

Bachelorthesis

Weiterentwicklung der Lernkartei-App 🍷cards zu einer vorlesungsbegleitenden Lernumgebung mit einer Bonusoption für kontinuierlich wiederholtes Lernen

zur Erlangung des akademischen Grades

Bachelor of Science (B. Sc.)

vorgelegt dem


Fachbereich Mathematik, Naturwissenschaften und Informatik
der Technischen Hochschule Mittelhessen


Curtis Adam


im Dezember 2018

Referent: Prof. Dr. Klaus Quibeldey-Cirkel
Korreferent: Christoph Thelen

Zusammenfassung


Die wissenschaftliche Arbeit über das Thema "Weiterentwicklung der Lernkartei-App cards zu einer vorlesungsbegleitenden Lernumgebung mit einer Bonusoption für kontinuierlich wiederholtes Lernen" für die Erlangung des akademischen Grades "Bachelor of Science", wurde von Curtis Adam verfasst.

Das Ziel dieser Arbeit war die Weiterentwicklung der Lernkartei-App cards 3.0 aus der Produktreihe ARSnova, um diese produktiv im Unterricht für das kontinuierlich wiederholte Lernen einzusetzen. Hierbei wurden unter anderem die Benutzeroberfläche vollständig überarbeitet und die Funktionalität der Software erweitert. In den folgenden Kapiteln werden zuerst die neuen Funktionen vorgestellt und mit anderen Lernkartei-App Produkten verglichen. Die Ergebnisse werden danach in einer zusammengefassten Form gegenübergestellt und betrachtet.

Im Anschluss folgt eine Entwickler-Dokumentation über die Besonderheiten von cards. Darunter zählt das Einrichten der Entwicklungssoftware, die Dateistruktur für Templates und der API, der Aufbau von gängigen Objekten wie Kartei-Typen, Karten oder Filter, der Ablauf des Wiederholungs-Algorithmus von Leitner sowie der Aufbau der Akzeptanz-Tests.

Zuletzt kommt eine Schlussbetrachtung über noch ausstehende Änderungen und das Fazit.

Danksagung

Ich möchte mich bei folgenden Personen für die Erstellung dieser Bachelorarbeit bedanken: Zuerst Prof. Dr. Klaus Quibeldey-Cirkel, für das ins Leben rufen der Lernkartei-Plattform cards und anderer ARSnova-Projekte sowie das vorantreiben neuer Funktionen. Zusätzlich gilt mein Dank Daniel Gerhardt, Tom-Käsler und Christoph Thelen, welche mir bei Server- / Datenbank oder **Meteor** Problemen und Fragen zur Verfügung standen. Ohne sie wäre das Projekt nicht so schnell vorangeschritten.

Für das Korrekturlesen, geben von Verbesserungsvorschlägen oder Hilfestellungen gilt man Dank Benjamin Ellen und Gabi Stiller.

Zuletzt möchte ich mich auch bei meinen Eltern Gerd Peter und Kerstin Adam bedanken, dass Sie mir während meiner Studienzeit in Notfällen immer beiseite standen.

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Ort, Datum

Unterschrift

Inhaltsverzeichnis

1	Einführung	1
1.1	Motivation	1
1.2	Ziel	1
2	Über cards 4.0	2
2.1	Demo-Kartei	2
2.2	Wortwolke	2
2.3	Karteien	2
2.4	Repetitorium	3
2.5	Karten	3
2.6	Import und Export	4
2.7	Filter	4
2.8	Präsentationsmodus	4
2.9	Wiederholungs-Algorithmen	5
2.10	Pomodoro-Timer	5
2.11	Benutzer-Rollen und Kartei-Zugänge	6
2.12	Backend	6
3	Vergleich mit anderen Lernkartei-Apps	7
3.1	Anki	7
3.2	Anki-Benutzeroberfläche	8
3.3	Mnemosyne	9
3.4	Mnemosyne-Benutzeroberfläche	10
3.5	Quizlet	11
3.6	Quizlet-Benutzeroberfläche	12
3.7	SuperMemo	13
3.8	SuperMemo-Benutzeroberfläche	14
3.9	Zusammenfassung	15
3.9.1	Fazit	15
4	Getting started	16
4.0.1	Hardwareanforderungen	16
4.0.2	Installation	16
4.0.3	Hinweis zu Meteor updates	16
4.0.4	Datenbank aktualisieren	16
4.0.5	Die Applikation starten	17
4.1	Sonstige Installationen	17
4.1.1	Testdatenbank laden	17
4.1.2	Google, Facebook und Twitter	17
4.1.3	Braintree setup, für PayPal	17
4.1.4	Web-Push-Notifikationen	17
4.2	WebStorm	18
4.2.1	Vorbereitung	18
4.2.2	Einrichtung von Webstorm für cards	18
4.2.3	Codestyle überprüfen	19
4.3	Chimpy	19
4.3.1	Installation für Linux	19
4.4	Guidelines	20
4.4.1	i18n	20

4.4.2	Meteor – Block Helper und JavaScript Hilfsklassen	20
4.4.3	Theme-Switcher	21
4.4.4	Merge-Request und Commits	22
5	Aufbau	23
5.1	Struktur	23
5.2	Server-Initialisierung	24
5.3	Hilfe	24
5.3.1	Template	24
5.4	Pomodoro-Timer	25
5.4.1	SweetAlert2-Benachrichtigungen	26
5.5	Wortwolke	27
5.6	Karte	28
5.6.1	Anlegen eines neuen Karteityp	29
5.6.2	Editor & Markdeep	29
5.6.3	Markdeep Aktualisieren	30
5.6.4	Templates	31
5.6.5	Verfügbare Hintergrundfarben für Karten	33
5.7	Kartei	34
5.7.1	Index	34
5.7.2	Repetitorium	34
5.7.3	Demo-Kartei und The-Making-of-cards	34
5.7.4	Templates	35
5.8	Filter	36
5.8.1	Templates	37
5.9	Benutzer	38
5.9.1	Rollen	38
5.9.2	Bezahlung & Preview-Karten	39
5.9.3	URL & Subscriptions	39
5.10	Lernmodus	40
5.10.1	Leitner	40
5.10.2	Aktivieren von Karten (Wiederholungs-Algorithmus von Leitner)	40
5.10.3	Lernstand-Kartei	41
5.10.4	Lernstand-Benutzer	41
5.10.5	Wozniak	41
5.11	Backend	42
5.11.1	Hochschulen	42
5.11.2	Dashboard	42
5.11.3	Server-Einstellungen	42
5.11.4	Zeige den Pomodoro-Timer statt Wortwolke auf der Landing-Page	42
5.11.5	Versenden von E-Mail-Benachrichtigungen	42
5.11.6	Benachrichtigungen testen	43
5.12	Acceptance-Test	43
5.12.1	Acceptance-Tests ausführen	43
5.12.2	Backdoor Benutzer für die Testdatenbank	44
5.12.3	Hilfsfunktionen	44
5.12.4	Aufbau	44
6	Schlussbetrachtung	45
6.1	Codequalität	45

6.2	Performance	45
6.3	Aktualisieren von Erweiterungen	45
6.4	Antwortoptionen	45
6.5	Offline-Modus	46
6.6	Drag and drop für den Kartei-Index	46
6.7	3D-Ansicht für Karten	46
6.8	Fazit	46

1 Einführung

cards 3.0 ist eine Webapplikation, welche für das erstellen und lernen von Lernkarten konzipiert wurde. In ihr können Studenten Lernkarten erstellen, mit dem Wiederholungs-Algorithmus von Leitner oder Woźniak lernen [QU18] oder ihre Werke an andere Studenten verkaufen. Der Nutzen des Systems hatte sich in seiner derzeitigen Implementation als ungenügend herausgestellt. Die implementierten Lernalgorithmen müssen manuell angestoßen werden und der mögliche Inhalt einer Lernkartei entspricht nicht dem Anforderungsniveau einer Hochschule. Dies liegt zu einem daran, dass eine Lernkartei nur maximal zwei Seiten beinhalten kann und die Formatierung des Inhaltes die Markup-Sprache Markdown verwendet, welches viele Einschränkungen mit sich bringt.

1.1 Motivation

Durch den Ausbau der Lernalgorithmen und Aufbau der Karten, kann cards produktiv für die Vorlesungen eingesetzt werden. Studenten können mit dem neuen System Lernkarten für die Klausuren lernen, Notizen für Vorlesung erstellen und auch Ihre Hausaufgaben darüber tätigen. Anstelle eines Stifts und Papier benötigen die Studenten nur noch ein Smartphone oder ein anderes Gerät mit Internetzugang, um dem Unterricht aktiv folgen zu können. Der Ausbau der Wiederholungsalgorithmen bereitet die Studenten zudem besser auf die anstehende Klausur vor, da diese durch regelmäßige Benachrichtigungen motiviert werden, in regelmäßigen Abständen den Inhalt des Moduls zu lernen.

1.2 Ziel

Die Lernplattform cards wird mit vorlesungsbegleitenden Tools erweitert:

- Präsentationsmodus für Karten [TH18]
- Demo-Kartei
- Repetitorium [TH18]
- Verschiedene Karteitypen mit bis zu 6 Seiten Pro Karte [TH18]
- Schwierigkeits-Stufen [TH17b] für bestimmte Karteitypen
- Karten-Tools: Minutensprunghuhr und Pomodoro-Timer
- Eine Wortwolke [TH18] zur Darstellung von besonders wertvollen Karteien und Repetitorien
- Eine überarbeitete Benutzeroberfläche zur einfachen Navigation auf Mobilgeräten mit Hilfe von Bootstrap [OT16]
- Ein Karteneditor mit Vorschau für Desktop und Mobilgeräte
- Markdeep, eine Weiterentwicklung der Markdown-Syntax mit mehr Funktionalitäten [MC18] und MathJax für den Inhalt einer Karte
- Wiederholungs-Algorithmus von Leitner mit Server Benachrichtigungen über neu zu lernende Karten
- Drag and drop für das Sortieren des Kartei-Index*
- 3D-Ansicht für Karten*

*Tools werden spätestens nach dem release von cards 4.0 zur Verfügung stehen.

2 Über cards 4.0

2.1 Demo-Kartei

Die Demo-Kartei dient als Einstiegspunkt für neue **cards** Benutzer. Der Inhalt dieser Kartei zeigt Beispiele für alle Karteitypen und den Funktionsumfang der Karten-Navigation. Der Inhalt der Demo-Kartei kann für den Server angepasst werden.



Abbildung 1: Die cards Demo-Kartei [QU18]

2.2 Wortwolke

Die Wortwolke zeigt die von der Redaktion als hochwertig gekennzeichnete Karteien und Repetitorien. Der Benutzer kann durch einen Klick auf den Namen ein Modal öffnen, welches alle Informationen über die Kartei oder das Repetitorium anzeigt.

2.3 Karteien

Karteien können vier verschiedene Zugriffsarten besitzen: Privat, Frei, THM oder Pro. THM und Pro Karteien müssen vorerst von Standard-Mitgliedern gekauft werden, bevor diese alle Karten einsehen können. Für Mitglieder von Hochschulen stehen alle Karteien bis auf Privat und Pro zur Verfügung.

2.4 Repetitorium

Ein Repetitorium ist vom Aufbau her genau wie eine Kartei, jedoch referenziert es nur seinen Inhalt auf bereits bestehende Karteien. Dies ermöglicht es, dass der Inhalt einer Kartei in verschiedenen Repetitorien verwendet werden kann. Falls eine Änderung in der referenzierten Kartei stattfindet, wird diese automatisch im Repetitorium angezeigt.

2.5 Karten

Lernkarten werden mit Hilfe von Karteien verwaltet. Eine Kartei kann nur einen Kartentyp verwalten. Diese Einschränkung lässt sich jedoch umgehen, indem ein Repetitorium angelegt wird.



Es gibt derzeit die Möglichkeit Kartentypen von 18 unterschiedlichen Typen zu erstellen. Diese unterscheiden sich unter anderem durch folgende Punkte:

Kartentyp	Seiten	Schwierigkeits-Stufen	Lernziel Taxonomie	Spaced Learning	Rückwärts Lernen *	Besonderheiten
1. Mitschrift	3	X				
2. Lerneinheit	4	X	X	X		
3. Aufgabensammlung	2	X		X		
4. Glossar	3			X	X	
5. Anweisungssatz	3	X		X	X	
6. Formelsammlung	4	X		X		TEX-Formelsatz mit MathJax in SVG-Qualität
7. Entwurfsmuster	6	X		X		
8. Zielerreichung	2			X		
9. Inverses Fragen	2	X		X	X	Frage A: »Wie sieht das UML-Klassendiagramm des Entwurfsmusters Observer aus?« Frage B: »Welches Entwurfsmuster beschreibt dieses UML-Klassendiagramm?«
10. Quiz	2	X		X		
11. Prüfung	3	X	X	X		
12. Vokabelkartei	2	X		X	X	Integration des Wörterbüches BEOLINGUS und des Übersetzungsdienstes DeepL
13. Fotokartei	2					Lightbox
14. Zitatensammlung	3			X	X	
15. Exzerpte	1					
16. Notizen	1					
17. To-dos	1					Task-Liste à la GitHub
18. Vortrag	1					für die Beamer-Präsentation optimiert: Vollansicht, Textskalierung, Tastensteuerung, direkter Aufruf der Audience-Response-Apps ARSnova und arsnova.click


Tabelle 1: Die cards Ausstattungsmerkmale der Kartentypen [QU18]

* Bei Kartentypen mit dem Ausstattungsmerkmal “Rückwärts Lernen” hat der Benutzer die Auswahl, ob die Vorderseite oder Rückseite zuerst angezeigt werden soll.

2.6 Import und Export

Es besteht die Möglichkeit Karten und Karteien zu importieren und exportieren. Das hierbei verwendete .json Format ist dabei für die cards Anwendung konzipiert, jedoch können auch Karten aus dem Programm Mnemosyne über eine .csv Datei importiert werden. Das Format von cards wurde so konfiguriert, dass der Kartentyp nach dem Anlegen verändert werden kann. Nicht benutzte Seiten werden nur ausgeblendet und bleiben in der exportierten Datei oder Datenbank enthalten.

2.7 Filter

Mithilfe einer Filterfunktion können Benutzer nach Karteien eines bestimmten Themengebiets suchen. Hierbei bietet cards zwei unterschiedliche Darstellungsmöglichkeiten für die Ergebnisse an: Eine Listenansicht und eine Wortwolke. Über diese Listen kann der Benutzer die Details einer Kartei ansehen und gegebenenfalls den Inhalt zu seinem Lernpensum hinzufügen, falls er die benötigten Zugangsberechtigungen besitzt.

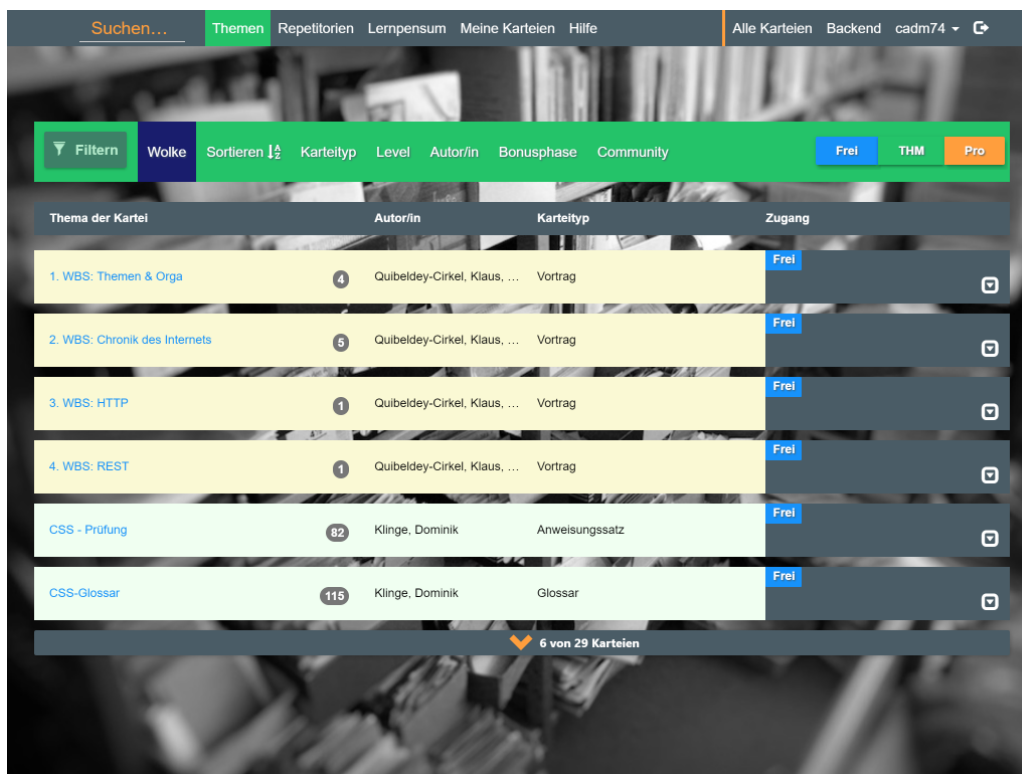


Abbildung 2: cards Filter für öffentliche Karteien

2.8 Präsentationsmodus

Studenten und Dozenten haben die Möglichkeit, ihre Karten in einem Präsentationsmodus anzeigen zu lassen. Hierbei werden alle Elemente bis auf die Karte und die Karten-Navigation ausgeblendet. Zusätzlich nimmt die Karte die gesamte Bildschirmhöhe ein, damit so viel Platz wie möglich für den Karteninhalt verwendet wird. Falls der Platz nicht ausreicht, hat der Benutzer die Möglichkeit die Schriftgröße des Karteninhaltes, mit Hilfe einer eingebauten Zoom-Funktion anzupassen.

2.9 Wiederholungs-Algorithmen

cards bietet zwei Wiederholungs-Algorithmen: Leitner und Woźniak. Der Wiederholungs-Algorithmus von Leitner wird für die Vergabe von Bonuspunkten benutzt und ist serverseitig gesteuert. Der SuperMemo Wiederholungs-Algorithmus von Woźniak bietet momentan keine Bonuspunkte Vergabe. Der Benutzer kann sich durch E-Mails und Web-Push Notifications [TH17a] über neu vorgelegte Karten benachrichtigten lassen. Diese Option ist Standardmäßig deaktiviert und muss in den Profileinstellungen aktiviert werden.

Für beide Wiederholungs-Algorithmen wird dem Benutzer ein Pomodoro-Timer angeboten, welchen er selbständig konfigurieren kann. In einer Bonusphase wird die Lernzeit vom Dozenten bestimmt.

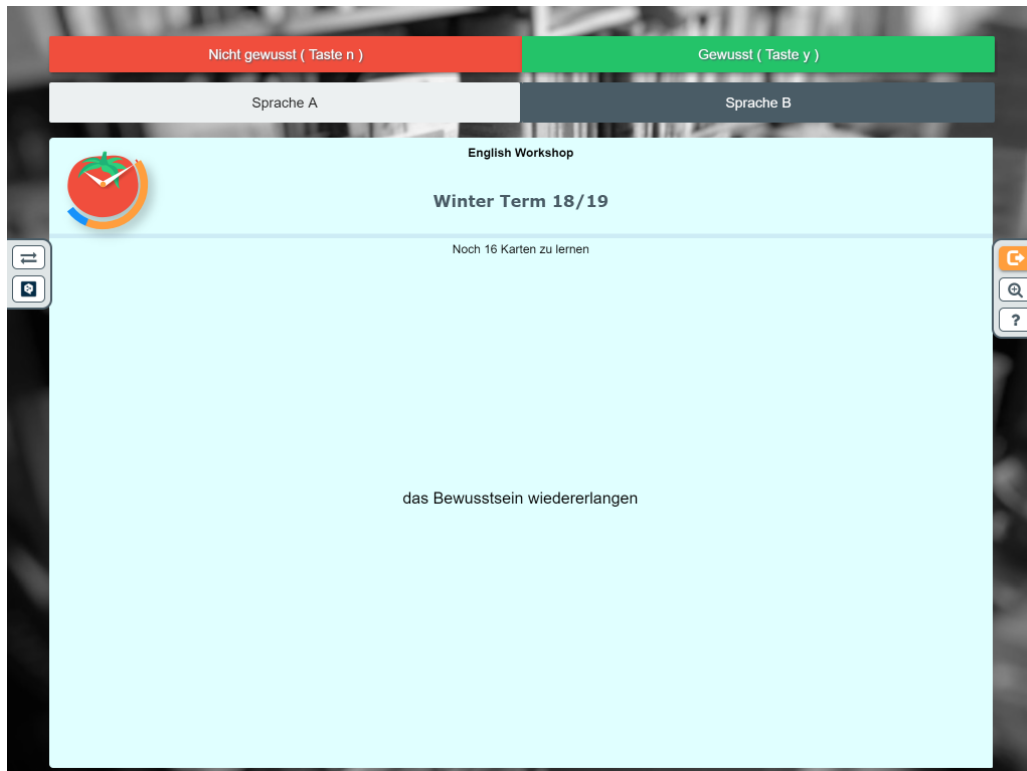


Abbildung 3: Leitner- Wiederholungs-Algorithmus von cards

2.10 Pomodoro-Timer

Der Benutzer kann für beide Wiederholungs-Algorithmen einen Pomodoro-Timer benutzen, um die Zeit für das Lernen besser zu managen. Hierbei wird die Lernzeit in mehrere Pomodoro-Einheiten zerlegt. Der Timer bietet folgende Einstellungsmöglichkeiten:

Anzahl der Pomodori am Tag, Arbeitszeit und Pause innerhalb einer Pomodoro. Wenn der Benutzer in keiner Bonusphase eingeschrieben ist, kann er seine Pomodoro Lernzeit selbst definieren. Bei aktiven Bonusphasen wird die Lernzeit vom Dozenten bestimmt. Der Benutzer muss bei dieser Situation die Lernzeit abarbeiten, damit seine aktiven Lernkarten in ein höheres Fach wandern können.

Bricht der Benutzer vor dem Ablauf der Bonus-Pomodori die Lerneinheit ab, wird sein Lernstand nicht aktualisiert.

