

# **Konzeption und Umsetzung einer Erweiterung zur Meteor basierten Webapplikation "Projektor"**

## **Bachelor-Thesis**

zur Erlangung des akademischen Grades B.Sc.

**Adrian Abbassian**

2143421



Hochschule für Angewandte Wissenschaften Hamburg  
Fakultät Design, Medien und Information  
Department Medientechnik

Erstprüfer: Prof. Dr. Torsten Edeler

Zweitprüfer: Prof. Dr. Andreas Plaß

Hamburg, 30. August 2017

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>5</b>
1.1	Ausgangssituation . . . . .	5
1.2	Ziel der Bachelorarbeit . . . . .	5
1.3	Aufbau . . . . .	5
<b>2</b>	<b>Meteor</b>	<b>6</b>
2.1	Geschichte . . . . .	6
2.2	Überblick der aktuellen Webentwicklung . . . . .	7
2.3	Grundstruktur . . . . .	7
2.3.1	Die sieben Grundsätze von Meteor . . . . .	8
2.3.2	Isomorphe Frameworks . . . . .	9
2.3.3	Blaze . . . . .	9
2.3.4	MongoDB . . . . .	10
2.3.5	Package: . . . . .	11
2.3.6	Distributed Data Protocol . . . . .	11
2.3.7	Templates . . . . .	12
2.3.8	Spacebars . . . . .	13
2.3.9	Routing . . . . .	13
<b>3</b>	<b>Anforderungsanalyse</b>	<b>15</b>
3.1	Projektfindung an deutschen Hochschulen . . . . .	15
3.1.1	Umfrage zur Projektfindung und Verwaltung an Hochschulen . . . . .	16
3.1.2	Auswertung der Umfrageergebnisse . . . . .	21
3.2	Bestehende Systeme an der HAW . . . . .	22
3.2.1	EMIL . . . . .	23
3.2.2	Helios . . . . .	23
3.3	Webapplikation Projektor . . . . .	24
3.3.1	Funktionalität . . . . .	25
3.3.2	Umsetzung . . . . .	26
3.4	Erweiterung der Zielgruppe . . . . .	26
3.5	Ergebnis der Anforderungsanalyse . . . . .	27
<b>4</b>	<b>Konzeption und Implementierung</b>	<b>28</b>
4.1	Beschreibung der Kurserweiterung . . . . .	28
4.1.1	Kursverwaltung . . . . .	29

## *Inhaltsverzeichnis*

4.1.2	Differenzierung der Nutzer . . . . .	34
4.1.3	Projektverwaltung im Kurs . . . . .	35
4.1.4	Mitgliederverwaltung . . . . .	37
4.2	Bewertungssystem . . . . .	37
4.3	Generierung von Excel-Tabellen . . . . .	40
4.4	Datenbankstruktur . . . . .	44
4.5	Sicherheit . . . . .	46
4.5.1	Kommunikation zwischen Server und Client . . . . .	46
4.5.2	Publish . . . . .	48
4.5.3	Subscribe . . . . .	49
<b>5</b>	<b>Evaluation</b>	<b>51</b>
5.1	Fazit . . . . .	51
5.2	Ausblick . . . . .	52
	<b>Listings</b>	<b>54</b>
	<b>Abbildungsverzeichnis</b>	<b>55</b>
	<b>Tabellenverzeichnis</b>	<b>56</b>
	<b>Literaturverzeichnis</b>	<b>57</b>

## **Abstract**

The aim of this Bachelor thesis is to extend and improve the project-search platform „Projektor“ using the Meteor Framework, upon which this technology is based. An extension is developed and integrated with the Projektor system architecture that has been based on analysis of questionnaire responses from particular topic areas and the university's existing project management structure. The class extension will provide a long-lasting and sustainable improvement to the existing project-finding and managing process for the course planning platform.

## **Zusammenfassung**

Das Ziel der Bachelorarbeit ist es, die Projektfindungs-Plattform „Projektor“ mit dem Framework Meteor, worauf ihre Grundtechnologie beruht, zu erweitern und zu verbessern. Durch Umfragen und Analysen aus bestimmten Themenbereichen des Projektmanagements an Hochschulen, wird eine Erweiterung entwickelt, welche konzipiert und in die momentane Systemarchitektur des Projektors integriert wird. Die Kurserweiterung soll das Finden und Managen von Projekten innerhalb eines Kurses nachhaltig verbessern und fördern.

# 1 Einleitung

Das folgende Kapitel beschäftigt sich zunächst mit der zugrundeliegenden Ausgangssituation. Anschließend werden Ziele und Aufbau der Bachelorarbeit dargestellt.

## 1.1 Ausgangssituation

Zu Beginn wird eine grobe Zusammenfassung über die Struktur der Arbeit und deren Ziele gegeben sowie eine Vertiefung in das verwendete Framework „Meteor“. Darauf folgt eine Anforderungsanalyse, aus der sich die Konzeption und Umsetzung dieser Erweiterung entwickelt. Abschließend wird eine Evaluation zur Verwirklichung ebendieser Erweiterung und deren Zukunft dargestellt.

## 1.2 Ziel der Bachelorarbeit

Das Ziel der Bachelorarbeit ist eine voll integrierte Erweiterung in die „Teamfindungs-Plattform Projektor“, ohne den eigentlichen Sinn, das Finden von Projekten und Projektpartnern, zu verändern. Es wird das Framework Meteor genutzt, mit dem auch der Projektor selbst umgesetzt wurde. Beschrieben wird eine Herangehensweise, mit der eine sinnvolle Erweiterung, von deren Entstehung bis hin zu deren Entwicklung, umgesetzt wird. In dieser Arbeit werden alle erlernten Fähigkeiten, welche man sich während des Studiums angeeignet hat, praktisch umgesetzt. Zudem wurden neue Fähigkeiten wie Meteor und neue Technologien erlernt.

## 1.3 Aufbau

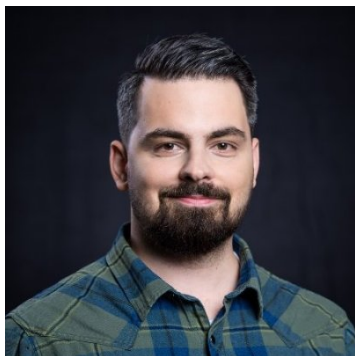
Dadurch das eine Erweiterung entwickelt wird, für eine schon vorhandenen Plattform, beginnt Kapitel 2 mit der Vertiefung in das Framework Meteor. Als nächstes folgt Kapitel 3, mit der Anforderungsanalyse. In Kapitel 4 wird sich mit der Konzeption und Umsetzung der Erweiterung befasst. Zum Schluss folgt Kapitel 5, mit der Evaluation der Arbeit.

## 2 Meteor

In diesem Kapitel wird das **Meteor Framework** mit all seinen Facetten erläutert. Zunächst wird auf die Entstehung von Meteor eingegangen. Anschließend folgt eine Vorstellung des Programms sowie eine Unterscheidung zu anderen Frameworks. Zum Schluss werden die Grundstruktur im Ganzen und deren Funktionalitäten im Einzelnen erklärt.

### 2.1 Geschichte

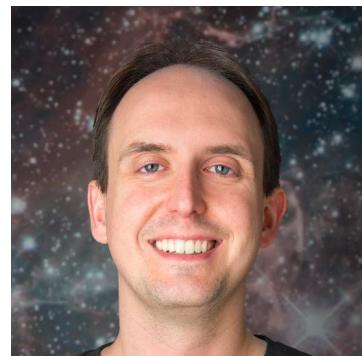
Ursprünglich hatten die Gründer der Meteor Development Group (MDG) Nick Martin (2.1(a)), Matt DeBergalis (2.1(b)) und Geoff Schmidt (2.1(c)) vor, mit einer Reise-Empfehlungsseite erfolgreich zu werden. Diese sollte von Y Combinator unterstützt werden.



(a) Nick Martin



(b) Matt DeBergalis



(c) Geoff Schmidt

**Abbildung 2.1:** Meteor Development Group Gründer

Y Combinator unterstützt Jungunternehmen mittels eines Mentorenprogramms und der Hilfe beim Crowdfunding dabei, auf dem Arbeitsmarkt Fuß zu fassen. In Betracht der großen Konkurrenz realisierten sie jedoch rasch, dass die Aussichten auf Erfolg gering waren. So entschieden sie sich an einer neuen Idee zu arbeiten, „**Skybreak**“. Hierbei handelte es sich um eine *Open Source Plattform*. Diese boten sie einer Sound Stiftung an, mit dem Versprechen, dass es eine Webapplikation sein wird, welche genau so flüssig und sauber wie eine Desktopapplikation laufen wird. Im Dezember 2011 kündigte die Meteor Development Group den ersten Preview Release von Skybreak an. Skybreak wurde aber kurz darauf zu **Meteor** umbenannt.

Kein Jahr später sahen viele große Konzerne, unter anderem Peter Levine (ehemaliger CEO von XenSource), Dustin Moskovitz (Mitgründer von Facebook) und Rod Johnson (Gründer von SpringSource) sehr viel Potential in dem Projekt und unterstützten es mit 11.2 Millionen Dollar. Seitdem gehört das Meteor GitHub Repository zu den Top 20 der beliebtesten Repositorys auf GitHub [\[HOC\]](#).

## 2.2 Überblick der aktuellen Webentwicklung

In den letzten Jahren hat sich die Webentwicklung rapide weiterentwickelt und steuert zwei klare Trends an. Webapplikationen werden immer schneller und flexibler. Es lassen sich immer weniger Unterschiede zu herkömmlichen Desktop Anwendungen ausmachen. Nutzer einer Applikation interessiert es nicht mehr welche Technologien für diese genutzt werden. Es geht ihnen lediglich darum, dass die Anwendung leicht und schnell zu verstehen ist. Der zweite Trend bezieht sich auf die zu schnelle Entwicklung von neuen Bibliotheken, Programmiersprachen und Frameworks. Kein Entwickler ist dazu in der Lage bei dieser Entwicklung mitzuhalten [\[HOC\]](#). Kurze Zusammenfassung der Trends :

- Nutzer erwarten selbsterklärende Applikationen
- Überforderung für Entwickler durch ständig neue Technologien.

Um den Entwicklern diesen Druck zu nehmen, ist Meteor die perfekte Alternative [\[TUR\]](#):

- JavaScript reichen aus, um moderne, in Echtzeit laufende Webapplikationen, sowohl für den Desktop als auch für mobile Plattformen, zu entwickeln.
- Als Einsteiger kann man ganz schnell, professionell wirkende Applikationen entwickeln.
- Es gibt eine sehr aktive *Community*, welche eigenständig weltweit Treffen organisiert, frei zugängliches Material veröffentlicht und jedem Hilfe z.B. durch Foreneinträge anbietet.

## 2.3 Grundstruktur

Meteor ist eine Open-Source Web und mobile Entwicklerplattform, welche auf dem MEAN<sup>1</sup> Schema basiert. Der Entwickler ist in der Lage, mit JavaScript als einzige Programmiersprache, eine komplette, in Echtzeit zwischen Client und Server kommunizierende und reaktive Applikation zu entwickeln [\[HOC\]](#).

---

<sup>1</sup>MEAN bezieht sich auf alle Applikationen, welche auf MongoDB, Node.js, Angular und Express.js aufgebaut sind. Es gibt viele Abwandlungen von MEAN, wie MEEN: MongoDB, Ember.js, Express und Node.js [\[HOC\]](#).

Die wichtigsten Bestandteile von Meteor sind [\[GAN\]](#):

- **Server:** Eine Full-Stack<sup>2</sup> Plattform wie Meteor, benötigt immer einen Webserver.
- **Datenbank:** Meteor liefert eine voll integrierte Datenbank, die **MongoDB**.
- **Kommunikation:** Für die Kommunikation zwischen Server und Client wurde ein selbst entwickeltes Datenprotokoll integriert, **DDP (Distributed Data Protocol)**.
- **Frontend UI:** Meteor nutzt hauptsächlich **Blaze** für die Nutzung der *Template-Engine*. Zusätzlich unterstützt es auch Angular und React<sup>3</sup>.

Außerdem bietet Meteor einen öffentlichen Package-Server an, auf dem alle Packages liegen, sowohl von Meteor selbst oder von Nutzer entwickelte. Um diesen Server nach bestimmten Packages leicht durchsuchen zu können, wird empfohlen das Web-Interface „Atmospherejs“ zu nutzen [\[HOC\]](#). Meteor funktioniert aber auch in Kombination mit NPM (Node Package Manager).

### 2.3.1 Die sieben Grundsätze von Meteor

Meteors Grundstruktur beruht auf sieben Grundsätze [\[BAN\]](#) :

- *Data on the Wire.* Don't send HTML over the network. Send data and let the client decide how to render it.
- *One Language.* Write both the client and the server parts of your interface in JavaScript.
- *Database Everywhere.* Use the same transparent API to access your database from the client or the server.
- *Latency Compensation.* On the client, use prefetching and model simulation to make it look like you have a zero-latency connection to the database.
- *Full Stack Reactivity.* Make realtime the default. All layers, from database to template, should make an event-driven interface available.
- *Embrace the Ecosystem.* Meteor is open source and integrates, rather than replaces, existing open source tools and frameworks.
- *Simplicity Equals Productivity.* The best way to make something seem simple is to have it actually be simple. Accomplish this through clean, classically beautiful APIs.

---

<sup>2</sup>Jede Ebene einer Anwendung, von der Datenbank bis zum HTML, reagiert auf jede Veränderung die passiert [\[LIE\]](#)

<sup>3</sup>JavaScript-Webframeworks



### 2.3.2 Isomorphe Frameworks

Meteor gehört zu den isomorphen Frameworks. Dies bedeutet, dass der gleiche JavaScript Code sowohl auf dem Client als auch auf dem Server und selbst auf der Datenbank durchlaufen wird. Viele andere Frameworks sind nicht in der Lage auf beiden Ebenen mit dem gleichen Code zu arbeiten, da die Schnittstelle nicht gut genug integriert wurde. Zum Beispiel könnten Angular (für das Frontend) und Express.js (für das Backend) nicht gut zusammenarbeiten. Durch die Isomorphie ist Meteor in der Lage eine einheitliche API<sup>4</sup> zum Laufen zu bringen, in der alle Kernprozesse des Frameworks auf allen Ebenen ausgeführt werden. Dadurch müssen sich Entwickler keine unterschiedlichen Frameworks für die einzelnen Ebenen der Anwendungen aneignen [HOC].

