauch. Wir können beispielsweise unser “Hello World” Programm wie folgt vereinfachen:

```
1  class HelloWorld extends React.Component {  
    render() {  
        return (<p>Hello {this.props.who}</p>);  
    }  
5  }  
  
ReactDOM.render(  
    <HelloWorld who="World" />, document.getElementById('root')  
);
```

Die Absicht hinter der `HelloWorld`-Komponente wird jetzt unmittelbar klar: Sie erzeugt ein `p`-tag, das begrüßt was auch immer an die Property `who` der Klasse übergeben wurde. Bei einfachen Komponenten wie dieser, die lediglich einige Argumente erhalten und in eine HTML-Vorlage einfüllen, können wir sogar noch eine einfachere Version schreiben:

*Dynamisch
anpassbares
HTML*

```
1  const HelloWorld = props => (<p>Hello {props.who}</p>);
```

Das Resultat dieser Funktion ist nur abhängig von den *props* die ihr übergeben werden. In diesem Fall wird von funktionalen Komponenten gesprochen, welche die Eigenschaft der referenziellen Transparenz erfüllen sollten. Für die selben *props* sollte also stets auch die gleiche Komponente ausgegeben werden. Funktionale Komponenten können auch aus der `React.PureComponent`-Klasse abgeleitet werden. [11] Diese Art von Komponenten kann in manchen Fällen eine bessere Performance bieten, wird hier jedoch vor allem auch aus semantischen Gründen genutzt, um zu verdeutlichen das die Komponente selbst nur als Darstellungselement verwendet wird.

*Funktionale
Komponenten*

Für funktionale Komponenten werden in dieser Arbeit als Konvention Arrow-Funktionen genutzt, dies ist aber generell nicht notwendig. Wichtig zu erwähnen ist, dass JSX-Ausdrücke bei Zuweisungen und als Argument von `return` von Klammern umgeben sein müssen.

Hinweise

7.1.2. State in React

Es reicht allerdings nicht aus nur statischen HTML Code zu generieren. Für komplexere Anwendungen müssen wir auch einen App-Zustand speichern und verändern können. Dazu könnten wir natürlich versuchen einfache Variablen für den Zustand anzulegen und diese über Event-Handler der HTML Elemente zu verändern (z.B. `onClick` in `button` Elementen). Wir würden allerdings feststellen das sich die UI nicht verändert, auch wenn die von uns angelegten Variablen durchaus korrekt geupdatet werden.

*Kein State in
einfachen
Variablen*

Der Grund dafür ist, das React versucht den Rendervorgang zu optimieren indem nur Elemente geupdatet werden, die auch tatsächlich verändert wurden. Dieser Prozess kann sogar kurz aufgeschoben werden um mehrere Updates in einem Renderpass zu kombinieren. Dies verhindert redundantes Rendern der Nutzeroberfläche. Dazu halten React-Komponenten einen eigenen internen Zustand unter ihrem `state` Attribut. Außerhalb eines Konstruktors darf dieser nur via der `setState` Methode angepasst werden. Dadurch wird garantiert das die Komponente über Veränderungen ihres Zustands in Kenntnis gesetzt wird und dementsprechend updaten kann.[12]

*React optimiert
Rendervorgang
über
State-Updates*

Eine simple Komponente die einen Zähler repräsentiert könnte zum Beispiel so aussehen:

```
1  class Counter extends React.Component {
    constructor(props) {
      super(props);
      this.state = { count: 0 };
5   this.countUp = this.countUp.bind(this);
    }

    countUp() {
      this.setState({ count: this.state.count + 1 });
10  }

    render() {
      return (
        <div>
15      Count: {this.state.count}
        <button onClick={this.countUp}> ++ </button>
        </div>
      );
    }
20 }
```

7.1.3. Redux

Mit diesem Komponenten-lokalen Zustand haben wir bereits alle notwendigen Werkzeuge um vollständige Applikationen zu schreiben, allerdings wird mit unserer momentanen Zustandsverwaltung schnell Probleme offensichtlich:

*Probleme mit
lokalem
Zustand*

- Zustand und seine Mutatoren müssen oft über mehrere Kinder-Komponenten weitergereicht werden
- Wenn Informationen in einer parallelen Komponente gebraucht wird muss der State in eine Eltern-Komponente umverlegt werden
- Komponenten-Zustände können verschwinden wenn eine Komponente neu geladen wird (nicht aber beim neu rendern)
- Diskrepanzen in Daten (z.B. aus asynchronen Aufrufen) sind wahrscheinlicher, wenn mehrere Komponenten diese selbst aufrufen anstatt sie sich zu teilen

Es wäre also besser wenn wir einen globalen Speicher hätten, der die ganze App mit einem zentral verwalteten Zustand versorgt und auf den von jeder Komponente aus einfach zugegriffen werden kann. Hierbei spricht man von einem Single Point of Truth (oder SPOT), ein Speicher der konsistenten Zustand für die gesamte App liefert.

*Single Point of
Truth*

Zu diesem Zweck hat Facebook ein Entwurfsmuster für Applikationen entworfen, die Flux³ Applikationsarchitektur.

Flux

Flux stellt eine Abweichung von der weitverbreiteten MVC Architektur dar. Wie in Abbildung 7.1 dargestellt, fließen Daten zwischen vier verschiedenen Komponenten:

*Dataflow in
Flux*

- **Actions:** Objekte mit einem Aktionstypen und optionalen Parametern, die gegebenenfalls von als *Action creator* bezeichneten Hilfsfunktionen erzeugt werden.

³ <https://facebook.github.io/flux/>

- **Dispatcher:** Verteilen Aktionen an alle verfügbaren Stores. Zu diesem Zweck registrieren Stores einen Callback bei dem Dispatcher.
- **Stores:** Übernimmt die Verwaltung des Appzustands. Stores konsumieren Aktionen um ihren derzeitigen Zustand anzupassen.
- **Views:** Für Nutzer sichtbarer Teil der Applikation. Unter Flux sind die View und Controller Aspekte der MVC Architektur vereinigt. Nutzeraktionen (aber z.B. auch asynchrone Aktionen wie der Abschluss einer Datenbankanfrage) können weitere Actions erzeugen.

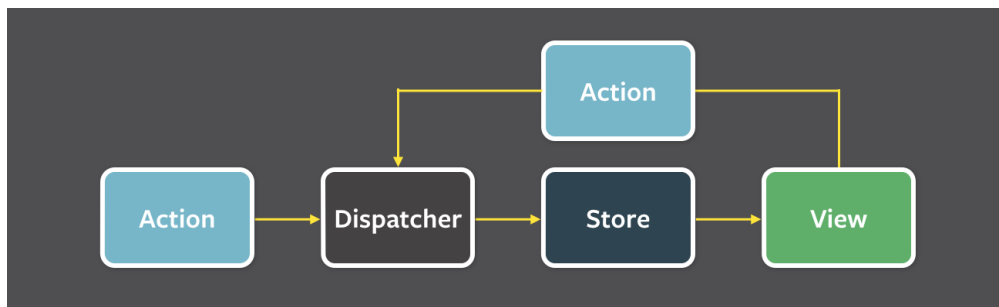


Abbildung 7.1.: Dataflow in Flux[7]

Für dieses Projekt wird Redux verwendet, was von der Flux-Architektur inspiriert ist, sich aber in einigen Aspekten davon unterscheidet. Unter anderem ist Redux nicht nur für React konzipiert, sondern als allgemeine State-Management Lösung.[21]

Einige nennenswerte Unterschiede zu Facebooks Flux-Pattern existieren natürlich. Konventionell verwenden Redux-Applikationen lediglich einen zentralen Store, modelliert durch ein JavaScript Objekt als eine Art State-Baum. Weiterhin hat Redux keinen separaten Dispatcher, sondern ist dieser ein fester Bestandteil des Stores. Der Store darf unter Redux auch keinen veränderlichen Zustand haben, im Gegensatz zu Flux.

Flux vs. Redux

Diese Unveränderlichkeit hängt mit einem weiteren Unterschied zusammen: den *Reducers*. Reducer berechnen den Zustand des Stores als Funktion über den momentanen Zustand und eine frisch eingetroffene Action. Dabei müssen Reducer lediglich den vorherigen Zustand kopieren und die modifizierte Kopie weitergeben. [18]

Reducer statt Dispatcher

Actions

Wie in Flux sind Actions die primäre Schnittstelle zwischen Nutzer und Store. Eine Action ist ein simples JavaScript Objekt, das einen Typen und eine optionale Nutzlast definiert. Ein *Action creator* ist dementsprechend eine einfache Funktion die ein Objekt dieser Art erzeugt. Dabei ist es eine Konvention den Action Typen als String-Konstante zu definieren, was es schwerer macht Tippfehler zu übersehen oder zu vergessen einen veränderten Typennamen anzupassen, da dies zu tatsächlichen Fehlermeldungen führt.

```
1  const CHANGE_COUNT = 'CHANGE_COUNT';
   const changeCount = (amount) => ({ type: CHANGE_COUNT, payload:
     amount });

   const RESET = 'RESET';
5  const reset = () => ({ type: RESET });
```

Reducer

Reducer konsumieren Actions um den aktuellen Programmzustand zu berechnen. Wenn Actions beschreiben was in der App passiert, dann legen Reducer fest wie sich ein Ereignis im Appzustand äußert.