2.11 Benutzer-Rollen und Kartei-Zugänge

cards bietet folgende Benutzerprofile:

- **Super-Admin:** Hat Zugriff auf das Backend und uneingeschränkte Rechte im System. Er kann alle Inhalte und Benutzer verwalten.
- **Admin:** Ein Super-Admin mit eingeschränkten Rechten. Er kann keine neuen Admins anlegen oder die Profile von Super-Admins verwalten.
- **Dozent:** Kann THM-Karteien erstellen und Pro-Karteien freischalten. Zusätzlich hat der Dozent Zugriff auf alle Karteien im System
- **Pro:** Verfügbar durch Abschließen eines Abonnement. Ein Pro Benutzer hat Zugriff auf alle Karteien und kann selbst Pro Karteien anlegen
- **THM:** Benutzer die sich mit einem CAS-Login anmelden bekommen automatisch dieses Profil. Es gestattet Ihnen kostenlosen Zugriff auf THM-Karteien
- **Standard:** Benutzer die sich mit einem Google, Twitter oder Facebook-Login anmelden erhalten dieses Profil. Es gestattet Ihnen kostenlosen Zugriff auf Frei-Karteien.
- **FirstLogin:** Ein Profil, das generiert wird, wenn der Benutzer sich zum ersten Mal anmeldet. Beim Abschließen oder Abbrechen der Registrationen wird dieser Status entfernt.
- **Blocked:** Blockierte Benutzer haben keinen Zugriff auf jegliche Funktionen der cards Applikation.

		Standard	THM	Pro	Dozent	Admin	Super-Admin
Privat	anlegen und lernen	X	X	X	X	X	X
Frei	anlegen und lernen	X	X	X	X	X	X
THM	lernen anlegen	(X)	X	X	X	X	X
					X	X	X
Pro	lernen anlegen prüfen	(X)	(X)	X	X	X	X
				X		X	X
					X	X	X

Tabelle 2: Benutzerrollen für cards

Mit (x) gekennzeichnete Flächen setzen voraus, dass der Benutzer die Kartei durch einen Kauf erworben hat.


2.12 Backend

Über das Backend kann der Super-Admin oder Admin die Benutzer und Studiengänge verwalten, eine Gesamtstatistik des Leitner-Lernstandes einsehen, Benachrichtigungen oder API-Zugänge verwalten und die E-Mail Benachrichtigungen aktivieren / deaktivieren. Das Backend benutzt noch das cards 3.0 Design und wird mit einer späteren Version überarbeitet. Ein Fokus für das neue Backend ist die Listenansicht für Benutzer mit Bootstrap zu stylen [OT16], um diese auch über Mobilgeräte verwalten zu können.



3 Vergleich mit anderen Lernkartei-Apps

3.1 Anki

Anki ist ein Karteikarten-Lernsystem [EL18], welches das Erinnern erleichtert. Es dient zum wiederholten Lernen von unterschiedlichen Themen-Gebieten, unter anderem Sprachen, Medizin und Jura Klausuren oder Gedichte. Die Software ist Open-Source und für Windows, Mac, Linux oder iPhone/Android verfügbar. Anki wird in 49 Sprachen angeboten darunter Deutsch und Englisch. Die Software lässt sich mit Third-Party Plugins nach Belieben erweitern.

Die App erlaubt das Anlegen von mehreren Benutzern und kann Karteien mit bis zu 100.000+ Karten verwalten. Karteien lassen sich miteinander verschachteln, welches dem Repetitorium System von cards ähnelt. Eine Karte an sich besitzt nur eine Vorderseite und Rückseite, der Inhalt wird jedoch durch ein Template generiert, welches dem Benutzer erlaubt aufwändigere Karten zu erstellen. Es werden MathJax und Latex unterstützt mit der Einschränkung, dass beide Formate in eine Bilddatei umgewandelt werden. Die Suche ist auf die Vorderseite, Rückseite oder dem Schlagwort einer Karte eingeschränkt. Schlagwörter können pro Karte angelegt werden und sind nur von Relevanz für die Suchfunktion.

Für das Erstellen von Vokabel Karten wird keine Übersetzung zur Verfügung gestellt. Der Benutzer muss hierbei einen Übersetzungsdienst aufrufen, falls er seine Vokabeln automatisch übersetzen lassen möchte.

Wie bei cards ist auch in Anki das Importieren und exportieren von Karten möglich. Es erlaubt zudem das Synchronisieren der Daten zwischen mehreren Geräten, diese Funktion ist bei cards derzeit nicht nötig, da die Daten Online gespeichert werden.

Der Lernmodus hat ein Tagespensum und die Antwortoptionen sind auf 4 Möglichkeiten beschränkt:

- **Normal:** Legt die Karte für 10 Minuten zurück.
- **Schwer:** Die Karte wird erst wieder in 3 Tagen vorgelegt.
- **Gut:** Die Karte wird erst wieder in 12 Tagen vorgelegt.
- **Einfach:** Die Karte wird erst wieder nach einem Monat vorgelegt.

Oft nicht gewusste Karten (Schwer) werden ab einer bestimmten Anzahl an "Nicht Gewusst" Antworten als Leeches markiert und können vom Benutzer aus dem Lernmodus entfernt werden. Der Benutzer kann seine Statistiken über ein spezielles Fenster einsehen. Die Statistiken werden mit Balkendiagrammen dargestellt und zeigen die "Anzahl der Wiederholungen", "Lerndauer", "Intervalle", "Welche Antwort wurde gewählt" und die Kartentypen als Kuchendiagramm. Vor dem Lernbeginn wird der Benutzer informiert, wie viele Karten es zu lernen gibt und wie viele davon Neu oder Wiederholungen sind.

Über das Menü "Optionen" können Änderungen für den Lernmodus vorgenommen werden. Hierbei können separate Einstellungen für neue Karten, Wiederholungen und Fehlschläge festgelegt werden. Standardmäßig werden die Einstellungen für alle Stapel übernommen, jedoch können durch das Anlegen von Optionen Gruppen diese für einzelne Stapel spezifiziert werden.

3.2 Anki-Benutzeroberfläche

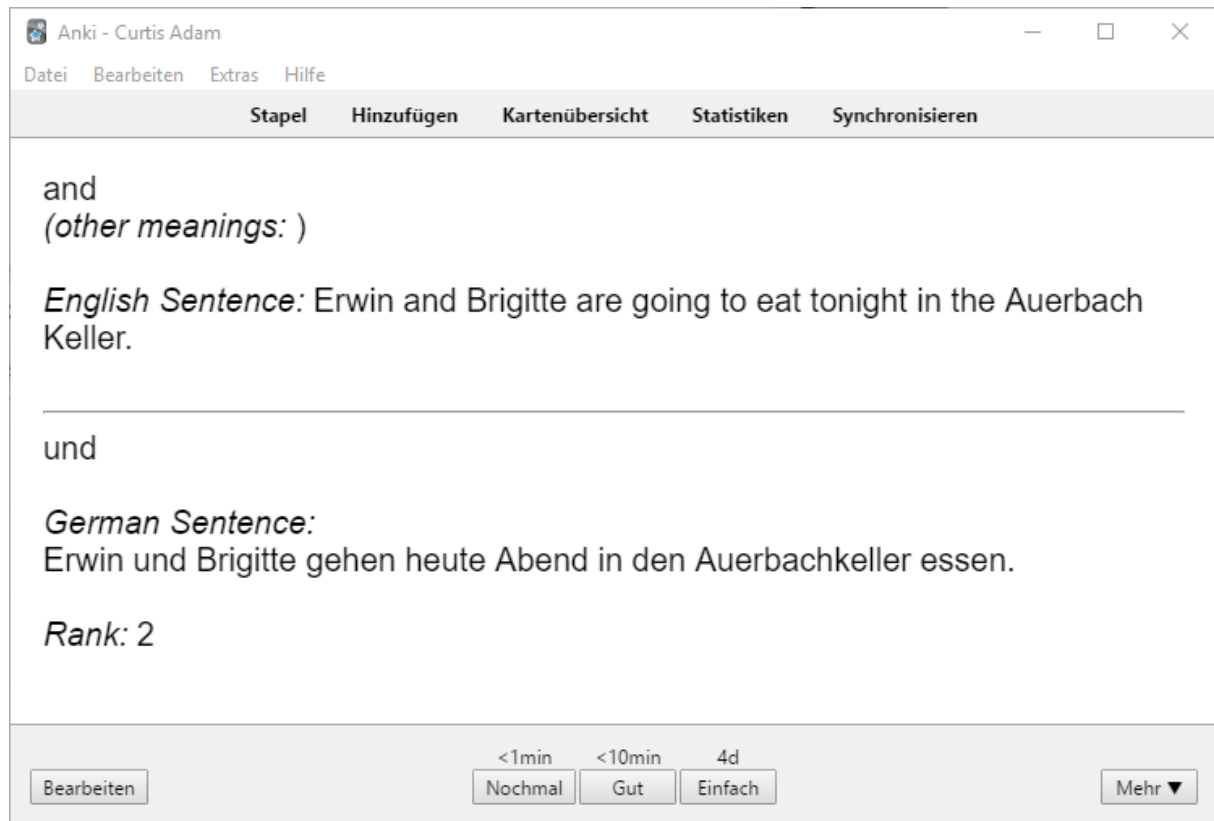


Abbildung 4: Lernmodus von Anki (Desktop)

Die Oberfläche erweckt den Anschein, dass die Funktionalität im Vordergrund liegt und die User-Experience eine Nebenrolle spielt. Die Benutzeroberfläche für Windows, Linux und Mac entspricht einer klassischen Desktopanwendung. Ein Großteil der Applikation ist in Menüleisten versteckt. Die Anzeige der Stapel und Karten ist zudem minimalistisch. Karten werden als Textfeld dargestellt und ähneln nicht dem Erscheinungsbild einer echten Lernkarte.

Die Antwortoptionen und Statistiken sind grau eingefärbte Knöpfe, welche über einem grauen Hintergrund liegen. Es kann zwar hilfreich sein, dass der Benutzer gleichzeitig die Frage und Antwort einsehen kann, jedoch ist es auf den ersten Blick nicht offensichtlich, zu welcher Seite der Inhalt gehört. Während des Lernen sollte zudem die Ansicht nicht mit unwichtigen Navigationselementen überladen werden.

Für die Bearbeitung von Karten existiert keine Vorschau. Der Bearbeiter muss die Karte speichern und danach öffnen, um seine Änderungen einzusehen.

3.3 Mnemosyne

Bei Mnemosyne handelt es sich um ein kostenloses Lernkarten Tool [BI18], welches den Lernprozess optimiert und als Forschungsprojekt über die Natur des Langzeitgedächtnisses dient. Es verfügt auch wie Anki über einen Windows, Mac, linux und Android Installer. Die Software ist Closed-Source, bietet jedoch die Installation von Third-Party Plugins und Scripts. Sie ist in 30 verschiedenen Sprachen erhältlich, darunter Deutsch und Englisch.

Der Benutzer kann standardmäßig zwischen drei verschiedenen Kartentypen wählen: "Front-Back", "Swapped" und Vokabeln. Der Inhalt der Karten kann aus Latex, HTML, Flash oder JavaScript bestehen. Zusätzlich können pro Karte Tags für die Sortierung und Notizen angelegt werden. Die Seiten einer Karte bestehen aus einem Fremdwort, Aussprache und Übersetzung. Es können Textausrichtung, Font und Farbe pro Karte definiert werden.

Wie auch bei Anki werden für Vokabeln keine Übersetzungen angeboten. Diese muss der Benutzer selbst einfügen.

Karten können nach verschiedenen Kriterien innerhalb einer Tabelle sortiert werden:

- Frage
- Antwort
- Tags
- Schwierigkeitsgrad
- Nächstes und letztes Datum für die Wiederholung

Tags und der Kartentyp können als Filter verwendet werden. Die für die Karte verwendete Hintergrundfarbe wird in der Tabelle angezeigt.

Es besteht eine große Auswahl an Mnemosyne Karteien, die sich importieren lassen. Neben dem Mnemosyne Format, können auch andere Formate wie das für die Software Anki, SuperMemo oder plain text (csv) importiert werden. Verschiedene Installationen lassen sich miteinander synchronisieren.

Karten können mit unterschiedlichen Lernsystemen gelernt werden. Darunter fallen die Priorisierung zwischen gewussten und nicht gewussten Antworten und das Lernen aller Karten ohne Sortierung. Für die Option der Priorisierung gibt es sechs verschiedene Antwortoptionen:

- **0:** Blackout
- **1:** Ähnlich 0 aber bereits vertrauter
- **2:** Erinnerung vielleicht noch morgen möglich
- **3:** Erinnerung sollte morgen möglich sein
- **4:** Erinnerung vielleicht noch in zwei Tagen möglich
- **5:** Erinnerung sollte noch in zwei Tagen möglich sein

Höhere Zahlen zeigen die Karte seltener an als niedrigere Zahlen.

Der Benutzer kann seine Lernstatistik wie bei Anki detailliert, mit Hilfe eines Balkendiagramms, darstellen lassen. Zu den Darstellungsoptionen zählen unter anderem: "Lernplan", "Gedächtnisgrad" und "Bewertung".

3.4 Mnemosyne-Benutzeroberfläche

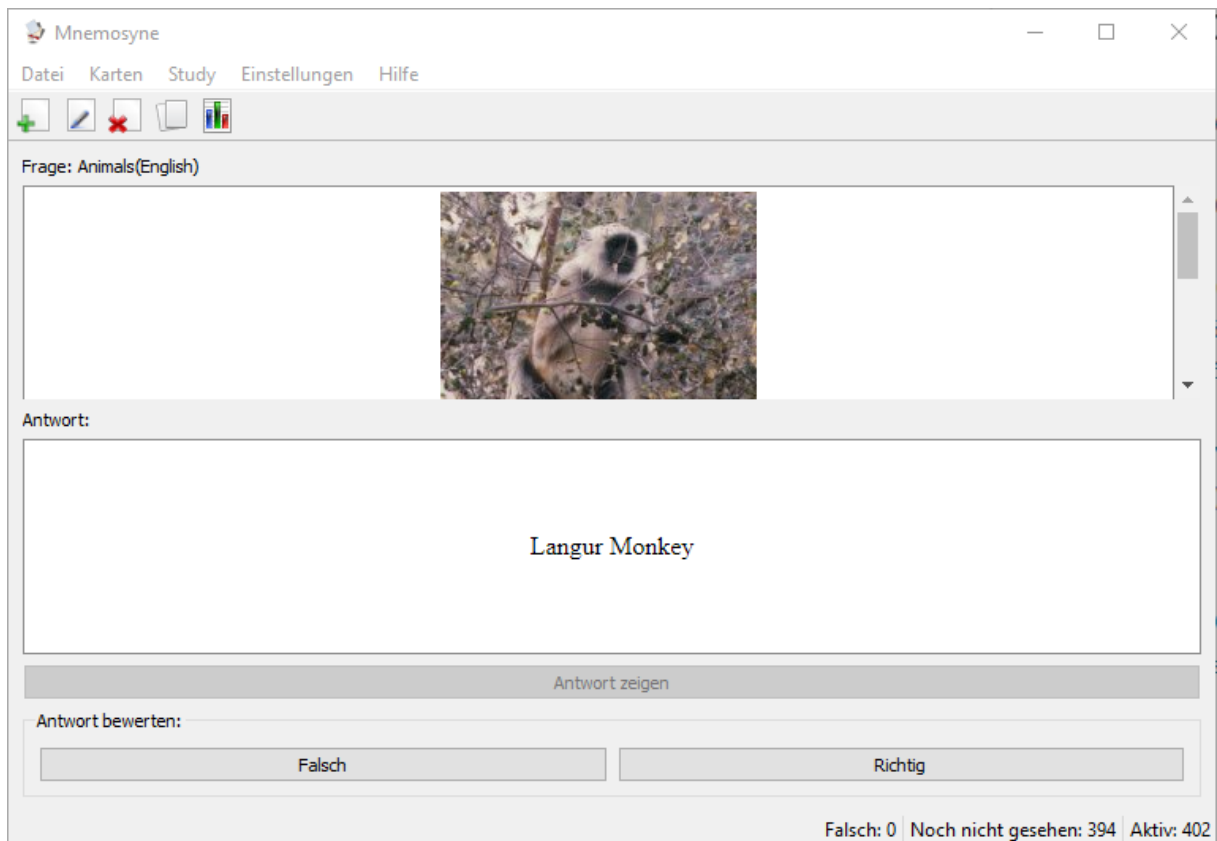


Abbildung 5: Lernmodus von Mnemosyne

Die Oberfläche ist auch wie bei Anki mehr auf Funktionalität statt User-Experience ausgelegt. Die Windows Version benutzt ein Standard 90'iger Desktop-App Design, welches den Großteil des Programms in Dropdown-Menüs versteckt. Im Gegensatz zu Anki, welches alle Knöpfe nur mit Text auszeichnete, hebt Mnemosyne diese grafisch hervor.

Es gibt eine bessere Trennung zwischen den Seiten und es wird mit einem Label hervorgehoben, um welchen Inhalt es sich handelt. Im rechten unteren Teil der Anwendung kann die Statistik der derzeit aktive Lernpensum eingesehen werden. Hierbei wird die Anzahl an nicht gewussten Fragen, die noch verbleibenden Fragen und die Gesamtanzahl der Karten für das Lernpensum hervorgehoben.

Für das Bearbeiten der Karten existiert keine direkte Vorschau. Der Bearbeiter muss hierfür auf den Knopf "Vorschau" klicken, um eine Vorschau für den derzeitigen Inhalt zu generieren.

3.5 Quizlet

Bei Quizlet handelt es sich derzeit um die weltweit größte Online-Lerngemeinschaft für Schüler und Lehrer [SUT18], welche in 18 verschiedenen Sprachen zur Verfügung steht. Der Quellcode für diese Software liegt nicht offen und es wird eine Registrierung für die Benutzung vorausgesetzt. Als Logins stehen Google, Facebook und eine interne Methode zur Verfügung. Die Kontoart kann als Schüler oder Lehrer gesetzt werden.

Es können Karteien, genannt Sets, angelegt werden, welche mehrere Karten beinhalten. Der Aufbau einer Karte besteht meistens aus einer Vorder und Rückseite, welche entweder aus reinem Text besteht oder ein Bild beinhaltet. Die Koordinaten eines Bilds können mit Areas markiert werden, welche durch ein Mouse Hover einen Tooltip anzeigen. Für das Schreiben der Vokabeln wird keine Übersetzung angeboten, jedoch kann die Aussprache festgelegt werden.

Zur Verfügung stehen mehrere Lernmodi: Ein normales Lernen der Vorder und Rückseite, das Schreiben der Antwort, ausschreiben was man hört (benötigt eine Soundausgabe), das Zuordnen von den Antworten per Drag and Drop, ein Lückentext oder auf spielerische Art durch Abschießen von herabfallenden Kometen durch eintippen der richtigen Antwort "Schwerkraft".

Es stehen über 271 Millionen Sets zur Verfügung. Diese können in Ordner und Kurs kombiniert werden. Zusätzlich besteht die Möglichkeit Freunde zu eigenen Kursen einzuladen. Um mehrere Features zu benutzen, muss der Benutzer ein Abonnement abschließen. Quizlet bietet hierzu folgende Optionen an:

Quizlet Plus: 1,67 € im Monat / 19,99 € im Jahr

- Offline Funktion
- Nachtmodus
- Hochladen von eigenen Bildern
- Unbegrenzte Anzahl an Areas definieren für Bildern
- Eigene Sprachaufzeichnungen erstellen

Quizlet Go: 1 € im Monat / 11,99 € im Jahr

- Offline lernen
- Nachtmodus
- Lernen ohne Werbung

Innerhalb der Benutzereinstellungen befinden sich Optionen für die Benachrichtigungen der Lernerinnerungen. Hier kann der Benutzer die Uhrzeit inklusive der Zeitzone festlegen, in der die Benachrichtigungen per E-Mail versendet werden. Neben der Lernerinnerung kann das Benachrichtigungssystem für folgende Events genutzt werden:

- Quizlet Newsletter
- Werbe E-Mails
- "Zuordnen" oder "Schwerkraft" Highscore wurde geschlagen

Es können Einstellungen zur Privatsphäre vorgenommen werden, oder man kann das Konto vollständig von Quizlet entfernen. Dieser letzte Schritt kann nach der Ausführung nicht mehr rückgängig gemacht werden.

3.6 Quizlet-Benutzeroberfläche

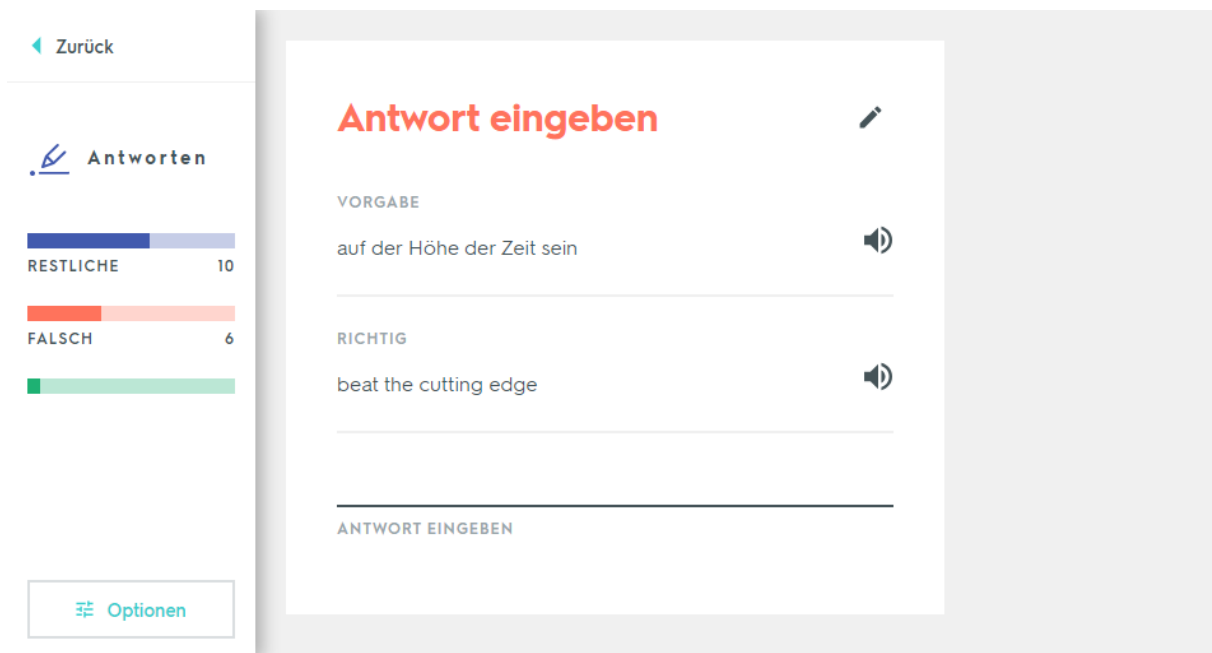



Abbildung 6: Lernmodus von Quizlet (Antworten)

Das Interface der Web-App wurde für Mobilgeräte optimiert (Mobile-First). Die Oberfläche passt sich automatisch der Bildschirmgröße an und verlagert auf kleineren Bildschirmen, wie z. B. auf Mobilgeräten, die Navigation in Ausklappbare Elemente. Für die Lernansicht werden in der linken Navigation die Statistiken der aktiven Sitzung angezeigt. Darunter zählen die noch zu lernenden Karten und die Anzahl an richtig und falsch beantworteter Fragen. Im Menüreiter "Optionen" lassen sich die Einstellungen für den aktiven Lerntyp vornehmen.