### 2.3.3 Blaze

Blaze ist eine *reactive templating engine*<sup>5</sup>. Die Hauptaufgabe von Blaze ist es bestimmte Daten an das dazugehörige HTML-Template zu binden und damit die *Reactivity* zu sichern. Blaze reichen einfache Befehle aus, die ihm sagen „was mit den Daten zu tun ist“ und nicht „wie es zu tun ist“ [BVJ] (siehe Abb.: 2.4).

Fullstack Reactivity	
Traditional programming	Reactive programming
<pre>var a = 2; var b = 5; var c = a + b; console.log(c); # c is 7</pre>	<pre>var a = 2; var b = 5; var c = a + b; console.log(c); # c is 7</pre>
<pre>a = 5; console.log(c); # c is still 7</pre>	<pre>a = 5; console.log(c); # c is magically 10</pre>
<pre>c = a + b; console.log(c); # c is finally 10</pre>	

Abbildung 2.2: Fullstack Reactivity [FUL]

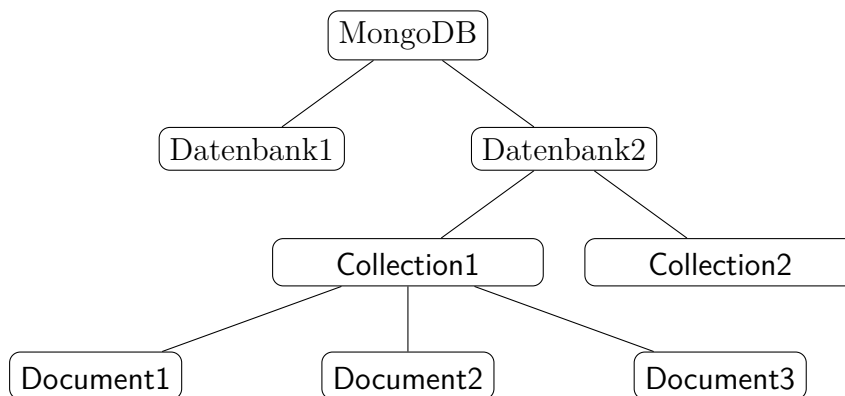
Dadurch muss die Seite nicht neu geladen werden, wenn sich Daten ändern, sondern die Daten ändern sich sofort, sowohl im Backend als auch im Frontend.

<sup>4</sup>application programming interface: Programmierschnittstelle für externe Software [BLO]

<sup>5</sup>Daten und visuelle Inhalte werden durch Veränderungen in der Datenbank direkt neu gerendert, ohne die Seite neu Laden zu müssen [BVJ]

### 2.3.4 MongoDB

MongoDB ist die mitgelieferte Datenbank von Meteor. Sie ist eine NoSQL Datenbank. Jeder Eintrag, welcher gespeichert wird, befindet sich in einem sogenannten *Document* und diese Documents gehören immer einer *Collection* an (Abb. 2.3). Collections sind das Gleiche wie *Tables* bei SQL Datenbanken. MongoDB gehört zu den *in-memory JSON-based databases*. Dadurch kann sie schnellere Abfragen tätigen als andere Datenbanken, da diese mit JSON-Formaten<sup>6</sup> arbeitet. MongoDB ist keine *Reactive* Datenbank. Um sie aber als solche verwenden zu können, wird **Livequery** genutzt. Dies ist eine Funktion von Meteor, die die Datenbank regelmäßig nach Veränderungen aktualisiert [BVJ].



**Abbildung 2.3:** MongoDB Architektur

**MiniMongo:** Während MongoDB auf dem Server einer Applikation läuft, läuft MiniMongo auf dem Client. In dieser clientseitigen Datenbank werden alle nötigen Daten gespeichert, welche vorher vom Server abgerufen wurden. Somit ergibt sich die Möglichkeit, auch auf clientseitiger Ebene die Datenbank zu durchsuchen, ohne vorher auf die Serverantwort zu warten. Dadurch erhält der Browser die Daten viel schneller. Wenn sich in MongoDB Daten verändern, werden diese Veränderungen sofort an den Client geschickt, damit MiniMongo auf dem aktuellsten Stand bleibt [BVJ].

<sup>6</sup>JSON steht für JavaScript Object Notation und ist ein Datenaustauschformat. Es repräsentiert Daten mit einem Attribut und einem dazugehörigen Wert. Für Webapplikationen wird meistens ein JSON genutzt, wenn es zur Kommunikation zwischen Server und Client kommt[RIS]

**Query:** *Query*s sind Suchabfragen, die sowohl clientseitig als auch serverseitig ausgeführt werden.

```
1 | db.inventory.find( {} )
```

**Listing 2.1:** Beispiel: Finden aller *Documents* der *inventory Collection*

```
1 | db.inventory.findOne( {} )
```

**Listing 2.2:** Beispiel: Finden eines einzelnen *Document* der *inventory collection*

### 2.3.5 Package:

*Packages* bilden die Grundlage für das rapide Wachstum von Meteor. Das Framework ist so aufgebaut, dass der Entwickler in der Lage ist eigenständige Packages zu entwickeln und diese lokal zu nutzen oder diese über Atmosphere online zu stellen, sodass auch andere Entwickler von diesem Package profitieren können. Ebenfalls können NPM Packages in die Applikation integriert werden [HOC].

Ein Package enthält meistens eine einzige Funktionalität. Meteor bietet beispielsweise selbst ein Package an, welches „Accounts-ui“ heißt. Accounts-ui ist ein vollständiges Login und Registrierungs-System, welches man nach Belieben zuschneiden kann, damit es der Applikation entspricht. Meteor bietet sehr viele eigens entwickelte Packages an, welche in ihrem GitHub Repository aufgelistet werden<sup>7</sup> [BVJ].

### 2.3.6 Distributed Data Protocol

Das Distributed Data Protocol(DDP) gehört zu einem der sieben Prinzipien von Meteor. „Data on the Wire: Don't send HTML over the network. Send data and let the client decide how to render it.“ DDP ist ein Web-Socket<sup>8</sup>, welches JSON-Daten zwischen Server und Client hin und her transferieren kann. Meteor benutzt Socket.io<sup>9</sup>, um eine bidirektionale<sup>10</sup> Socket Verbindung zwischen Client und Server einzurichten [HOC].

<sup>7</sup><https://github.com/meteor/meteor/tree/devel/packages>

<sup>8</sup>Ein Web-Socket ist ein auf TCP(Transmission Control Protocol) basierendes Netzwerkprotokoll, das entworfen wurde, um eine Verbindung, welche in beide Richtungen zwischen einer Webanwendung und einem WebSocket-Server bzw. einem Webserver, der auch WebSockets unterstützt, herzustellen.[ORA]

<sup>9</sup>[Socket.io](#) ist eine JavaScript Bibliothek für Echtzeit Webapplikationen

<sup>10</sup>Eine Datenübertragung, welche in beide Richtungen stattfindet [W11]

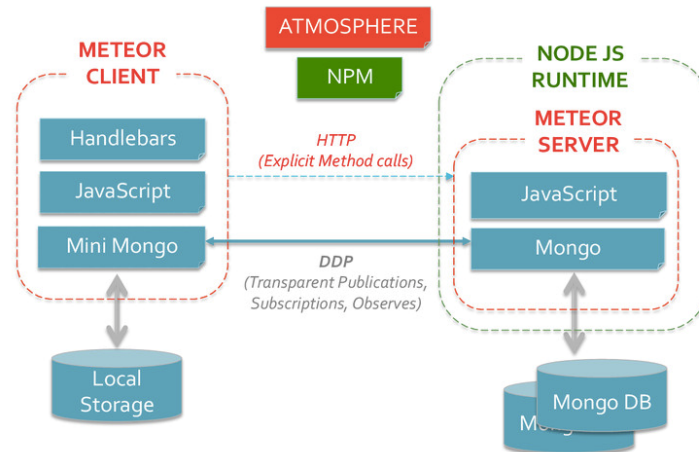


Abbildung 2.4: Distributed Data Protocol [AMA]

### 2.3.7 Templates

Meteor Applikationen gehören zu den **Single-Page-Webanwendungen**. Diese bestehen in der Regel aus einem einzigen HTML-Dokument, dessen Inhalte immer dynamisch nachgeladen werden. In diesem Fall wären es die Meteor HTML-Templates, die nach einer Interaktion geladen werden [HOC]. Bei einer guten Umsetzung, wird die Applikation so entwickelt, dass jedes Template ein bestimmtes Feature beinhaltet, wie z.B. eine Galerie. In der Regel wird das jeweilige Template nach seiner Funktionalität benannt. Diesen Namen bekommen auch die dazugehörigen HTML und JavaScript Dateien.

```

1 <template name="layout">
2   <header>
3     {{> navigationBar }}
4   </header>
5   <main>
6     {{> galerie }}
7     {{> bewertung }}
8   </main>
9 </template>

```

Listing 2.3: HTML-Template

In diesem Beispiel wird ein *Template* mit dem Namen „layout“ erstellt. Dieses *Template* ruft mit der Syntax `{{> TemplateName }}` andere Templates auf, welche dann ausgeführt werden. Im *Header* der HTML-Datei wird eine Navigationsbar erscheinen und in der *Main* eine Galerie mit Bewertungen.

### 2.3.8 Spacebars

Meteor bietet sogenannte *Spacebars* an, welche es ermöglichen innerhalb von HTML-Codes, Schleifen und if/else Bedingungen auszuführen sowie reaktive Daten anzuzeigen [ROB].

```

1 | Template.layout.helpers({
2 |   value(){
3 |     //return random integer between 0 and 1
4 |     var boolean = Math.floor(Math.random() * 2);
5 |     if (boolean === 1){
6 |       return true;
7 |     } else {
8 |       return false;
9 |     }
10 |   }
11 | });

```

**Listing 2.4:** Beispiel eines *Helpers*

```

1 | <template name="layout">
2 |   <header>
3 |     {{> navigationBar }}
4 |   </header>
5 |   <main>
6 |     {{#if value}}
7 |       {{> galerie }}
8 |       {{> bewertung }}
9 |     {{else}}
10 |       {{> keksmaschine }}
11 |     {{/if}}
12 |   </main>
13 | </template>

```

**Listing 2.5:** HTML-Template mit Spacebars

In diesem Beispiel wird veranschaulicht wie *Spacebars* in der Praxis umgesetzt werden. Durch die `{{#if}} ... {{/if}}` Syntax wird entschieden, welche Templates geladen werden sollen. Die `if` Bedingung bekommt eine Funktion/Methode(`value`) übergeben, welche ein `true` oder `false` als Rückgabewert beinhaltet. Bei `true` werden die Templates „galerie“ und „bewertung“ geladen, bei `false` „keksmaschine“.

### 2.3.9 Routing

Dadurch, dass Meteor Applikationen Single-Page-Webanwendungen sind, gibt es nur eine einzige URL. Wenn ein Nutzer einer Meteor-Applikation, z.B. einen Artikel liest, welchen er durch ein HTML-Template geladen hat, kann er diesen Artikel einem anderen Nutzer nicht schicken, da sich die URL nie ändert. Um dieser Problematik

entgegenzuwirken, wurde von Meteor ein Package mit einem Routingsystem entwickelt, mit dem der Entwickler selbständig URLs definieren kann. Jede *Route* (URL) die definiert wird, bekommt eine Zugehörigkeit zu einem HTML-Template mit dem jeweiligen Datacontext [\[HOC\]](#).

## 3 Anforderungsanalyse

In diesem Kapitel wird eine Analyse durchgeführt, die sich am Anfang mit der allgemeinen Projektfindung an Hochschulen befasst. Darauf folgen eine Umfrage zur Projektfindung und Verwaltung an Hochschulen und deren Auswertung. Abschließend wird die Herangehensweise für eine sinnvolle Erweiterung beschrieben.

### 3.1 Projektfindung an deutschen Hochschulen

Jeder Studierende wird im Laufe seines Studiums mit Projekten konfrontiert. Jede Hochschule bereitet ihre Studierenden auf die Phase der Projektarbeit vor [LÜC]. Es gibt drei Abläufe während eines Projektes:

- Projektfindung
- Projektphase
- Projektpräsentation

Bei zwei dieser Abläufe (Projektphase und Projektpräsentation) werden die Studierenden meist von Professoren und Tutoren unterstützt und begleitet. Der Fachbegriff für diese beiden Abläufe heißt *Projektmanagement*. Den Studierenden werden die Grundstrukturen des Projektmanagements beigebracht. Viele Hochschulen legen lediglich eine Vorgehensweise, wie mit Projekten während der Projektphase umgegangen werden soll, fest. (Siehe: [LÜC], [SCH], [WEI], [HOS]) Es wird jedoch in keiner dieser Quellen beschrieben, wie die Projektfindung abläuft oder wie Projekte verwaltet werden. In der Phase der Projektfindung ist der Studierende auf sich alleine gestellt. Er hat weder die Übersicht darüber welche Projekte es gibt, noch wer alles daran teilnimmt.

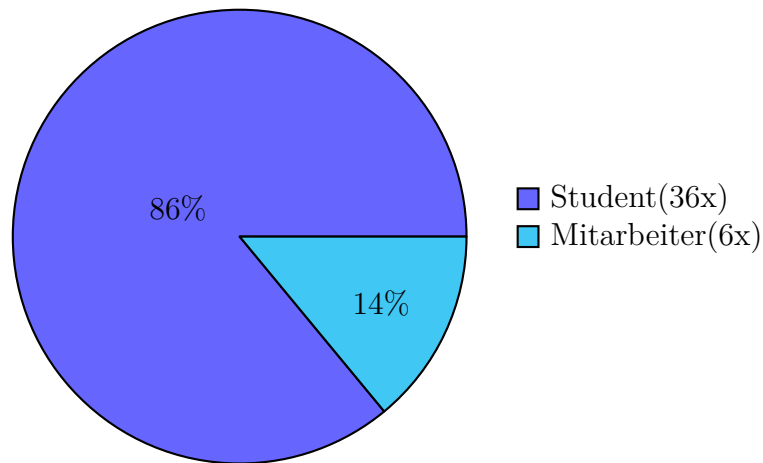
Die nachfolgende Umfrage beschäftigt sich damit, wie Studierende, Lehrende und Mitarbeiter die gegenwärtige Situation an ihrer Hochschule empfinden, wenn es um die Findung und Verwaltung von Projekten geht.