Dazu erhalten sie den alten Programmzustand und die vorderste Action auf dem Action-Queue. Als Resultat liefert der Reducer schließlich einen Wert zurück, der den alten Zustandswert ersetzt. Actions propagieren durch alle Reducer.

```

1  const count = (state = 0, { type, payload }) => {
    switch (type) {
      case RESET:
        return 0;
5     case CHANGE_COUNT:
        return state + payload;
      default:
        return state;
10    }
  }

```

*Reducer
berechnen neue
Zustände aus
Actions*

Mittels der in Redux definierten `combineReducers` Funktion können mehrere Reducer kombiniert werden, sodass letzten Endes nur ein übergeordneter Reducer existiert, der als Basis für den Zustand des Stores verwendet wird.

Bei diesem Prozess wird eine Action zur Zeit abgearbeitet. Alle Actions werden in der Reihenfolge in der sie ausgelöst werden abgearbeitet, was den Appzustand reproduzierbar und konsistent macht. [22]

*Wurzel-
Reducer
erzeugen*

Interaktion mit dem Store

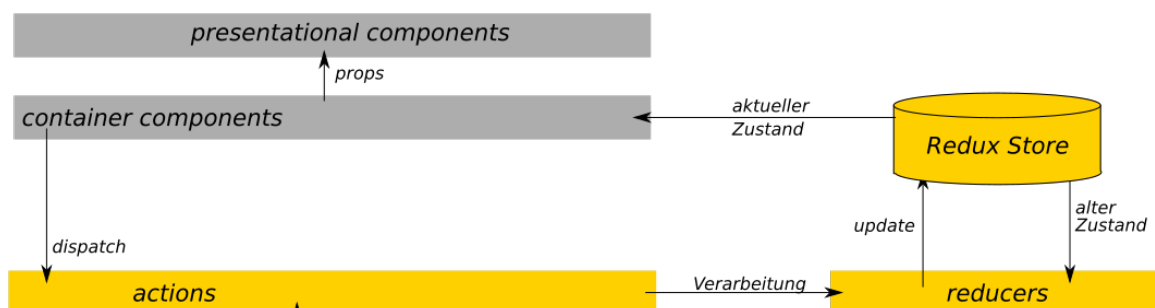


Abbildung 7.2.: Zusammenspiel von React und Redux im Frontend

Schließlich müssen wir noch Daten aus dem Store auslesen bzw. Actions auslösen können. Für React mit Redux ist es dabei wichtig zwischen zwei Arten von React-Komponenten zu unterscheiden: Container-Komponenten und Presentational-Komponenten.[23]

*Komponenten-
arten in
React-Redux*

Präsentationskomponenten sind Komponenten, die nicht mit dem Store in Verbindung stehen. Sie erhalten also alle ihre Attribute von einem übergeordneten Komponente und müssen von Hand geschrieben werden.

*Presentational
Components*

Andererseits sind Containerkomponenten von React-Redux generierte Komponenten, die eine Präsentationskomponente umhüllen und mit dem Store verbinden. Das generieren erfolgt Dank der `connect` Funktion, die zwei Funktionen und die zu verbindende Präsentationskomponente als Argumente erwartet.

*Container
Components*

```
1  const Counter = ({ countUp, countDown, reset, count }) => (  
    <div>  
      <button onClick={countDown}>--</button>  
      { count }  
5    <button onClick={countUp}>+</button>  
      <button onClick={reset}>Reset</button>  
    </div>  
  );  
  
10 const mapStateToProps = state => ({ count: state.count });  
    const mapDispatchToProps = dispatch => ({  
      countUp: () => dispatch(changeCount(1)),  
      countDown: () => dispatch(changeCount(-1)),  
      reset: () => dispatch(reset())  
15  });  
    const ConnectedCounter =  
      connect(mapStateToProps, mapDispatchToProps)(Counter);
```

Als Erstes wird `mapStateToProps` erwartet, die das State-Objekt des Stores erhält und ein Objekt zurückliefert, das beliebig zusammengesetzt sein kann aber sinnvollerweise Einträge des State-Objektes verwendet.

Danach folgt `mapDispatchToProps`, welche eine `dispatch`-Funktion vom Store erhält, die ermöglicht Actions auszulösen. Hier werden dann einige Funktionen definiert, die zur Interaktion mit dem Store notwendig sind.

Beide resultierenden Objekte werden in die Eigenschaften der Komponente eingemischt und können dort einfach verwendet werden.

Teil IV

IMPLEMENTIERUNG

Kapitel 8 Software Ökosystem

Im Rahmen dieser Arbeit wurde eine Full Stack Webapplikation entwickelt. Die technischen Aspekte dieses Prozesses sind Fokus dieses Abschnittes. Dazu wird erst die Applikationsarchitektur erörtert und im Detail erklärt was im Front- und Backend der Anwendung abläuft.

Dann wird auf die Projektstruktur eingegangen, was unter Anderem für die Wartung und Erweiterung der Applikation eine Rolle spielt.

Schließlich wird diskutiert was für die Nutzung der App aus technischer Sicht notwendig ist. In Abbildung 8.1 dargestellt ist der schematische Aufbau der Applikation. Dabei

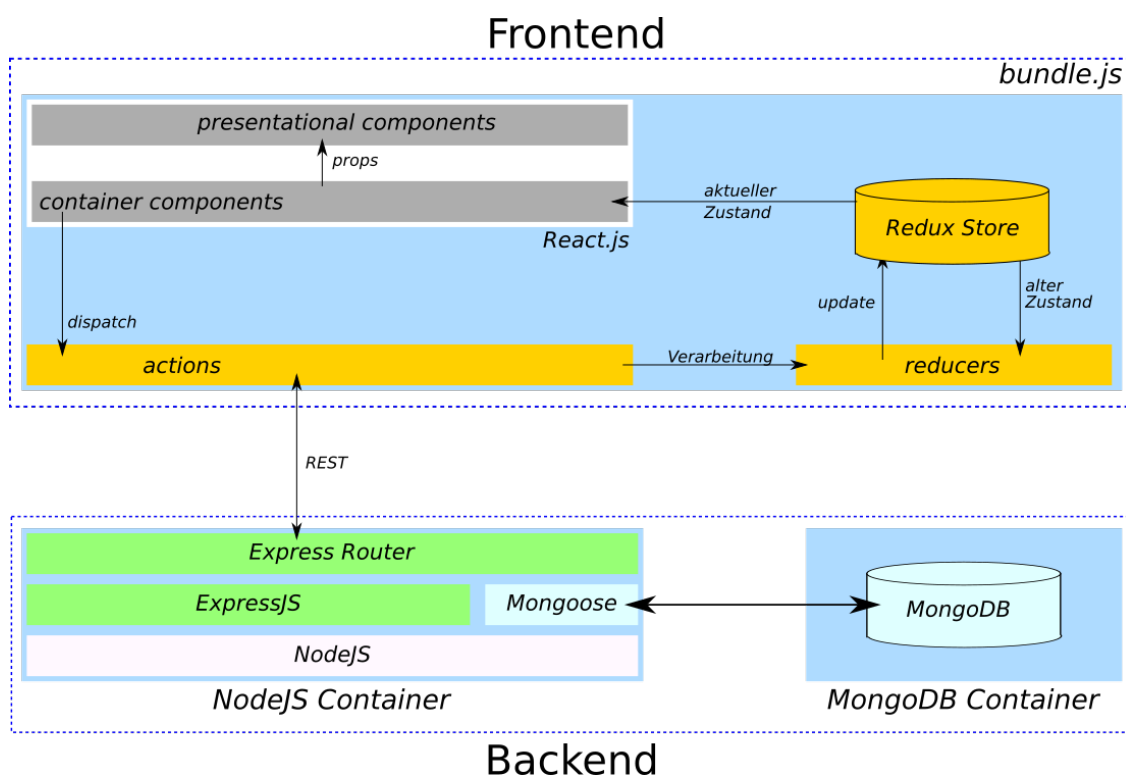


Abbildung 8.1.: Graphische Darstellung der Applikationskomponenten

lässt sich das Projekt in Front- und Backend aufteilen, wie für viele Web-basierte Applikationen üblich.

8.1. Backend

Das Backend stellt die Kernlogik der Applikation und verwaltet äußere Zugriffe auf die Datenbank des Webservers. In unserem Fall besteht das Backend hauptsächlich aus zwei Docker-Containern, welche lokal miteinander kommunizieren können. Dadurch muss

nur einer der beiden Container tatsächlich einen Zugriff nach draußen preisgeben. Der Datenbank-Container, ist nicht nach außen hin offen, sondern steht nur dem Webserver zur Verfügung. Hier läuft eine Instanz von MongoDB.

8.1.1. Docker

Das Backend macht Gebrauch von dem Containerisierungs-Tool *Docker*. Dazu wird ein Dockerfile benutzt um den Webserver aufzusetzen und alle notwendigen Node-Module zu installieren. Das von Docker bereitgestellte *Compose*-Tool wird dann verwendet um den Webserver zusammen mit der Datenbank zu initialisieren und zu starten. Dadurch muss die Datenbank sich nicht im Netzwerk preisgeben, was die Sicherheit erhöht.

*Backend nutzt
Docker*

Der Datenbank-Container und im Development-Modus auch der Webserver-Container machen zusätzlich Gebrauch von Docker Volumes, Verbindungen zwischen dem Dateisystem des Containers und ihrer Host-Maschine. Dadurch bleibt die Datenbank auch beim Löschen und Neuaufsetzen der Container bestehen. Im Development-Modus hilft dies dabei Änderungen im Quellcode an den laufenden Container zu übermitteln sodass dieser nicht ständig neugestartet werden muss um Änderungen geltend zu machen.