3.7 SuperMemo

SuperMemo ist wie Quizlet eine Closed-Source Web-App. Sie wird in 12 verschiedenen Sprachen angeboten [SUP18], welche Deutsch und Englisch beinhalten.

Die App beinhaltet Freie und Premium Kurse. Um an allen Premium Kursen teilnehmen zu können, muss der Benutzer entweder eine monatliche Gebühr von 9,90 € bezahlen, oder er kann einen uneingeschränkten Zugang auf einen einzelnen Kurs für 99,00 € erwerben. Es stehen über 550 Kurse zur Verfügung, wovon über 60% kostenlos sind.

Kurse sind ähnlich dem Aufbau von Karteien in cards. Sie bestehen aus einer Beschreibung und meist mehreren Karten. Die Karten selbst besitzen eine Vorder und Rückseite. Der Inhalt kann Text, Bilder oder Audio-Dateien beinhalten. Es ist zu beachten, dass eine Kartei maximal 50.0 MB umfassen darf. Dieser Wert wird dem Ersteller im Editor angezeigt. Es besteht die Möglichkeit, eine Karte mit einer Notiz zu versehen. Diese Notiz kann denselben Inhaltstypen wie der Rest der Karte besitzen und fällt unter das gleiche Limit.

Im Vergleich zu anderen Produkten bietet SuperMemo ein Wörterbuch und einen Übersetzungsdienst. Für das Wörterbuch wird ein selbst entwickeltes System verwendet, welches Audio-Dateien zur Aussprache und dazugehörige Beispielsätze anbietet. Für die Übersetzungen wird Google-Translate verwendet.

Für den Lernmodus sind die Antwortoptionen auf 3 Knöpfe beschränkt:

- Gewusst (Grün)
- Fast Gewusst (Gelb)
- Nicht gewusst (Rot)

Die ausgewählten Antworten werden innerhalb eines Fortschrittsbalkens oberhalb der Antwortoptionen farblich gekennzeichnet.

Über einen Kalender kann der Benutzer seinen Lernstatus einsehen. Dabei zeigt der Kalender für jeden Tag die gelernten Karten und Wiederholungen, die damit verbrachte Zeit und die Anzahl der demnächst vorgelegten Karten an. Zusätzlich kann pro Kurs das Tagespensum festgelegt werden. Hierbei berechnet SuperMemo die durchschnittliche Lernzeit pro Tag und die Anzahl an Tagen, um diesen Kurs abzuschließen. Die Statistiken enthalten dieselben Daten wie der Kalender. Es können darüber die Gesamtstatistiken für einen Tag, Woche, Monat oder der gesamte Verlauf angezeigt werden.

Kurse können nach vordefinierten Kriterien gefiltert werden, darunter zählen:

- Kategorie
- Sprache der Anweisungen
- Kostenlos oder kostenpflichtig
- Autor

Neben den Vordefinierten Kriterien gibt es zusätzlich die Option die Kurse mit einem Suchfeld nach dem Inhalt zu filtern.

3.8 SuperMemo-Benutzeroberfläche

Die Benutzeroberfläche wurde mit dem Ansatz Mobile First entwickelt. Es werden in der Ansicht für Mobilgeräte nur die Kurse angezeigt. Die Lernstatistik oder das Erstellen und bearbeiten werden innerhalb einer Ausklappbaren Navigation verborgen.

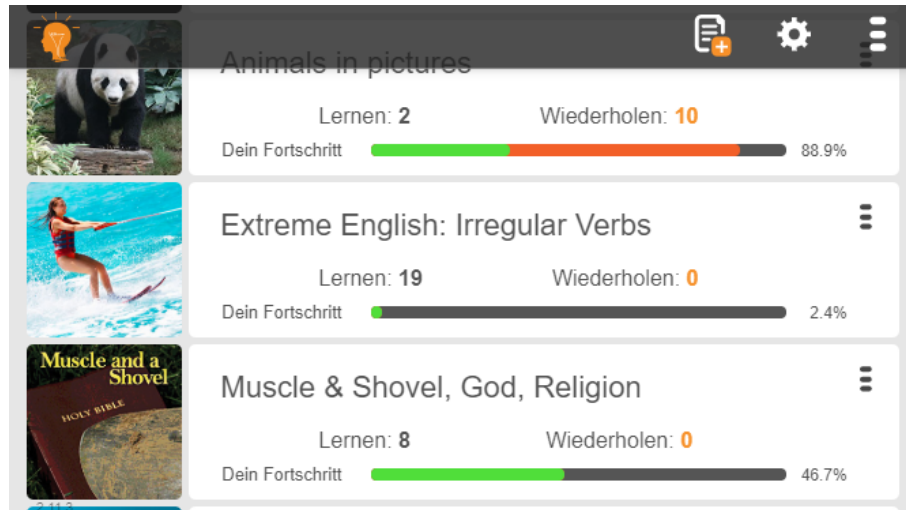


Abbildung 7: Lernpensum von SuperMemo

Das Lernpensum benutzt eine Listenansicht für die einzelnen Kurse. Jeder Kurs zeigt einen Fortschrittsbalken und die Anzahl an Karten, die gelernt oder wiederholt werden müssen.

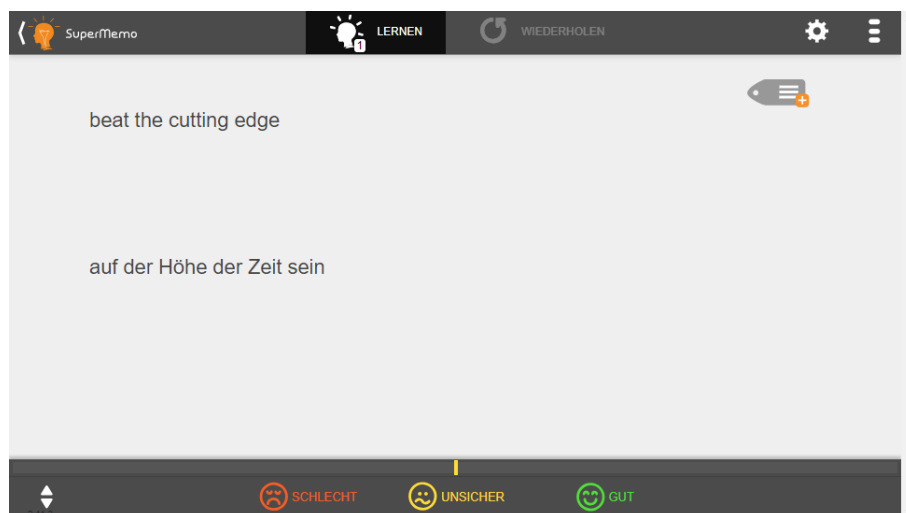


Abbildung 8: Lernmodus von SuperMemo

Der Lernmodus zeigt nur die für das Lernen relevanten Navigationselemente. Die Antwortoptionen werden Visuell hervorgehoben. Der Verlauf der bereits beantworteten Karten wird in einem Fortschrittsbalken über den Antwortoptionen farblich markiert.

3.9 Zusammenfassung

Eine Zusammenfassung der oben genannten Applikation in Tabellenform:


Features	●cards	Anki	Mnemosyne	Quizlet	SuperMemo
OS	Web-App	Windows, Linux, Mac Android, iPhone	Windows, Linux, Mac Android, iPhone	Web-App	Web-App
Open Source	X	X	X		
Mobile First	X			X	X
Öffentliche Inhalte	<100 Karteien		<400 Karteien		574 Kurse, davon 388 kostenlos
Karteitypen / Kartentypen	18	3	3	1	1
Sprachen	1	49	30	< 18	12
Inhalt der Karte	Markdeep, MathJax	HTML, Template, Latex, MathJax	HTML, flash, JavaScript	Plain Text, Bild	Plain Text, Bilder, Video
Erweiterungen		X	X		
Übersetzungsdienste	DeepL BEOLINGUS				Google-Translate
Lernoptionen	Leitner, Woźniak	Woźniak	Woźniak	Verschiedene*	Woźniak
Kostenpflichtige Inhalte	Abonnement, Karteien	Keine	Keine	Abonnement	Kurse

Tabelle 3: Zusammenfassung für Vergleich der Lernkartei-Apps

3.9.1 Fazit

●cards bietet im Vergleich zur Konkurrenz nur eine Sprache. Sobald die Entwicklung von Version 4.0 abgeschlossen ist, werden mehrere Sprachen zur Verfügung gestellt. Es gibt zudem einige nützliche Features, die in anderen Apps benutzt werden, jedoch in ●cards noch nicht vorhanden sind. Darunter zählt zu einem das Markieren von Koordinaten innerhalb des Kartenbereiches mit einem Tooltip sowie ein Verlauf des Lernstands über die Anzeige eines Kalenders. Auch die Möglichkeit Antworten zu definieren sollte dem regelmäßigen lernen und vor allem der Vergabe von Bonuspunkten von großem Nutzen sein.

4 Getting started

Für die Entwicklung von cards wird die Benutzung einer Linux distribution vorgeschlagen. Windows Betriebssysteme haben größere Probleme mit **Meteor** oder den Chimp Acceptance-Tests [BT18].


4.0.1 Hardwareanforderungen

- 4 GB RAM
- 10 GB freier Speicherplatz, 20 GB mit einer WebStorm IDE [JB18] Installation
- Ubuntu 18.10 oder neuer



4.0.2 Installation

1. Installiere **Meteor**
2. Installiere git mit dem Befehl `sudo apt install git`
3. Installiere python2 mit dem Befehl `sudo apt install python` , neue Ubuntu Versionen werden nur mit python 3 ausgeliefert
4. Installiere die g++ essential `sudo apt install g++ build-essential`
5. Vergewissere dich, dass du deinen SSH Schlüssel für dieses Projekt eingerichtet hast. Dies ist nur notwendig, wenn du dich aktiv an der Entwicklung des Projekts beteiligen möchtest
6. Erstelle eine Kopie des Repository [TH15] mit `git clone git@git.thm.de:arsnova/cards.git`
7. Wechsel in das neu erstellte Verzeichnis mit dem Namen "cards" und führe in diesem den Befehl `meteor npm install` aus, um die benötigten Node.js packages zu installieren
8. Setze in der Konfigurationsdatei `settings.json` oder `settings_debug.json` deinen CAS-Login als admin, falls du diesen mit Admin-Rechten ausstatten willst

4.0.3 Hinweis zu **Meteor** updates

Das Aktualisieren der cards Anwendung sollte den Projektbesitzern überlassen werden. Neuere Versionen von **Meteor** bringen meist Änderungen mit sich, welche eine Überarbeitung der derzeitigen Implementierung benötigen oder das Aktualisieren der Datenbank voraussetzen.

4.0.4 Datenbank aktualisieren

1. Vergewissere dich, dass du folgende Kriterien erfüllst, bevor du die Datenbank aktualisierst:
 - Du bist auf dem staging branch `git checkout staging`
 - Dein staging branch ist auf dem aktuellsten Stand: `git pull`
 - Die cards läuft Anwendung im Hintergrund
 - Dein Terminal befindet sich innerhalb der Anwendung cards Ordner
2. Installiere die **MongoDB**-Tools mit `sudo apt-get install mongo-tools`
3. Setze dein Repository [TH15] auf den letzten Stand der alten Version:
 - **Meteor** 1.6: `git checkout 3a1751a2838ad595eda7d3ccd9e9191a3bf32271`
4. Erstelle einen Datenbankdump:
`mongodump -h "localhost-port "3001d "meteo /cardsBackup/`
Vergewissere dich, dass der Datenbankdump einen Ordner und Dateien erzeugt hat
5. Beende **Meteor**

6. Setze die Datenbank deiner **Meteor** Installation zurück `meteor reset`
7. Setze deinen staging branch auf die aktuellste Version `git checkout staging`
8. Aktualisiere die Node.js packages `meteor npm install`
9. Starte **Meteor**
10. Benutze den Datenbankdump um deine Daten in der aktualisierten **Meteor** Installation wiederherzustellen:
`mongorestore --drop -h "localhost-port "3001d "meteor-/cardsBackup/meteor`

4.0.5 Die Applikation starten

Um die Applikation zu starten, wird vorausgesetzt, dass du dich im **cards** Verzeichnis befindest. Folgende Befehle sind für die lokale Entwicklung relevant:

- Für die Entwicklung: `meteor --settings settings_debug.json`
- Für die Produktion: `meteor --settings settings.json`

Die Applikation lässt sich nach erfolgreichem starten über `http://localhost:3000` aufrufen.

4.1 Sonstige Installationen

4.1.1 Testdatenbank laden

1. Vergewissere dich, dass du die **MongoDB** Community Tools installiert hast (4.0 oder neuer): <https://docs.mongodb.com/manual/administration/install-community/>
2. Starte den Server mit Development-Settings, damit der Backdoor-Login zur Verfügung steht
3. Um die Daten der Test-Datenbank zu laden, musst du innerhalb des Hauptordners der **cards** Applikation folgenden Befehl ausführen: `./tests/loadTestDatabase.sh`

4.1.2 Google, Facebook und Twitter

Für die verschiedenen Login-Funktionen werden API-Schlüssel benötigt. Diese müssen innerhalb der verwendeten settings.json hinterlegt werden:

- Google+ api-Schlüssel
- Facebook api-Schlüsse
- Twitter api-Schlüsse


4.1.3 Braintree setup, für PayPal

1. Erstelle einen Account bei Braintree, ein mobile und web payment System [PA18]
2. Füge den API-Schlüssel in `settings.json` (`BT_MERCHANT_ID`) und (`BT_PUBLIC_KEY`) hinzu

4.1.4 Web-Push-Notifikationen

1. Erstelle einen Account bei Firebase, eine mobile und web application Plattform [GO18]
2. Hole dir den FCM API-Schlüssel über **Project settings -> cloud messaging -> server key**
3. Füge den API-Schlüssel in `settings.json` (`FCM_API_KEY`) hinzu


4.2 WebStorm

Für die Entwicklung von cards wird die IDE WebStorm von JetBrains [JB18] empfohlen. Das ARSnova-Team stellt hierfür Dateien bereit, um über dieses Programm alle für den Code-Style relevanten Pipeline-Tests lokal durchzuführen.

4.2.1 Vorbereitung

1. Installiere WebStorm (Kostenlos für Studenten der THM): <https://www.jetbrains.com/webstorm/>
2. Setze das Zeilenende für git auf LF **Dies Betrifft nur Windows Benutzer**: <https://help.github.com/articles/dealing-with-line-endings/>
3. Installiere die npm packages innerhalb der App mit dem Befehl `meteor npm install`

4.2.2 Einrichtung von Webstorm für cards

1. Importiere die von uns vorbereitete `webstorm_settings.jar` unter **File -> Import Settings...** Du findest die .jar Datei im Hauptverzeichniss von cards.
2. Überprüfe, dass die Konfiguration von den Einstellungen des Screenshots entspricht. Die Versionsunterschiede der **Node.js** Version können hierbei ignoriert werden.

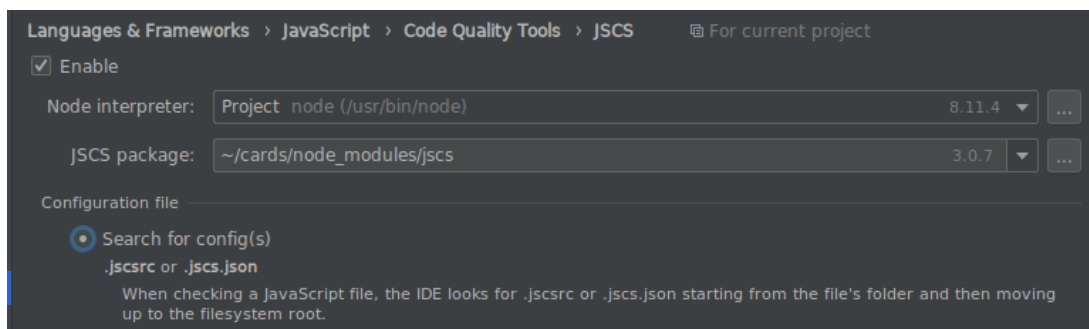


Abbildung 9: WebStorm: JSCS Einstellungen

3. Für **JSHint** muss "Use config file" mit dem Wert **Default** gesetzt sein.

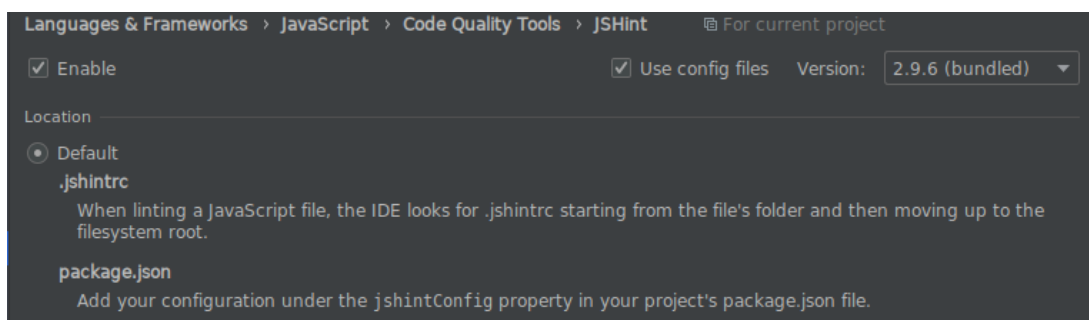



Abbildung 10: WebStorm: JSHint Einstellungen

4. Verweise WebStorm auf die von cards mitgelieferte und verborgene **.gulpfile.js** Datei.

4.2.3 Codestyle überprüfen

Wähle die Option **default codeCheck** über **Tools -> Run Gulp/Grunt/npm Task -> Gulp**
Webstorm überprüft dann anschließend den Code Style des Projekts:

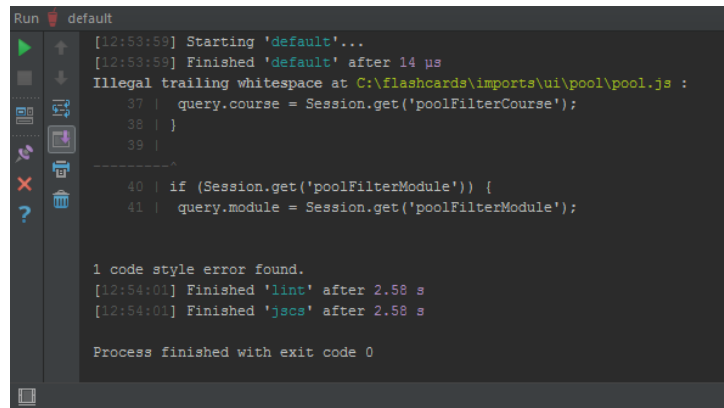


Abbildung 11: Gulp: Test fehlgeschlagen

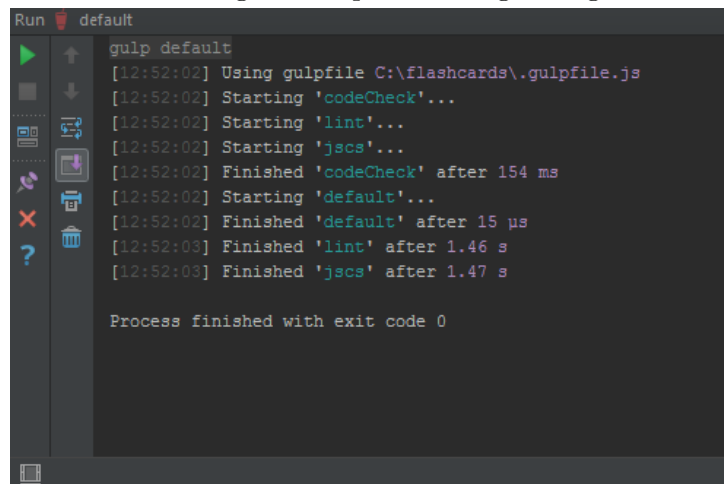


Abbildung 12: Gulp: Test erfolgreich

4.3 Chimp

4.3.1 Installation für Linux

1. Vergewissere dich, dass die **Node.js** packages des Projekts installiert sind. Führe dazu den Befehl `meteor npm install` innerhalb des Hauptverzeichnisses des Projekts aus
2. Für die Datenbank-Tests werden die **MongoDB**-Tools **mongoimport** und **mongoexport** benötigt. Diese Tools kannst du mit folgendem Befehl auf deinem System installieren: `sudo apt-get install mongo-tools`
3. Installiere die neueste Version von **Node.js** 8 [NO18b], 9 oder neuer sind derzeit nicht kompatibel mit chimp [BT18]: <https://nodejs.org/en/download/package-manager>
4. Installiere Java 8. Neuere Versionen funktionieren nicht mit chimp
5. Installiere Chrome oder Chromium

4.4 Guidelines

4.4.1 i18n

Texte werden in den i18n [SP18] Dateien hinterlegt. Diese befinden sich im Verzeichnis `/i18n/`. Der Inhalt wird Sinnhaftig gruppiert:

```
1 "connection": {
2   "title": "Um __lastAppTitle__ online nutzen zu koennen, ist eine WebSocket-Verbindung zum Server
        erforderlich.",
3   "websocket": "WebSocket-Check",
4   "connected": "okay",
5   "disconnected": "keine Verbindung"
6 },
```

Listing 1: i18n Template für den WebSocket-Verbindung Status

Wörter die mit zwei Unterstrichen beginnen und aufhören, werden als Platzhalter interpretiert. Als Beispiel folgt der Aufruf des Felds **Title** über ein HTML-Template und durch JavaScript:

```
1 //Template Variante (String)
2 {{_ 'connection.title' lastAppTitle=cards}}
3 //Template Variante (HTML)
4 {{{_ 'connection.title' lastAppTitle=cards}}}
5 //JavaScript Variante
6 TAPi18n.__('connection.title', {lastAppTitle: cards})
```

Listing 2: Aufruf eines i18n Strings innerhalb eines Templates und mit JavaScript

Der Inhalt für Bereiche wie das Impressum ist zu groß, um diesen Effizient in der i18n Datei zu hinterlegen. Für solche Fälle wird ein **Meteor – Block Helper** benutzt, um die entsprechende Übersetzung zu laden:

```
1 {{#if isActiveLanguage 'de'}}
2 //DEUTSCHER INHALT
3 {{else}}
4 //ENGLISCHER INHALT
5 {{/if}}
```

Listing 3: **Meteor – Block Helper** für Übersetzungen

4.4.2 **Meteor – Block Helper** und JavaScript Hilfsklassen

Um die Code-Redundanz zu reduzieren und die Wartbarkeit zu vereinfachen, ist es ratsam, öfters benötigte **Meteor – Block Helper** oder JavaScript Funktionen auszulagern. **Meteor – Block Helper** können unter `/imports/startup/client/registerhelper.js` registriert werden. Dies bringt folgende Vorteile mit sich:

- Der Inhalt des **Meteor – Block Helper** befindet sich an einer Stelle, was das Aktualisieren vereinfacht und Fehler über die gesamte Applikation minimiert.
- Neue Templates haben direkten Zugriff und benötigen keinen speziellen Aufruf.