### 3.1.1 Umfrage zur Projektfindung und Verwaltung an Hochschulen

Die Umfrage wurde durch die Plattform „survio“ (<https://my.survio.com/>) erstellt und ausgewertet.

An dieser Umfrage haben 42 Personen teilgenommen:

**Frage 1:** Was ist/war deine Rolle an der Hochschule?



**Abbildung 3.1:** Ergebnis der Frage 1

**Frage 2:** Aus welcher Hochschule kommst bzw. kamst du?

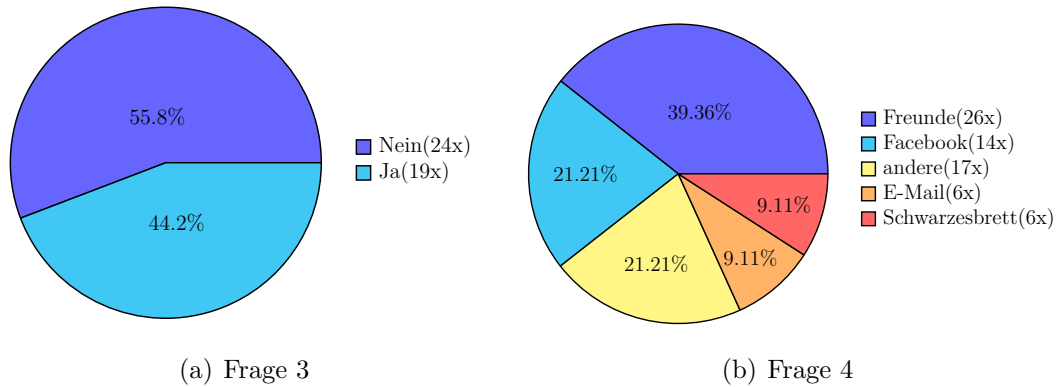
- |                               |                                    |
|-------------------------------|------------------------------------|
| • 32 x HAW Hamburg            | • 1x HWP                           |
| • 2x FH Lübeck                | • 1x TUHH                          |
| • 1x Universität Bremen       | • 1x Leuphana Universität Lüneburg |
| • 1x iba Darmstadt            | • 1x Universität Bremen            |
| • 1x Uni Hamburg / HFBK       | • 1x SAE Institute Hamburg         |
| • 1x Hochschule 21, Buxtehude |                                    |



### 3 Anforderungsanalyse

**Frage 3:** Gibt es eine Plattform an deiner Hochschule um Projekte zu finden und zu verwalten? (Resultat: Abb.3.2(a))

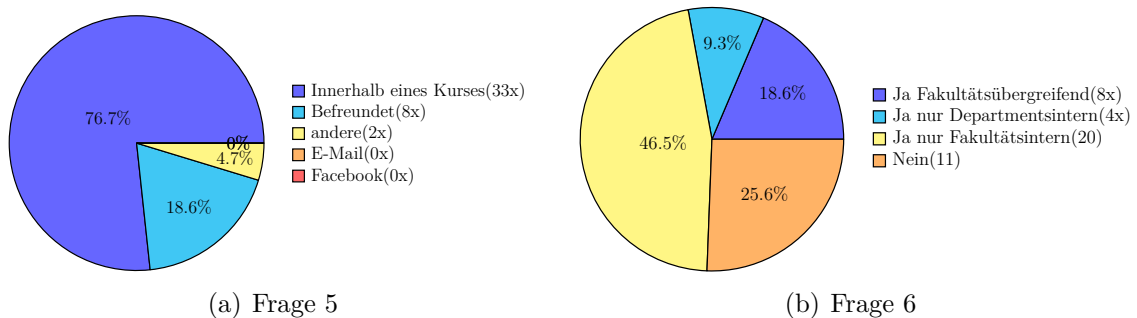
**Frage 4:** Worüber hast du bis jetzt deine Projekte gefunden?(Mehrfach Antwortmöglichkeit) (Resultat: Abb.3.2(b))



**Abbildung 3.2:** Ergebnisse der Fragen 3 und 4

**Frage 5:** 5. Wie hast du deine bisherigen Projektpartner kennengelernt? (Resultat: Abb.3.3(a))

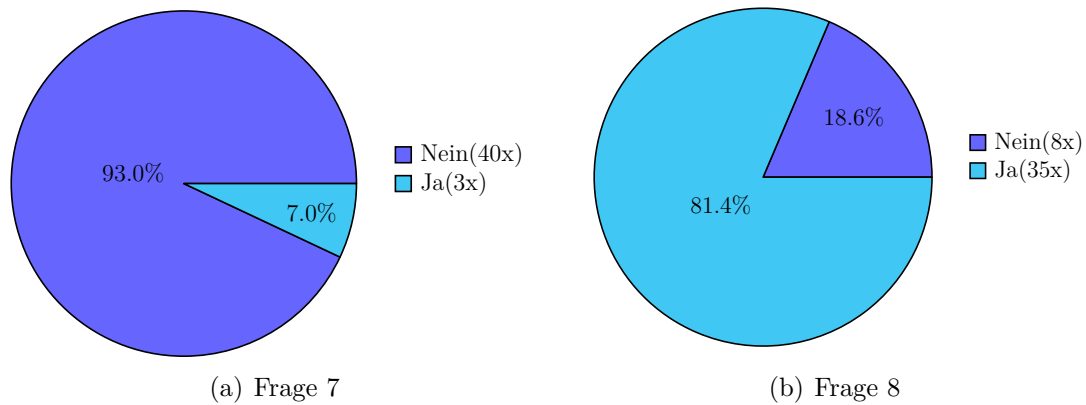
**Frage 6:** 6. Werden an deiner Hochschule Studiengangsübergreifende Projekte gefördert (z.B. Maschinenbauer mit Informatiker)? (Resultat: Abb.3.3(b))



**Abbildung 3.3:** Ergebnisse der Fragen 5 und 6

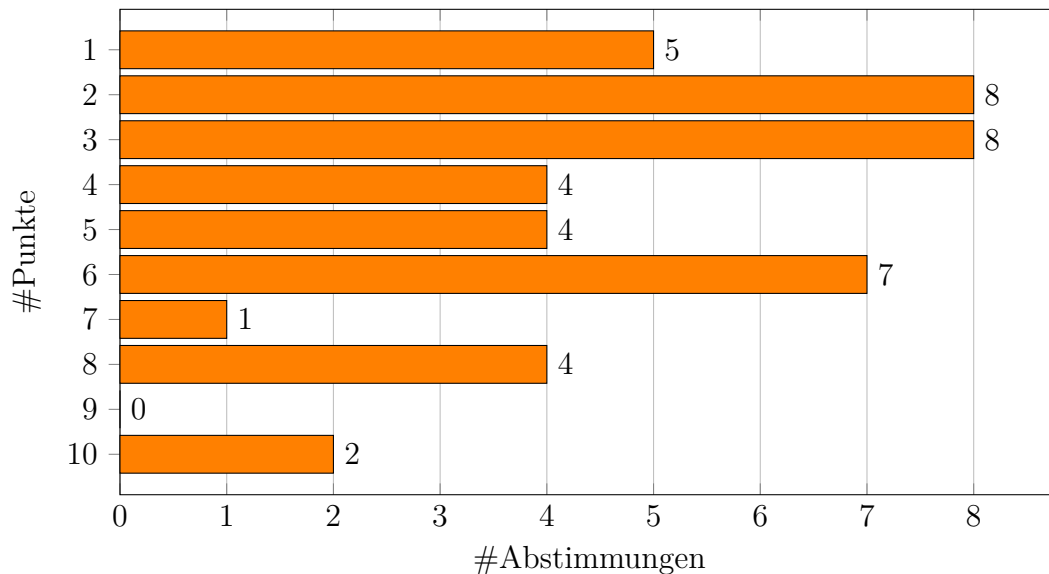
**Frage 7:** 7. Gibt es eine Übersicht von Kurs-internen Projekten, z.B. wenn ein Student krank ist und er sehen möchte wo noch Platz für ihn ist. (Resultat: Abb.3.3(a))

**Frage 8:** 8. Gäbe es eine Anlaufstelle nur für die Projektfindung und Projektverwaltung an deiner Hochschule, würdest du diese nutzen?(Resultat: Abb.3.3(b))



**Abbildung 3.4:** Ergebnisse der Fragen 7 und 8

**Frage 9:** 9. Wie würdest du die Projektfindung und Projektverwaltung an deiner Hochschule bewerten? (Resultat: Abb.3.1.1)



**Abbildung 3.5:** Ergebnisse der Frage 9

### 3 Anforderungsanalyse

**Frage 10:** 10. Hast du noch weitere Ideen, die die Projektfindung/-verwaltung verbessern können? (Resultat: Abb.3.6)

<i>„Es wäre toll, wenn es eine Plattform geben würde um seine Projektpartner nicht nur in seinem Departement, sondern in seiner gesamten Universität finden könnte. Ohne sowas gestaltet sich die Projektfindung sehr mühselig.“</i>
<i>„Dadurch dass die iba eine private Berufsakademie ist und die Studentenzahl relativ gering ist gestaltet sich die Projektverwaltung über das "Hören-Sagen". Evtl. macht es keinen Sinn dafür eine extra Stelle einzurichten.“</i>
<i>„Eine Plattform mit aktuellen Projekten, eventuell hochschulübergreifend wäre hilfreich für eine bessere Übersicht.“</i>
<i>„Digitale Tools sind für sowas immer nett, werden aber in der Regel sowieso nicht genutzt. Projektsteckbriefe direkt an einem schwarzen Brett mit einem Status und Ansprechpartner wären eher hilfreich sowohl bei der Projekt- als auch Ideenfindung.“</i>
<i>„mehr Projekte zu den jeweiligen Themenschwerpunkten meines Studiums (Licht, Bild, Ton, Elektrotechnik) - interessante und lehrreiche Projekte nicht nur für ausgewählte Studenten - faire Verteilung“</i>
<i>„Web-Portal als erste Anlaufstelle für Teamfindung, virtuelles Schwarzes Brett“</i>
<i>„Projektkoordinationsbüro“</i>
<i>„Sammelbecken erstellen“</i>

**Abbildung 3.6:** Antworten der Frage 10

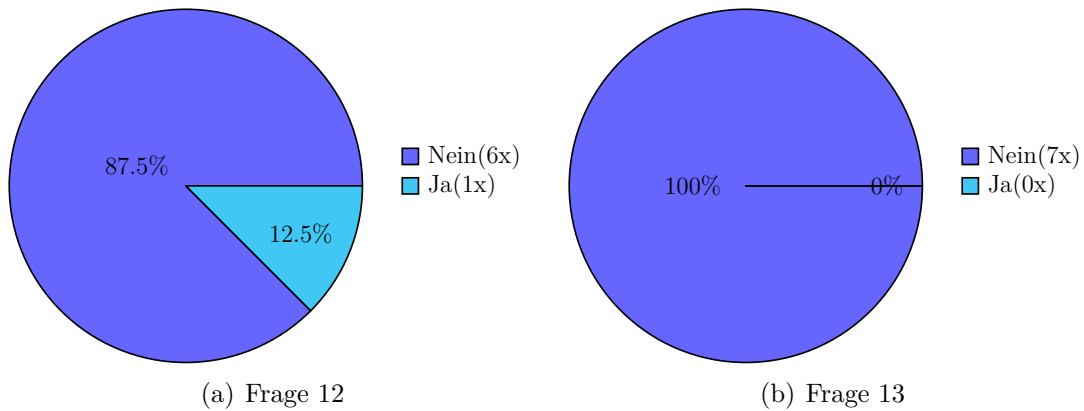
**Frage 11:** 11. (Mitarbeiter) Wie managen Sie die Projektverwaltung innerhalb Ihrer Kurse? (Resultat: Abb.3.7)

<i>„Planungstreffen am Anfang des Semesters.“</i>
<i>„Über Hören und sprechen“</i>

**Abbildung 3.7:** Antworten der Frage 11

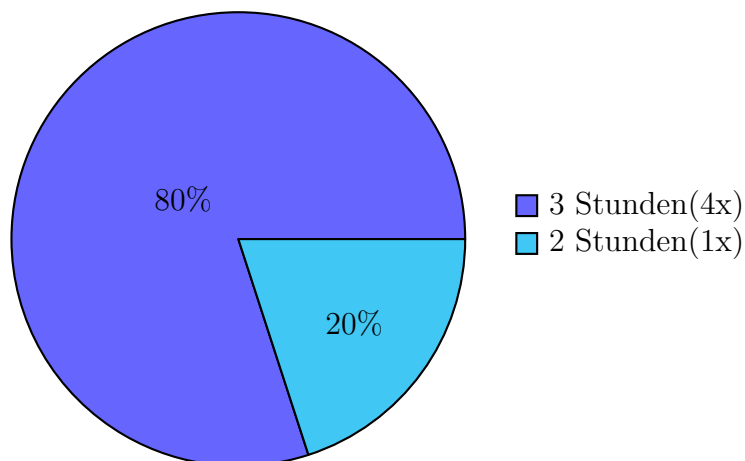
**Frage 12:** 12. (Mitarbeiter) Haben Sie eine visuelle Übersicht um zu sehen, welcher Student in welchem Projekt ist? (Resultat: Abb.3.8(a))

**Frage 13:** 13. (Mitarbeiter) Sind Sie immer auf dem laufenden wie es in den Projekten Ihrer Studenten vorangeht?(Resultat: Abb.3.8(b))



**Abbildung 3.8:** Ergebnisse der Fragen 12 und 13

**Frage 14:** 14. (Mitarbeiter) Wie lange benötigen Sie (in Stunden) die Projektnoten den Studenten zuzuordnen und eine Exceltabelle zu erstellen mit allen nötigen Informationen(Vorname, Nachname, Matrikelnummer, Prüfungsdatum, Note, etc.)(Resultat: Abb.3.1.1)



**Abbildung 3.9:** Antworten auf die Frage 14

### 3.1.2 Auswertung der Umfrageergebnisse

An der Umfrage haben nur Studierende und Mitarbeiter deutscher Hochschulen teilgenommen. Die Umfrage besteht aus einem Allgemeinteil, welcher von jedem Teilnehmer ausgefüllt wurde und einem Mitarbeiterteil, welcher nur von Lehrenden und Tutoren ausgefüllt wurde. Der größte Teil der Teilnehmer kommt aus der HAW (siehe: 3.1.1).