*Volumes für
Datenbank und
Debugging*

8.1.2. Webserver

Das Backend lässt sich aufteilen in zwei Komponenten: den auf Node.js basierenden Webserver und die durch MongoDB getriebene Datenbank.

Node.js

Bei Node.js handelt es sich um eine JavaScript-basierte Ausführungsumgebung, die auf der vom Chrome-Browser verwendeten *V8 Javascript Engine* aufgebaut ist. [10] Ein elementarer Vorteil von Node.js ist, das JavaScript eine recht populäre Programmiersprache ist, insbesondere auch im Bereich der Webtechnologien. Zum Beispiel ist JavaScript konsistent die populärste (oder zumindest meist genutzte) Programmiersprache auf GitHub. [20] Dadurch verbessert sich die Chance das projektfremde Entwickler bereits mit der Umgebung vertraut sind und sich somit schneller eindenken können.

*Popularität
von JavaScript*

Für Full Stack Projekte lohnt sich JavaScript umso mehr, da die Logik des gesamten Projektes in der selben Sprache formuliert, und gegebenenfalls auch geteilt werden kann. Dadurch ist es nicht notwendig allzu oft zwischen Sprachen und Konzepten umzudenken.

Express.js

Für die Implementierung des Webserver ist die Entscheidung auf Express.js gefallen, welches ein offenes Web-Framework der Firma *StrongLoop, Inc.* von IBM ist. [4] Da der Webserver für diese Arbeit hauptsächlich dazu dient zwischen der Datenbank und dem Nutzer zu vermitteln war es nicht notwendig ein komplexes Framework auszuwählen. Express bietet sich somit an, da es eine angenehme Syntax zum definieren einer REST API (via des Express Routers) liefert, ohne einen Überhang an Struktur zu erwarten, der für den Umfang dieses Projektes übermäßig gewesen wäre. Da außerhalb der Verwendung einer REST API keine tiefgreifenden Commitments über die Webserverstruktur getroffen wurden, kann der Express-Teil auch bei Bedarf ohne größere Umstände ersetzt werden.

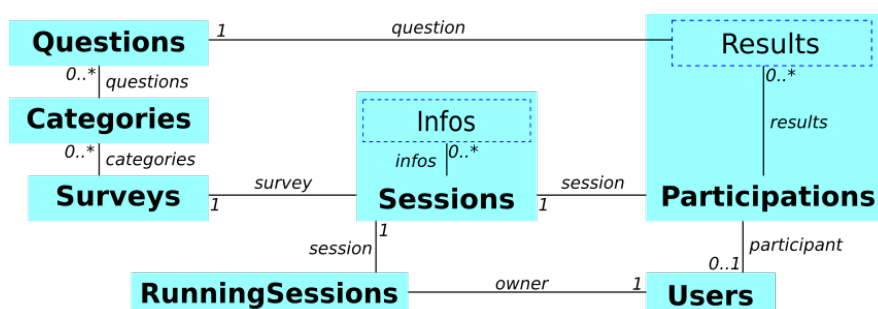


Abbildung 8.2.: Vereinfachte Darstellung der Datenbankschemas

Mongoose / MongoDB

Als Datenbank wurde sich auf MongoDB festgelegt, eine dokumentenorientierte Datenbank der MongoDB, Inc. Da Einträge (bzw. Dokumente) in MongoDB lediglich JSON Objekte sind, bietet sich die Verwendung mit einer JavaScript-Umgebung bereits an. Das von *LearnBoost* implementierte offene Objektmodellierungswerkzeug Mongoose erleichtert den Umgang mit MongoDB noch einmal.

Mongoose erlaubt es Schemas für Dokumente und Operationen auf diesen Objekten zu definieren. Dadurch kann die gesamte Modellierung im Kontext der Datenbankschemas stattfinden. Zusätzlich können Schemas validiert werden, inklusive eigener Validatoren in Form von Middleware, falls notwendig.

Ein weiterer Vorteil ist, dass Dokumente größtenteils wie normale JavaScript Objekte manipuliert werden können. Änderungen am Objekt werden dann einfach durch Aufruf der `save`-Methode des Dokuments verwirklicht. Dokumente können auch "bevölkert" werden, was alle Referenzen auf andere Dokumente in einem angegebenen Teil des Dokuments durch die entsprechenden fremden Dokumente ersetzt.

Die Kombination von Mongoose und MongoDB erlaubt also eine simple Interaktion zwischen Datenbank und Server, was unter anderem Vorteile für die Wartbarkeit der Codebase hat.

8.1.3. Datenbank

Für die Datenbank selbst ist ein Blick auf die verwendeten Schemas und ihre Vernetzung interessant. Abbildung 8.2 stellt eine Vereinfachte Sicht der Datenbank dar, die nur auf die Relationen der einzelnen Sammlungen (bzw. *collections*) eingeht.

Der Graph orientiert sich lose an UML, wobei Assoziationen Ausschlag darüber geben, dass eine Referenz zwischen den beiden Sammlungen existieren. Die Quantität steht an der referenzierten Sammlung. Die Sammlungen *Infos* und *Results* sind Subschemas, die Teil einer anderen Sammlung sind.

Umfragesammlungen

Die Sammlungen *Questions*, *Categories* und *Surveys* bilden zusammen die von Nutzern definierten Umfragen. Fragen speichern dabei unterschiedliche Formulierungen je nach Nutzergruppe, Kategorien verweisen auf die mit ihnen assoziierten Fragen und speichern zusätzlich einen Titel, der in der UI für Nutzer sichtbar ist. Umfragen schließlich haben eine Liste von Kategorien, sowie einen für Nutzer lesbaren Titel.

Zusätzlich besitzen Umfragen einen Parameter, der sie als schreibgeschützt festlegt, wel-

Questions,
Categories,
Surveys

Schreibge-
schützte
Umfragen

cher beim erstellen einer Stunde mit dem jeweiligen Fragebogen gesetzt wird. Ein weiteres Cache-Attribut speichert die Umfrage in einer Baum-Form, die in der Darstellung der Umfrage im Frontend Verwendung findet. Dadurch muss diese nur ein Mal pro Umfrage erstellt werden.

User

Die Nutzersammlung besteht aus einem einfachen Schema das für Nutzer relevante Daten, etwa ihre Logininformationen. Ihr Passwort wird dabei mittels der *bcrypt*-Library gehasht.

Session und RunningSession

Im Kern der Funktionalität steht die Stundenverwaltung, die über die *Sessions*-Sammlung abgewickelt wird. Über Sessions (oder Stunden) werden Informationen über eine konkrete Unterrichtsstunde gespeichert, insbesondere auch welche Umfrage verwendet wurde und wer die Stunde erstellt hat (hauptsächlich für administrative Zwecke).

Stunden stehen im engen Zusammenhang mit der Hilfssammlung *RunningSessions*, die dazu dient Stunden zu identifizieren die noch aktiv sind. Dazu besitzt sie einen einzigartigen Sessioncode, der von Nutzern eingegeben werden kann um an einer Stunde teilzunehmen.

Participation

Letztlich verbleibt die *Participations*-Sammlung, die Feedback-Daten mit Nutzern assoziiert. SchülerInnen werden dabei in einem *Participation*-Dokument zusammengefasst um ihre Anonymität zu wahren. Lehrkräfte und Beobachter hingegen werden über ihre Nutzer-ID mit ihren eigenen Angaben assoziiert, sodass diese bei Bedarf (solange die Umfrage nicht geschlossen ist) jederzeit anpassen können.

In späteren Updates kann diese Sammlung benutzt werden um interessantere Auswertungsmöglichkeiten als eine simple Ergebnismöglichkeit zu unterstützen, z.B. indem die Teilnahmen an mehreren Stunden mit der selben Umfrage verrechnet werden um Ergebnisse über einen Zeitraum zu akkumulieren oder Trends zu verfolgen.

8.2. Frontend

Das Frontend bildet den für Nutzer sichtbaren Teil der Applikation und ist Verantwortlich für die mit Nutzerinteraktion verantwortliche Logik. Frontendcode wird in einem Vorverarbeitungsschritt in einer einzelnen JavaScript-Datei zusammengefasst, die mit einer simplen HTML-Datei vom Server ausgeliefert wird.

Das Frontend verwendet eine Kombination aus der React-Library, die dem beschreiben der Interfacelogik dient und der Redux-Library, deren Aufgabe es ist den Zustand der Applikation als lokaler Store zu versorgen.

React

Bei React handelt es sich Dank der in Kapitel 7 erwähnten Syntaxerweiterung durch JSX um ein sehr intuitives und lesbares Werkzeug zur Beschreibung von Nutzeroberflächen. React erlaubt es im Prinzip selbstdefinierte HTML-Komponenten zu schreiben, die beliebig angeordnet und verschachtelt werden können.

Redux

Um Reacts Probleme mit dem Verteilen und Verwalten des Programmszustands zu kompensieren wurde die Redux-Bibliothek mit einbezogen, die es erlaubt den Programmszustand in einen zentralen Store auszulagern. Das `react-redux`-Paket bildet dabei eine Brücke zwischen den beiden Bibliotheken und erlaubt es React über die Containerkomponenten mit dem Redux-Store zu interagieren.

Redux wurde gewählt da es React auf angenehme Weise um einen Store ergänzt, der als "Single Point Of Truth" für die Applikation dient. Der Store kann auch persistiert werden um den Zustand zwischen Seitenaufrufen zu erhalten. Ohne Redux müssten React-Komponenten ihren eigenen Zustand verwalten. Da die selben Informationen oft in Komponenten gebraucht wird, die in völlig unterschiedlichen Zweigen des DOM-Baums leben, führt dies schnell zu Problemen in der Lesbarkeit und führt schnell zu schwer auffindbaren Fehlern.