Für JavaScript Funktionen ist es Hilfreich diese in Klassen auszulagern. Diese Klassen werden in `/imports/api/` abgelegt. Es wird vorgeschlagen folgenden Aufbau zu verwenden:

```


1 let editorFullscreenActive = false;
2
3 export let CardVisuals = class CardVisuals {
4
5     static isEditorFullscreen () {
6         return editorFullscreenActive;
7     }
8 };

```

Listing 4: Aufbau der Hilfsklasse für Karten-Effekte

Nachdem die Klasse mit dem Befehl `import {CardVisuals} from ...` importiert wurde, kann die Funktion `isEditorFullscreen` mit dem Befehl `CardVisuals.isEditorFullscreen();` aufgerufen werden. WebStorm importiert in der Regel die dafür benötigte Klasse automatisch.

4.4.3 Theme-Switcher

Die Farben von cards werden mit dem Theme-Switcher definiert. Dieser besteht aus folgenden Dateien:

- Die Zuordnung der Farbe an ein HTML-Tag: `/client/themeSwitcher.scss`
- Die Definition der Farben: `/client/themes/`

Als beispiel wird eine Farbe für den facebook login button definiert, hierfür werden 4 neue Variablen angelegt:

```

1 "login_facebook_background": $primary,
2 "login_facebook_background_hover": $button_background_hover,
3 "login_facebook_text": white,
4 "login_facebook_text_hover": $button_text_hover,

```

Listing 5: Default Theme Variablen für den Facebook-Login Button

Es werden für Buttons immer einer Hintergrundfarbe und eine für den Inhalt (Text und Icons) gesetzt. Zusätzlich werden Varianten für das Mouse Hover Event angelegt. Die Variablen verwendet das snake_case Format (Verbinden der einzelnen Wörter mit einem Unterstrich). Falls bei dem Mouse Hover Event die Default-Einstellung für Button Events verwendet werden soll, werden die Variablen `$button_background_hover` und `$button_text_hover` übergeben.

```

1 $login_facebook_background: map-get($map, "login_facebook_background");
2 $login_facebook_background_hover: map-get($map, "login_facebook_background_hover");
3 $login_facebook_text: map-get($map, "login_facebook_text");
4 $login_facebook_text_hover: map-get($map, "login_facebook_text_hover");

```

Listing 6: Mapping der neuen Farben innerhalb von themeSwitcher.scss

Die Variablen aus den einzelnen Themes werden am Anfang der themeSwitcher.scss Datei aufgerufen und an eine neue Variable innerhalb des Theme-Switcher zugewiesen. Diese neuen Variablen sollte am besten dieselbe Bezeichnung wie aus den Theme Dateien verwenden. Zum Schluss kann die Variable an das jeweilige HTML-Tag zugeordnet werden, hierbei ist zu beachten, dass das **!important** Attribut gesetzt werden muss, um ein Überschreiben durch Third Party packages zu verhindern.

4.4.4 Merge-Request und Commits

Merge-Requests werden immer über Branches erstellt, die automatisch durch ein Issue erzeugt wurden. Diese Branches werden aufgrund ihrer Namensgebung automatisch mit einem Review-Server verknüpft, welcher es vereinfacht, die Änderungen Online einzusehen.

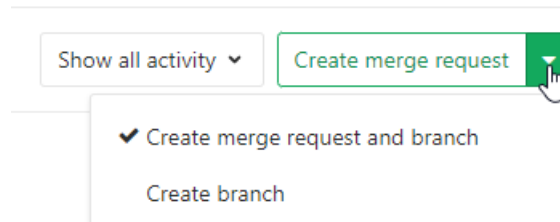


Abbildung 13: GitLab: Branch und Merge-Request anlegen

Solange die Bearbeitung des neuen Features nicht vollendet wurde, wird der Name des Merge-Requests mit dem Wort "WIP:" angeführt, dies übernimmt Git-Lab beim Erstellen automatisch. Aus Sicherheitsgründen ist es ratsam, Commits die aus Zeitmangel nicht fertiggestellt werden konnten, mit einem WIP: Tag zu versehen und diese auf dem Server zu hinterlegen. Nachdem dieses Commit zu einem späteren Zeitpunkt fertiggestellt wird, kann die Commit Message durch rebasen korrigiert werden.

Ein Merge-Request wird automatisch abgelehnt, solange er nicht alle stages der Pipeline, mit Ausnahme der Review-App, erfolgreich durchläuft. Du kannst den Status der Pipeline über die Seite des Merge-Requests überprüfen.

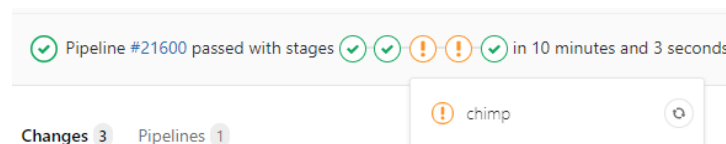


Abbildung 14: GitLab: Pipeline Status für Merge-Request

Wenn das neue Feature vollendet wurde, wird das Wort "WIP:" aus dem Merge-Request entfernt und ein Reviewer über das Assignee Auswahlfeld hinzugefügt. Zusätzlich sollte die Checkbox für das Löschen des Branches nach Akzeptieren des Merge-Request aktiviert werden.

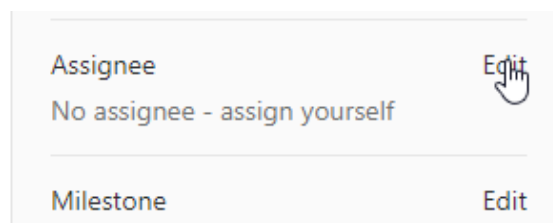


Abbildung 15: GitLab: Assignee für Merge-Request setzen





Als Assignee sollte immer ein Entwickler mit Maintaner oder Owner Berechtigungen gesetzt werden, solange nichts anderes verlangt wird. Setze niemals dich selbst als Assignee für das Merge-Request.

5 Aufbau

5.1 Struktur

Im Folgenden die Ordnerstruktur für 🍷cards. Der Hauptteil der Programmierung befindet sich im Ordner `/imports/`. Die Aufteilung entspricht der **Special directory conventions** [GL17].

Legende

-  **C** # Client Dateien
-  **S** # Server Dateien
-  **CS** # Client und Server Dateien
-  **R** # Repository Dateien

🍷cards

-  `client` **C** # Client Ressourcen und Theme-Switcher
-  `i18n` **CS** # Die Übersetzungsdateien
-  `imports` **CS** # Geteilte Client und Server Ressourcen
 -  `api` **CS** # Hilfsklassen und `Meteor - Method's`
 -  `startup` **CS**
 -  `client` **C** # Service Worker und Routen für die einzelnen Ansichten
 -  `server` **S** # Initialisierung der Datenbank und Browser-Policy
 -  `ui` **C** # Templates und Events der einzelnen Ansichten von 🍷cards
-  `private` **S** # Newsletter Demo und The-Making-of 🍷cards Karteien
-  `public` **CS** # Icons Bilder und Videos
-  `server` **S** # Cronjob E-Mail Benachrichtigungen und Leitner-Algorithmus
-  `tests` **R** # Acceptance-Tests

5.2 Server-Initialisierung

Pfad	/imports/ui/server/initialize.js
------	----------------------------------

Innerhalb der Datei `initiliaze.js` werden wichtige Funktion für den Server-Start ausgeführt:

- Die Datenbank-Einträge werden der aktiven Spezifikation angepasst
- Die Primärschlüssel werden für **MongoDB** gesetzt mit Hilfe der Funktion **ensureIndex** [MO16]
- Der Inhalt für die Demo-Kartei wird angelegt
- Der Benutzer und Inhalt für das Testen der Notifikationen wird angelegt

5.3 Hilfe

Template-Pfad	/imports/ui/help/
Hilfsklasse	MainNavigation

Durch die **Meteor – Session helpFilter** wird der nicht benötigte Teil der Hilfe ausgeblendet. Der Inhalt der **Meteor – Session** wird innerhalb von `/imports/startui/client/route` gesetzt. Falls nach einer Kategorie gefiltert werden soll, wird ein entsprechender String an die **Meteor – Session** beim Aufruf der entsprechenden Route übergeben. Beim Zurücksetzen des Filters wird die **Meteor – Session** auf `undefined` gesetzt:

```
1 Router.route('/create', {
2   name: 'create',
3   template: 'filterIndex', \\ Benutzt das Wort filterIndex fuer den Filter der Hilfe
```

Listing 7: Beispiel: Filtern der Hilfe für Meine Kartein

Die gesetzten Filter werden in dem Template innerhalb des Verzeichnisses

`/imports/ui/help/content/helpContent` angewendet.

Das Aufrufen der Hilfe mit der F1-Taste wird durch die Hilfsklasse **MainNavigation** getätigt.

5.3.1 Template

```
cards/imports/ui/help
├── content # Der Inhalt für die Hilfe
│   └── categories # Die einzelnen Kategorien der Hilfe
└── modal # Das mit F1 erreichbare Modal
```

5.4 Pomodoro-Timer

Template-Pfad	/imports/ui/pomodoro/
Hilfsklasse	PomodoroTimer

Die Basis des Pomodoro-Timer wurde aus dem Code-Pen Projekt von John Wilkos [WI18] entnommen und in der Hilfsklasse **PomodoroTimer** hinterlegt. Die Templates befinden sich unter `/imports/ui/pomodoroTimer` .

Folgende Seiten verwenden derzeit einen Pomodoro-Timer:

- Präsentationsmodus
- Leitner und Woźniak Lernansicht
- Demo-Kartei
- Wortwolke

Die Grundeinstellungen des Formulars werden in einem Objekt hinterlegt, welches sich am Anfang der Datei `/imports/api/pomodoroTimer.js` befindet:

```
1 let defaultSettings = {
2   goal: 2, // Das Pomodori-Ziel fuer die Sitzung
3   work: { // Die Lernzeit einer Pomodoro-Einheit
4     length: 25, // Standardeinstellung in Minuten
5     max: 45, // Maximale Laenge in Minuten
6     min: 15, // Minimale Laenge in Minuten
7     step: 5 // Abstufung des Schiebereglers in Minuten
8   },
9   break: { // Die Pause einer Pomodoro-Einheit
10    length: 5, // Standardeinstellung in Minuten
11    max: 15, // Maximale Laenge in Minuten
12    min: 5, // Minimale Laenge in Minuten
13    step: 5 // Abstufung des Schiebereglers in Minuten
14  },
15  longBreak: { // Grosse Pause nach X Pomodori
16    goal: 4, // Anzahl der zu erreichenden Pomodori
17    length: 30 // Laenge der grossen Pause in Minuten
18  },
19  sounds: { // Sounds fuer unterschiedliche Events
20    bell: true, // Beginn und Ende einer Pause
21    success: true, // Beenden der Lernphase nach Erreichen des Ziels
22    failure: true // Beenden der Lernphase vor Erreichen des Ziels
23  }
24 };
```

Listing 8: Beispiel: Standardwerte für das Pomodoro-Timer Formular

Für den Bonus Pomodoro-Timer werden diese Einstellungen innerhalb des Cardset **MongoDB – Document** gespeichert und dessen Einstellungen für die Teilnehmer automatisch geladen:

```

1 pomodoroTimer {
2   quantity: 2 // Anzahl der zu erreichenden Pomodoro-Einheiten
3   workLength: 30 // Laenge der Lernzeit in Minuten
4   breakLength: 15 // Laenge der Pause in Minuten
5   soundConfig: [true, false, true] // Array aus Booleans fuer die einzelnen Soundeinstellungen.
6   [bell, success, failure]
7 }

```

Listing 9: Bonus Pomodoro-Timer Objekt innerhalb eines Cardset **MongoDB – Document**

Um die Position des Pomodoro-Timer auf der Landing-Page zu bestimmen, wird eine Abfrage an die **adminSettings MongoDB – Collection** durchgeführt. Hierfür gibt es ein **MongoDB – Document** mit dem Eintrag `name: "wordcloudPomodoroSettings"` welches ein Boolean **MongoDB – Field** **enabled** für die Darstellung benutzt. Ist dieses auf true, wird die Wortwolke auf der Landing-Page deaktiviert und an dessen stelle der Timer angezeigt. Bei false ist die Wortwolke im Desktop-Modus sichtbar und der Timer wird an der linken unteren Ecke platziert.

5.4.1 SweetAlert2-Benachrichtigungen

Hilfsklasse	SweetAlertMessages
-------------	--------------------

Für die Benachrichtigungen der Pause oder des beenden der Lerneinheit wird die Java-Script Bibliothek SweetAlert2 [ED18] verwendet. Die Aufrufe der Dialoge befinden sich in der Datei `/imports/api/sweetAlert.js`. Die Übersetzungen sind in den i18n Dateien hinterlegt unter **pomodoro.sweetAlert**:

```

1 'pomodoro' {
2   'sweetAlert': {
3     'user': { // Dialoge fuer Benutzer ohne Bonus
4     },
5     'bonus': { // Dialoge fuer Benutzer mit Bonus
6     }
7     'presentation': { // Dialoge fuer den Praesentationsmodus und der Demo-Kartei
8     }
9   }
10 }

```

Listing 10: SweetAlert2 Nachrichten des Pomodoro-Timer in den i18n Dateien

5.5 Wortwolke

Template-Pfad	/imports/ui/wordcloud/
Hilfsklasse	WordcloudCanvas
Collection	cardsets

Für die Wortwolke wird die JavaScript Bibliothek **wordcloud2.js** [GU18] verwendet. Sie ist die Standardansicht für die Landing-Page.

Es werden in der Wortwolke alle Karteien und Repetitoren angezeigt, die den Wert `wordcloud: true` benutzen. Die Wortwolke kann im Backend für die Landing-Page deaktiviert werden. Hierbei wird in der **MongoDB – Collection** **adminSettings** das **MongoDB – Field** **enabled** für das **MongoDB – Document** **wordcloudPomodoroEnabled** auf `true` gesetzt. In den einzelnen Filter-Ansichten wird eine spezielle Konfiguration für die Darstellung der Wortwolke verwendet. Sie wird mit der Funktion **setConfig** der Hilfsklasse **WordcloudCanvas** abgerufen. Die Größe der Wortwolke wird mit **setCanvasSize** bei jeder Änderung der Browser-Fenster Größe neu berechnet. Hierbei wird die Höhe aller Elemente Ober und unterhalb der Wortwolke addiert und die Gesamtgröße mit der Höhe des Browserfensters abgezogen. Dadurch wird garantiert, dass bei den Ansichten mit einer Wortwolke das HTML-Dokument nicht größer als das Sichtfeld ist und der gesamte Inhalt "Above the Fold" dargestellt wird.

Es kann einer von zwei Datensätzen auf der Landing-Page angezeigt werden:

1. **fakeStatistics:** Falls innerhalb der Settings-Datei der Anwendung der Wert **public.fakeStatistics** auf `true` gesetzt wurde, benutzt die Landing-Page vordefinierte Karteinamen anstatt diese aus der Datenbank abzurufen. Diese sind unter `/public/fakeStatistics/wordcloud.json` hinterlegt. Aufgrund der geringen Informationsmenge der Fake-Statistik ist das Modal für weitere Details deaktiviert.
2. **Datenbank:** Wenn die Fake-Statistik innerhalb der Settings-Datei deaktiviert wurde, werden die Werte aus der Datenbank geladen.

Für die Erzeugung der Wortwolke werden folgende Werte übergeben:

```
1 clearCanvas = true; // Sollen die alten Werte vor dem Zeichnen geloescht werden?
2 drawOutOfBound = false; // Zeige nur Elemente an die in das Canvas passen
3 gridSize = 24; // Der Abstand zwischen den Text-Elementen
4 weightFactor = 24; // Die Gewichtung der Kartenzahl eines Objekts. Je groesser der Wert, desto
   groesser der unterschied zwischen kleinen und grossen Karteien
5 rotateRatio = 0; // Die Drehung eines Elements. 0 bedeutet dass der Text horizontal ausgerichtet
   ist
6 fontFamily = 'Roboto Condensed, Arial Narrow, sans-serif';
7 color = "random-light"; // Zufaelliche Farbauswahl fuer die Elemente
8 backgroundColor = 'rgba(255,255,255, 0)'; // Hintergrundfarbe des Canvas. Fuer die Applikation .
   cards ist dieser Wert Transparent
9 wait = 400; // Die Laenge der Pause in Millisekunden fuer das Zeichnen eines einzelnen Elements
10 hover: WordcloudCanvas.wordcloudHover, // Aufruf der Funktion wordcloudHover(), falls die Maus
   ueber einem Text-Element ist
11 click: WordcloudCanvas.wordcloudClick, // Aufruf der Funktion wordcloudClick(), falls der
   Benutzer auf ein Element klickt
```

Listing 11: Beispiel: Daten für das Erzeugen der Wortwolke

5.6 Karte


Template-Pfad	/imports/ui/card/
Hilfsklassen	CardVisuals, CardNavigation
Collection	cards

Die Eigenschaften eines Karteityps werden in `/imports/api/cardType.js` definiert. Mit Hilfe von Arrays kann die Funktionalität für den Typ eingestellt werden. Die Zahlen innerhalb des Arrays entsprechen der Nummer des Karteityps:

```
1 let cardTypesWithDifficultyLevel = [0, 1, 2];
2 let cardTypesWithLearningModes = [0, 1];
3 let cardTypesOrder = [{cardType: 1}, {cardType: 0}, {cardType: 2}];
```

Listing 12: Beispiel: Globale Eigenschaften für Karteityps

In diesem Beispiel wird angegeben, welche Karteitypen einen **Schwierigkeitsgrad** (**cardTypesWithDifficultyLevel**) und **Lernmodus** (**cardTypesWithLearningModes**) besitzen. Die letzte Eigenschaft **cardTypesOrder** gibt die Reihenfolge für die Auswahllisten an.

Karten bestehen meist aus mehreren Seiten mit unterschiedlichen Eigenschaften. Hierfür befindet sich in der Datei ein Array **cardTypeCubeSides**, welches Arrays mit einem Objekt pro Kartenseite beinhaltet. cards unterstützt bis zu 6 Seiten mit unterschiedlichem Inhalt pro Karte. Um später geplante Features wie die 3D-Würfel Ansicht zu verwenden, sollte diese Anzahl nicht überschritten werden.

```
1 //1: Vokabelkartei / Vocabulary
2 [
3   {
4     "contentId": 1,
5     "side": "front",
6     "defaultStyle": "default",
7     "defaultCentered": true
8   },
9   {
10    "contentId": 2,
11    "side": "back",
12    "defaultStyle": "default",
13    "defaultCentered": true,
14    "isAnswer": true,
15    "isAnswerFocus": true
16  }
17 ]
```

Listing 13: Beispiel: Seiten Eigenschaften für eine Vokabel Kartei

Alle Werte bis auf **contentId** und **side** sind optional. Für **defaultStyle**, **defaultCentered** und **defaultTextAlign** werden die Default-Werte der Variablen innerhalb der Datei benutzt, falls kein Wert angegeben ist. Der Inhalt einer Karte kann aus folgenden Werten bestehen:

Wert	Typ	Eigenschaft
contentId	Int32	Inhalt für die Seite, Wert von 1 - 6
side	String	Position auf 3D-Würfel (front back left right top bottom)
defaultStyle	String	Hintergrundfarbe aus dem Theme-Switcher
defaultCentered	Boolean	Ist der Text standardmäßig Vertikal-Zentriert?
defaultTextAlign	Int32	Horizontale-Zentrierung für Text (0 = left 1 = center 2 = right 3 = justify)
gotLearningGoal Placeholder	Boolean	Wird als Platzhalter der Text für das Lernziel benutzt?
isAnswer	Boolean	Ist die Seite teil der Antwort?
isAnswerFocus	Boolean	Sprung auf Seite beim anzeigen der Antwort

Tabelle 4: Seiten-Aufbau einer Karte

5.6.1 Anlegen eines neuen Karteityp

Um einen neuen Karteityp anzulegen, müssen folgende Kriterien erfüllt sein:

- Es müssen Seiten innerhalb **cardTypeCubeSides** existieren.
- Die globalen Einstellungen müssen am Anfang der Datei gesetzt sein. Die benötigte Zahl ist die Position des Objekts innerhalb **cardTypeCubeSides**.
- Die i18n Felder wurden angelegt. Diese befinden sich unter **card.cardTypeN**, wofür N für die Zahl des Typs steht.

Damit die Übersetzungsdateien richtig geladen werden, muss der **Meteor**-Server komplett neu-gestartet werden. Ein automatischer Neubau reicht nicht aus.

5.6.2 Editor & Markdeep

Template-Pfad	/imports/ui/card/editor/, /imports/ui/markdeep/editor/
Hilfsklassen	CardEditor, MarkdeepEditor, MarkdeepContent
Collection	cards

Der Pfad des Karteneditors ist in zwei Teile aufgeteilt: `/imports/ui/card/editor/` beinhaltet alle Elemente, die spezifisch zum Bearbeiten der Karte sind, darunter zählen:

- Bearbeiten des Titels einer Karte
- Setzen der Kompetenzstufe
- Hinzufügen einer Lerneinheit

Der Zweite Teil, Markdeep, wird zusätzlich für den Kartei-Editor benutzt und befindet sich in einem eigenen Verzeichnis `/imports/ui/markdeep/editor/`. Die Elemente des Markdeep-Editors sind die Navigationsleiste für das Styling und das Textfeld.