**Auswertung des Allgemeinteils:** Durch den dominierenden Anteil der HAW Teilnehmer, fällt bei der Betrachtung der einzelnen Antworten eine Besonderheit zur dritten Frage auf (3.2(a)). *Survio* ermöglicht es, jede einzelne Antwort jedes Teilnehmers einzusehen (Abb. 3.10). Knapp 50% der HAW Teilnehmer sind sich unschlüssig, ob sie eine Plattform für die Projektfindung und Verwaltung haben. Daraus lässt

22	<input checked="" type="radio"/> JA <input type="radio"/> JA <input type="radio"/> JA	23.07.2017, 08:11:30	Student	HAW Hamburg	Nein
21	<input checked="" type="radio"/> JA <input type="radio"/> JA <input type="radio"/> JA	25.07.2017, 00:37:04	Student	HWP	Ja
20	<input checked="" type="radio"/> JA <input type="radio"/> JA <input type="radio"/> JA	25.07.2017, 00:27:32	Student	Haw Hamburg	Nein
19	<input checked="" type="radio"/> JA <input type="radio"/> JA <input type="radio"/> JA	25.07.2017, 00:23:45	Mitarbeiter	TUHH	Nein
18	<input checked="" type="radio"/> JA <input type="radio"/> JA <input type="radio"/> JA	24.07.2017, 22:51:00	Student		Ja
17	<input checked="" type="radio"/> JA <input type="radio"/> JA <input type="radio"/> JA	24.07.2017, 21:48:04	Student	HAW Hamburg	Ja
16	<input checked="" type="radio"/> JA <input type="radio"/> JA <input type="radio"/> JA	24.07.2017, 20:43:28	Student	HAW	Nein
15	<input checked="" type="radio"/> JA <input type="radio"/> JA <input type="radio"/> JA	24.07.2017, 20:40:01	Student	HAW Hamburg	Nein
14	<input checked="" type="radio"/> JA <input type="radio"/> JA <input type="radio"/> JA	24.07.2017, 20:27:55	Student	Uni Hamburg / HFBK	Ja
13	<input checked="" type="radio"/> JA <input type="radio"/> JA <input type="radio"/> JA	24.07.2017, 20:25:39	Student	Haw Hamburg DMI	Nein
12	<input checked="" type="radio"/> JA <input type="radio"/> JA <input type="radio"/> JA	24.07.2017, 20:15:57	Student	HAW Hamburg	Nein
11	<input checked="" type="radio"/> JA <input type="radio"/> JA <input type="radio"/> JA	24.07.2017, 20:12:25	Student	University of York	Nein
10	<input checked="" type="radio"/> JA <input type="radio"/> JA <input type="radio"/> JA	24.07.2017, 19:41:16	Student	HAW Hamburg	Nein
9	<input checked="" type="radio"/> JA <input type="radio"/> JA <input type="radio"/> JA	24.07.2017, 19:08:07	Student	Hochschule 21, Buxtehude	Nein
8	<input checked="" type="radio"/> JA <input type="radio"/> JA <input type="radio"/> JA	24.07.2017, 19:03:15	Student	HAW Hamburg	Nein
7	<input checked="" type="radio"/> JA <input type="radio"/> JA <input type="radio"/> JA	24.07.2017, 19:02:58	Student	iba Darmstadt	Nein
6	<input checked="" type="radio"/> JA <input type="radio"/> JA <input type="radio"/> JA	24.07.2017, 18:22:55	Student	Universität Bremen	Ja
5	<input checked="" type="radio"/> JA <input type="radio"/> JA <input type="radio"/> JA	24.07.2017, 17:37:28	Student	HAW Hamburg	Ja

**Abbildung 3.10:** Ausschnitt der Auswertung von survio

sich schließen, dass die Teilnehmer durch die Hochschule nicht gut genug aufgeklärt werden, wofür deren hochschulinterne Plattformen sind. Durch das Einbeziehen **aller** Teilnehmer in der dritten Frage, überwiegt die Antwort, dass es keine Plattform an deren Hochschulen gibt für die Findung und Verwaltung. Durch die Fragen 4(3.2(b)),

5(3.3(a)) und 7(3.4(a)) wird erkennbar, dass es für die Teilnehmer keine klare Anlaufstelle gibt, wenn diese ein Projekt finden oder planen wollen. Bei Frage 6 sind sich fast alle Teilnehmer einig, dass, wenn es zur Projektfindung innerhalb eines Kurses kommt, abwesende Studierende (z.B. Krankheitsbedingt) keine Möglichkeit haben einen Überblick über die Projekte, welche sich in dem Kurs zusammengefunden haben, zu bekommen. Dementsprechend bewerten die Teilnehmer im Durchschnitt die Projektfindung und Projektverwaltung an ihrer Hochschule mit 4.2 von 10 möglichen Punkten (Abb. 3.1.1).

**Auswertung des Mitarbeiterteils:** Die Professoren und Tutoren sind sich fast alle einig, dass sie keine Kontrolle über die Projekte in Ihren Kursen haben (Abb. 3.8(a)). Die Kursbetreuer haben weder einen Überblick über den Stand der Projekte, noch über die Teilnehmeranzahl. Selbst die Zuordnung, welcher Studierende an welchem Projekt teilnimmt, ist teilweise unbekannt. Dementsprechend haben Studierende, welche bei der Findung nicht anwesend waren, keinen Ansprechpartner, der ihnen einen kurzen und schnellen Überblick verschaffen kann, in welchem Projekt noch Platz wäre und welchen Studierenden er ansprechen könnte. Den Überblick über die Projekte erhält der Kursbetreuer erst, wenn es zur Benotung der Projekte oder der Studierenden kommt. Für diese Zuordnung, welche in den meisten Fällen digitalisiert wird, benötigen die Mitarbeiter im Schnitt 2-3 Stunden.

**Gesamtauswertung:** Die Hochschulen haben ein großes Defizit, wenn es um die Projektfindung und Verwaltung geht. Sowohl bei Projekten, welche von Professoren oder Tutoren organisiert wurden als auch bei Projekten, die eine Kurszugehörigkeit haben, fällt es schwer einen Überblick zu behalten. Die Hochschulen versuchen intern studiengangübergreifende Projekte zu fördern (Abb. 3.3(b)), aber ohne eine vernünftige Anlaufstelle ist dies ein schwieriges Unterfangen. Die Teilnehmer der Umfrage sind sich nicht im Klaren wer oder was<sup>1</sup> der richtige Ansprechpartner ist, wenn sie ein Projekt suchen.

## 3.2 Bestehende Systeme an der HAW

Die HAW hat zwei Systeme, die sowohl von den Studierenden als auch von den Mitarbeitern der Hochschule genutzt werden. Es geht um die Plattformen EMIL und Helios.

---

<sup>1</sup>könnte eine Plattform sein

#### 3.2.1 EMIL

EMIL steht für Elektronische Medien, Information und Lehre. Es ist ein Dienst der HAW zur Ergänzung der Präsenzlehre, für multimediale Inhalte und Lehrmethoden. Lehrende können hier, begleitend zur Lehrveranstaltung, einen individuellen, frei gestaltbaren, virtuellen Lernraum erstellen. In Form einer interaktiven Homepage zur Veranstaltung, werden dann folgende Features zur Verfügung gestellt [EMI]:

- Ressourcen wie Dokumente und Folien hochladen und veröffentlichen
- Online-Tests zur Selbstkontrolle erstellen
- Aufgaben wie z.B. Praktikumsberichte oder Hausarbeiten einsammeln und bewerten
- Elektronisch über Foren, Chat und Mail kommunizieren
- Kollaboratives Arbeiten

Studierende erhalten einen vom Kursersteller festgelegten Einschreibeschlüssel, um der Veranstaltung beitreten zu können. Teilnehmer haben so eine komplette Übersicht über alle Mitglieder. Außerdem können sie auf alle Materialien, welche von Kursbetreuern zur Verfügung gestellt werden, zugreifen. Auch Nutzungsrechte für Foren und Wikis können auf diese Weise erworben werden.[BET]

#### 3.2.2 Helios

HELIOS (HAW - elektronisches Informations- und Organisationssystem) ist das Hochschulportal für Studierende, Lehrende und Mitarbeiter der HAW. Über dieses System läuft die gesamte Studiums- und Prüfungsverwaltung. Die Prüfungsverwaltung ist der interessanteste Bereich für alle HAW-Beteiligten[LI1]. Für die Studierenden sind es die Bereiche **Prüfungsan/- und abmeldung**, **Notenübersicht** und **Studienmatrix**. Für die Lehrenden und die Mitarbeiter hingegen unterteilt sich der Bereich in drei andere Unterpunkte: **Prüfungsanmeldungen**, **Prüfungsorganisation** und **Leistungserfassung**[LI2]. Durch die Umfrage(siehe: 3.1.1) hat sich ergeben, dass die Leistungserfassung im Schnitt bis zu 3 Stunden in Anspruch nimmt, wenn Projektarbeiten eines Kurses komplett bewertet und digitalisiert werden. Die Leistungserfassung beinhaltet:

- Projekte den Studenten zuordnen
- Bewertungen den jeweiligen Studenten zuordnen (sofern es eine individuelle Bewertung gibt)
- Alles in einer Excel Tabelle festhalten

### 3.3 Webapplikation Projektor

Der Projektor wurde von zwei Studierenden der HAW, des Studiengangs Media Systems konzipiert und umgesetzt. Die Webapplikation wird kontinuierlich weiterentwickelt (Abb. 3.11). Sie dient der vereinfachten Findung und der Förderung von studiengangübergreifenden Projekte, an der HAW. Um diese Plattform nutzen zu können, muss der Nutzer ein immatrikulierter Student oder Angestellter der HAW sein. Externe Nutzer können lediglich auf die Startseite zugreifen.

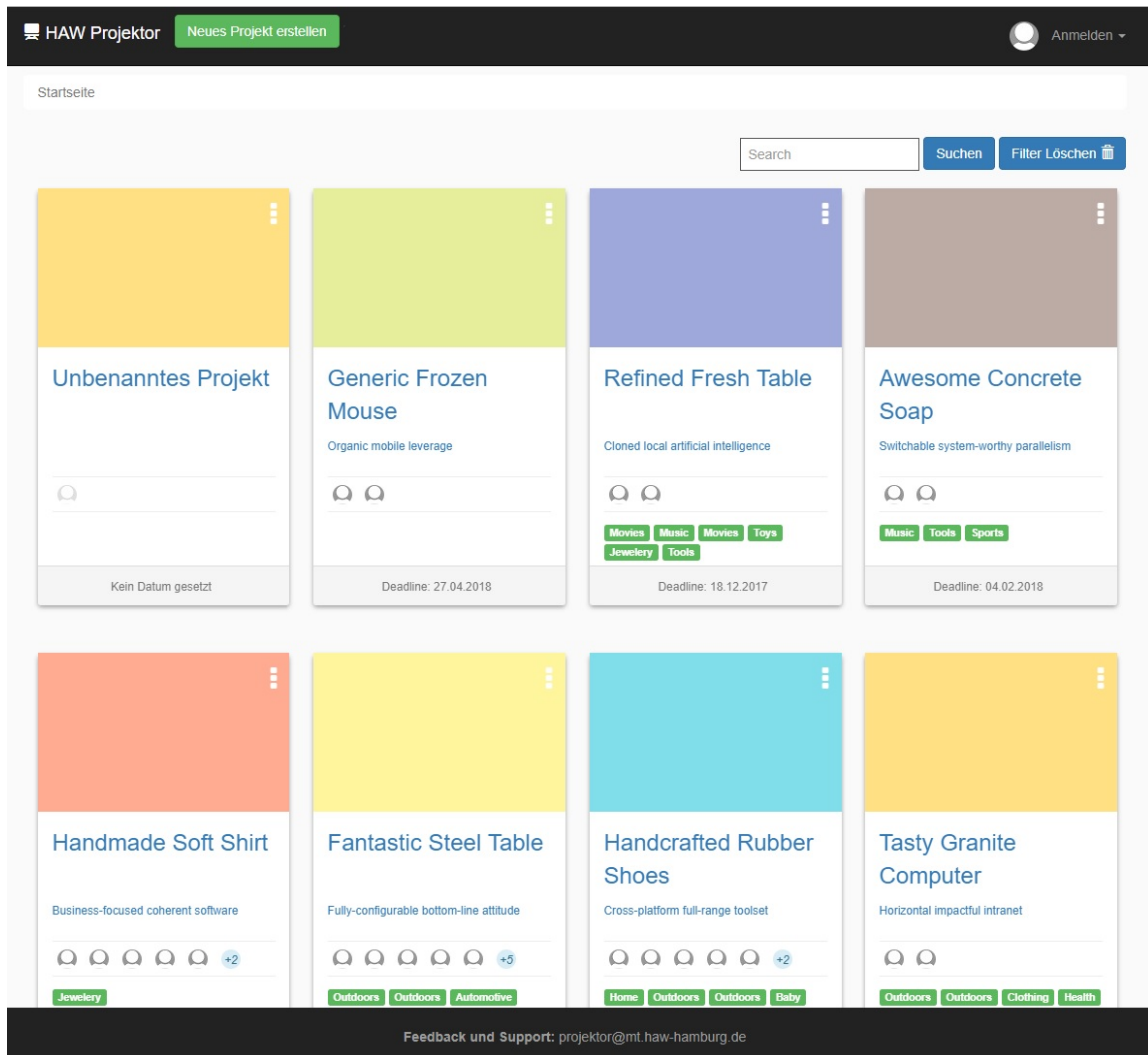


Abbildung 3.11: Startseite des Projektors



#### 3.3.1 Funktionalität

Der Projektor bietet viele verschiedene Funktionalität zur Findung von Projekten und Mitgliedern sowie deren Umsetzung und Organisation.