Für React bedeutet dies, das Komponenten jetzt in die zwei Arten aufgeteilt werden müssen, die in den technischen Grundlagen beschrieben wurden: *Presentational components*, die den für Nutzer sichtbaren Teil beschreiben und *container components*, die Zustandsinformationen an *presentational components* weitergeben und *actions* auslösen.

*Redux als
Single Point of
Truth*

React Router

Eine weitere nützliche Library ist die *React Router* Library, die es ermöglicht REST-ähnliche Routen für ein Frontend zu definieren. [13] Das erlaubt zum Beispiel die Navigation mittels den Navigationselementen des Browsers ohne zwangsläufig die Seite neu laden zu müssen. Das hat den Vorteil das die verhältnismäßig großen Bandbreitkosten die mit dem laden dieser Anwendung vermieden werden können ohne auf gängige Methoden der Browsernutzung verzichten zu müssen.

Die wichtigsten Elemente hierbei sind das `Link`-Element, das wie ein Hyperlink funktioniert, allerdings beschränkt auf den Adressraum von *React Router* und das `Route` Element, welches mit einem spezifischen Pfad definiert wird und gegebene Komponenten nur dann rendert, wenn der gegebene Pfad derzeit aktiv ist.

*Browsernavigation ohne
Refresh,
Konditionelles
Rendern via
Routen*

Bootstrap

Schließlich verlässt sich das Frontend für viele Layoutfragen und für die Mobilverträglichkeit auf die Bootstrap Frontend Komponenten. [2] Bootstrap liefert viele flexible Komponenten die dabei helfen eine saubere "mobile-first"-Applikation zu designen.

Kapitel 9 Projektaufbau

Wir befassen uns nun mit dem Aufbau der Codebase und wie die Applikation letzten Endes auf technischer Seite zu bedienen ist. Eine aktuelle Version der Applikation mit detaillierterer Dokumentation findet sich auch in der zugehörigen GitLab-Repository.¹

9.1. Die Codebase

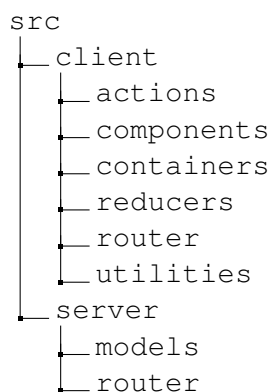


Abbildung 9.1.: Vereinfachter Aufbau des Projektverzeichnis.

Dieser Abschnitt befasst sich mit dem Aufbau des `src`-Ordners, der die gesamte Logik der Applikation enthält. Dabei ist der Ordner in einen Frontend-Teil im `client`-Ordner und einem Backend-Teil im `server`-Ordner aufgeteilt.

9.1.1. client

Der Aufbau des Frontend-Teils ist an Redux orientiert. Die Ordner `actions`, `components`, `containers` und `reducers` enthalten Definitionen von entsprechenden Redux Konstrukten. Dabei sind *presentational components* in `components` zu finden und *container components* dementsprechend in `containers`. Asynchrone Aktionen sind als Redux-Aktionen implementiert, finden sich also ebenfalls unter `actions`.

Zu beachten ist, dass die Trennung zwischen Komponententypen hier nicht so streng gesehen wird wie sich in diversen ähnlichen Projekten finden ließe. Die meisten Containerkomponenten in `container` bringen ihre eigene Präsentationskomponente mit, anstatt sich auf einer Komponente in `components` zu beziehen. Grund dafür ist, dass viele Komponenten ohnehin einen spezifischen Zweck in der UI haben, die Unterteilung also zusammengehörigen Code eher voneinander getrennt hätte als dass es zu einer erhöhten

¹ <https://git.rwth-aachen.de/learntech-lufgi9/bama/ba-cordova-feedbackapp>

Wiederverwendbarkeit führt. Containerkomponenten können bei Bedarf jederzeit aufgeteilt und der präsentationelle Teil nach `components` ausgelagert werden ohne bestehenden Code ungültig zu machen. Der `router`-Ordner enthält Komponenten die entweder direkt von der *React Router Library* dargestellt werden oder Routen-Komponenten aus der *React Router Library* verwenden.

9.1.2. server

Der Backend-Teil der Applikation umfasst zum einen *mongoose* Schemas im `models`-Ordner und zum anderen die Konfiguration der REST Routen in `router`.

Für Modelle der Datenbank kann und sollte dabei Gebrauch von der Schema-Syntax, welche durch *mongoose* mitgebracht wird, gemacht werden. Diese erlaubt das definieren von Methoden auf Dokumenten und gesamten Sammlungen, [9] was dazu führt das der Code für Datenbankinteraktionen deutlich lesbarer wird.

9.2. Aufsetzen der Applikation

Wir müssen nun noch darauf eingehen wie die Applikation tatsächlich zum Laufen gebracht wird. Dabei machen wir uns die durch Docker gegebene Containerisierung zu Nutzen. Voraussetzung dafür, diese Applikation zu starten ist also idealerweise eine lauffähige Instanz von Docker.

Um die Applikation zu starten kann schließlich der Befehl

```
1 docker-compose up --build
```

verwendet werden. Dabei existiert die Möglichkeit optional eine Entwicklungsvariante des Containers zu starten, die unter anderem den `src`-Ordner durchgereicht erhält um auf Änderungen in der Codebase zu reagieren. Dazu muss der Befehl

```
1 docker-compose -f docker-compose.dev.yml up --build
```

verwendet werden. Das `--build`-Flag kann auch weggelassen werden sobald eine lauffähige Instanz des Containers bereits erstellt wurde.

Docker führt die notwendigen Schritte zum Starten, konkret das installieren der notwendigen Node Module, das Starten und Initialisieren der Datenbank und das Hosten des Servers auf einem in der Datei `Dockerfile` definierten Port.

Für die Nutzerstudie wurde die Applikation auf `http://feedback.elearn.rwth-aachen.de` gehostet, wo auch zukünftige Versionen verfügbar sein sollen.

Eine neu gehostete Applikation startet mit komplett leeren Datenbanken, es wird allerdings ein Standard-Admin angelegt, dessen Accountdaten in der `.env`-Datei in der Projektwurzel definiert wird. Sollte diese Datei nicht existieren werden der Standardname `admin` mit Passwort `password` verwendet. Außerdem wird eine Warnung beim Start des Servers ausgegeben die auf die Verwendung dieser Standardeingaben verweist. Mit diesem Admin-Konto können dann neue Nutzer und Umfragen nach belieben angelegt werden.

Teil V

FAZIT UND AUSBLICK

Kapitel 10 Fazit

Diese Arbeit befasste sich mit der iterativen Entwicklung einer Applikation zur Sammlung von Feedback-Daten im Schulunterricht als Hilfsmittel in der Ausbildung von Lehrkräften. Dazu wurde die Applikation unter regelmäßiger Absprache von simplen Papierprototypen zu einer voll funktionstüchtigen Applikation entwickelt, die schließlich in einer echten Unterrichtsumgebung getestet wurde. Dieser Nutzertest warf einige Probleme technischer und konzeptioneller Natur auf, erwies aber auch das die Applikation zu dem derzeitigen Stand bereits größtenteils einsatztauglich war. Daraufhin wurde die restliche Arbeitszeit in das Korrigieren der aufgetretenen Probleme und das Verfeinern der bestehenden Nutzeroberfläche, sowie ein grundlegendes Neudesign der unterliegenden Datenbank investiert. Die resultierende Anwendung löst einige der grundlegenden Probleme aus dem Nutzertest, ist jedoch keineswegs perfekt. Vor allem weitere Feldtests der Applikation, die sich nicht zeitlich im Rahmen dieser Arbeit verantworten ließen, sind zwingend notwendig um das Resultat weiter zu verfeinern.

Unsere Applikation bietet einen funktionstüchtigen Nachfolger der vorherigen Feedback App und erweitert diese um eine Reihe von Features, die sowohl verbesserte Nutzbarkeit als auch komplett neue Interaktionen ermöglichen. Insbesondere betrifft dies die Einführung einer Nutzerverwaltung die es den Nutzern nun möglich macht Ergebnisse aus vorherigen Stunden durch simples einloggen einzusehen. Weiterhin steht Nutzern nun ein Umfrageeditor zur Verfügung der das Erstellen und Bearbeiten von Umfragen über eine einfache Nutzeroberfläche unterstützt. Alle grundlegenden Verwaltungsaktionen wie das Erstellen und Löschen von Nutzern können über die Nutzeroberfläche getätigt werden und benötigen keinen direkten Zugriff auf die Serverumgebung mehr.

*Erweiterung
um
Umfrageeditor,
Nutzerverwal-
tung,
etc.*

Auch die Visualisierung wurde komplett erneuert und ersetzt das alte System, welches die durchschnittlichen Antwortwerten jeder Frage gebildet hat, durch eine neue anteilmäßige Darstellung. Dadurch ist auf einen Blick zu sehen wie gut oder schlecht eine Umfrage, Kategorie oder Frage von der Schülerschaft gewertet wurde ohne Informationen über die genaue Verteilung der Antworten zu verlieren. Weiterhin intendiert die neue Auswertung von Lehrkräften und Beobachtenden einen effektiveren Dialog in den Evaluationsgesprächen zu schaffen. Dazu werden angemeldete Nutzer direkt mit ihren Antworten assoziiert, was es einfacher macht die Positionen der Teilnehmer eines Evaluationsgesprächs zu identifizieren. Nutzer können einfach Fragen wenn ein Resultat für sie heraussticht.