Die Konvertierung des Markdeep Text wird mit einem **Meteor – Block Helper** und einer Hilfsklasse getätigt. Diese Klasse befindet sich unter `/imports/api/markdeep`. Bevor der umgewandelte Markdeep Text an den Client ausgegeben wird, müssen folgende Schritte abgearbeitet werden:

- **Der Inhalt muss auf Schadcode überprüft werden:** Um die volle Funktionalität von Markdeep ausschöpfen zu können, muss inline styling und scripting erlaubt werden. Damit Benutzer den Inhalt der Karten nicht für missbräuchliche Zwecke nutzen, wird der Inhalt mit der Erweiterung DOMPurify [CU18] überprüft.
- **Bilder mit Lightbox verknüpfen:** Markdeep hat standardmäßig keine Unterstützung für Lightbox [DH17], daher müssen nach der Konvertierung des Textes alle Bilder in einem Lightbox Wrapper eingebunden werden.
- **Media-Controls einstellen:** Für eingebettete Videos wird die Navigationsleiste aktiviert. Für iframes wird der Vollbild-Modus deaktiviert, da dies ansonsten mit dem Vollbild-Modus von ●cards in Konflikt gerät.
- **Text-Zoom:** Das Eltern-Element für den Inhalt wird mit einer festen **font-size** in Pixeln definiert. Der Wert wird aus der **Meteor – Session** der ●cards Zoomfunktion entnommen. Dadurch wird sichergestellt, dass der Text vor der Darstellung die richtige Schriftgröße benutzt
- **Links modifizieren:** Damit der Anwender beim Lernen die Anwendung nicht versehentlich vollständig verlässt, werden alle Links mit dem Attribut **target="__blank"** ausgestattet
- **iFrame Höhe anpassen:** Durch die Verwendung von **Screenfull.js** [SO18] kann der Vollbild-Modus für YouTube Videos nicht benutzt werden. Um die Videos dennoch für den Präsentationsmodus groß darstellen zu können, wird die Höhe automatisch an der Größe des sichtbaren Karteninhaltes angepasst.
- **Markdeep Style:** Die von Markdeep mitgelieferten Styles werden am Anfangsbereich des umgewandelten Inhaltes gesetzt

Am Ende wird der zurückgegebene Text mit MathJax umgewandelt. Die mit Markdeep ausgelieferte Variante von MathJax kann nicht benutzt werden, da der Markdeep Inhalt erst nach dem Erzeugen des **DOM** hinzugefügt wird. Dies ist auch der Grund, warum die Markdeep Link-Anchor mit einer für ●cards ausgelegten JavaScript Funktion angesprochen werden.

5.6.3 Markdeep Aktualisieren

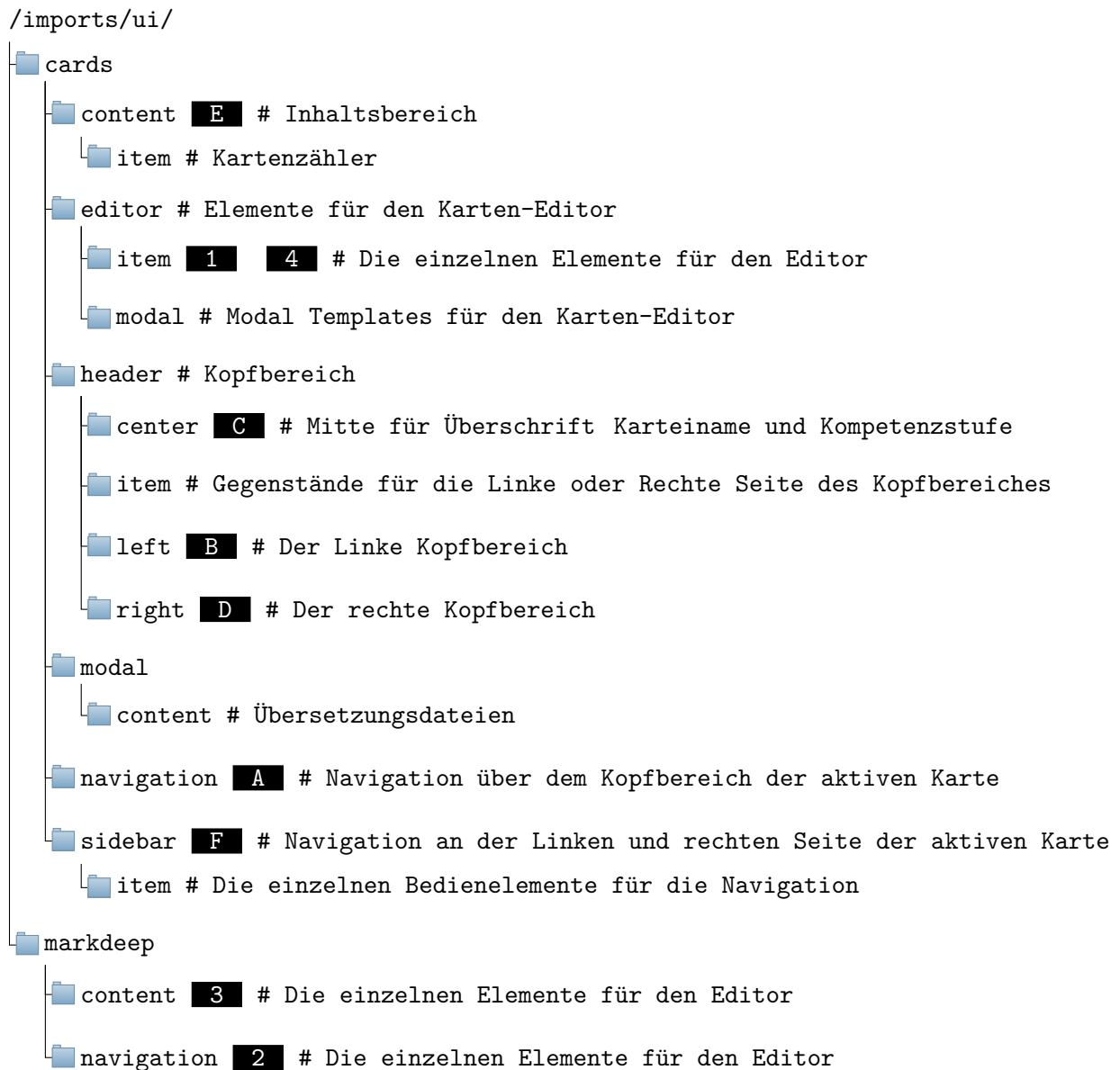
Ein direktes Aktualisieren von Markdeep ist aufgrund der Konfiguration von ●cards nicht möglich. Es müssen daher folgende Schritte für die Vollständige Aktualisierung ausgeführt werden:

1. Lade die aktuelle Version von markdeep.js herunter:
<https://casual-effects.com/markdeep/latest/markdeep.min.js>
2. Such innerhalb des beigefügten highlight.min.js Datei (Beginn ab Zeile 7 von markdeep.min.js) nach der Zeile: `"object"==typeof self&&self;"undefined"!=typeof exports?` und ersetze den Typ `exports` am Ende des Befehls mit `undefined`. Das Ergebnis sollte wie folgt aussehen: `"object"==typeof self&&self;"undefined"!=typeof undefined?`
3. Aktualisiere die MathJax-Einstellungen mit den neuen Werten aus Markdeep: <https://casual-effects.com/markdeep/latest/markdeep.js>. Du kannst die neuen Werte finden, indem du nach der Variable **MATHAX__Config** suchst.
Benutzerdefinierte Definitionen gehen zu **customMathJaxDefinitions** unter `/imports/api/markdeep.js`
Die MathJax-Einstellungen gehen zu **MeteorMathJax.defaultConfig** unter `/imports/api/markdeep.js`

Vergewissere dich, dass deine lokale Installation funktioniert, bevor du diese auf den Staging-Server überträgst.

5.6.4 Templates

Die Templates für Karten sind in zwei Pfade aufgeteilt: `/imports/ui/cards` und `/imports/ui/markdeep`. Der letzte Pfad wird zusätzlich für Karteien benutzt und passt daher nicht in den Pfad für Karten.



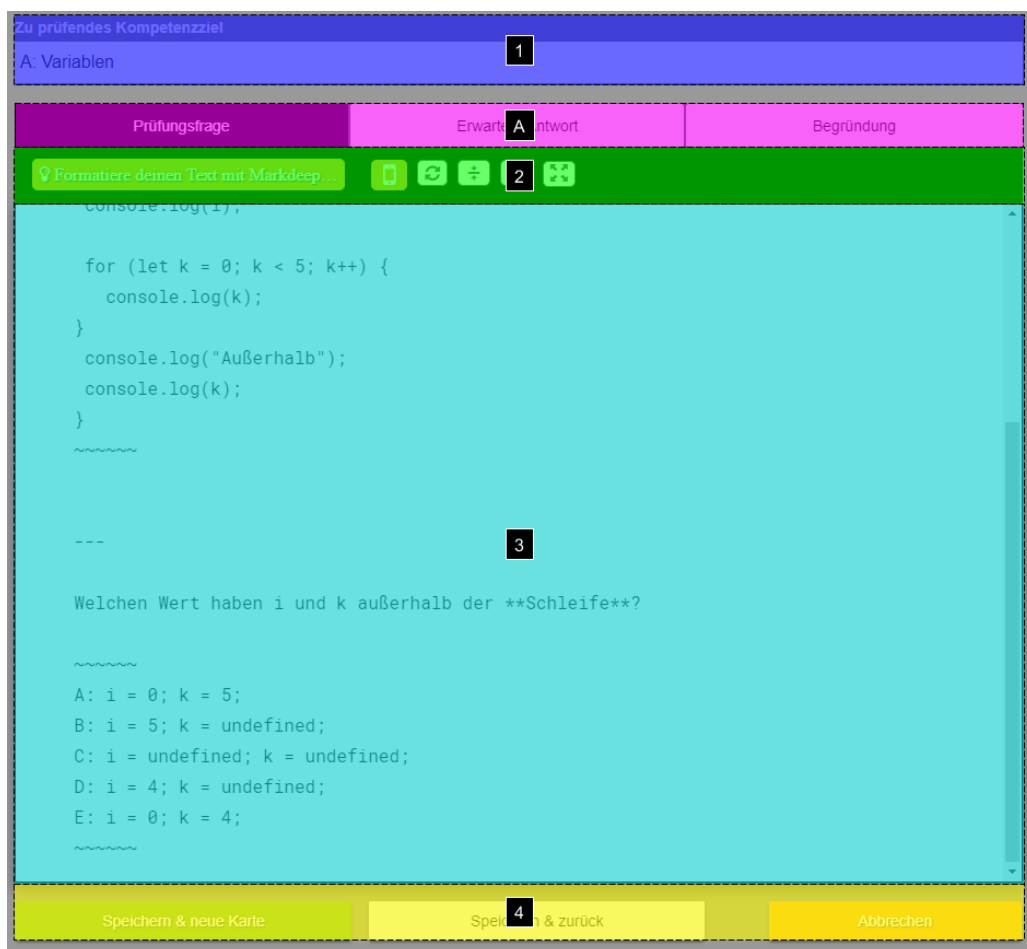
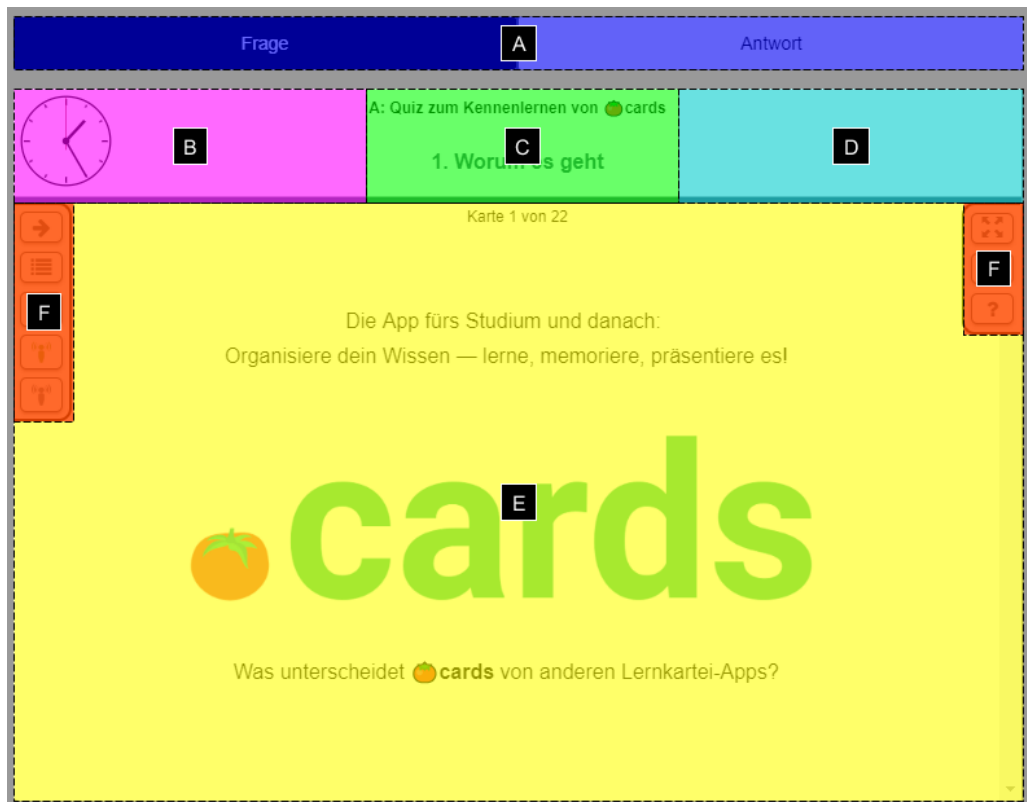


Abbildung 16: Karte & Karten-Editor: Template

5.6.5 Verfügbare Hintergrundfarben für Karten

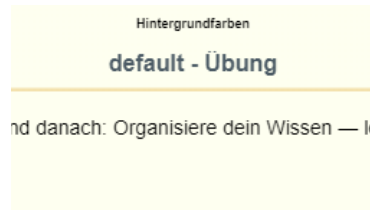
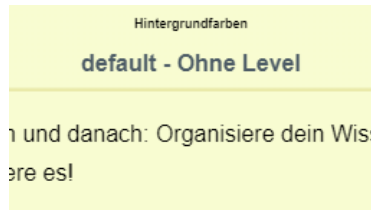


Abbildung 17: Hintergrundfarben für Karten der Schwierigkeitsstufe 0 (Ohne Level)

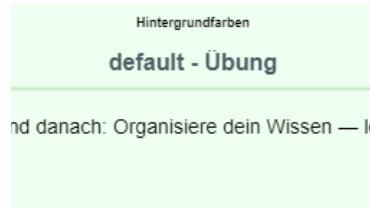
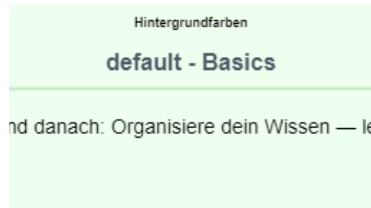


Abbildung 18: Hintergrundfarben für Karten der Schwierigkeitsstufe 1 (Basic)

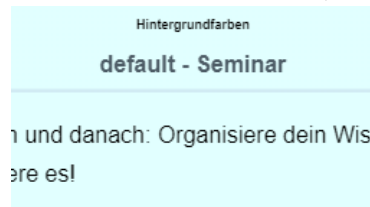
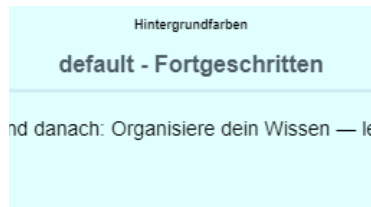


Abbildung 19: Hintergrundfarben für Karten der Schwierigkeitsstufe 2 (Fortgeschritten)

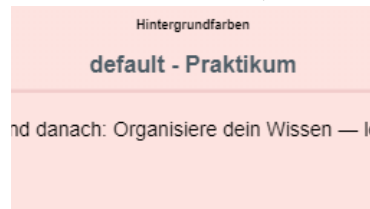
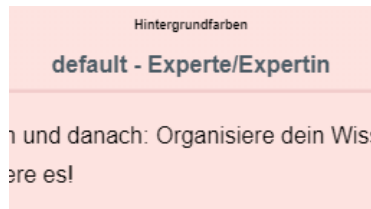


Abbildung 20: Hintergrundfarben für Karten der Schwierigkeitsstufe 3 (Experte/Expertin)

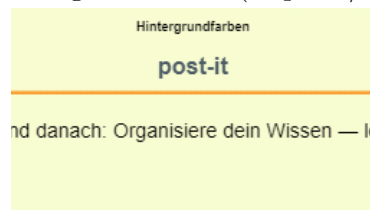
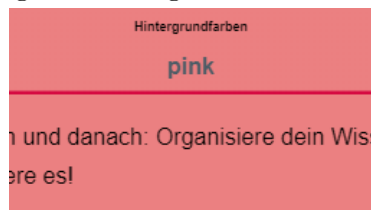


Abbildung 21: Hintergrundfarben für Notizen

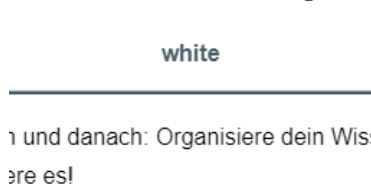



Abbildung 22: Hintergrundfarbe für den Präsentationsmodus

5.7 Kartei

Template-Pfad	/imports/ui/cardset/
Hilfsklassen	CardsetVisuals, CardsetNavigation, CardIndex
Collection	cardsets

5.7.1 Index

Eine Kartei besteht meist aus mehreren Karten. Damit es nicht zu Performance-Problemen in größeren Karteien kommt rendert cards nur maximal 3 Karten gleichzeitig. Hierfür wird ein **ID-Index** bestehend aus den Karten der aktiven Kartei erstellt. Der Inhalt des Karten-Karussells wird nach dem Wechsel der Karte aktualisiert. Die Funktion **initializeIndex** der Hilfsklasse **CardIndex** generiert die notwendige Liste und speichert diese in der Variable **cardIndex**. Diese Liste kann dann durch die Funktion **getCardIndex** abgerufen werden.

Aufgrund fehlender Funktionen für die **Meteor – MiniMongoDB** [MA16], ist eine case insensitive Sortierung nicht möglich. Dieses Problem wird zu einem späteren Zeitpunkt mit Hilfe des manuellen Sortierens gelöst.


5.7.2 Repetitorium

Ein Repetitorium ist vom Aufbau identisch mit einer Kartei. Durch das Setzen des Wertes **isShuffled** wird bestimmt, ob eine Kartei als Repetitorium gespeichert wurde. Beide Typen werden in der **MongoDB – Collection** **Cardsets** abgelegt.

In dem Array **cardGroups** werden die ID's der referenzierten Karteien abgelegt, über die das Repetitorium die Karten bezieht. Die Zugriffsberechtigungen auf die referenzierten Karten werden mit den Einstellungen des Repetitoriums überschrieben. Dies erlaubt es, Karten aus Karteien mit dem Zugang "Privat" zu veröffentlichen, ohne die Einstellungen der referenzierten Kartei zu verändern.

5.7.3 Demo-Kartei und The-Making-of-cards

Meteor-Methoden	deletedemoCardsets importDemoCardsets
-----------------	---------------------------------------

Eine automatisch generierte Kartei. Sie wird bei jedem Start des Server neu erstellt. Eine Demo-Kartei ist ein Repetitorium, welches seinen Inhalt aus dem Pfad `/private/demo` bezieht. Der Inhalt besteht aus exportierten Karteien welche das cards Format verwenden. The-Making-of-Cards benutzt dasselbe System wie die Demo-Kartei und befindet sich unter dem Pfad `/private/makingOf`.

5.7.4 Templates

cards/imports/ui/cardset

- info **A** # Kopfbereich der Kartei für den Titel und Beschreibung
 - box **B** **C** # Tabellen mit Informationen über die Kartei / Repetitorium
 - item # Die einzelnen Felder der Tabellen
- index
 - bonus # Bonus-Lernstatistik aller Teilnehmer
 - editors # Liste an Teilnehmern mit Bearbeitungsrechten für die Kartei
 - cards # Kartei-Index
- navigation **D** # Navigation unterhalb der Kartei Beschreibung
 - item # Die einzelnen Navigationselemente
 - modal # Modale für die Kartei Navigation

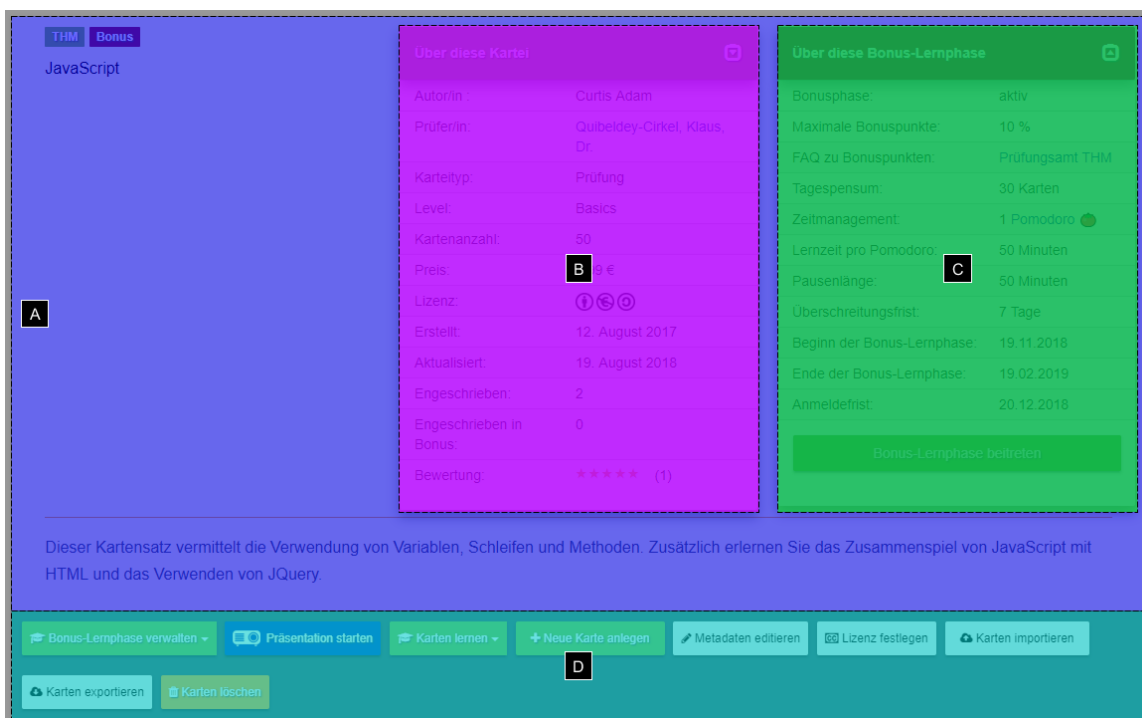


Abbildung 23: Kartei: Template

5.8 Filter

Template-Pfad	/imports/ui/filter/, /imports/ui/cardsets
Hilfsklassen	Filter, FilterNavigation
Collection	cardsets

Die Liste für die Kategorien "Themen", "Meine Karteien", "Repetitorien", "Mein Lernpensum", "Repetitorium erstellen" und "Alle Karteien" wird durch die Filter am Anfang der Navigation gesteuert. Die Änderungen der Einstellungen können durch Arrays am Anfang der Datei `/imports/api/filterNavigation.js` vorgenommen werden:

```
1 //0: Themen-Pool / Pool
2 //1: Kartei anlegen / My Cardsets
3 //2: Repetitorien / Repetitorium
4 //3: Lernpensum / Learning
5 //4: Alle Karteien / All Cardsets
6 //5: Kartei mischen / Shuffle
7 let filtersWithResetButton = [0, 1, 2, 3, 4, 5];
8 let filtersWithDisplayModeButton = [0, 2, 4];
9 let filtersWithSortButton = [0, 1, 2, 3, 4, 5];
```

Listing 14: Ausschnitt aus der Konfiguration für die Filter-Liste

In diesem Ausschnitt sieht man an dem Array `filtersWithDisplayModeButton`, dass durch die Zahlen 0, 2 und 4 festgelegt wird, welche Ansichten einen Modus für das Umschalten für die Liste oder Wortkwolke besitzen. Der Filter ist in zwei Klassen aufgeteilt:

- **Filter** zu finden in `/imports/api/filter`
- **FilterNavigation** zu finden in `/imports/api/filterNavigation`

Über die FilterNavigation werden die Filteroptionen für die 5 unterschiedlichen Seiten gesteuert. Die Klasse Filter erstellt anhand dieser Informationen den Kartei-Filter für die **MongoDB**-Query. Jede Seite verwendetet seine eigene **Meteor – Session**, um die Filter-Einstellungen zu speichern:

```
1 Session.setDefault('poolFilter', undefined); // Themen
2 Session.setDefault('myCardsetFilter', undefined); // Meine Karteien
3 Session.setDefault('repetitoriumFilter', undefined); // Repetitorien
4 Session.setDefault('workloadFilter', undefined); // Mein Lernpensum
5 Session.setDefault('allCardsetsFilter', undefined); // Alle Karteien
6 Session.setDefault('shuffleFilter', undefined); // Repetitorium Referenzen bearbeiten
```

Listing 15: Sessions für das speichern der einzelnen Filtereinstellungen

Für die Auswahllisten der einzelnen Filter wird jeweils ein **MongoDB**-Query ausgeführt. Dies ist notwendig, um alle möglichen Filterkriterien anzuzeigen. Diese Filter aktualisieren sich automatisch bei jeder Änderungen durch die Benutzung des **Meteor – Session** Elements. Der Wert der **Meteor – Session** bleibt solange erhalten bis entweder die Sitzung geschlossen wird oder der Benutzer den Filter zurücksetzt.