Alle Funktionalitäten im Überblick:

- Startseite (Auflistung aller Projekte)
- Auflistung aller Mitglieder
- Zwei integrierte Suchen: eine zum Durchsuchen der Projekte und eine zum Durchsuchen der Mitglieder
- Login-System über die HAW-Kennungen, kein Zugriff für Außenstehende
- Erstellen von Projekten
- Bearbeiten von Projekten
  - Titel/Untertitel des Projektes
  - Projektbeschreibung
  - Gesuchte Fähigkeiten
  - Kontaktdaten angeben
  - Mitglieder hinzufügen und entfernen, sowie Rechte vergeben
  - Betreuer hinzufügen und entfernen, sowie Rechte vergeben
  - Projektanlass und andere Details angeben
  - Galerie erstellen zum Hochladen von Bildern oder Hinzufügen von Videos
- Individuelle Nutzerprofile
  - Auflistung der Projekte in denen der Nutzer Mitglied ist
  - Nutzer kann seine Fähigkeiten auflisten
  - Profilbild hochladen
  - Text über sich selbst schreiben
  - Kontaktdaten eintragen

#### 3.3.2 Umsetzung

Der Projektor wurde mit dem Meteor Framework umgesetzt und läuft auf einem Linux Server, welcher von der HAW zur Verfügung gestellt wurde. Zum Starten der Applikation auf dem Server, wird ein Skript ausgeführt, welches Meteor-up (mup)[ZOD] heißt. In dem Skript werden Docker Container<sup>2</sup> definiert in denen unter anderem der Projektor selbst enthalten ist oder externe Software wie z.B. [Elasticsearch](#), eine auf Java basierte Suchmaschine.

Die Datenbank des Projektors besteht aus fünf Collections:

- **Projects Collection:** Enthält alle Projekte, welche erstellt worden sind.
- **Drafts Collection:** Enthält alle Projektentwürfe
- **Images Collection:** Enthält alle hochgeladenen Bilder
- **Studies Collection:** Enthält alle Studiengänge, Departements und Fakultäten der HAW
- **Users Collection:** Enthält alle Nutzer der HAW, welche sich mindestens einmal eingeloggt haben

#### 3.4 Erweiterung der Zielgruppe

Zu jedem Produkt, welches vom Konsumenten genutzt wird, gibt es eine Erwartungshaltung bzw. Verhaltensweise, durch die die jeweilige Zielgruppe definiert wird [CHR]. Die aktuelle Zielgruppe des Projektors besteht hauptsächlich aus Studierenden, da diese die Plattform zum Suchen und Erstellen von Projekten nutzen sowie zum Finden von passenden Projektpartnern. Die Erweiterung muss eine weitere Zielgruppe erreichen, welche aber nicht die momentane Zielgruppe beeinflusst. Die zweitgrößte Zielgruppe an Hochschulen stellen die Mitarbeiter dar:

- Professoren
- Wissenschaftliche Mitarbeiter
- Tutoren

Dementsprechend wird eine Erweiterung konzipiert, die diese Zielgruppe mit einbinden soll.

---

<sup>2</sup>Software, Module, Bibliotheken, etc. können isoliert, in eigene Container gepackt werden. Dies soll die Trennung und Verwaltung von genutzten Ressourcen und eine bessere Leistung des Servers gewährleisten.

## 3.5 Ergebnis der Anforderungsanalyse

Aus all den Punkten, die die Anforderungsanalyse hergibt, wird eine Erweiterung konzipiert und umgesetzt. Diese differenziert zwischen Projekten, die in einem Kurs durchgeführt werden und Projekten, die keine Kurszugehörigkeit haben. Ein komplettes Kursmanagement Tool, welches in seiner Struktur EMIL entspricht, aber im Gegensatz zu EMIL nur für Projektfindung und Verwaltung innerhalb eines Kurses dient. So erhalten sowohl Studierende als auch Betreuer einen Überblick über alle anstehenden Projekte innerhalb des Kurses.

## 4 Konzeption und Implementierung

Dieses Kapitel enthält das Resultat aus der Anforderungsanalyse. Zunächst wird die Kurserweiterung mit ihren Bestandteilen beschrieben. Darauf folgt eine detaillierte Beschreibung der technischen Umsetzungen bestimmter Bestandteile. Abschließend wird auf die neue Datenbankstruktur des Projektors und die Sicherheit mit dem Umgang von Daten eingegangen.

### 4.1 Beschreibung der Kurserweiterung

Die Kurserweiterung wird eine umfangreiche Kurs- und Projektverwaltung beinhalten, ohne die aktuelle Funktionalität des Projektors zu beeinträchtigen. Sie dient dazu, noch schneller und besser kursspezifische Projekte zu finden und zu managen. Somit werden auf der Startseite des Projektors nur noch Projekte angezeigt, die keine Kurszugehörigkeit haben. Jeder Nutzer kann den Kursen beitreten, wenn er den dazugehörigen Einschreibeschlüssel hat oder durch andere Nutzer mit bestimmten Rechten hinzugefügt wird. Je nach Nutzer werden die Anzahl der verfügbaren Funktionalitäten sowie die Rechtevergabe in den Kursprojekten angepasst.

### 4.1.1 Kursverwaltung

Die Kursverwaltung besteht aus zwei Ebenen. Die erste Ebene beinhaltet die Verwaltung aller Kurse. Jeder Nutzer ist selbst dafür verantwortlich, in welchen Kursen er eingeschrieben ist. Während die Dozenten des Kurses in der Lage sind, sowohl Kursen beizutreten als auch solche zu erstellen (Abb. 4.1(a)), können die Studierenden Kursen nur beitreten (Abb. 4.1(b)).



**Abbildung 4.1:** Differenzierung auf der ersten Kursebene

Zum Beitreten eines Kurses benötigt der Nutzer einen Einschreibeschlüssel (Abb. 4.2), welcher beim Erstellen der jeweiligen Kurse vom Kursersteller festgelegt wird. Somit wird verhindert, dass Unbefugte beitreten können.

Kurs beitreten ×

Möchtest du den Kurs beitreten?

Wenn du diesem Kurs beitreten möchtest, benötigst du den zugehörigen Einschreibeschlüssel!

Mathe 2 SS2016 Media Systems ▼

Einschreibeschlüssel:

☒ ☐

**Abbildung 4.2:** Modal zum Beitreten eines Kurses

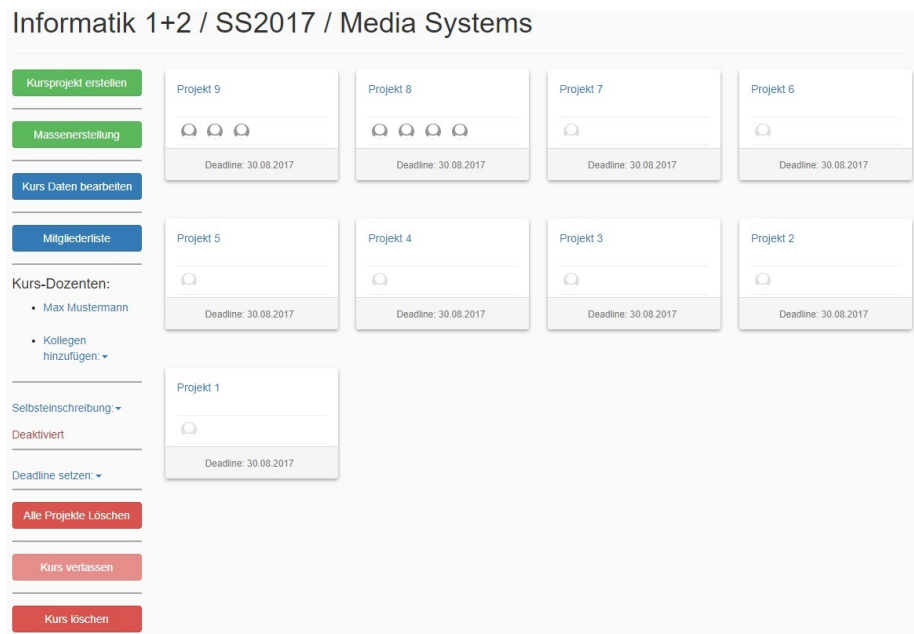
Zum Erstellen eines Kurses gibt es vier Pflichtfelder die ausgefüllt werden müssen:

1. Name des Kurses
2. Semester
3. Studiengang
4. Einschreibeschlüssel

Nach dem Erstellen/Beitreten eines Kurses, erscheint eine Kurskarte, die alle Informationen, den Einschreibeschlüssel ausgeschlossen, enthält. Zusätzlich ist auf der Kurskarte die Anzahl aller Projekte und aller Studierenden im Kurs zu sehen (Abb. 4.1(b)). Dadurch haben die Dozenten immer einen genauen Überblick über den jeweiligen Kurs.

Die zweite Ebene ist die interne Kursverwaltung. Die Berechtigung, Änderungen am und im Kurs vorzunehmen, haben nur der Ersteller des jeweiligen Kurses sowie vom Ersteller hinzugefügte Dozenten.

Folgende Änderungen können die Dozenten vornehmen (Abb. 4.3):



**Abbildung 4.3:** Ansicht eines Kurses aus Sicht eines Dozenten

**Erstellen von Kursprojekten:** Das Erstellen eines Kursprojektes unterscheidet sich insofern von der normalen Projekterstellung des Projektors, dass diese Projekte eine sofortige Standardeinstellung haben. Dies bedeutet, dass die Felder „Kurs“ und „Betreuer“ sowie „Deadline“ (sofern diese im Kurs gesetzt ist), schon im Projekt vordefiniert sind.

**Massenerstellung:** Die Massenerstellung ist eine vereinfachte und schnellere Alternative zum Erstellen von mehreren Projekten, ohne großen Zeitaufwand. Der Nutzer muss in einem Textfeld den Titel für die jeweiligen Projekte eintippen. Jede Zeile entspricht einem Projekt mit dem jeweiligen Titel (Abb. 4.4). Sollte zwischen zwei Zeilen eine leere Zeile auftauchen, wird diese ignoriert.

Massen Projekt Erstellung

Möchtest du mehrere Projekte mit einem Klick erstellen?

Zum erstellen von mehrere Projekten wird nur der Titel des Projektes benötigt.

Beispiel:

Projekt 1  
Projekt 2  
Projekt 3  
...

Projekt 1  
Projekt 2  
Projekt 3  
Projekt 4  
Projekt 5  
Projekt 6

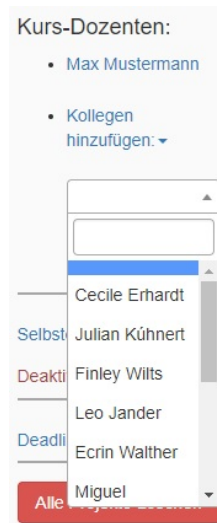
Abbrechen Projekte erstellen

**Abbildung 4.4:** Massenerstellung im Kurs

**Bearbeiten von Kursdaten:** Alle Dozenten, die einem Kurs hinzugefügt wurden, können die Daten des Kurses verändern. Dies beinhaltet den Einschreibeschlüssel, den Kursnamen, das Semester und den Studiengang. Das Bearbeiten der Kursdaten ermöglicht es Fehler zu berichtigen, die beim Erstellen des Kurses entstanden sind. Wenn eine studiengangübergreifende Projektarbeit stattfinden soll, kann der Kurs mit dem neuen Studiengang angepasst werden. Alternativ kann der Kurs in das neue Semester mitgenommen werden.

**Hinzufügen von Dozenten:** Bei studiengangübergreifenden Projekten sind in der Regel mehrere Dozenten involviert. Bei kursübergreifenden Projekten im gleichen Studiengang kann jeder Ersteller neue Dozenten hinzufügen. Durch eine *Dropdown* mit allen registrierten Dozenten des Projektors, welche sich mindestens einmal auf

dem Projektor eingeloggt haben<sup>1</sup>, soll die Suche vereinfacht werden. Durch eine integrierte Suche kann jeder Dozent direkt nach einem bestimmten Kollegen suchen.



**Abbildung 4.5:** Hinzufügen eines Dozenten

**Selbsteinschreibung:** Die Dozenten können durch einen Schieberegler die Selbsteinschreibung aktivieren bzw. deaktivieren (Abb. 4.6(a)). Da jedes Projekt eine einmalige URL hat und jeder Nutzer mit der richtigen URL auf das jeweilige Projekt zugreifen kann, gibt es die Möglichkeit sich **direkt** über ein Projekt in einen Kurs und gleichzeitig in das selbige Projekt einzuschreiben (Abb. 4.6(b)). Dies bedeutet, dass sich z.B. ein kranker Studierender, der die Phase der Projektfindung verpasst hat, durch Kommilitonen, welche ihm die URL zuschicken, direkt in den Kurs und das Projekt einschreiben kann, sofern der Dozent die Selbsteinschreibung aktiviert hat. Wenn der Dozent Projekte vorschlägt, (z.B. durch die **Massenerstellung** 4.1.1) kann sich jeder Student selbst den Projekten hinzufügen.



(a) Selbsteinschreibung aktivieren/deaktivieren

(b) Kursprojekt beitreten

**Abbildung 4.6:** Selbsteinschreibung

<sup>1</sup>Nutzerprofile werden automatisch nach dem ersten Einloggen mit den HAW-Daten angelegt.