*Verbesserte
Visualisierung
basierend auf
anteiliger
Darstellung*

Die Anwendung verwendet ein hybrides Design, das unabhängig von der Bildgröße eine geeignete Darstellung liefert und sogar spezielle Komponenten in der Nutzeroberfläche verwendet falls Bildschirme zu klein werden um die konventionelle Darstellung zu verwenden. Dazu wird Gebrauch von dem populären Mobile-first Paradigma gemacht, dass über die verwendeten Frameworks die Interfacegestaltung inspiriert hat.

Ein weiteres Ziel dieser Arbeit war einen gewissen Standard an Codequalität zu halten um zukünftige Arbeit an dem Projekt zu erleichtern. Dazu wurde ein weit verbreitetes

*Angestrebte
Codequalität
und Maintain-
barkeit*

Ökosystem basierend auf der populären Programmiersprache JavaScript gewählt um einen Einstieg in die Arbeit an der Codebase zu vereinfachen. Weiterhin wurde darauf geachtet unnötige Kopplung im Code zu vermeiden wenn möglich um Komponenten einfach austauschbar und erweiterbar zu machen. Nach der aus dem Nutzertest resultierenden Überarbeitung ist die Datenbank des Projektes nun ebenfalls besser für zukünftige Erweiterungen gewappnet, die Auslagerung von Ergebnissen in eine eigene Sammlung beispielsweise sollte zukünftige komplexere Visualisierungsmethoden deutlich zugänglicher machen als die vorherige Kopplung an einzelne Stunden.

Andererseits lassen sich auch einige Verfehlungen der Arbeit identifizieren, wie bereits angedeutet wurde. Diese zu beachten ist von besonderem Interesse, sei es um Lehren für zukünftige Projekte zu ziehen oder mögliche Verbesserungen und Lösungen zu finden.

Eine Auffälligkeit ist der Mangel einer tatsächlichen mobilen App-Version dieser Arbeit. Die ursprüngliche Intention war hierzu Cordova[1] zu verwenden, weswegen auch kein wesentlicher Aufwand mehr zu erwarten verbunden sein sollte diese zu vollenden. Schließlich liefert Cordova nur einen als Web-View bezeichneten speziellen Webbrowser aus, der eine Webseite vom Mobilgerät aus lädt. Aus Zeitgründen ist dies allerdings nicht in einem Maße möglich gewesen, das sich von der reinen Webversion abheben würde, weswegen diese in den Vordergrund der Entwicklung getreten ist.

*Keine
dedizierte
Smartphone
App*

Problematischer, aber weniger offensichtlich ist ein konzeptionelles Missgeschick. In Folge des Nutzertests sind einige Probleme dieser Natur bereits behoben worden, hauptsächlich dass zu viele unbegründete Annahmen über die Teilnehmerschaft einer Stunde getroffen wurden was zu dem Szenario führte das Lehrkräfte sich Nutzerkonten teilen mussten. Hier gemeint ist aber die momentane Interpretation von Antwortwerten in der Visualisierung. Momentan werden die Antworten in einem Farbverlauf von rot nach grün dargestellt. Dies macht sich zunutze, das die Kodierung als ein Verlauf von gut nach schlecht wahrgenommen wird, was es einfacher machen soll die Ergebnisse auf einen Blick zu interpretieren. Das kam auch in den Nutzertests als generell gut an. Dadurch ergibt sich jedoch ein kulturelles und ein logisches Problem: Kulturell ist die Assoziation der Farben zu den gewünschten Interpretationen nicht unbedingt gegeben und je nach Betrachter kann es z.B. durch Seheinschränkungen wie Farbenblindheit allgemein zu Interpretationsproblemen kommen; logisch hängt die Bedeutung der Antwortmöglichkeiten sehr von Framing der Frage ab und muss nicht einmal unbedingt eine qualitative Aussagekraft haben. Die einfachste Möglichkeit dieses Problem zu beheben wäre die Wertung von den Antworten zu lösen, also generell grün als die gute und rot als die schlechte Antwortmöglichkeit zu interpretieren, bei Fragen zu spezifizieren wie die Interpretation zu verstehen ist (Gewichtung der Auswahlmöglichkeiten) und hinzunehmen das die Metapher für manche Nutzer flach fällt und diese sich statt dessen auf eine Legende verlassen müssen. Andererseits müsste überdacht werden ob sich nicht bessere visuelle Metaphern oder Visualisierungsmöglichkeiten anbieten, was zumindest eine vertiefte Literaturrecherche oder sogar eine quantitative Studie zu dem Thema involvieren würde.

*Zukünftige
Probleme in der
Visualisierung*

Barrierefreiheit ist ebenfalls ein Thema mit dem sich im Rahmen der Applikation noch genauer beschäftigt werden muss. Viele der Komponenten besitzen bereits Daten um Nutzern mit Sichteinschränkungen zu helfen, allerdings ist die tatsächliche Eignung in Ermangelung von Nutzerfeedback zu diesem Thema noch unbekannt.

*Fragwürdige
Barrierefreiheit*

Schließlich ist noch erwähnenswert, das die Projektarchitektur nicht unbedingt zukunfts-fähig ist. React in Verwendung ist zwar ein robustes und weit verbreitetes Framework,

es ist aber auffällig geworden wie oft es zu Problemen in Kombination mit JavaScript gekommen ist. Ein Wechsel zu TypeScript oder ausführliche automatisierte Tests scheint sinnvoll um eine produktivere Arbeitsumgebung zu schaffen. Weiterhin bietet sich zumindest für den Redux-Teil der Applikation an von den eingebauten Datenstrukturen auf unveränderliche Datenstrukturen aus anderen Libraries aufzubauen, die den von Redux benötigten funktionalen Stil besser unterstützen.

Schließlich gibt es noch viele Stellen der Applikation die dem Nutzer kein ausreichendes Feedback für seine Aktionen geben oder noch nicht ausreichend bequem zu nutzen sind. Es ist unbedingt notwendig diese Komponenten zu überarbeiten um eine akzeptable Nutzererfahrung zu garantieren.

Außerhalb von Problemen der momentanen App-Version gibt es noch einige weitere Verbesserungsmöglichkeiten. Die administrativen Werkzeuge sollten auf jeden Fall erweitert werden, insbesondere in Hinblick auf das bequeme Verwalten und Sichern der Datenbank.

In der Visualisierung bietet es sich an interessantere Auswertungsmöglichkeiten hinzuzufügen. Es wäre etwa denkbar Trends über mehrere Stunden zu berechnen und zu visualisieren, die notwendigen Daten sind bereits einfach in der Datenbank verfügbar und auslesbar, es fehlt lediglich der Darstellungsteil. Andere Graphen als die momentanen anteilmäßigen Balkendiagramme wären ebenfalls eine denkbare Ergänzung.

Das klonen von bereits existierenden Stunden und Fragebögen für eine einfachere Wiederverwendung sind ebenfalls eine einfache Verbesserung der Nutzererfahrung. Eine etwas schwieriger zu implementierende Verbesserung wäre, den momentanen Umfrageeditor um eine Undo/Redo-Funktionalität zu erweitern, was das wiederherstellen von assoziierten Daten, also Fragen und Kategorien in der Datenbank, bedeuten würde. Die Nutzerverwaltung involviert sogar noch mehr Kollateralschaden im Fall einer versehentlichen Löschung und ist somit noch schwieriger wiederherzustellen. Für die Nutzer ist dies allerdings wesentlich angenehmer und sicherer als die Möglichkeit ihre Arbeit durch einen falschen Klick zu verlieren.

Weiterhin lassen sich diverse Komponenten finden, die nicht das notwendige Feedback bei fehlgeschlagenen Interaktionen oder Nutzerfehlern liefern. Das hat keine Bedeutung für die Funktionalität der Anwendung, ist aber weniger zuträglich für die User Experience und sollte dementsprechend unbedingt korrigiert werden. Auch auf Seite der Codebase ist ein gründlicheres Refactoring notwendig, da sich zweifelsfrei Stilfehler und Inkonsistenzen über den hektischeren Teil der Implementation eingeschlichen haben. Aus Codequalitätsgründen bietet sich gegebenenfalls ein die vorher angesprochenen automatisierten Tests und weitere Code Health Tools wie einen JavaScript Linter einzuführen, der Verletzungen in den Stilkonventionen automatisch erkennen kann.

Auch die Entwicklungswerkzeuge und Containerumgebung können noch ausgebessert werden. Ein Prozessmanager für den Server, welcher die Verwaltung des Node-Servers übernimmt und beispielsweise Ressourcenmanagement für diesen übernimmt, ist definitiv eine gute Idee. In der Entwicklung gab es augenscheinlich Inkompatibilitäten solcher Prozessmanager (im Test waren *pm2* und *forever*) mit der Ausführungsumgebung, die unbedingt analysiert und korrigiert werden sollten. Momentan übernimmt das Werkzeug *nodemon* diese Rolle, welches allerdings eher als Entwicklungswerkzeug gedacht ist und einen richtigen Prozessmanager nicht ersetzt. Ein besserer Debugging-Workflow wäre ebenfalls angebracht. Insbesondere Redux und React lassen sich über Browserplugins gut testen, aber ein guter Debugger, der nicht durch den Bundling-Prozess von Webpack gestört wird, wäre ideal.