5.8.1 Templates

cards/imports/ui/

```
└─ filter # Die Navigationsleiste für den Filter
   └─ index B # Index der einzelnen Ansichten
      └─ item # Elemente für die Darstellung einer Kartei
      └─ navigation A # Filter-Navigation
         └─ item # Die einzelnen Filteroptionen für die Leiste
```

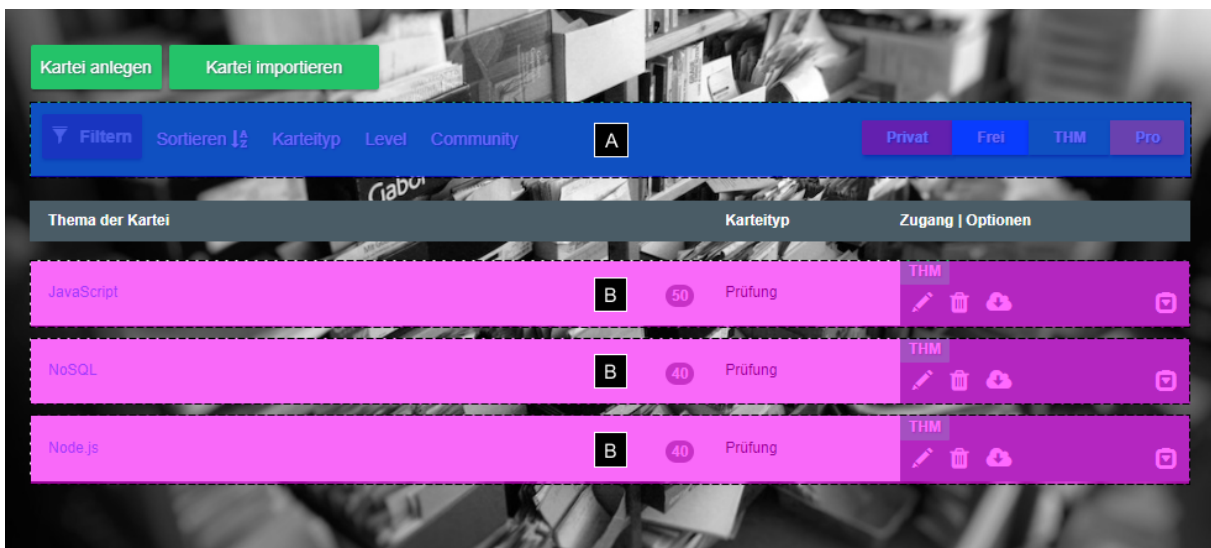


Abbildung 24: Filter: Template

Die derzeitige Version des Filters befindet sich noch in der Ausbauphase. Dieser wird in einem späteren Zeitpunkt alle Elemente in einzelne Templates und Dateien für den Inhalt von **B** zerlegen. Zusätzlich wird das Design überarbeitet, um die Darstellung mehrerer Labels zu ermöglichen:

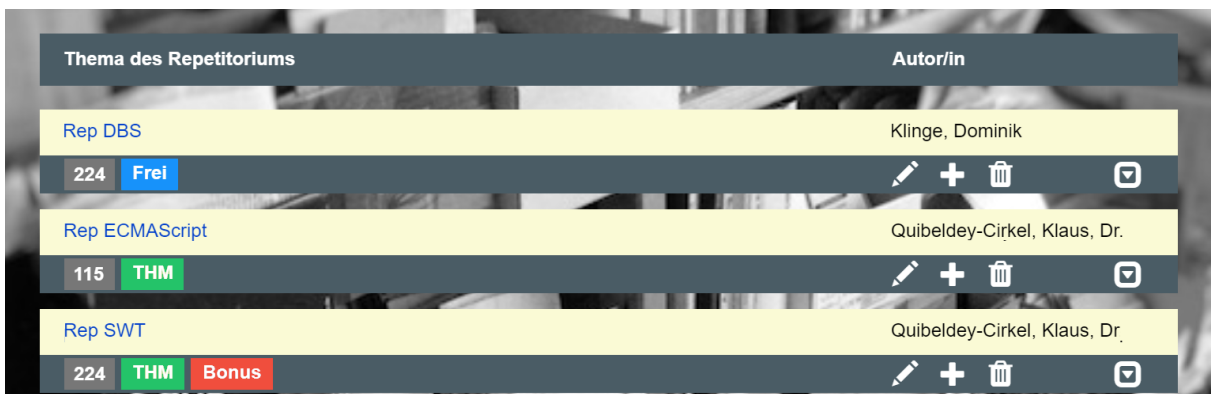


Abbildung 25: Filter: Das neue Design für die Ergebnisliste des Filters

5.9 Benutzer

Template-Pfad	/imports/ui/profile/
Hilfsklasse	UserPermissions
Collection	users

5.9.1 Rollen

Benutzerrollen werden mit der Hilfsklasse **UserPermissions** überprüft, welche sich unter `/imports/api/userPermissions` befindet. Mit ihr wird die Anzeige im Client-Modus und das Versenden der Datenbankeinträge innerhalb der **Meteor – Subscription** gesteuert. Die Anwendung verfügt über 9 verschiedene Berechtigungstypen. Ein Benutzer kann mehr als einen Berechtigungstyp verwenden:

- **firstLogin:** Beim Erstellen eines neuen Profils muss der Benutzer die AGB akzeptieren. In diesem Zeitraum ist er bereits mit einem Profil eingeloggt, welches mit dem **firstLogin** Berechtigungstyp markiert ist. Dieser verhindert, dass der Benutzer auf andere Seiten der Anwendungen zugreifen kann. Nachdem die AGB bestätigt wurden, wird dieser Berechtigungstyp entfernt. Beim Ablehnen der AGB wird das Profil gelöscht.
- **blocked:** Benutzer mit diesem Berechtigungstyp können auf keine Seite zugreifen. Sie werden stattdessen auf die URL `/blocked` umgeleitet. Dieser Berechtigungstyp kann nur durch einen Super-Admin oder Admin gesetzt und entfernt werden.
- **standard:** Dies entspricht dem Zugang "Frei". Dieser Berechtigungstyp wird automatisch an Benutzern vergeben, die ihr Profil ohne den CAS-Login erstellt haben.
- **university:** Entspricht dem Zugang "THM". Dieser Berechtigungstyp wird automatisch an Benutzern vergeben, welche ihr Profil mit dem CAS-Login erstellt haben.
- **pro:** Benutzer mit einem Pro-Abonnement erhalten automatisch diesen Berechtigungstyp. Er erlaubt Ihnen Zugriff auf alle öffentliche Karteien und das Erstellen von Pro-Karteien.
- **lecturer:** Muss von einem Super-Admin oder Admin vergeben werden. Dieser Berechtigungstyp entspricht dem Zugang "Dozent". Dozenten haben dieselben Zugriffsrechte wie Pro-Benutzer und können Anträge für die Freigabe von Pro-Karteien bearbeiten.
- **admin:** Ein Benutzer mit diesem Berechtigungstyp hat Zugriff auf alles. Der Super-Admin kann in der Settings-Datei des Servers bestimmt werden. Hierzu muss der Eintrag **admin.name** mit dem CAS Kürzel hinterlegt werden. Beim Registrieren des entsprechenden Kontos bekommt dieser Benutzer dann den admin Berechtigungstyp. Super-Admins können von normalen Administratoren nicht gelöscht oder bearbeitet werden.
- **editor:** Das normale "Admin" Konto. Ein Admin besitzt dieselben Rechte wie der Super-Admin, kann jedoch Super-Admins nicht löschen oder modifizieren.
- **server:** Ein Berechtigungstyp für versteckte Benutzerprofile der Anwendung. Dieser wird grundsätzlich nur als Filter benutzt, um Inhalte in der Anwendung zu verbergen. Das Benutzerkonto für den Besitzer der Demo-Kartei und Test-Notifikationen ist auf diesen Berechtigungstyp angewiesen.


5.9.2 Bezahlung & Preview-Karten

Collection	paid
------------	------

Pro Bezahlung wird ein Objekt in der **MongoDB – Collection** **paid** angelegt. Dieses Objekt wird für die Erstellung der Kartei und Repetitorium **Meteor – Subscription** benutzt:

```
1 "_id" : "756zQwtC3bnDwMGiE",
2 "cardset_id" : "49v9vKD3J4kTHxWaf", // ID der bezahlten Kartei oder Repetitorium
3 "user_id" : "maY94ik3rQLK555Sw", // ID des Kaeufers
4 "date" : ISODate("2018-01-24T18:14:11.795Z"), // Kaufdatum
5 "amount" : 4.99 // Preis
```


Listing 16: Beispiel: **MongoDB** Bezahlungs-Objekt

Für die Bezahlung wird derzeit der Sandbox-Modus von Braintree verwendet. Hierbei entstehen dem Benutzer keine Kosten. Sobald cards ein ausgereifteres Bezahlungssystem verwendet, wird dieser Sandbox-Modus deaktiviert.

Falls ein Benutzer nicht die benötigten Rechte für eine öffentliche Kartei oder einem Repetitorium besitzt, kann er diesen dennoch mit beschränktem Zugriff einsehen. Hierbei werden dem Benutzer eine beschränkte Anzahl an Karten zur Verfügung gestellt. Diese Karten werden per Zufall aus dem Pool entnommen. Die Funktion, die für die Zuweisung verantwortlich ist, heißt **getPreviewCards** und befindet sich am Anfang der Datei `/imports/api/cards.js`

5.9.3 URL & Subscriptions

Pfad	/imports/ui/startup/client/route.js
------	-------------------------------------

Für die Zuweisung der Templates zu den einzelnen Ansichten der Applikation cards wird das package **iron-router** [MA17] verwendet.

```
1 Router.route('demolist', {
2   name: 'demolist', \\ Name des Pfades
3   template: 'demo', \\ Name des Templates
4   subscriptions: function () {
5     return [Meteor.subscribe('demoCardsets'), Meteor.subscribe('demoCards')];
6   },
7   data: function () {
8     Session.set('helpFilter', "cardsetIndex");
9     return Cardsets.findOne({kind: 'demo', name: "DemoCardset", shuffled: true});
10  },
```

Listing 17: Kopfbereich für einen **iron-router** Pfad

Die Funktion **subscriptions** legt die **Meteor – Subscription** fest, die für die Darstellung der URL benötigt werden. Solange die benötigten **Meteor – Subscription** nicht geladen sind, wird ein Template mit einem Ladebalken ausgegeben. In der Funktion **data** wird die aktuelle **Meteor – Session** für die Hilfe gesetzt. Für gewisse Ansichten wird zusätzlich ein Objekt, meist als Resultat einer **MongoDB** Suche, zurückgegeben. Dieses Objekt kann in dem mit **template:** referenzierten **Meteor**-Template durch das Schlüsselwort **this** aufgerufen werden. Rückgaben aus der Funktion **data** sind in der Regel reaktiv und sollten nicht für den Editor benutzt werden, da dieser Ansonsten temporäre Änderungen bei einem Seitenaufruf durch ein anderes Tab oder Fenster verwirft. Es ist zudem anzuraten, die benötigten **Meteor – Subscription** für die Ansicht anzupassen, um die CPU Last und den Traffic durch zu viele reaktive Objekte zu minimieren [BA16].

5.10 Lernmodus

5.10.1 Leitner

Template-Pfad	/imports/ui/learn/
Collection	leitner

Der Großteil der Leitner-Methoden wird nur vom Server ausgeführt und befindet sich unter `/server/leitner.js`.

Für jede Karte einer Kartei oder eines Repetitorium wird ein Lernobjekt angelegt, welches folgende Eigenschaften enthält:

```
1 "_id": "5FnStoyhpugBHeHHb",
2 "cardset_id": "A3CZBYvfZJ6JkM6v2", // Zugehörige Kartei oder Repetitorium
3 "user_id": "AGmLaJx6FnKRPmTp7", // ID des Lernalters
4 "original_cardset_id": "AGmLaJx6FnKRPmTp7", // ID der Referenzierten Kartei falls Repetitorium
5 "box": 3, // Das derzeitige Fach in dem sich die Karte befindet
6 "active": true, // Muss der Lerner diese Karte derzeit beantworten?
7 "currentDate": DATE, // Datum an dem die Karte aktiviert wurde
8 "nextDate": DATE // Das naechstmögliche Datum ab dem die Karte wieder aktiviert werden kann
```

Listing 18: Beispiel: MongoDB Leitner-Objekt

5.10.2 Aktivieren von Karten (Wiederholungs-Algorithmus von Leitner)

Die `Meteor – Method setCards` ist verantwortlich für das Setzen der nächsten aktiven Karten. Im Folgenden eine Beschreibung des Ablaufs für das Setzen der Karten, als Beispiel wird folgender Lernstand verwendet:

Anzahl verfügbarer Karten pro Fach: [412, 5, 5, 0, 10]

Maximales-Tagespensum: 33

Fach Nr. 1: **412 Karten**, Fach Nr. 2: **5 Karten**, Fach Nr. 3: **5 Karten**, Fach Nr. 4: **0 Karten**, Fach Nr. 5: **10 Karten**

Im ersten Schritt wird die Anzahl der aktiven Karten anhand des Tagespensums festgelegt. Diese Anzahl besteht in der Regel aus **50%** der Karten aus Fach Nr. 1, **20%** aus Fach Nr. 2, **15%** aus Fach Nr. 3, **10%** aus Fach Nr. 4 und **5%** aus Fach Nr. 5.

Leere Fächer addieren Ihre Prozentzahl auf die nächst höheren. In unserem Beispiel ergibt sich dadurch folgende Anzahl an Karten:

Maximal zu holende Karten pro Fach: [(16,5), (6,6), (4,95), 0, 4,95]

Nach Runden: [17, 7, 5, 0, 5] = 34 Karten

Als Nächstes wird überprüft, ob die Summe dem Maximalem-Tagespensum entspricht. Falls diese nicht übereinstimmt, wird die Differenz am niedrigsten Fach ausgeglichen:

Nach Anpassung an Maximales-Tagespensum: [16, 7, 5, 0, 5]

Nun wird überprüft, ob die Anzahl der Karten vollständig belegt werden kann. In unserem Fall fehlen für Fach Nr. 2 und Fach Nr. 4 einige Karten.

Anzahl fehlender Karten: [-396, 2, 5, 0, -5] = 7 fehlende Karten

Um nun die fehlende Anzahl an Karten auszugleichen, wird die Verteilung der Fächer angepasst. Es werden vom niedrigsten Fach, in diesem Fall Fach Nr. 1, Karten entnommen. Falls Fach Nr.1 nicht genügend Karten enthält, wird der Rest aus dem nächst höheren Fach entnommen:

Nach aktualisieren der Verteilung: [23, 5, 0, 0, 5]

Zum Schluss werden die Karten anhand der Liste mit dem Wert **active: true** gekennzeichnet. Die Zuweisung der einzelnen Karten geschieht zufällig, nur die maximale Anzahl pro Fach wird berechnet.

5.10.3 Lernstand-Kartei

Template-Pfad	/imports/ui/cardset/index/bonus
Collections	workload, leitner

Die Liste wird beim Aufbau des Templates mit der **Meteor – Method** `getLearningData` generiert und das Ergebnis in der **Meteor – Session** `learnerStats` hinterlegt. Eine **Meteor – Subscription** ist aufgrund der Datenmenge ungeeignet und führt zu einem langen Aufbau der Liste. Wegen dieser Limitation aktualisiert sich die Liste nicht Automatisch bei einer Änderung. Für jede Aktion die den Inhalt der Liste verändert, muss die **Meteor – Session** `learnerStats` mit dem neuen Inhalt aktualisiert werden.

5.10.4 Lernstand-Benutzer

Template-Pfad	/imports/ui/learn/
Hilfsklasse	LeitnerProgress
Collections	workload, leitner

Für die Darstellung der Lernstandanzeige wird die JavaScript Bibliothek **chart.js** verwendet. Mit der Hilfsklasse **LeitnerProgress** wird die **MongoDB**-Query generiert und im Anschluss mit **chart.js** gerendert. Der Aufbau des Arrays für **chart.js** besteht aus mindestens 24 **MongoDB** aufrufen. Vier verschiedene Schwierigkeitsgrade mit je 6 Fächern. Diese Informationen müssen aus den einzelnen Leitner **MongoDB – Document** eines Benutzers entnommen werden, um die Belegung der einzelnen Leitner Fächer genau darstellen zu können.


5.10.5 Wozniak

Template-Pfad	/imports/ui/learn/
Collection	wozniak

Wie auch der Wiederholungs-Algorithmus von Leitner benutzt der Wiederholungs-Algorithmus von Woźniak für jede Karte ein dazugehöriges Woźniak-Objekt:

```
1 "_id": "5FnStoyhpugBHeHHb",
2 "cardset_id": "A3CZBYvfZJ6JkM6v2", // Zugehoerige Kartei oder Repetitorium
3 "user_id": "AGmLaJx6FnKRPmTp7", // ID des Lernalers
4 "ef": 3, // Berechneter Wert ueber die Schwierigkeit der Karte
5 "interval": 1, // Wann soll die Karte wieder vorgelegt werden?
6 "reps": 2, // Anzahl Wiederholungen
7 "nextDate": DATE // Das naechstmoeegliche Datum ab dem die Karte wieder aktiviert werden kann
```

Listing 19: Beispiel: **MongoDB** Wozniak-Objekt

Der Wiederholungs-Algorithmus von Woźniak befindet sich derzeit in der Entwicklungsphase und wird ab einer späteren cards Version vollständig zur Verfügung gestellt.

5.11 Backend

5.11.1 Hochschulen

Template-Pfad	/imports/ui/admin/university/
Collection	collegeCourses

Hochschulen werden zu Repetitorien zugewiesen. Aufgrund einer derzeitigen Umstrukturierung der Anwendung sind diese momentan unbenutzt:

```
1 "_id" : "4vaJFD6bLDWEjppQb",
2 "college" : "THM", //Hochschule
3 "course" : "Architektur (konsekutiv) (MA)" // Kurs der Hochschule
```

Listing 20: Beispiel: MongoDB CollegeCourse Objekt

5.11.2 Dashboard

Template-Pfad	/imports/ui/admin/dashboard/
Subscription-Pfad	/imports/api/serverInventory/

Für die Gesamtanzahl von Karten, Karteien, Repetitorien und Benutzern wird die Erweiterung **publish-counts** [PE16] verwendet. Mit ihr ist es möglich, eine Live-Ansicht der Gesamtanzahl von Inhalten zu generieren. Da eine genaue Berechnung der Anzahl eine hohe Server-Last verursacht, werden die Werte mit dem Attribut **fastCount** bereitgestellt. Dieses gibt ohne hohe Server-Last ein Ergebnis, welches sich nur um eine geringe Anzahl von dem Istzustand abweicht. Die Statistik wird zusätzlich auf der Landing-Page für unangemeldete Benutzer angezeigt.

5.11.3 Server-Einstellungen

Template-Pfad	/imports/ui/admin/settings/
Collection	adminSettings

5.11.4 Zeige den Pomodoro-Timer statt Wortwolke auf der Landing-Page

Speichert in der **adminSettings MongoDB – Collection** ein **MongoDB – Document** mit den Boolean Wert: `"name": "wordcloudPomodoroSettings"`, welcher die Anzeige für die Landing-Page bestimmt:


- **true:** Zeige nur den Pomodoro-Timer
- **false:** Zeige die Wortwolke

5.11.5 Versenden von E-Mail-Benachrichtigungen

Speichert in der **adminSettings MongoDB – Collection** ein **MongoDB – Document** mit den Boolean Wert: `"name": "mailSettings"`. Dieser Wert wird in der Server-Datei für den Wiederholungs-Algorithmus von Leitner vor dem Versenden von E-Mail Benachrichtigungen abgefragt.