**Setzen von Deadlines:** Durch das Setzen einer Deadline innerhalb eines Kurses, sind die Dozenten in der Lage die Deadline für jedes schon vorhandene oder zukünftige neu erstellte Projekt vorzudefinieren. In Zusammenhang mit dem *File-Upload*, welcher in jedem Projekt vorzufinden ist, können die Dozenten durch die Deadline eine Frist für eine Abgabe setzen. Durch einen Vergleich der Upload-Zeit der Datei und der Deadline, können die Dozenten feststellen, ob die Abgabe rechtzeitig erfolgt ist. Den Link für die jeweilige Abgabe sowie das Datum mit der Uhrzeit, können die Dozenten sich über eine Exceltabelle ansehen, welche als Download zur Verfügung steht. Sollte eine Abgabe über die Deadline hinausgehen, wird geprüft, ob möglicherweise eine Abgabe bereits vor der Deadline stattgefunden hat. In so einem Fall werden beide *Links* und deren *Upload-Datum* zur Verfügung gestellt. Damit können die Dozenten feststellen, welche Veränderungen zwischen den beiden Abgaben stattgefunden haben. Sie selbst entscheiden dann welche Version gewertet wird.

**Löschen aller Projekte:** Alle Projekte innerhalb eines Kurses werden unwiderruflich gelöscht. Damit dies nicht versehentlich bei einem einzelnen Klick passiert, erscheint zur Absicherung ein *Modal*<sup>2</sup> mit der Warnung:

„Möchtest Du wirklich alle Projekte löschen? Alle Projekte in diesem Kurs werden unwiderruflich gelöscht!.“

Erst wenn hier eine Bestätigung erfolgt, werden alle Projekte gelöscht.

**Verlassen des Kurses:** Den Kurs können alle Mitglieder jederzeit verlassen. Bevor ein Nutzer den Kurs endgültig verlässt, erscheint ein *Modal*<sup>2</sup> mit Warnhinweisen, in denen alle möglichen Konsequenzen aufgelistet werden.

1. Studierende werden aus jedem Projekt entfernt, in dem sie als Mitglied eingetragen sind.
2. Dozenten werden aus jedem Projekt entfernt, in dem sie als Mitglied oder als Betreuer eingetragen sind.
3. Dozenten verlieren alle Rechte im Kurs und den Projekten.

Mit Bestätigung des Dialogfensters werden alle Verbindungen vom Nutzer zum Kurs gelöscht. Einzig wenn ein Dozent der letzte eines Kurses ist, kann dieser den Kurs nicht verlassen.

**Löschen von Kursen:** Der gesamte Kurs mit seinen Mitgliedern, Betreuern und Projekten wird unwiderruflich gelöscht. Dies wird abermals durch ein *Modal* abgesichert.

---

<sup>2</sup>Dies ist ein Dialogfenster, welches sich nach einem Klick mit einem transparenten grauen Hintergrund über die eigentliche Seite legt und auf eine Eingabe des Nutzers wartet.[[BOO](#)]

### 4.1.2 Differenzierung der Nutzer

Es gibt zwei Nutzergruppen, Studierenden und die Mitarbeiter. Abhängig von der Nutzergruppe, wird eine begrenzte Anzahl an Funktionalitäten zur Verfügung gestellt. Die Differenzierung erfolgt durch die *User Collection* mit dem Feld *role*, welche einen String enthält: entweder *Student* oder *Mitarbeiter*. Die Documents für die Nutzer, mit allen relevanten Daten, werden nach ihrem ersten Login auf der Website in die *User Collection* gespeichert. Diese Daten werden vom ITSC (Informationstechnik Service Center) der HAW zur Verfügung gestellt. Durch Zugriff auf einen LDAP-Server (Lightweight Directory Access Protocol)<sup>3</sup> werden folgende Daten übermittelt:

- Rolle an der Universität (Studierender oder Mitarbeiter)
- Tätigkeit an der Universität (Wissenschaftliche Mitarbeiter, Tutor, Studierender, etc.)
- Fakultät
- Departement
- Studiengang
- Matrikelnummer
- E-Mail
- Vorname
- Nachname
- Voller Name
- Geschlecht

Die Differenzierung wird unter anderem clientseitig mit einem erweiterten Spacebar Package kontrolliert [HEN]:

```

1 | {{#if $eq currentUser.profile.role "Mitarbeiter"}}
2 |   <button type="button" class="btn btn-primary
   |       btn-create-course">Kurs Erstellen</button>
3 | {{/if}}
```

**Listing 4.1:** Spacebar Erweiterung

Mit dieser Syntax wird kontrolliert, ob der momentan eingeloggte Nutzer ein Mitarbeiter ist. Somit wird der visuelle Inhalt dem Nutzer angepasst.

---

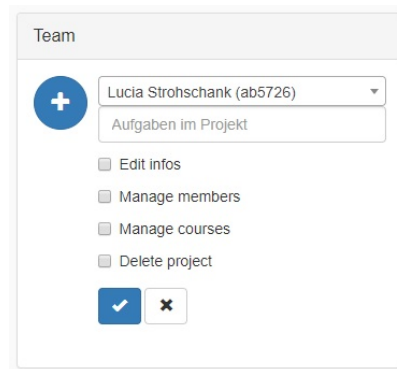
<sup>3</sup>Verzeichnisdienst

### 4.1.3 Projektverwaltung im Kurs

Die Verwaltung der Kursprojekte, ist allein durch die Betreuer des Kurses möglich. Diese haben die Befugnis jedes Kursprojekt individuell zu bearbeiten, solange sie als Betreuer des Projektes eingetragen sind. Unabhängig vom Ersteller des Kursprojektes, sind Dozenten automatisch Betreuer und haben alle Bearbeitungsrechte. Vor der Erweiterung gab es drei vordefinierte Berechtigungen:

1. *Edit Infos*: Nur die Informationen im Projekt bearbeiten z.B. den Titel, die Tags, etc.
2. *Manage Members*: Nur die Mitglieder und Betreuer verwalten und bearbeiten.
3. *Delete Project*: Das Projekt löschen.

Durch das Hinzufügen des Kurses, wurden die Berechtigungen um **Manage Courses** erweitert. Manage Courses geben den Nutzern die Berechtigung ein normales Projekt, welches keine Kurszugehörigkeit hat, einem Kurs hinzuzufügen. Während die Dozenten alle vier Berechtigungen inne haben, müssen Studierende diese zunächst von einem Betreuer übertragen bekommen.((Abb. 4.7)



**Abbildung 4.7:** Verteilung der Berechtigung

Die Berechtigungen werden direkt im jeweiligen Projekt *Document* in der Datenbank abgespeichert. Im Listing 4.2 wird ein kleiner Teil eines Projekt *Documents* dargestellt. Die Speicherung der Nutzer mit Berechtigungen, werden in den jeweiligen Arrays abgespeichert.

```
1 Permissions{}:      //Objekt
2   editInfos[]:    //Array
3   userId         //Ids der Nutzer
4   manaMembers[]:  //Array
5   userId         //Ids der Nutzer
6   deleteProject[]: //Array
7   userId         //Ids der Nutzer
8   manageCourse[]: //Array
9   userId         //Ids der Nutzer
```

**Listing 4.2:** Speicherung der *Permissions*

Es gibt drei Szenarien in denen die Studierenden schon Berechtigungen haben.

**Szenario 1:** Durch das Selbsteinschreiben (siehe: 4.1.1) in ein Projekt, erhalten die Studierenden automatisch die *editInfos* Berechtigung. In allen anderen Berechtigungen werden sie nicht eingetragen. Sollten die Studierenden weitere Berechtigungen bekommen, kann dies jederzeit von einem Betreuer bearbeitet werden.

**Szenario 2:** Studierende erstellen selbst Kursprojekte, wodurch sie automatisch alle Berechtigungen haben, welche auch Dozenten haben. In diesem Falle haben Dozenten dennoch die Befugnis den jeweiligen Studierenden Berechtigungen wieder zu entziehen.















**Szenario 3:** Studierende erstellen selbst ein normales Projekt<sup>4</sup> und fügen einen Kurs hinzu. Dadurch behalten alle Studierende ihre vordefinierten Berechtigungen. Auch hier haben Dozenten die Möglichkeit die Berechtigungen zu entziehen.

---

<sup>4</sup>Ein Projekt ohne Kurszugehörigkeit

### 4.1.4 Mitgliederverwaltung

Die Verwaltung der Mitglieder läuft über die Mitgliederseite (Abb. 4.8). Durch die Differenzierung der Nutzer(4.1.2), sieht der visuelle Inhalt für die Mitglieder anders aus, als für die Dozenten des Kurses. Bei den Mitgliedern fehlen die Spalten **Matrikelnummer**, **Bewertung** und der Button zum Entfernen eines Mitgliedes.

Profil	Vorname	Nachname	Matr.-Nr.	Studiengang	Projekte	Bewertung	Rolle
	Max	Mustermann	123456	Media Systems			Mitarbeiter
	Linus	Roschinsky	69892	Media Systems			Mitarbeiter
	Jette	Burkhardt	67789	Media Systems		<input type="text" value="Bewertung"/>	Student 
	Fatima	Schielke	76800	Medientechnik		<input type="text" value="Bewertung"/>	Student 
	Lyn	Figura	93525	Media Systems		<input type="text" value="Bewertung"/>	Student 
	Eileen	Albrecht	94626	Media Systems		<input type="text" value="Bewertung"/>	Student 

**Abbildung 4.8:** Die Memberliste aus Sicht eines Dozenten

Die Dozenten können jederzeit ein Mitglied aus dem Kurs entfernen. Dadurch verliert dieses Mitglied all seine Zugehörigkeiten zu den Kursprojekten und hat keinerlei Verbindungen mehr zu diesem Kurs. Sollte ein Nutzer sich erneut in den selben Kurs einschreiben und z.B. ein Fehlverhalten vorweisen, haben die Dozenten die Möglichkeit, diesen Nutzer endgültig für den Kurs zu sperren. Dadurch hat er keinerlei Möglichkeiten mehr sich für diesen Kurs einzuschreiben.

## 4.2 Bewertungssystem

Die Erweiterung ermöglicht es den Dozenten die Projekte der Studierenden zu bewerten. Dies geschieht über die Mitgliederliste in der Spalte **Bewertung**. Im Laufe der Projektphase hat der Dozent die Möglichkeit über ein Notizfeld, welches in jedem Projekt enthalten ist, Notizen einzufügen und diese mit in die Bewertung einzubeziehen.

Das Bewertungssystem richtet sich nach der Studienordnung der jeweiligen Hochschule. Das aktuell verwendete System richtet sich nach der Studienordnung der HAW, des Departments Medientechnik [DEP]:

Dezimalzahlenbewertung	Note(Benotung)	Notenbeschreibung
0,7	= ausgezeichnet	= eine besonders herausragende Leistung
1,0 und 1,3	= sehr gut	= eine hervorragende Leistung
1,7/2,0 und 2,3	= gut	= eine Leistung, die erheblich über den durchschnittlichen Anforderungen liegt
2,7/3,0 und 3,3	= befriedigend	= eine Leistung, die durchschnittlichen Anforderungen entspricht
3,7 und 4,0	= ausreichend	= eine Leistung, die trotz ihrer Mängel noch den Anforderungen genügt
4,3 bis 5,0	= nicht ausreichend	= eine Leistung, die wegen erheblicher Mängel den Anforderungen nicht mehr genügt

Tabelle 4.1: Bewertungen an der HAW [DEP]

Auf der Mitgliederseite kann der berechtigte Nutzer jeden Einzelnen und dessen Projektzugehörigkeit sehen. Dementsprechend hat der Dozent einen schnellen Überblick darüber, wen er bewertet. Durch einen regulären Ausdruck wird jede Eingabe vom Nutzer kontrolliert, ob diese in die Sprache des Ausdrucks passt (Abb.: 4.2).

```
1 | /^$|^([1-4])([,.] (0|3|7))?$|^5([,.]0)?$|^0([,.]7$|
```

Mit diesem Ausdruck werden alle Dezimalzahlen aus dem Bewertungssystem der HAW (Tabelle: 4.1) validiert. Dies erfolgt sowohl mit einem **Komma** als auch mit einem **Punkt** und alleinstehenden Ziffern.

### Erklärung des Ausdrucks

```
1 | /^$
```

Mit dem **Slash** „/“ beginnt der reguläre Ausdruck (siehe Abb.: 4.2). Darauf folgt der obenstehende Teilausdruck. Er beginnt mit einem Einfügezeichen „^“, welches für den Zeilenanfang steht. Das Dollarzeichen „\$“, steht für das Zeilenende. Dieser Teilausdruck steht für eine leere Eingabe. Der Zeilenanfang wird initialisiert und sofort wieder beendet. Dies ist notwendig, wenn der Betreuer keine Bewertung für den Studierenden abgeben möchte.

```
1 | | ^ [1-4] ([,.] (0|3|7)) ? $
```

Dieser Teilausdruck beginnt mit einem Balken „|“. Der Balken stellt eine **oder** Bedingung dar, dies bedeutet das dieser Teilausdruck erst ausgeführt wird, wenn der vorige Teilausdruck als **false** deklariert wird. „[1-4]“ beinhaltet nur die Zahlen 1, 2, 3 und 4. Elemente, welche in einer runden Klammer stehen „([,.] (0|3|7))?“ , werden nur gemeinsam ausgeführt. Der Teilausdruck „[,.]“ beinhaltet nur ein Komma oder ein Punkt. Darauf folgt „(0|3|7)“, eine 0 oder 3 oder 7. Durch das Fragezeichen am Ende wird dieser Teil als optional angesehen. Zusammengefasst enthält dieser Teilausdruck die Zahlen 1, 2, 3, 4 und die Dezimalzahlen aus der Bewertungstabelle der HAW (Tabelle. 4.1), sowohl mit Punkt als auch mit Komma. Nur die Benotungen mit *ausgezeichnet* und *nicht ausreichend* sind nicht enthalten.

```
1 | | ^ 5 ([,.] 0) ? $
```

Dieser Teilausdruck beginnt wieder mit einer **oder** Bedingung und beinhaltet die Bewertung: 5, 5.0 und 5,0.

```
1 | | ^ 0 [,.] 7 $ /
```

Dieser Teilausdruck ist der Letzte des Gesamtausdruckes und beinhaltet die Bewertung 0,7 oder 0.7.