Alles in allem lässt sich sagen das die ihr primäres Ziel die bisherige Feedback App abzulösen zufriedenstellend erreicht hat. Aber eine lange Liste von Herausforderungen und Erweiterungsmöglichkeiten stehen noch bevor. Viele dieser Verbesserungen sind unproblematisch und nur eine Frage der Verfügbaren Zeit. Weitere Nutzertests sind zwingend notwendig um eine bessere Einschätzung der tatsächlichen Interfacequalität zu bekommen und effektive Verbesserungen vorzunehmen. Insbesondere dem Themenbereich Barrierefreiheit sollte möglichst früh stärkere Aufmerksamkeit gewidmet werden um zu verhindern das betroffene Schülerinnen und Schüler, aber auch andere Nutzer der Anwendung, von der Teilnahme ausgeschlossen werden.

Anhang A Literaturverzeichnis

- [1] Apache cordova. <https://cordova.apache.org/>. Zugriff: 13.09.2019.
- [2] Bootstrap · the most popular html, css and js library in the world. <https://getbootstrap.com/>. Zugriff: 13.09.2019.
- [3] Edkimo - lernen gestalten. digitale plattform für feedback & partizipation. <https://edkimo.com/de/>. Zugriff: 11.09.2019.
- [4] Express. <https://expressjs.com/de/>. Zugriff: 30.08.2019.
- [5] Farbwelt | corporate design | rwth aachen. http://www9.rwth-aachen.de/global/show_document.asp?id=aaaaaaaaaadpbhq. Zugriff: 11.09.2019.
- [6] Feedback app. <http://schuelerlabor.informatik.rwth-aachen.de/fragebogenapp/>. Zugriff: 11.09.2019.
- [7] Flux · application architecture for building user interfaces. <https://facebook.github.io/flux/>. Zugriff: 20.08.2019.
- [8] Google forms. <https://docs.google.com/forms/>. Zugriff: 11.09.2019.
- [9] Mongoose v5.7.1: Schemas - statics. <https://mongoosejs.com/docs/guide.html#methods>. Zugriff: 13.09.2019.
- [10] Node.js. <https://nodejs.org/en/>. Zugriff: 30.08.2019.
- [11] React top-level api - react. <https://reactjs.org/docs/react-api.html>. Zugriff: 15.09.2019.
- [12] React.component - react. <https://reactjs.org/docs/react-component.html#setState>. Zugriff: 15.09.2019.
- [13] Reacttraining/react-router: Declarative routing for react. <https://github.com/ReactTraining/react-router>. Zugriff: 13.09.2019.
- [14] Schüleruni informatik. <https://www.rwth-aachen.de/go/id/etzf>. Zugriff: 06.09.2019.
- [15] Surveymonkey. <https://www.surveymonkey.de/>. Zugriff: 11.09.2019.
- [16] *Der Nutzen von Schüler-Feedback*. Staatsinstitut für Schulqualität und Bildungsforschung, 2017.
- [17] *Methoden des Schüler-Feedbacks*. Staatsinstitut für Schulqualität und Bildungsforschung, 2017.

- [18] Samer Buna. The difference between flux and redux - jscomplete edgecoders - medium. <https://medium.com/edge-coders/the-difference-between-flux-and-redux-71d31b118c1>. Zugriff: 20.08.2019.
- [19] Statistisches Bundesamt Deutschland. Ausstattung privater haushalte mit informations- und kommunikationstechnik - deutschland. <https://www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Einkommen-Konsum-Lebensbedingungen/Ausstattung-Gebrauchsgueter/Tabellen/a-evs-infotechnik-d.html>, 2003. Zugriff: 23.08.2019.
- [20] Thomas Elliott. The state of the octoverse: top programming languages of 2018. <https://github.blog/2018-11-15-state-of-the-octoverse-top-programming-languages/>. Zugriff: 30.08.2019.
- [21] Dan Abramov et al. Prior art · redux. <https://redux.js.org/introduction/prior-art>. Zugriff: 22.08.2019.
- [22] Dan Abramov et al. Three principles · redux. <https://redux.js.org/introduction/three-principles>. Zugriff: 22.08.2019.
- [23] Dan Abramov et al. Usage with react · redux. <https://redux.js.org/basics/usage-with-react>. Zugriff: 22.08.2019.

Anhang B Abbildungsverzeichnis

3.1. Balkendiagramm durchschnittlicher Kategorie-Bewertungen.[6]	8
3.2. Slider-Komponente, die in der Umfrage verwendet wird.[6]	8
5.1. Hinter dem "Fragebogen auswählen"-Button verbirgt sich ein Pop-Up. . .	15
5.2. Nur der innere Kasten kann geklickt werden.	15
5.3. Ursprüngliche Homepage.	15
5.4. Anteilmäßige Visualisierung der Ergebnisse für eine Frage.	16
6.1. Homescreen der Feedback App	17
6.2. Zwei Kategorien mit unterschiedlichem Fortschritt.	18
6.3. Eine Frage mit zugehöriger Antwortauswahl.	18
6.4. Ausgeklapptes Navigationsmenü	19
6.5. Hauptansicht der Nutzerverwaltung	19
6.6. Pop-Up Fenster zum Bearbeiten von Nutzern	20
6.7. Dialog für das Löschen von Nutzer-Accounts	21
6.8. Auflistung aller existierenden Umfragen	21
6.9. Ansicht einer ausgewählten Umfrage mit abgeschnittener Kategorienliste.	22
6.10. Zwei Frage-Elemente mit unterschiedlichen Anzahlen an Formulierungen.	22
6.11. Bearbeitungsansicht einer Frage.	23
6.12. Nutzer können über dieses Element an beliebigen Stunden teilnehmen. . .	23
6.13. Exemplarisches Element für die Liste laufender Stunden, inklusive Label der Liste.	23
6.14. Beide Seiten des "Neue Stunde anlegen"-Modals.	24
6.15. Liste aller Stunden, an denen Nutzer selbst teilgenommen haben.	25
6.16. Übersicht der Schülerfeedback-Kategorien.	25
6.17. Exemplarisches Aussehen einer Schülerfeedback-Kategorie.	26
6.18. Schriftliches Feedback, unterteilt nach positivem und negativem Feedback.	26
6.19. Darstellung der Resultate für eine Frage.	27
7.1. Dataflow in Flux[7]	32
7.2. Zusammenspiel von React und Redux im Frontend	33
8.1. Graphische Darstellung der Applikationskomponenten	36
8.2. Vereinfachte Darstellung der Datenbankschemas	38
9.1. Vereinfachter Aufbau des Projektverzeichnisses.	41
C.1. Früher Papierprototyp der UI.	52
C.2. Ausgewählte Screens eines Softwareprototypen	53

D.1. Exemplarischer Home Screen eines Nutzers.	54
D.2. Volle Ansicht des Nutzerverwaltungswerkzeuges.	55
D.3. Screenshot der Umfrageverwaltung.	55
D.4. Übersicht der Kategorien im Umfrageeditor.	55
D.5. Übersicht der Fragen in einer Kategorie.	56
D.6. Exemplarische Ansicht einer Umfrage.	57
D.7. Auszug aus einem Fragebogen.	57
D.8. Exemplarische Ansicht eines ausgewerteten Fragebogens.	58
D.9. Auszug aus der Detailansicht der Auswertung.	59

Anhang C Prototypen

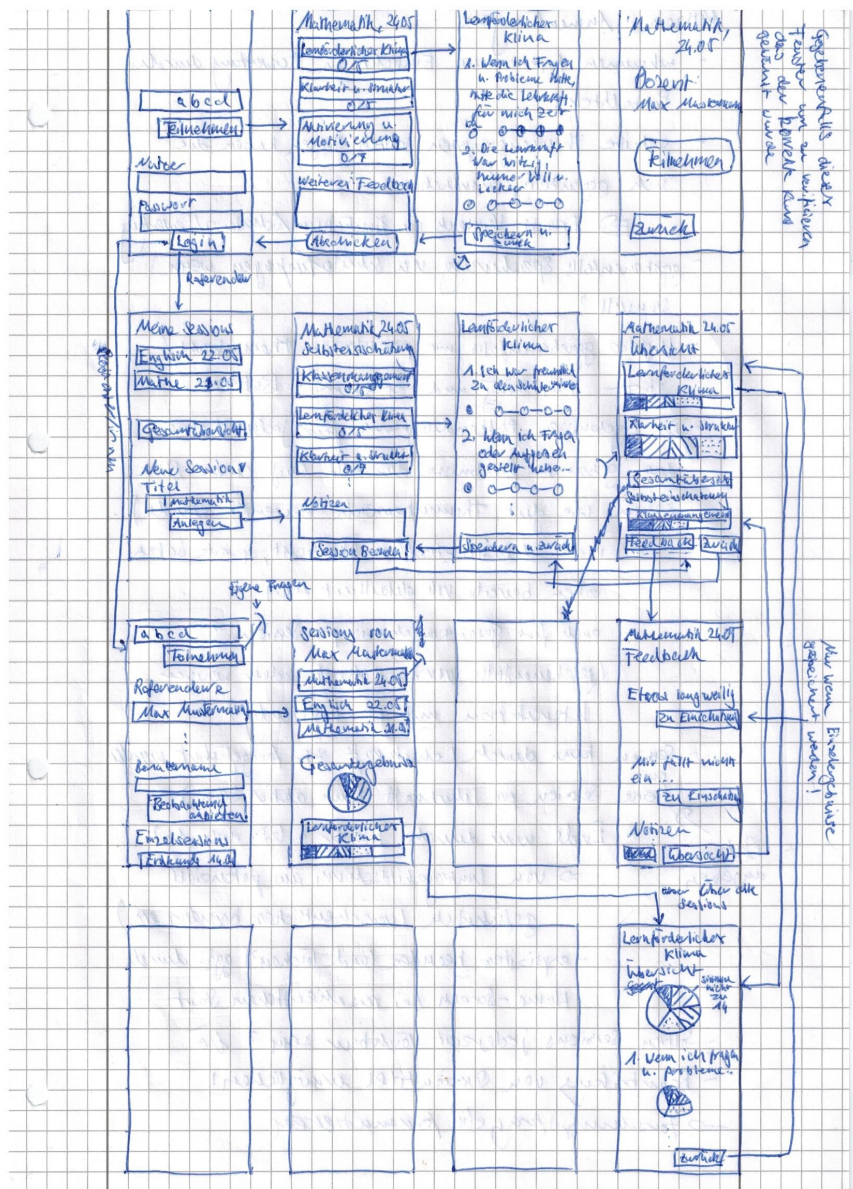


Abbildung C.1.: Früher Papierprototyp der UI.