- **true:** Erlaube das Versenden von E-Mail Benachrichtigungen
- **false:** Blockiere das Versenden von E-Mail Benachrichtigungen

5.11.6 Benachrichtigungen testen

Um die Funktionalität der E-Mail und Web-Push Benachrichtigungen einfach überprüfen zu können, legt cards beim Start der Anwendung eine versteckte Kartei mit Karten und Leitner-Objekten an. Diese sind unter `/imports/startup/server/initialize.js` am Anfang der Datei im JSON Format hinterlegt.

```
1 "_id" : "5J5wnMMomKYXsoyLd",
2 "name" : "testNotifications", // Name fuer den Abruf der Einstellungen
3 "target" : "2nK723ebLafKyDf69", // Die Ziel-ID des Benutzers fuer die Testnachricht, es koennen
  nur vorhandene Benutzer eingetragen werden
4 "testCardsetID" : "NotificationsTestCardset", // Die ID der versteckten Server Test-Kartei
5 "testUserID" : "NotificationsTestUser" // Die ID des versteckten Test-Users, relevant fuer das
  finden der Leitner-Objekte
```

Listing 21: Beispiel: AdminSettings **MongoDB** – Document für Test-Nachrichten


Die vom Server bereitgestellten Objekte werden vom Leitner-Cronjob ignoriert. Für das Versenden von Nachrichten werden E-Mail und Web-Push Klassen direkt angesprochen.

5.12 Acceptance-Test

Pfad	/tests/
------	---------

5.12.1 Acceptance-Tests ausführen

Um die Chimp-Tests ausführen zu können, müssen folgende Kriterien erfüllt sein:

- Du befindest dich im Hauptverzeichnis von cards
- Der **Meteor**-Server läuft im Hintergrund

Es werden 4 Bash-Skripts innerhalb des Verzeichnis `/tests/` zur Verfügung gestellt:

runTests.sh:

Führt alle Acceptance-Tests aus die sich in im Verzeichnis `/tests/features/` befinden. Das Terminal gibt am Ende der Testreihe aus, wie viel Acceptance-Tests erfolgreich durchgelaufen sind. Dieses Skript wird zusätzlich für die GitLab Runner verwendet.

singleTests.sh:

Ähnlich `runTests.sh`, jedoch kann der Benutzer auswählen, welchen Test er starten will.

dumpTestDatabase.sh:

Erstellt einen Datenbankdump der **MongoDB** des derzeit aktivierten **Meteor**-Server. Der Inhalt wird für die Tests als Startpunkt verwendet.

loadTestDatabase.sh:

Lädt den Datenbankdump in die derzeit aktive Datenbank. Eine Ausführung dieser Datei hat die Löschung aller Daten der aktiven Datenbank zur Folge.

5.12.2 Backdoor Benutzer für die Testdatenbank

Für den Login können folgende Profile benutzt werden. Diese sind vorkonfiguriert in dem Datenbankdump innerhalb des Verzeichnis `/tests/testDatabaseDump/meteor`. Um neue Profile hinzufügen zu können, muss der Backdoor-Login mit neuen Einträgen erweitert und daraufhin der Datenbankdump aktualisiert werden.

Benutzer mit Admin-Rechten

- "adminLogin" # Super-Admin
- "editorLogin" # Admin

Frontend Benutzer

- "standardLogin" # Google, Twitter oder Facebook Benutzer
- "universityLogin" # CAS Benutzer
- "lecturerLogin" # Dozent
- "blockedLogin" # Blockierter Benutzer
- "firstLogin" # Simulation einer Neuanmeldung

5.12.3 Hilfsfunktionen

Um das Schreiben von Acceptance-Test zu vereinfachen, werden öfters verwendete Abläufe, wie z. B. das Anmelden oder öffnen einer Kartei in Funktionen ausgelagert. Diese befinden sich unter `/tests/features_helper`.

Die Datei **global.js** beinhaltet die einzelnen URL's und den Timeout für einen Acceptance-Test. Die Datei **navigations** besitzt alle Events die nicht zu einer Karte oder einer Kartei gehören. Darunter zählen das Anmelden, Abmelden sowie Funktionen für Überprüfen von Inhalten oder der aktiven URL.

5.12.4 Aufbau

```
tests
├── filter # Die Tests für chimp
├── features_helper # Hilfsfunktionen um das erstellen für die Acceptance-Tests
├── helpers # Shell Skripte für den automatischen Testdurchlauf
└── testDatabaseDumpmeteor # Datenbankdump der für die Acceptance-Tests
```

6 Schlussbetrachtung

Im Anschluss eine Liste an noch offenen Punkten und Problemen, die während der Projektphase und Bearbeitung der Bachelorarbeit aufgetreten sind:

6.1 Codequalität

Während der Bearbeitung dieser Bachelorarbeit erhielt ein Großteil der Templates, zu finden unter `/imports/ui/`, ein Refactoring. Dies wurde durchgeführt, um die Navigation für Entwickler zu vereinfachen und den Aufbau der Templates besser beschreiben zu können. Bereiche wurden in kleine logische Abschnitte unterteilt und in entsprechenden Ordnern gruppiert. Es gibt jedoch noch einige Templates, die ein Refactoring benötigen. Darunter fallen das Main-Template, welches die Hauptnavigation beinhaltet, und das Backend. Neben dem Refactoring der Templates, gibt es noch redundanten JavaScript Code, der in Klassen ausgelagert werden kann.

6.2 Performance

Durch das Refactoring der **Meteor – Subscription**, welche gleichzeitig mit dem Refactoring der Templates stattfand, wurde der Traffic zwischen Server und Client stark reduziert. Anstatt nach der Anmeldung alle öffentlichen Daten in die **Meteor – MiniMongoDB** auszulagern, wird dies nur noch für den Inhalt der derzeit besuchten Kartei oder Repetitorium erledigt. Hierfür besteht noch weiteres Optimierungspotential, indem nur noch die Karten gesendet werden, die in der Ansicht aktiv sind. Dies wird das synchronisieren von 200 bis 400 Karten in einer Kartei reduzieren auf 2 – 3 Karten.

6.3 Aktualisieren von Erweiterungen

cards benutzt momentan teils veraltete Technologien. Diese sollten aktualisiert werden, um die Entwicklung neuer Funktionen zu vereinfachen. Zu diesen Technologien zählen:

- Bootstrap 3 für das responsive Design [OT16], hierfür gibt es bereits Version 4.
- FontAwesome 4 für Vector Icons [FA17], hierfür gibt es bereits eine neuere Version mit einem größeren Umfang an kostenlosen Icons.

6.4 Antwortoptionen

Die Antworten für den Wiederholungs-Algorithmus von Leitner sind derzeit auf zwei Optionen beschränkt: "Gewusst" und "Nicht Gewusst". Diese sollen durch folgende Antwortoptionen erweitert werden:

- Multiple choice Frage
- Single choice Frage
- Ja | Nein Antwort
- Freitext

Durch diese Änderung muss der Benutzer die richtige Antwort wissen, damit seine Karte in das nächst höhere Fach gelangt.

6.5 Offline-Modus

Aufgrund von Synchronisationsproblemen zwischen Client und Server, die durch das Überarbeiten der **Meteor – Subscription** entstanden sind, musste der Offline-Modus entfernt werden. Es ist geplant, diesen in einer Version nach **cards 4.0** wieder einzuführen. Welche Daten offline gespeichert werden oder wie der Offline-Modus aktiviert wird, steht noch zur Diskussion.

6.6 Drag and drop für den Kartei-Index

Diese Funktionalität sollte ursprünglich Bestandteil der Projektphase sein, wurde jedoch aufgrund wechselnder Prioritäten gestrichen. Es gestattet dem Besitzer einer Kartei die Reihenfolge über den Index per Drag and drop zu verändern. Die derzeitige Implementierung sortiert die Karten alphabetisch nach dem Inhalt der Karte. Aufgrund der Anzahl von aktiven Benutzern hat dieses Feature die höchste Priorität.

6.7 3D-Ansicht für Karten

Wie auch die Drag and drop Funktionalität war eine 3D-Ansicht für Karten während der Projektphase angedacht. Es steht noch in Frage, ob die endgültige 3D-Darstellung einem Würfel entsprechen soll, da solch eine Form die Fläche des Inhaltes für Breitbildschirme einschränkt. Eine alternative Darstellungsmöglichkeit wäre die eines vertikalen Karussells [DE18], welches es zusätzlich erlauben würde, mehr als 6 Inhalte für eine Karte darzustellen.

6.8 Fazit

Trotz der oben aufgelisteten Probleme wurde das Ziel, **cards** als vorlesungsbegleitende Lernumgebung weiterzuentwickeln, erreicht. **cards** wird derzeit aktiv an der Technischen Hochschule Mittelhessen für den Bachelorkurs "Webbasierte Systeme" eingesetzt, welcher ein Repetitorium mit Bonusvergabe auf <https://thm.cards> namens "WBS" und auf <https://linux.cards> namens "Linux Essentials" besitzt. Für das Sommersemester 2018 wurde **cards** bereits für den Bachelorkurs "Softwaretechnik" und den Masterkurs "Continuis Delivery" zur Bonusvergabe eingesetzt.

Die noch offenen Punkte sollten innerhalb der nächsten Jahre abgearbeitet sein. Teile von ihnen könnten später Bestandteil einer Masterarbeit werden.

Anhang A: Collections

Wert	Typ	Schlüssel	Verwendung
_id	String	Primary	
name	String		Filter Kriterium für unterschiedliche Einstellungen
enabled	Boolean		Einstellung aktiviert oder deaktiviert

Tabelle 5: MongoDB – Collection: adminSettings

Wert	Typ	Schlüssel	Verwendung
_id	String	Primary	
cardset_id	String		Kartei für den Zugang
token	String		

Tabelle 6: MongoDB – Collection: apiAccess

Wert	Typ	Schlüssel	Verwendung
_id	String	Primary	
subject	String	Primary	Titel der Karte
difficulty	Int32		Level
front	String		Inhaltsbereich Nr.1
back	String		Inhaltsbereich Nr.2
lecture	String		Inhaltsbereich Nr.3
hint	String		Inhaltsbereich Nr.4
top	String		Inhaltsbereich Nr.5
bottom	String		Inhaltsbereich Nr.6
cardset_id	String	Primary	die ID der zugehörigen Kartei
cardType	String		Kartentyp
centerTextElement	[Boolean]		Vertikale Zentrierung pro Seite
alignType	[Int32]		Horizontale Zentrierung pro Seite
date	Date		Erstelldatum
dateUpdated	Date		Aktualisierungsdatum
learningGoalLevel	Int32		Lernziel
learningIndex	String		Lerneinheit
backgroundStyle	Int32		Style für den Hintergrund
originalAuthorName	Object		Original Autor, relevant für importierte Karteien

Tabelle 7: MongoDB – Collection: cards

Wert	Typ	Schlüssel	Verwendung
_id	String	Primary	
name	String		Titel
description	String		Beschreibung
date	Date		Erstelldatum
dateUpdated	Date		Aktualisierungsdatum
owner	String		Ersteller
visible	Boolean		Freigeschaltet
ratings	Boolean		Anzahl Bewertungen
kind	String		Zugangsart
price	Double		Preis
reviewed	Boolean		Überprüft durch Dozent
reviewer	String		Überprüfer
request	Boolean		Überprüfungsantrag offen
relevance	Double		Durchschnittliche Bewertung
quantity	Int32		Anzahl Karten
license	[String]		Verwendete Lizenzen
userDeleted	Boolean		Ist das Konto des Erstellers gelöscht?
learningActive	Boolean		Bonus aktiv?
maxCards	Int32		Tagespensum
daysBeforeReset	Int32		Überschreitungsfrist
learningStart	Date		Beginn der Bonus-Lernphase
learningEnd	Date		Ende der Bonus-Lernphase
registrationPeriod	Date		Anmeldefrist
learningInterval	[Int32]		Wiederholungs-Intervalle in Tagen
wordcloud	Boolean		Wird dieser inhalt auf der Landing-Page angezeigt?
shuffled	Boolean		Ist dieser Inhalt ein Repetitorium oder eine Kartei?
cardGroups	[String]		Referenzierte Karteien des Repetitoriums
cardType	Int32		Karteityp
difficulty	Int32		Level
noDifficulty	Boolean		Level - Allgemein
originalAuthorName	Object		Autor der importierten Kartei
workload	Object		Einstellungen für Leitner und Wozniak
pomodoroTimer	Object		Einstellungen für den Pomodoro-Timer der Kartei

Tabelle 8: MongoDB – Collection: cardsets

Wert	Typ	Schlüssel	Verwendung
_id	String	Primary	
college	String		Hochschule
Course	String		Studiengang

Tabelle 9: MongoDB – Collection: collegesCourses

Wert	Typ	Schlüssel	Verwendung
_id	String	Primary	
name	String		Name des Themes

Tabelle 10: MongoDB – Collection: colorThemes

Wert	Typ	Schlüssel	Verwendung
_id	String	Primary	
card_id	String		Karte des Leitner-Objekts
cardset_id	String	Primary	Kartei des Leitner-Objekts
originalCardset_id	String	Primary	Referenzierte Kartei bei Repertorium
user_id	String	Primary	Lerner
box	Int32		Derzeitiges Fach von 1 - 6
active	Boolean		Wurde die Karte zum lernen Freigeschaltet?
nextDate	Date		Nächstmöglicher Termin zur Freischaltung
currentDate	Date		Letztes Datum indem die Karte beantwortet wurde

Tabelle 11: MongoDB – Collection: leitner

Wert	Typ	Schlüssel	Verwendung
_id	String	Primary	
target	String		Lecturer oder Admin
origin	String		ID des Senders
receiver	String		ID des Empfängers
type	String		Pro-Freigabe oder Meldung
text	String		Text für Meldung
date	Date		Erstelldatum der Nachricht
read	Boolean		Wurde von Empfänger gelesen?
cleared	Boolean		Wurde durch Empfänger bestätigt?
link_id	String		ID der Kartei auf die referenziert wird

Tabelle 12: MongoDB – Collection: notifications

Wert	Typ	Schlüssel	Verwendung
_id	String	Primary	
cardset_id	String	Primary	ID der Kartei für die erworbene Lizenz
user_id	String	Primary	Benutzer der die Kartei gekauft hat
date	Date		Kaufdatum
amount	Double		Betrag

Tabelle 13: MongoDB – Collection: paid

Wert	Typ	Schlüssel	Verwendung
_id	String	Primary	
cardset_id	String	Primary	Kartei für die Wertung
user	String	Primary	ID des Bewerter
rating	Int32		Bewertung

Tabelle 14: MongoDB – Collection: ratings

Wert	Typ	Schlüssel	Verwendung
_id	String	Primary	
createdAt	Date		Erstelldatum
username	String		Login
roles	[String]		Benutzerrechte
services	Object		Anmeldetyp
status	Object		Ist der Benutzer online?
visible	Boolean		Ist der Benutzer sichtbar für andere?
lastOnAt	Date		Letzter Tag der Anmeldung
selectedColorTheme	String		Aktives Theme
mailNotification	Boolean		Sind Mail-Notifikationen aktiv?
webNotification	Boolean		Sind Web-Push-Notifikationen aktiv?
profile	Object		Vorname, E-Mail und Nachname

Tabelle 15: MongoDB – Collection: user

Wert	Typ	Schlüssel	Verwendung
_id	String	Primary	
subscriptions	[Object]		
userId	String	Primary	ID des Benutzers

Tabelle 16: MongoDB – Collection: webPushSubscriptions

Wert	Typ	Schlüssel	Verwendung
_id	String	Primary	
cardset_id	String	Primary	Einstellungen zutreffend auf dieser Kartei
user_id	String	Primary	Einstellungen zutreffend auf diesen Benutzer
leitner	Object		Einstellungen für Leitner-Einstellungen der Kartei

Tabelle 17: MongoDB – Collection: workload

Wert	Typ	Schlüssel	Verwendung
_id	String	Primary	
card_id	String		Karte für das Wozniak-Objekt
cardset_id	String	Primary	Kartie für das Wozniak-Objekt
user_id	String	Primary	Benutzer für das Wozniak-Objekt
ef	Double		Berechneter Wert über die Schwierigkeit der Karte
intervals	Int32		Wann soll die Karte wieder vorgelegt werden?
reps	Int32		Wie oft wurde die Karte wiederholt?
nextDate	Date		Nächster Tag für die Freischaltung

Tabelle 18: MongoDB – Collection: wozniak

Wert	Typ	Verwendung
“bonus.maxPoints”	Int32	Maximale Anzahl erreichbarer Bonuspunkte
“bonus.count”	Int32	Anzahl der eingeschriebenen Benutzer in Bonus
“normal.count”	Int32	Anzahl der eingeschriebenen Benutzer ohne Bonus

Tabelle 19: MongoDB – Collection: cardset workload Objekt

Wert	Typ	Verwendung
breakLength	Int32	Länge der Pause in Minuten
quantity	Int32	Anzahl der Pomodori um das Lernziel zu erfüllen
soundConfig	[Boolean]	Anzahl der eingeschriebenen Benutzer ohne Bonus
workLength	Int32	Länge der Lerneinheit in Minuten

Tabelle 20: MongoDB – Collection: cardset pomodoroTimer Objekt

Anhang B: Diagramme

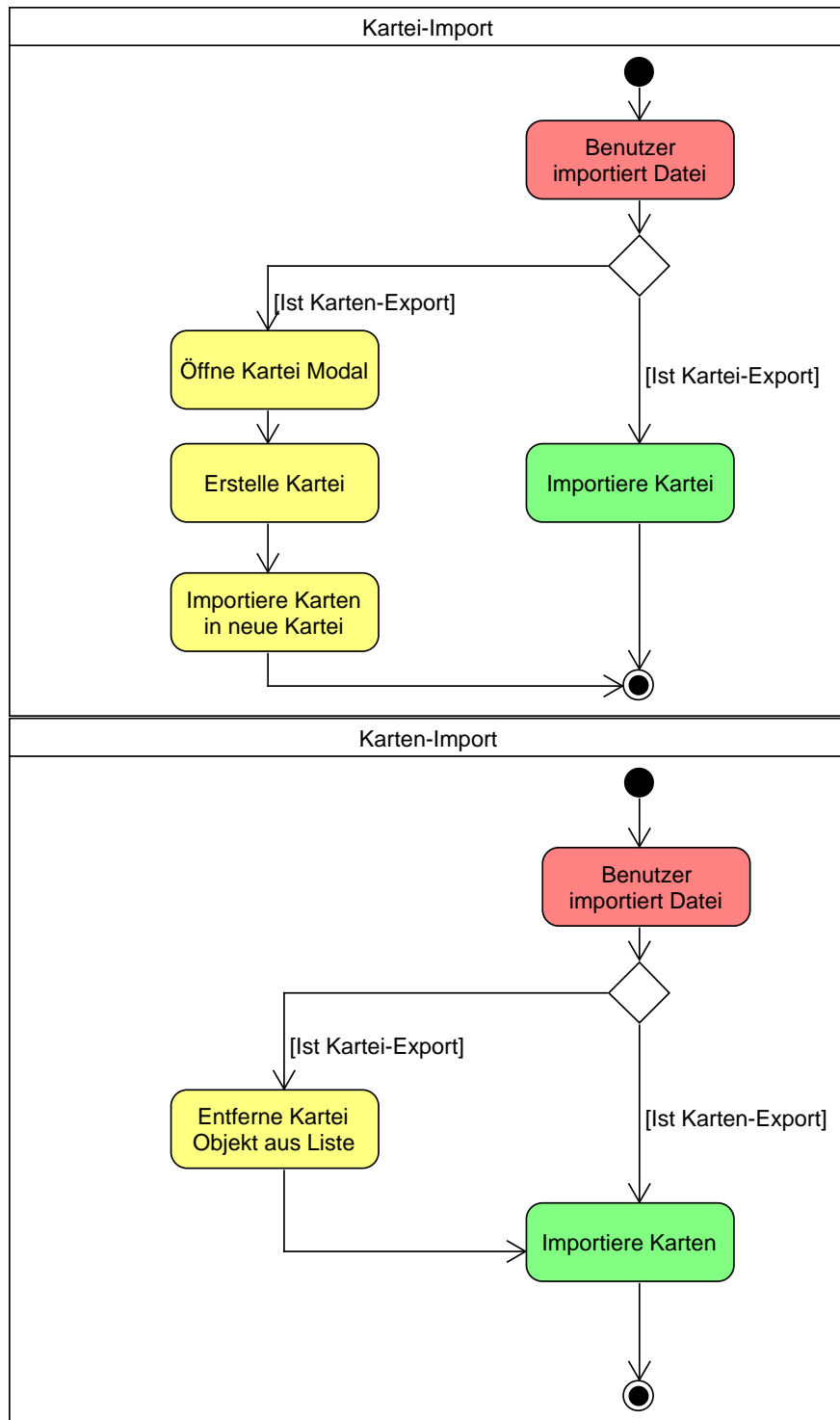
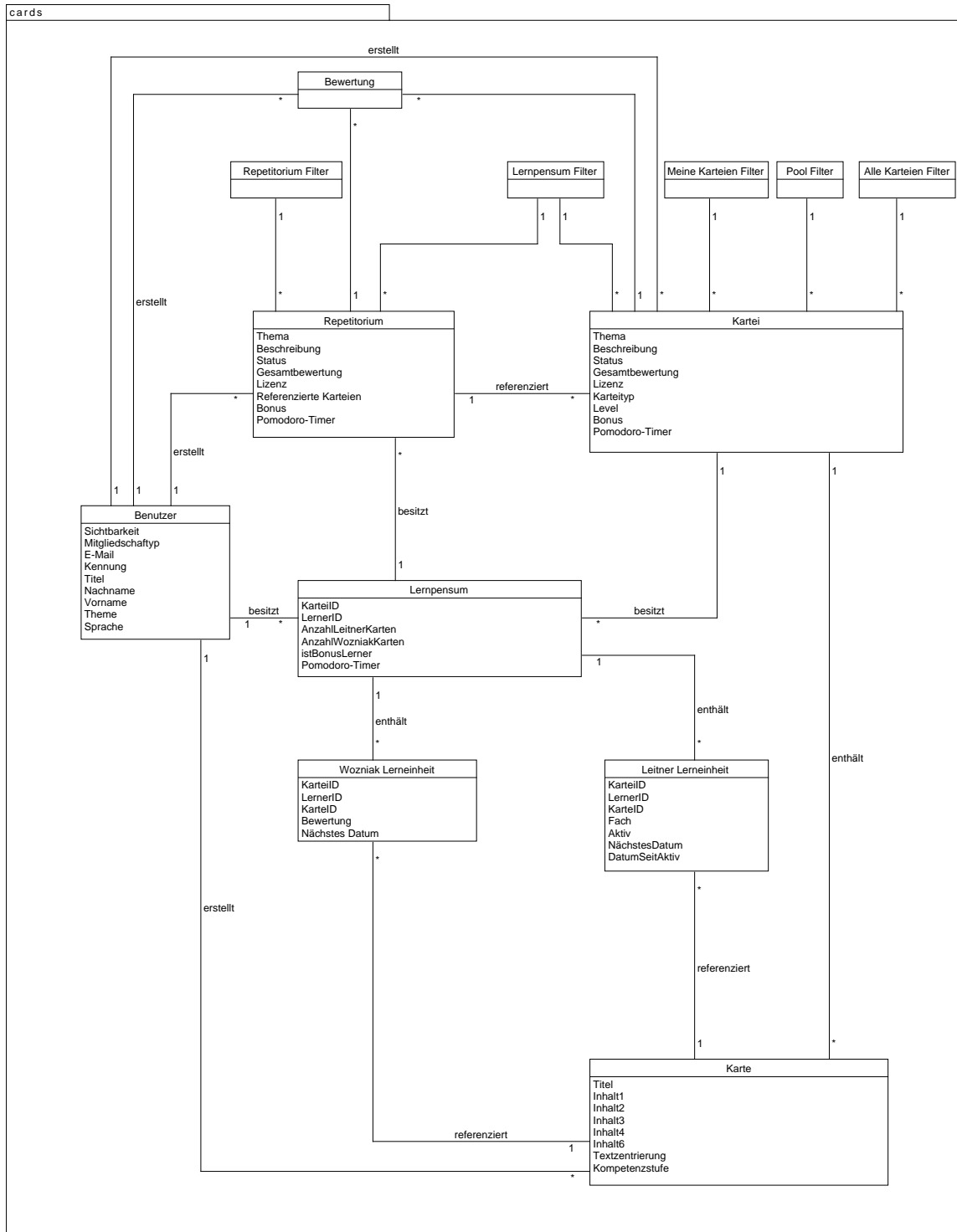