Sollte eine Eingabe bei der Bewertung stattfinden, welche nicht in die Sprache des regulären Ausdruckes passt, wird dies dem Nutzer mitgeteilt und das entsprechende Feld rot markiert.

### 4.3 Generierung von Excel-Tabellen

Auf der Mitgliederseite haben die Dozenten die Möglichkeit sich alle Daten über die Studenten ihres Kurses als Exceltabelle herunterzuladen. Diese enthält jeweils eine Tabelle, mit extra Spalten (Tabelle: 4.2), die die Mitgliederseite widerspiegelt und eine Tabelle, die der Heliosliste der HAW angepasst wird (Tabelle: 4.3).

Projektname	Matr.Nr	Vorname	Nachname	Studiengang
Zeitmaschine	94626	Eileen	Albrecht	Media Systems
Keksmaschine	93525	Lyn	Figura	Media Systems
Keksmaschine	67789	Jette	Burkhardt	Medientechnik
Keksmaschine	76800	Fatima	Schielke	Medientechnik

Rechtzeitiger Upload	<- Datum	Letzter Upload	<- Datum
<a href="http://www.link1.de/pdf">www.link1.de/pdf</a>	23.08.2017 23:57	<a href="http://www.link2.de/pdf">www.link2.de/pdf</a>	24.08.2017 02:43
<a href="http://www.linkA.de/pdf">www.linkA.de/pdf</a>	23.08.2017 23:23	-	-
<a href="http://www.linkA.de/pdf">www.linkA.de/pdf</a>	23.08.2017 23:23	-	-
<a href="http://www.linkA.de/pdf">www.linkA.de/pdf</a>	23.08.2017 23:23	-	-

**Tabelle 4.2:** Beispielansicht einer Heliostabelle

Die Tabelle 4.2 gibt dem Dozenten die Möglichkeit sich eine umfassende Übersicht über seinen Kurs einzuholen. Zusätzlich zur Mitgliederseite, gibt es vier extra Spalten, welche die Links zur Datei und das Abgabedatum anzeigen. Dies bedeutet, wenn für einen Kurs eine *Deadline* gesetzt wird, bis zu der eine Abgabe stattgefunden haben soll, können die Betreuer über diese Tabelle den Link zur Abgabe einsehen und feststellen, ob die Abgabe rechtzeitig stattgefunden hat. Es wird immer die letzte Datei vor der Deadline angezeigt, egal wie viele Dateien vorher hochgeladen wurden. Werden Dateien nach der Deadline hochgeladen, wird der Link zur Datei in die Spalte Letzter Upload hinzugefügt. Somit kann der Prüfer selbst entscheiden, ob er diese Abgabe noch akzeptiert oder nur die Abgaben, welche vor der Deadline getätigt wurden.



startHISsheet					
mtknr	bewertung	pdatum	nachname	vorname	geschl
	100	28.01.2016			
	130	28.01.2016			
	170	28.01.2016			
	170	28.01.2016			
endHISsheet					

					endHISsheet
pstatus	pversuch	stgsem	abschl	stgdtxt	pbeginn

**Tabelle 4.3:** Beispielsicht der Mitglieder als Excel Tabelle

Tabelle 4.3 zeigt den Aufbau einer Excel Tabelle, welche im Heliossystem importiert werden kann. So eine Tabelle kann der Dozent über die Mitgliederseite herunterladen. Folgende Spalten werden durch das Generieren der Tabelle ausgefüllt:

1. Matrikelnummer
2. Bewertung(sofern abgegeben, siehe 4.2)
3. Prüfungsdatum

Alle restlichen Spalten werden automatisch nach dem Importieren der Heliostabelle, durch das Heliossystem der HAW, ausgefüllt. Durch das Ausführen des Downloads der Tabellen, wird zuerst eine *.xlsx* Datei erstellt, welche auf dem Server gespeichert wird. Alle Metadaten dieser Datei werden in der *xlsFiles Collection* als *Document* abgespeichert. Erst dann steht die Datei als Download zur Verfügung. Für diesen Ablauf werden zwei *Packages* verwendet: *mongoxlsx* [NIC] und *Veilov, File Upload* [DIM].

**mongo-xlsx:** Mit diesem *Package* wird sowohl aus einem *JSON Array* eine **.xlsx** Datei erstellt als auch umgekehrt. Sowohl für die Heliostabelle als auch für die Mitgliederübersicht, wird ein JSON-Array erstellt (siehe 4.3).

```

1 | const projectObject = {
2 |   Projektname: "Muffinmaschine",
3 |   'Matr.Nr': "222222",
4 |   Vorname: "Max",
5 |   Nachname: "Mustermann",
6 |   Studiengang: "Media Systems" ,
7 |   Notizen: "Ist dies keine Tolle Bachelorarbeit?",
8 |   'Rechtzeitiger Upload': "www.Projektor.mt.haw-hamburg.de\
   | \file1.pdf",
9 |   '<- Datum': 29.08.2017,
10 |   'Letzter Upload': "www.Projektor.mt.haw-hamburg.de\
   | file2.pdf",
11 |   '<-- Datum': 30.08.2017,
12 | };
13 | data.push(projectObject);

```

**Listing 4.3:** Beispiel eines JSON

In Listing 4.3 wird ein JSON-Objekt mit allen benötigten Werten erstellt. Dieses JSON wird am Ende in ein Array gepushed. Durch eine Schleife wird immer ein neues JSON-Objekt erstellt, welches in das Array „data“ gepushed wird. Die Schleife läuft solange durch, bis für jeden Studierenden ein JSON-Objekt erstellt wurde. Wenn das JSON-Array mit allen nötigen Daten fertig erstellt ist, wird dieses in die folgende Funktion als Parameter übergeben.

```

1 | mongoXlsx.mongoData2Xlsx(data, model, function(err, data)

```

**Listing 4.4:** Funktion zur Erstellung einer **.xlsx** Datei

Mit dieser Funktion wird die Excel-Datei generiert und erstellt. Da die Funktion asynchron<sup>5</sup> ausgeführt wird, kann keine weitere Funktion, welche von dieser abhängig ist, hinzugefügt werden. Um der Asynchronität entgegenzuwirken, wird die Funktion, welche die Exceldatei zum Server hinzugefügt und als Download zur Verfügung stellt, innerhalb der mongo-xlsx Funktion implementiert (siehe: 4.5).

<sup>5</sup>Organisation des zeitlichen Ablaufs während der Ausführung der Programme[GUN]

```
1  mongoXlsx.mongoData2Xlsx(data, model, function(err, data)
2  {
3      XlsFiles.addFile(data.fullPath, {
4          name: data.fullPath,
5          type: 'application/
6              vnd.openxmlformats-officedocument.
7              spreadsheetml.sheet',
8          userId: courseId,
9          meta: {},
10     });
11 });
```

**Listing 4.5:** Funktion zum Erstellen einer **.xlsx** Datei und hinzufügen zum Server

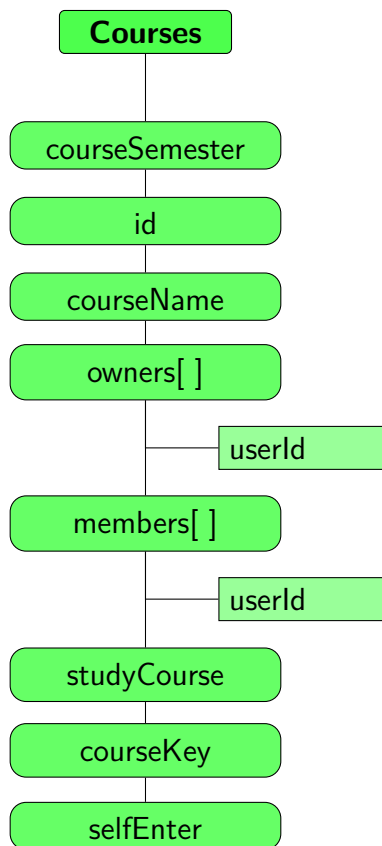
Nach jedem Download wird die gesamte *XlsFiles Collection* gelöscht, damit keine unnötigen Dateien auf dem Server liegen. Excel Tabellen werden durch jeden neuen Download neu generiert, da sich die Daten der Studierenden jederzeit ändern können.

## 4.4 Datenbankstruktur

Für die Erweiterung mussten neue *Collections* erstellt und bestehende *Collections* erweitert werden (siehe: 4.11, 4.9 und 4.10). Die *ProjectFiles* und *XlsFiles* *Collection* werden für die Verwaltung der Excel-Dateien und der hochgeladenen Dateien der Studierenden, innerhalb ihrer Projekte genutzt. Anders als in der *Projects Collection* oder *Courses Collection* (4.10 oder 4.10), wird in der Grafik kein konkretes Collections-Schema gezeigt, da die Collections durch die Packages *mongoxlsx* [NIC] und *Veilov, File Upload* [DIM], ein vordefiniertes Schema haben.



**Abbildung 4.9:** ProjectFiles und XlsFiles Collection



**Abbildung 4.10:** Courses Collection

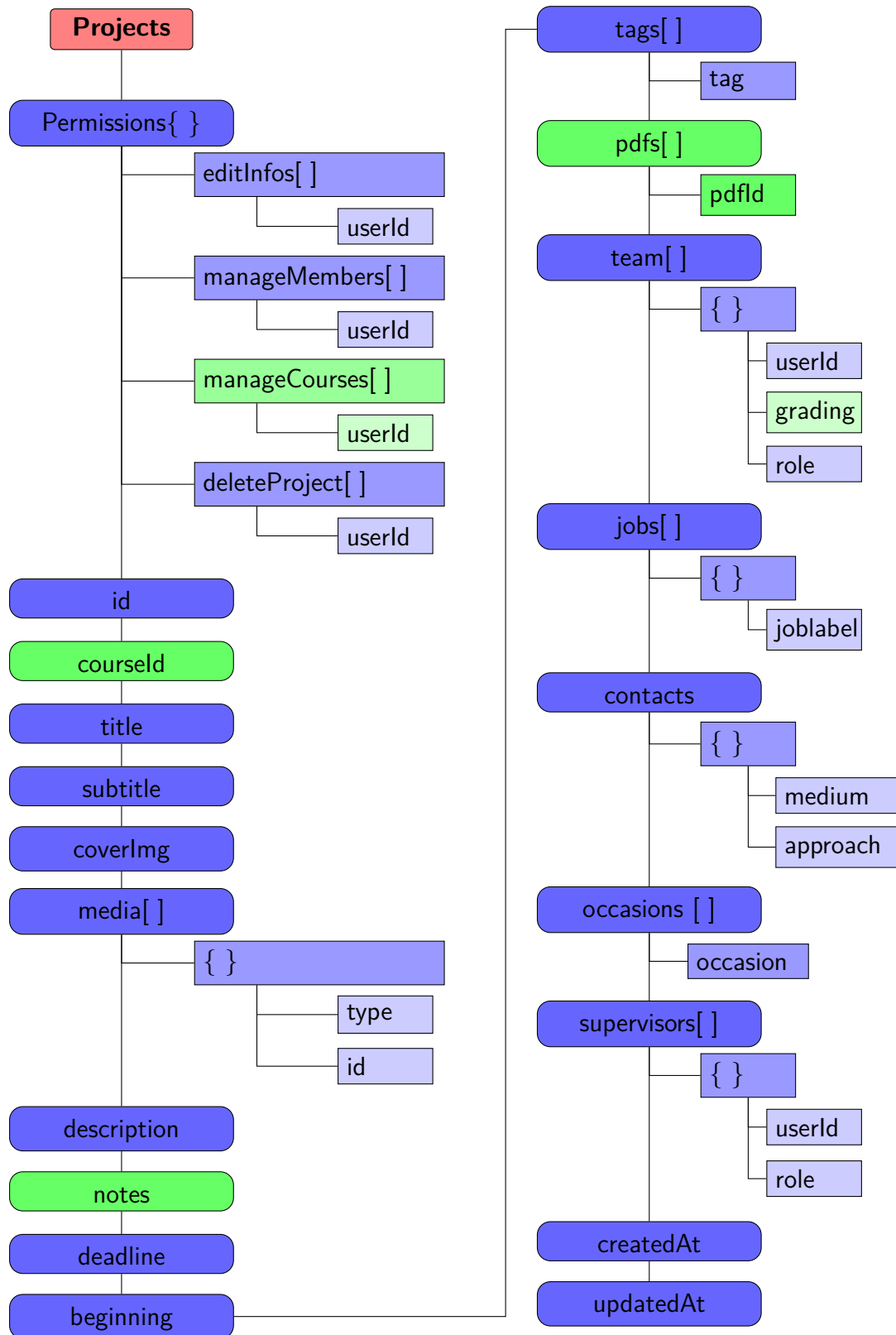


Abbildung 4.11: Projects Collection

Die *Courses Collection* beinhaltet als Schema die gesamte Kursverwaltung. Alles was im und am Kurs verändert wird, läuft über diese Collection.

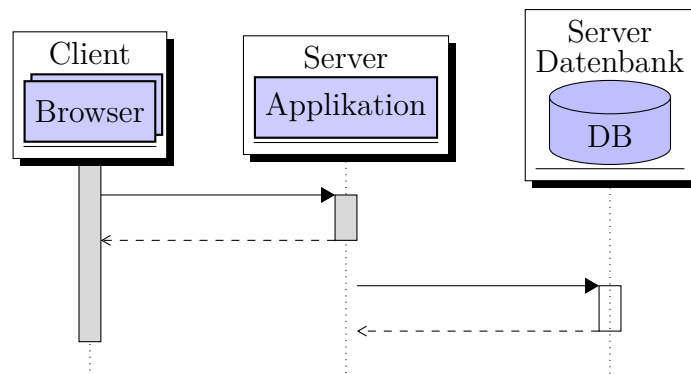
Die *Projects Collection* gab es schon vor der Kurserweiterung. Sie wurde während der Entwicklung an der Kurserweiterung immer wieder ergänzt. Alle neu hinzugefügten Felder sind in Abbildung 4.11 grün dargestellt. Durch diese Ergänzungen wurde eine Schnittstelle zwischen den Kursen und den Projekten geschaffen, was wiederum neue Möglichkeiten für die Verwaltung von Projekten innerhalb eines Kurses eröffnete.