Abbildung C.2.: Ausgewählte Screens eines Softwareprototypen

Anhang D Screenshots

Hallo, Admin

Selbsteinschätzung abgeben

Stunde beobachten

Sessioncode

Teilnehmen

Meine laufenden Stunden

Neue Stunde anlegen

Neu laden

Sessioncode: ZZBG

12.09.2019

Teststunde

Fach Informatik

Klassengröße 42

Stunde schließen

Als Lehrkraft teilnehmen

Sessioncode: WQJT

13.09.2019

Test

Fach Informatik

Klassengröße 12

Stunde schließen

Als Lehrkraft teilnehmen

Meine Stunden

Unbeantwortete anzeigen

Unbeantwortete verbergen

Meine Stunden durchsuchen...

Stunden neu laden

Meine erste Stunde

12.09.2019

Fach Informatik

Klassengröße 12

%

Ergebnisse anzeigen

Teststunde

12.09.2019

Fach Informatik

Klassengröße 42

13%

19%

36%

38%

Ergebnisse anzeigen

Meine Umfrage

15.09.2019

Fach Informatik

Klassengröße 32

13%

43%

26%

17%

Ergebnisse anzeigen

Abbildung D.1.: Exemplarischer Home Screen eines Nutzers.

Tobias Möhring - Entwicklung einer Cross-Platform Feedback-App zur Verbesserung von Unterrichtsstunden von Lehramtsstudierenden und Referendaren

54

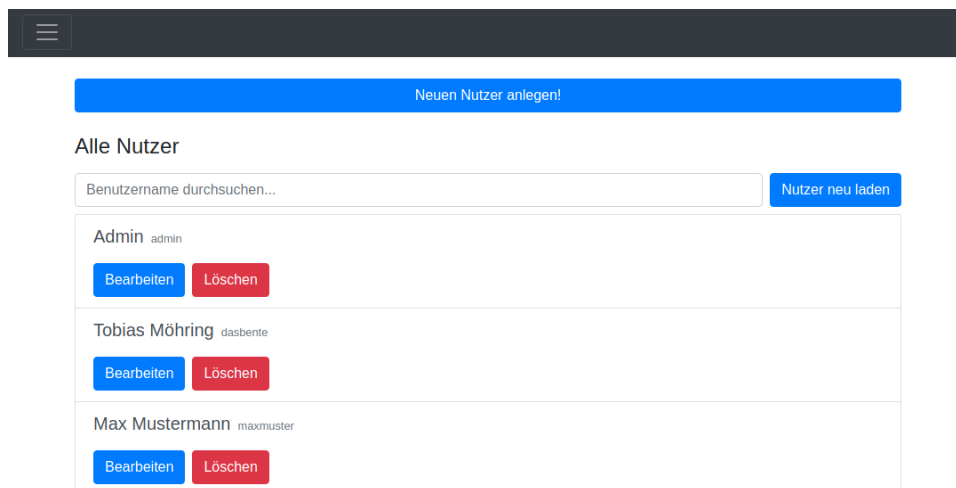


Abbildung D.2.: Volle Ansicht des Nutzerverwaltungswerkzeuges.

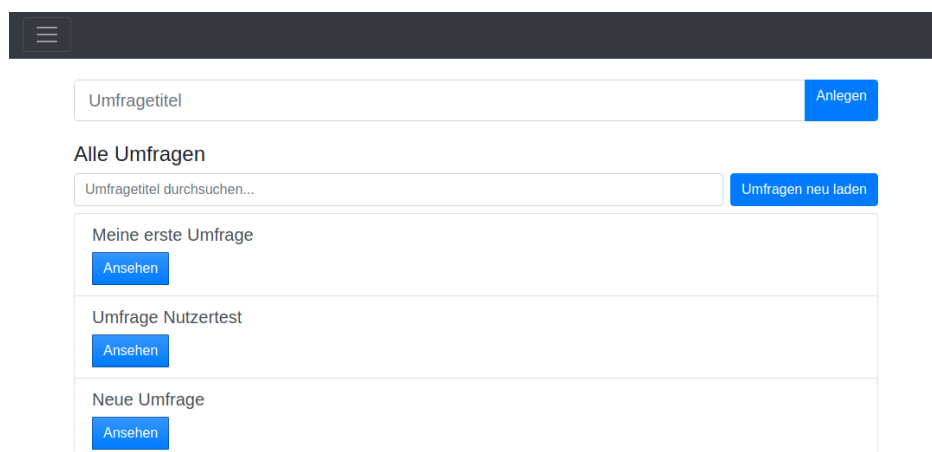


Abbildung D.3.: Screenshot der Umfrageverwaltung.

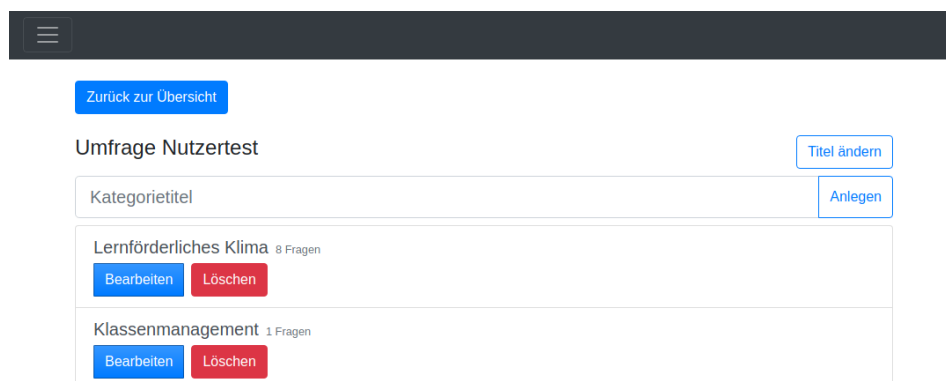


Abbildung D.4.: Übersicht der Kategorien im Umfrageeditor.

Kategorieübersicht

Umfrageübersicht

Lernförderliches Klima

Titel ändern

Neue Frage erstellen

SchülerInnen

Wenn ich Fragen oder Probleme hatte, hatte die Lehrkraft für mich Zeit.

Bearbeiten

Löschen

SchülerInnen

Die Lehrkraft war witzig, humorvoll und locker.

Bearbeiten

Löschen

SchülerInnen

Die Lehrkraft nahm Rückmeldungen von uns an.

Bearbeiten

Löschen

SchülerInnen

Wenn die Lehrkraft eine Frage oder Aufgabe gestellt hat, hatte ich ausreichend Zeit zum Überlegen.

Bearbeiten

Löschen

Lehrkräfte

Wenn ich Fragen oder Aufgaben gestellt habe, habe ich den Schüler-innen ausreichend Zeit zum Überlegen gelassen.

Bearbeiten

Löschen

Beobachtende

Wenn die Lehrkraft eine Frage oder Aufgabe gestellt hat, hatten die Schüler-innen ausreichend Zeit zum Überlegen.

Bearbeiten

Löschen

SchülerInnen

Ich habe mich in dieser Unterrichtsstunde wohl gefühlt.

Bearbeiten

Löschen

Lehrkräfte

Die Schüler-innen haben sich in dieser Unterrichtsstunde wohl gefühlt.

Bearbeiten

Löschen

Beobachtende

Die Schüler-innen haben sich in dieser Unterrichtsstunde wohl gefühlt.

Bearbeiten

Löschen

Lehrkräfte

Ich war freundlich zu den Schüler-innen.

Bearbeiten

Löschen

Beobachtende

Die Lehrkraft war freundlich zu den Schüler-innen.

Bearbeiten

Löschen

Lehrkräfte

Ich habe Fehler nicht kritisiert, sondern als Lerngelegenheit betrachtet.

Bearbeiten

Löschen

Beobachtende

Fehler wurden von der Lehrkraft nicht kritisiert, sondern als Lerngelegenheit betrachtet.

Bearbeiten

Löschen

Lehrkräfte

Ich habe die Schüler-innen ausreden lassen, wenn sie dran waren.

Bearbeiten

Löschen

Beobachtende

Die Lehrkraft hat die Schüler-innen ausreden lassen, wenn sie dran waren.

Bearbeiten

Löschen

Abbildung D.5.: Übersicht der Fragen in einer Kategorie.

Lernförderliches Klima	Klassenmanagement
--	---

Was hat dir besonders gefallen?

Was hat dir nicht gefallen?