Abbildung 26: cards 3.6 – Aktivitätsdiagramm: Import von Karten und Karteien



Projekt:	cards
Projektort und -zeitraum:	THM, FB MNI, SoSe 2017, v2 WiSe 2018/2019
Projekt-Repository:	https://git.thm.de/cards
Release:	v3.6
Staging Server:	http://staging.arsnova.cards
Produktionsserver:	http://thm.cards
UML-Diagrammart:	Domänendiagramm
UML-Tool:	UMLet v14.3
Modellversion:	v2.1
Erstelldatum:	31.05.2017
Ersteller/in:	Simon Heuser, Hermann Lallah
Letzte Änderung:	20.12.2018
Letzter Bearbeiter:	Curtis Adam
E-Mail:	curtis.adam@mni.thm.de

Abbildung 27: cards 3.6 – Domänenmodell

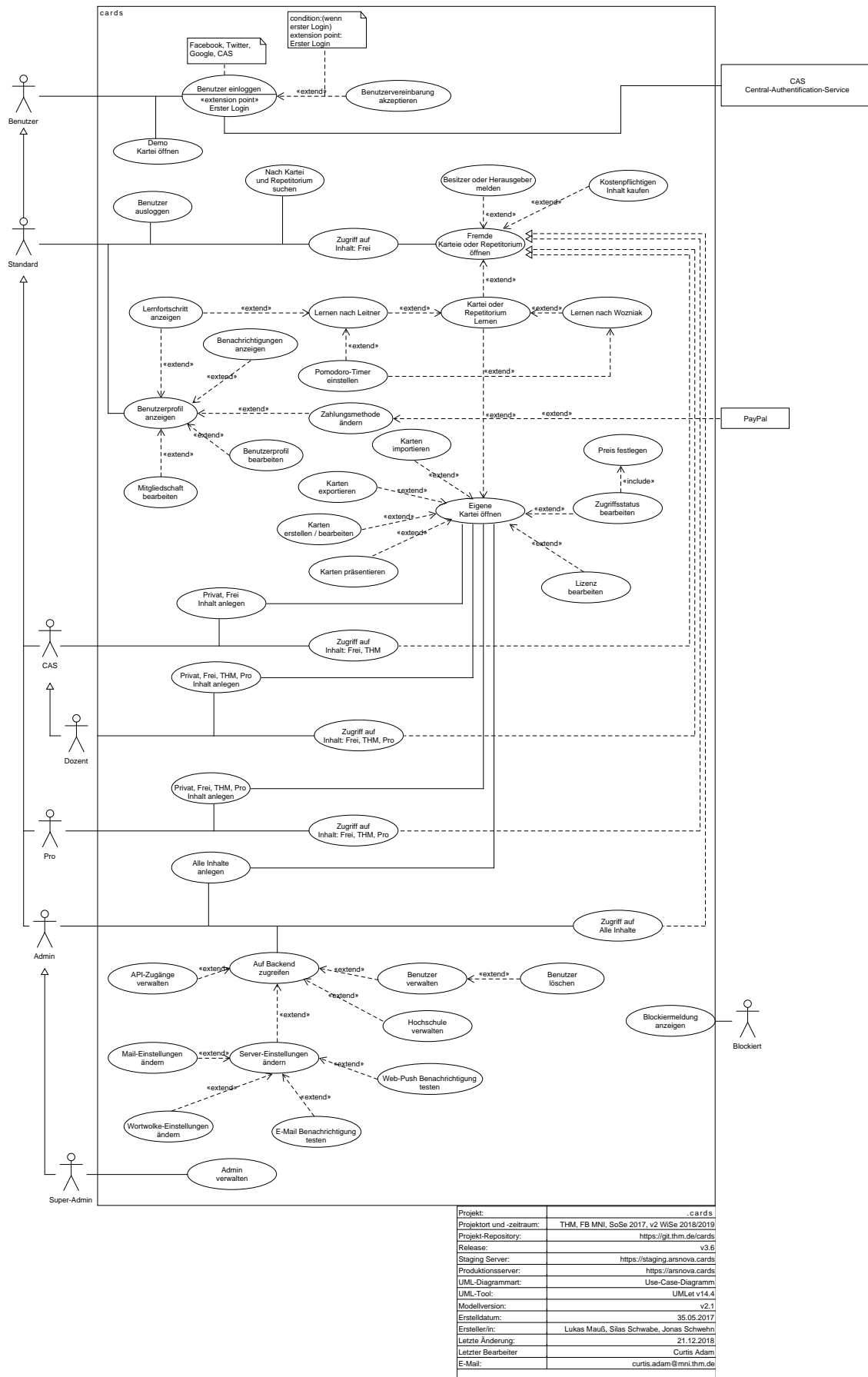
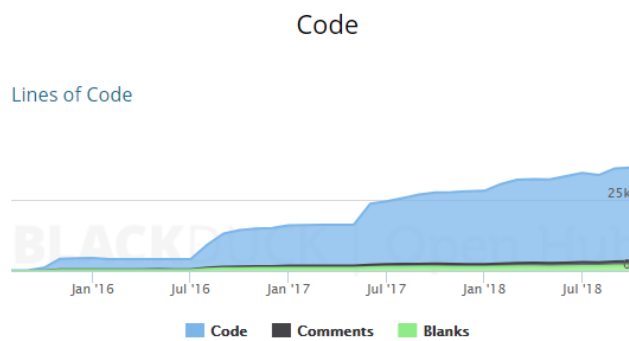
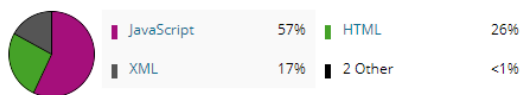


Abbildung 28: cards 3.6 – Anwendungsfalldiagramm



Languages



30 Day Summary

Sep 14 2018 — Oct 14 2018

137 Commits
9 Contributors

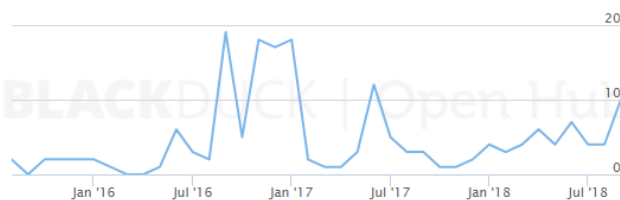
12 Month Summary

Oct 14 2017 — Oct 14 2018

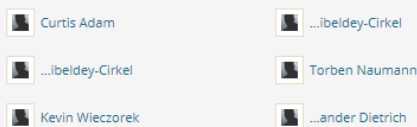
2445 Commits
Up + 1450 (145%) from previous 12 months
20 Contributors
Down -20 (50%) from previous 12 months

Community

Contributors per Month



Most Recent Contributors



Ratings

1 user rates this project:
★★★★★ 5.0/5.0

Click to add your rating

☆☆☆☆☆
Review this Project!

Abbildung 31: cards 3.6 – OpenHub-Statistiken

Abbildungsverzeichnis

1	Die cards Demo-Kartei [QU18]	2
2	cards Filter für öffentliche Karteien	4
3	Leitner- Wiederholungs-Algorithmus von cards	5
4	Lernmodus von Anki (Desktop)	8
5	Lernmodus von Mnemosyne	10
6	Lernmodus von Quizlet (Antworten)	12
7	Lernpensum von SuperMemo	14
8	Lernmodus von SuperMemo	14
9	WebStorm: JSCS Einstellungen	18
10	WebStorm: JSHint Einstellungen	18
11	Gulp: Test fehlgeschlagen	19
12	Gulp: Test erfolgreich	19
13	GitLab: Branch und Merge-Request anlegen	22
14	GitLab: Pipeline Status für Merge-Request	22
15	GitLab: Assigene für Merge-Request setzen	22
16	Karte & Karten-Editor: Template	32
17	Hintergrundfarben für Karten der Schwierigkeitsstufe 0 (Ohne Level)	33
18	Hintergrundfarben für Karten der Schwierigkeitsstufe 1 (Basic)	33
19	Hintergrundfarben für Karten der Schwierigkeitsstufe 2 (Fortgeschritten)	33
20	Hintergrundfarben für Karten der Schwierigkeitsstufe 3 (Experte/Expertin)	33
21	Hintergrundfarben für Notizen	33
22	Hintergrundfarbe für den Präsentationsmodus	33
23	Kartei: Template	35
24	Filter: Template	37
25	Filter: Das neue Design für die Ergebnisliste des Filters	37
26	cards 3.6 – Aktivitätsdiagramm: Import von Karten und Karteien	vi
27	cards 3.6 – Domänenmodell	vii
28	cards 3.6 – Anwendungsfalldiagramm	viii
29	cards 3.6 – Aktivitätsdiagramm: Frontend	ix
30	cards 3.6 – Aktivitätsdiagramm: Backend	x
31	cards 3.6 – OpenHub-Statistiken	xi

Tabellenverzeichnis






1	Die cards Ausstattungsmerkmale der Karteitypen [QU18]	3
2	Benutzerrollen für cards	6
3	Zusammenfassung für Vergleich der Lernkartei-Apps	15
4	Seiten-Aufbau einer Karte	29
5	MongoDB – Collection: adminSettings	i
6	MongoDB – Collection: apiAccess	i
7	MongoDB – Collection: cards	i
8	MongoDB – Collection: cardsets	ii
9	MongoDB – Collection: collegesCourses	iii
10	MongoDB – Collection: colorThemes	iii
11	MongoDB – Collection: leitner	iii
12	MongoDB – Collection: notifications	iii
13	MongoDB – Collection: paid	iv
14	MongoDB – Collection: ratings	iv
15	MongoDB – Collection: user	iv
16	MongoDB – Collection: webPushSubscriptions	iv
17	MongoDB – Collection: workload	v
18	MongoDB – Collection: wozniak	v
19	MongoDB – Collection: cardset workload Objekt	v
20	MongoDB – Collection: cardset pomodoroTimer Objekt	v

Listings

1	i18n Template für den WebSocket-Verbindung Status	20
2	Aufruf eines i18n Strings innerhalb eines Templates und mit JavaScript	20
3	Meteor – Block Helper für Übersetzungen	20
4	Aufbau der Hilfsklasse für Karten-Effekte	21
5	Default Theme Variablen für den Facebook-Login Button	21
6	Mapping der neuen Farben innerhalb von themeSwitcher.scss	21
7	Beispiel: Filtern der Hilfe für Meine Kartein	24
8	Beispiel: Standardwerte für das Pomodoro-Timer Formular	25
9	Bonus Pomodoro-Timer Objekt innerhalb eines Cardset MongoDB – Document	26
10	SweetAlert2 Nachrichten des Pomodoro-Timer in den i18n Dateien	26
11	Beispiel: Daten für das Erzeugen der Wortwolke	27
12	Beispiel: Globale Eigenschaften für Karteitypen	28
13	Beispiel: Seiten Eigenschaften für eine Vokabel Kartei	28
14	Ausschnitt aus der Konfiguration für die Filter-Liste	36
15	Sessions für das speichern der einzelnen Filtereinstellungen	36
16	Beispiel: MongoDB Bezahlungs-Objekt	39
17	Kopfbereich für einen iron-router Pfad	39
18	Beispiel: MongoDB Leitner-Objekt	40
19	Beispiel: MongoDB Wozniak-Objekt	41
20	Beispiel: MongoDB CollegeCourse Objekt	42
21	Beispiel: AdminSettings MongoDB – Document für Test-Nachrichten	43

Literatur

- [BA16] Marty Banting, Project Ricochet Team, Top 10 Meteor Performance Problems <https://projectricochet.com/blog/top-10-meteor-performance-problems/>. September 2016.
- [BI18] Peter Bientsman, The Mnemosyne Project. Homepage. <https://mnemosyne-proj.org/>. Version 2.6.1, August 2018.
- [BT18] TheBrain team. Chimp. GitHub Repository. <https://github.com/TheBrainFamily/chimp>. Version 0.54.0, August 2018.
- [CU18] Cure53 Team. DOMPurify. GitHub Repository und API Dokumentation. <https://github.com/cure53/DOMPurify>. Version 1.0.8, November 2018.
- [DE18] Dave DeSandro. Code-Pen Projekt. Carousel - dynamic. <https://codepen.io/desandro/pen/wjeBpp>. Release 2018.
- [DH17] Lokesh Dhakar. Lightbox2. API Dokumentation. <https://lokeshdhakar.com/projects/lightbox2/>. Version 2.1.0, November 2017.
- [ED18] Tristan Edwards, Limon Monte. SweetAlert2. API Dokumentation. <https://sweetalert2.github.io>. Version 7.28.2, September 2018.
- [EL18] Damien Elmes. Anki Karteikarten-Lernsystem. Homepage. <https://apps.ankiweb.net/>. Version 2.1.4, September 2018.
- [FA17] Fonticons, Inc.. Font Awesome. Homepage. <https://fontawesome.com>. Version 4.7.0, Dezember 2017.
- [GL17] Ryan Glover. The Meteor Chef, Organize your Meteor Porject. <https://themetorchef.com/tutorials/organizing-your-meteor-project/> April 2017.
- [GO18] Google LLC. Firebase. Homepage. <https://console.firebase.google.com/>. Release 2018.
- [GU18] Timothy Guan-tin Chien. WordCloud2. GitHub Repository und API Dokumentation. <https://github.com/timdream/wordcloud2.js> Version 1.1.0, Februar 2018.
- [JB18] JetBrains, Webstorm. IDE Download-Page. <https://www.jetbrains.com/webstorm/>. Version 2018.3, Dezember 2018.
- [MA16] Kyle Banker, Peter Bakkum, Shaun Verch, Douglas Garrett, Tim Hawkins. *Meteor in Action - Second Edition, Manning Publications Co. - 2016*. Kapitel 4.4.6.
- [MA17] Chris Mather. iron-router. GitHub Repository und API Dokumentation. <http://iron-meteor.github.io/iron-router/>. Version 1.1.2, April 2017.
- [MC18] Morgan McGuire. Markdeep. API Dokumentation. <https://casual-effects.com/markdeep/>. Version 1.03, Dezember 2018.
- [ME18] Meteor Development Group Inc.. Meteor. Dokumentation. <https://docs.meteor.com>. Version 1.8.0.1, November 2018.

- [MO16] Stephan Hochhaus, Manuel Schoebel.
MongoDB in Action - Second Edition, Manning Publications Co. - 2016. Kapitel 8.2.2.
- [MO18] MongoDB, Inc. MongoDB. Dokumentation.
<https://docs.mongodb.com>. Version 4.0.4, November 2018.
- [NO18a] Node.js Foundation. Node.js. API Dokumentation.
<https://nodejs.org/en/docs>. Version 11.4.0, Dezember 2018.
- [NO18b] Node.js Foundation. Node.js. Version 8 Download.
<https://nodejs.org/dist/latest-v8.x/>. Version 8.14.0, November 2018.
- [OT16] Mark Otto, Jacob Thornton. Bootstrap. API Dokumentation.
<https://getbootstrap.com/docs/3.3/>. Version 3.3.7, Juli 2016.
- [PA18] PayPal. Braintree. Homepage.
<https://www.braintreepayments.com/de>. Release 2018.
- [PE16] Percolate Studio. Publish Counts. GitHub Repository und API Dokumentation.
<https://github.com/percolatestudio/publish-counts>. Version 0.8.0, Dezember 2016.
- [QU18] Klaus Quibeldey-Cirkel, cards Demo-Kartei.
<https://thm.cards/demo>. Release 3.6, Dezember 2018.
- [SO18] Sindre Sorhus. Screenfull.js. GitHub Repository und API Dokumentation.
<https://sindresorhus.com/screenfull.js/>. Version 3.3.3, September 2018.
- [SP18] Marcus Spiegel. i18n. GitHub Repository und API Dokumentation.
<https://github.com/mashpie/i18n-node> Version 0.8.3, Juni 2018.
- [SUP18] SuperMemo World. SuperMemo. Homepage.
<https://www.supermemo.com>. Release 2018.
- [SUT18] Andrew Sutherland. Quizlet Inc.. Quizlet. Homepage.
<https://quizlet.com/mission> Release 2018.
- [TH15] ARSnova-Team der TH Mittelhessen. Git-Lab cards Repository.
<https://git.thm.de/arsnova/cards>. August 2015.
- [TH17a] ARSnova-Team der TH Mittelhessen. Release-Notes für cards 3.1
<https://git.thm.de/arsnova/cards/tags/v.3.1.0>. Januar 2017.
- [TH17b] ARSnova-Team der TH Mittelhessen. Release-Notes für cards 3.1.5
<https://git.thm.de/arsnova/cards/tags/v.3.1.5>. Juni 2017.
- [TH18] ARSnova-Team der TH Mittelhessen. 3.5 Release-Notes für cards
<https://git.thm.de/arsnova/cards/tags/v.3.5>. September 2018.
- [WI18] John Wilkos. Code-Pen Projekt. Pomodoro-Timer Template.
<https://codepen.io/kevyman/pen/ZGNBBN>. Release 2018.

Glossar

- Meteor** Ein auf [Node.js](#) basierendes Web-Framework mit dem Fokus auf eine konsistente JavaScript-API zwischen Client und Server. Legt einen besonderen Wert auf Echtzeit, Reaktivität, Rapid Prototyping und Wiederverwendung von Code [ME18]. 16, 17, 29, 39, 43
- Meteor – Block Helper** Eine von [Meteor](#) bereitgestellter Spacebars Helfer, der einen HTML-Block nimmt und darstellen kann. Ein Helfer wird mit `{{#NAME}}` geöffnet und mit `{{/NAME}}` geschlossen, um eine reine String-Ausgabe zu erzeugen. Für die Ausgabe von HTML müssen drei geschweifte Klammern verwendet werden. Von [Meteor](#) bereits vordefinierte Helfer sind **each**, **with** und **if** [MO18]. 20, 29
- Meteor – Method** [Meteor](#)’s Remote Procedure Call (RPC) System, welches Benutzereingaben und Daten des Clients auf dem Server speichert. Sie dient als API-Endpunkt des [Meteor](#)-Servers und kann auf dem Client oder Server definiert werden. Einer Methode können mehrere Daten übergeben werden und sie kann einen Rückgabewert enthalten [ME18]. 23, 40, 41
- Meteor – MiniMongoDB** Eine nicht persistente, In-Memory Miniaturdatenbank Variante der [MongoDB](#), welche in der Regel eine Teilmenge der aktuellen Serverdaten verwaltet und in reinem JavaScript geschrieben ist. Sie ist nicht abhängig vom HTML5 Local-Storage und existiert nur im Speicher des Browsers. Mit ihr steht nur ein beschränkter Funktionsumfang gegenüber der [MongoDB](#) zur Verfügung [MO18]. 34, 45
- Meteor – Session** Ein Globales [Meteor](#) Objekt, welches einen beliebigen Satz von Schlüssel-Werten speichern kann und über die gesamte Browser-Sitzung aktiv ist. Die Besonderheit der Session ist die Reaktivität. Templates die den Inhalt durch Aufruf von `Session.get()` verwenden, werden automatisch aktualisiert, falls dieser sich durch `Session.set()` ändert [ME18]. 24, 30, 36, 39, 41
- Meteor – Subscription** Eine Abonnement auf eine Publikation des [Meteor](#)-Servers. Eine Publikation stellt einen Satz von Daten bereit, die aus einer [MongoDB – Collection](#) stammen. Dieser Datensatz wird im Cache der [Meteor – MiniMongoDB](#) gespeichert und wird automatisch mit den Daten des Servers synchronisiert [ME18]. 38, 39, 41, 45, 46
- MongoDB** Eine Dokumenten-orientierte NoSQL-Datenbank, welche Sammlungen von JSON-ähnlichen Dokumenten verwalten kann. Sie ist standardmäßig die Datenbank für [Meteor](#) Anwendungen [MO18]. xvi, 16, 17, 19, 24, 36, 39–43
- MongoDB – Collection** Eine Ansammlung von [MongoDB – Document](#). Diese ist Analog zu den Tabellen in relationalen Datenbanken [MO18]. i–v, 26, 27, 34, 39, 42
- MongoDB – Document** Ein Datensatz innerhalb einer [MongoDB](#), welcher aus [MongoDB – Field](#) und Werten besteht. Sie ähneln den Aufbau eines JSON-Objekts [MO18]. 26, 27, 41–43
- MongoDB – Field** Ein Wert innerhalb eines [MongoDB – Document](#). Werte können [MongoDB – Document](#), Arrays oder Arrays von anderen [MongoDB – Document](#) beinhalten [MO18]. 26, 27
- Node.js** Eine serverseitige Plattform, die mit der JavaScript-Laufzeitumgebung ”V8” ausgeführt wird. [Meteor](#) enthält Kopien der node.js binaries [NO18a]. 18, 19