Zufrieden mit den Angaben? Rückmeldung abschicken!

Abbildung D.6.: Exemplarische Ansicht einer Umfrage.

Lernförderliches Klima

Wenn ich Fragen oder Aufgaben gestellt habe, habe ich den Schüler-innen ausreichend Zeit zum Überlegen gelassen.

k.A.

stimme nicht zu

stimme eher nicht zu

stimme eher zu

stimme zu

Die Schüler-innen haben sich in dieser Unterrichtsstunde wohl gefühlt.

k.A.

stimme nicht zu

stimme eher nicht zu

stimme eher zu

stimme zu

Ich war freundlich zu den Schüler-innen.

k.A.

stimme nicht zu

stimme eher nicht zu

stimme eher zu

stimme zu

Ich habe Fehler nicht kritisiert, sondern als Lerngelegenheit betrachtet.

k.A.

stimme nicht zu

stimme eher nicht zu

stimme eher zu

stimme zu

Ich habe die Schüler-innen ausreden lassen, wenn sie dran waren.

k.A.

stimme nicht zu

stimme eher nicht zu

stimme eher zu

stimme zu

Zurück

Übersicht

Weiter

Abbildung D.7.: Auszug aus einem Fragebogen.

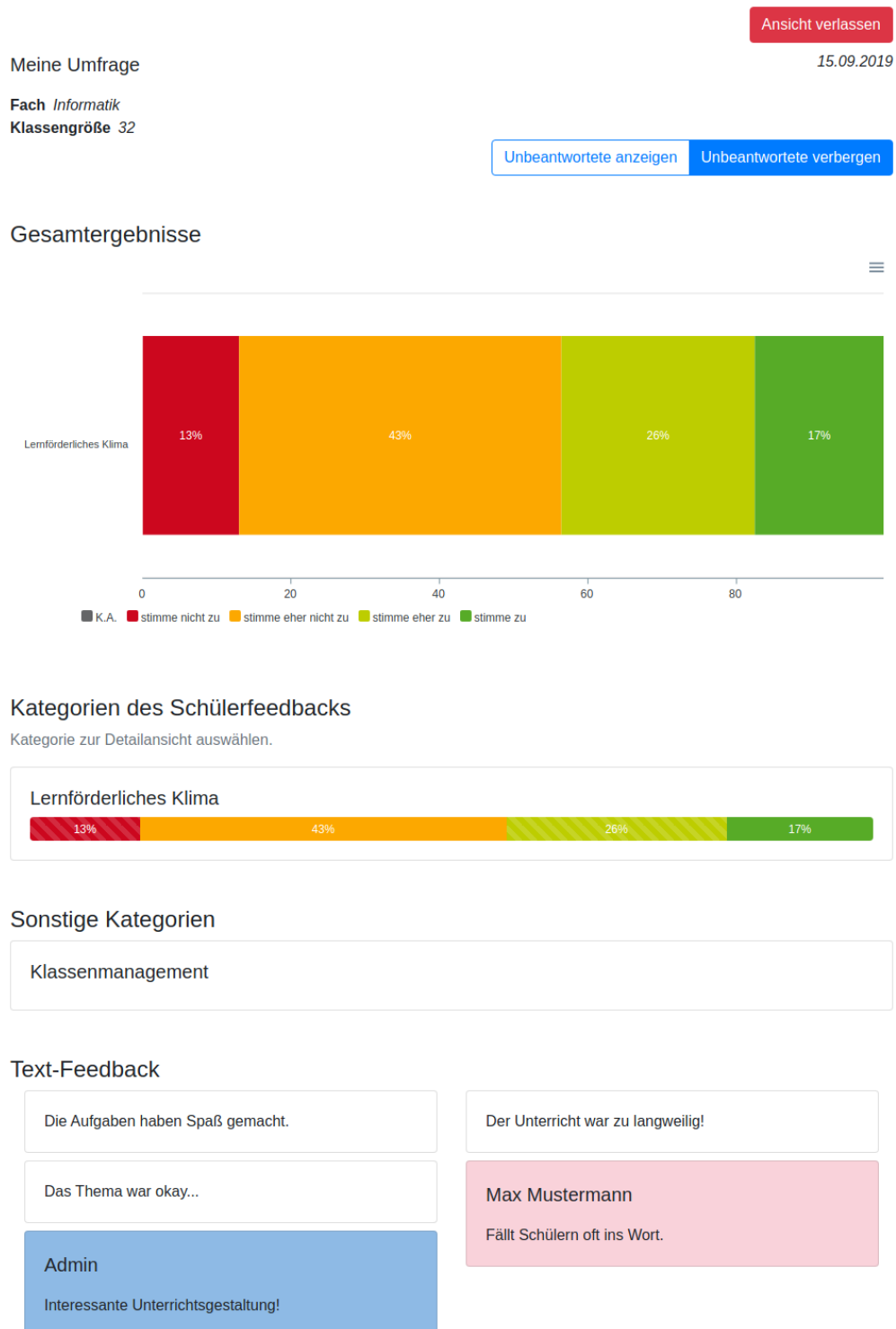


Abbildung D.8.: Exemplarische Ansicht eines ausgewerteten Fragebogens.

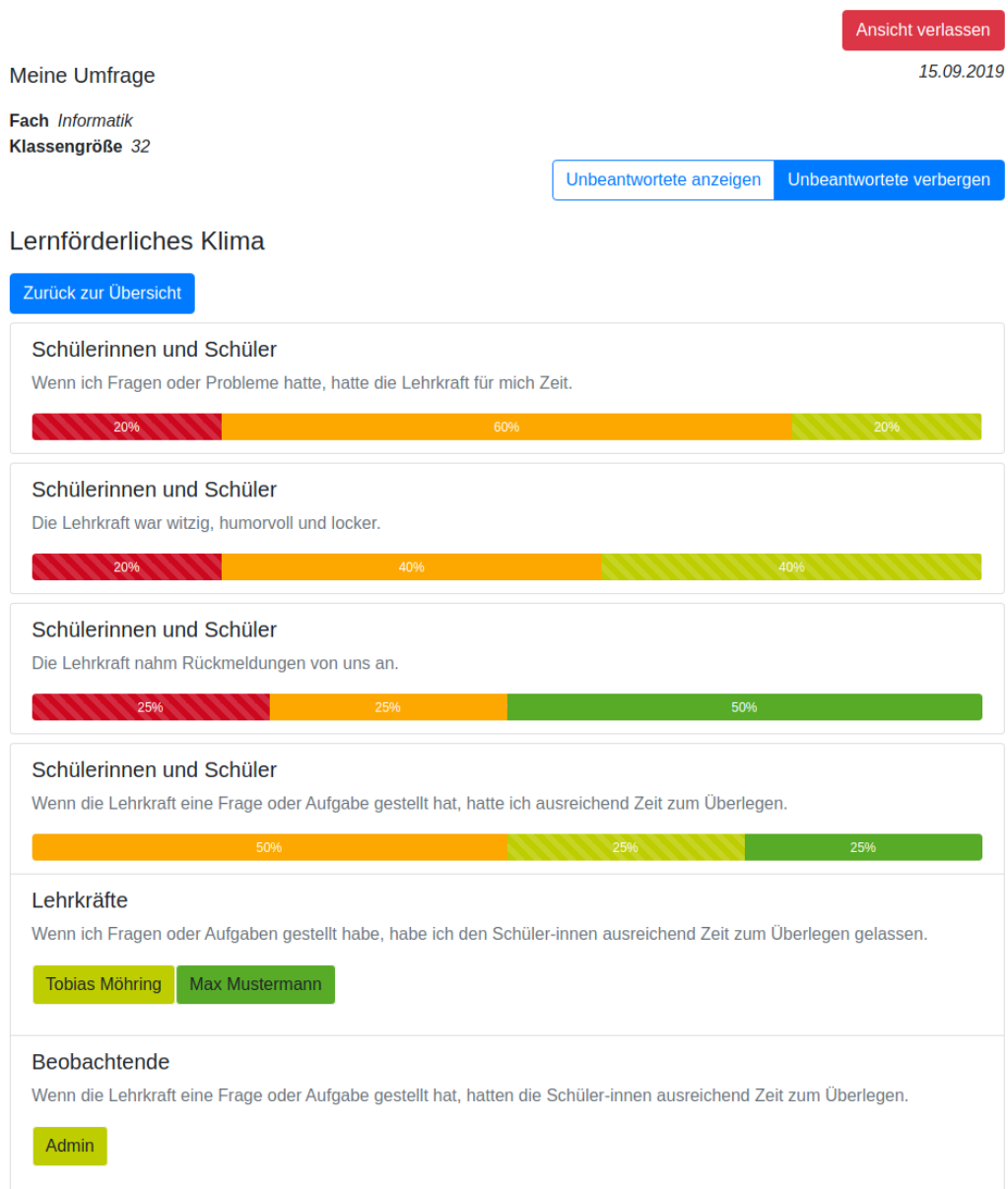


Abbildung D.9.: Auszug aus der Detailansicht der Auswertung.

Eidesstattliche Versicherung

Möhring, Tobias

345522

Name, Vorname

Matrikelnummer (freiwillige Angabe)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/
Masterarbeit* mit dem Titel

Entwicklung einer Cross-Platform Feedback-App zur Verbesserung von Unterrichtsstunden von Lehramtsstudierenden und Referendaren

selbstständig und ohne unzulässige Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Aachen , 15. September 2019

Ort, Datum

Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Strafflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Aachen , 15. September 2019

Ort, Datum

Unterschrift