## 4.5 Sicherheit

Die Erweiterung arbeitet mit vielen sensiblen Daten der Studierenden, wie z.B. mit deren Matrikelnummern, Mail-Adressen, Hochschulkennungen oder Benotungen. Würde man nicht kontrollieren, welche Daten, zu welchen Zeitpunkt, an welchen Nutzer gesendet werden, könnte jeder mit ganz einfachen Tools, wie z.B. *Meteor DevTools*, an die Daten sämtlicher Nutzer gelangen. Dieses Tool zeigt welche *Documents* vom Server an den Client geschickt werden und welche Daten in diesen *Documents* enthalten sind. Daher wird ganz genau darauf geachtet wie die Kommunikation zwischen Server und Client abläuft.

### 4.5.1 Kommunikation zwischen Server und Client

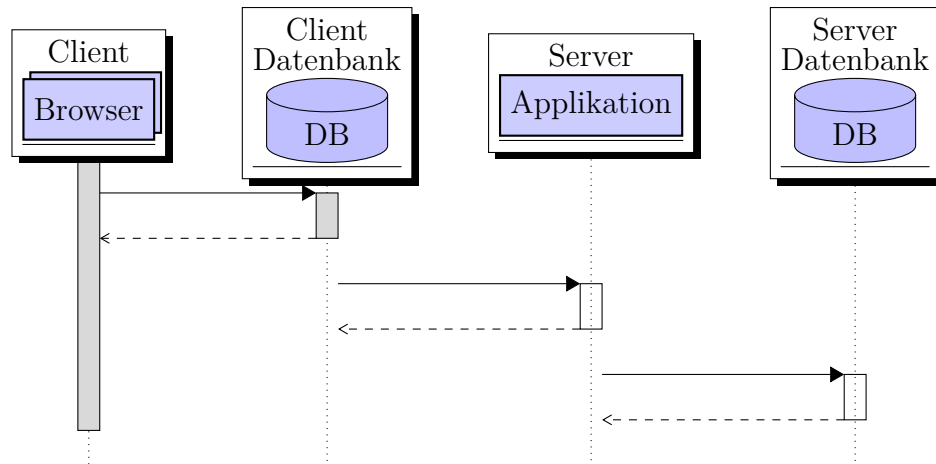
Standardmäßig läuft die Kommunikation zwischen Server und Client wie folgt ab (Abb. 4.12):



**Abbildung 4.12:** Kommunikation zwischen Server und Client (ohne Meteor)

Der Client, in der Regel der Browser, sendet eine Anfrage an die Applikation, welche sich immer auf dem Server befindet. Die Anfrage enthält Informationen darüber, was der Nutzer sehen möchte. Der Server schickt eine weitere Anfrage an die Datenbank, um auf die nötigen Daten zugreifen zu können. Diese werden dann an den Server zurückgeschickt und weiter zurück zum Client. Erst jetzt werden die Daten angezeigt. Meteor hat diese Struktur um einen Schritt erweitert. Zwischen dem Client und dem

Server gibt es jetzt eine sogenannte *Client-Datenbank*(Abb. 4.13), die schon erwähnte **Minimongo** 2.3.4.



**Abbildung 4.13:** Kommunikation zwischen Server und Client(mit Meteor)

Mit dieser Struktur werden die Daten, welche vom Server zurückgeschickt werden, temporär so lange in die Minimongo abgespeichert, bis ein neues *Template* geladen wird. Dann werden neue Daten temporär abgespeichert. Dies geschieht ebenfalls, wenn sich in der MongoDB selbst etwas ändert. Dieses System nennt man auch DDP(Distributed Data Protocol). Hierbei handelt es sich um ein von Meteor selbst entwickeltes Protokoll, welches die Daten durch *Subscribe* und *Publishen* filtert, bevor sie bei der Client-Datenbank ankommen. Der Vorteil bei diesem System ist der unmittelbare Zugriff auf die Daten, ohne vorher auf die Antwort des Servers warten zu müssen[COL].

### 4.5.2 Publish

Durch den *Publish* wird ermöglicht, dass vom gesamten Datenbankinhalt nur ausgewählte Daten an den Client gesendet werden. Ein *Publish* kann wie folgt aussehen(4.6):

```

1 Meteor.publish('singleCourse', function
  singleCoursePublication(courseId) {
2   const course = Courses.find(courseId, {
3     fields: {
4       _id: 1,
5       courseName: 1,
6       courseSemester: 1,
7       studyCourse: 1,
8       owner: 1,
9       member: 1,
10      courseKey: 1,
11      selfEnter: 1,
12      deadline: 1,
13
14    },
15    });
16    return course;
17  });

```

**Listing 4.6:** Publish Beispiel

Dieser *Publish* bekommt einen festen Namen „singleCourse“ und einen vordefinierten Übergabewert „courseId“ (dieser ist Optional). Mit diesem Publish sollen die Daten eines spezifischen Kurses an den Client gesendet werden. Zu diesem Zweck wird die „Kurs ID“ des Kurses benötigt, welche für den Publish vom zugehörigen Subscribe übermittelt wird (siehe: 4.6). In der Publish Funktion selbst wird eine Query ausgeführt, die in der *Course Collection* nach dem angeforderten *Document*(Kurs) sucht. In der Query selbst kann angegeben werden, welche Felder des Documents an den Client geschickt werden. Somit wird nicht das gesamte Document an den Client gesendet, sondern nur ein ausgewählter Teil.



### 4.5.3 Subscribe

Durch den *Subscribe* wird eine Anfrage für Daten an den Server geschickt. Auf dem Server wird die dazugehörige Publish Funktion ausgeführt. Die Subscribes finden in der Kurserweiterung immer in der *onCreated*<sup>6</sup> Funktion von Meteor statt (Abb. 4.7). In Listing 4.7 sind drei Subscribes für die Startseite des Kurses definiert. Der Aufbau eines Subscribes ist relativ einfach. Als Parameter wird der Name des dazugehörigen Publishes angegeben. Optional können bestimmte Daten mit übergeben werden.

```

1 | Template.currentCourse.onCreated(function
   |   currentCourseOnCreated() {
2 |   this.subscribe('userSupervisor');
3 |   this.subscribe('singleCourse', FlowRouter.getParam('
   |     courseId'));
4 |   this.subscribe('courseProjects', FlowRouter.getParam('
   |     courseId'));
5 | });

```

**Listing 4.7:** Subscribe Beispiel

```

1 |   this.subscribe('userSupervisor');

```

**Listing 4.8:** Subscribe der Dozenten

Bei Subscribe 4.8 werden nur die Daten für alle Dozenten angefragt.

```

1 |   this.subscribe('singleCourse', FlowRouter.getParam('
   |     courseId'));

```

**Listing 4.9:** Subscribe eines einzelnen Kurses

Bei *Subscribe* 4.9 wird das *Document* für einen spezifischen Kurs angefragt. Zu diesem Zweck wird als Parameter die *ID* des Kurses mitgeschickt.

```

1 |   this.subscribe('courseProjects', FlowRouter.getParam('
   |     courseId'));

```

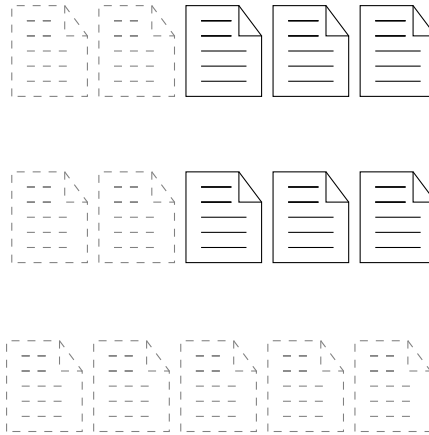
**Listing 4.10:** Subscribe bestimmter Kursprojekte

Bei *Subscribe* 4.10 werden alle Daten von Projekten angefragt, die eine Zugehörigkeit zu diesem Kurs haben. Auch hier wird als Parameter die *ID* des Kurses mitgeschickt.

---

<sup>6</sup>onCreated wird immer als Erstes ausgeführt, wenn ein *Template* geladen wird. Erst danach werden alle anderen Funktionen und Methoden ausgeführt.

Abbildung 4.14 veranschaulicht eine Collection, in der 15 Documents enthalten sind. Diese liegen auf der Server-Datenbank, wobei nur sechs der 15 vorhandenen Documents benötigt werden. Somit werden durch den Subscribe genau diese sechs Documents an den Client geschickt.



**Abbildung 4.14:** Beispiel: Subscribe von bestimmten *Documents*

# 5 Evaluation

Die Evaluation beinhaltet ein Fazit über die Umsetzung der Erweiterung und ihr mögliches Potential zur Verbesserung des **Projektors**. Darauf folgt ein Ausblick über Verbesserungsvorschläge und Weiterentwicklungsmöglichkeiten in Verbindung zur Modularisierung der Erweiterung.

## 5.1 Fazit

Durch die Anforderungsanalyse wurden viele Defizite im Bereich der Projektfindung und Verwaltung an Hochschulen offengelegt. Durch die Aufdeckung dieser Defizite, konnten Ideen und Möglichkeiten zur erfolgreichen Erweiterung des Projektors entwickelt werden. Die Erweiterung sollte den Projektor sowohl technologisch als auch zielgruppenorientiert vorantreiben, ohne von dessen Grundidee/Grundfunktion abzuweichen. Durch die übersichtliche Struktur, mit der der Projektor entwickelt wurde und dem ausführlich dokumentierten Framework Meteor, konnte man sich gut in den zugrundeliegenden „Ist-Zustand“ einarbeiten und einen Überblick über mögliche Änderungen verschaffen. So konnte die Erweiterung in den Projektor integriert und damit der angestrebte „Soll-Zustand“ erreicht werden. Folgende Punkte ermöglicht der Projektor nach der Kurserweiterung:

- Übersicht von Projekten innerhalb eines Kurses, sowohl für Studierende als auch für Betreuer
- Abwesende Studierende finden nachträglich schneller ein Projekt
- Betreuer können die Projekte der Studenten bewerten
- Betreuer können Excel-Listen herunterladen
- Excel-Listen sind kompatibel mit dem Helios-System der HAW
- Kurse erstellen und bearbeiten
- Komplette Kursverwaltung
- Neues Rechtesystem
- Übersicht der Kursteilnehmer
- Separieren von Kursprojekten und Projekten die keine Kurszugehörigkeit haben

Somit wurde das Ziel der Anforderungsanalyse erreicht. Eine Plattform in der man erfolgreich sowohl nach kursinternen als auch offenen Projekten suchen kann. Die Kurserweiterung ermöglicht es durch eine ausgeklügelte Verwaltung, sich einen Überblick über Projekte innerhalb eines Kurses sowie dessen Mitglieder zu verschaffen.

Dank des DDP war, anders als ursprünglich angenommen, weniger Beschäftigung mit der Datensicherheit nötig, da man die verfügbaren Daten jederzeit überblicken konnte. Die komplette Kurserweiterung kann auf [www.dev.projektor.mt.haw-hamburg.de](http://www.dev.projektor.mt.haw-hamburg.de), in Kombination mit dem Projektor aufgerufen und eingesehen werden.

## 5.2 Ausblick

Verbesserungsvorschläge und Ideen der Erweiterung:

- Semestereingabe bei der Kurserstellung als Dropdown
- Gleiches Projekt zu verschiedenen Kursen hinzufügen
- Besseres Management von gelöschten Kursprojekten (Studierende möchte Projekt vielleicht behalten)
- Besser gestaltete Exceltabellen
- Kurszugehörigkeit ins Profil der Nutzer integrieren
- Automatisierte Tests nach einem neuen Feature
- Sidebar responsive machen
- Mehrere Deadlines setzen für verschiedene Abgaben
- Mehrere Abgaben hochladen können im Projekt
- Als Betreuer Kursinhalte hochladen können
- Ankündigungen innerhalb des Kurses machen
- Als eigenständiges Package weiterentwickeln

Durch das Packagesystem von Meteor, gibt es die Möglichkeit Applikationen komplett modular zu entwickeln. Das bedeutet, dass jedes Feature als Package entwickelt werden kann. Diese Packages werden so programmiert, dass sie unabhängig von den anderen Packages funktionieren. Sollte ein Feature deaktiviert werden, wird dieses Package aus der Package.js von Meteor entfernt oder auskommentiert. Trotzdem kann die Applikation voll funktionsfähig weiterlaufen. Somit können ganz einfach neue Features hinzugefügt werden.

Die Bachelorarbeit von Finn Schröder mit dem Titel [SCR],

*„Erweiterbare Webanwendungen unter Verwendung des Meteor-Frameworks.  
Implementierung eines Plugin-Systems am Praxis-Beispiel der  
Teamfindungs-Plattform Projektor.“*

befasst sich mit der Modularisierung des Projektors. In dieser Arbeit gibt es unter anderem eine Anleitung zur Erstellung eines Packages für den Projektor. Mit dieser Anleitung ist es möglich, die komplette Kurserweiterung in ein Package zu integrieren und es so der modularisierten Version des Projektors anzupassen.

# Listings

2.1	Beispiel: Finden aller <i>Documents</i> der inventory <i>Collection</i> . . . . .	11
2.2	Beispiel: Finden eines einzelnen <i>Document</i> der <i>inventory collection</i> . .	11
2.3	HTML-Template . . . . .	12
2.4	Beispiel eines <i>Helpers</i> . . . . .	13
2.5	HTML-Template mit Spacebars . . . . .	13
4.1	Spacebar Erweiterung . . . . .	34
4.2	Speicherung der <i>Permissions</i> . . . . .	36
4.3	Beispiel eines JSON . . . . .	42
4.4	Funktion zur Erstellung einer <b>.xlsx</b> Datei . . . . .	42
4.5	Funktion zum Erstellen einer <b>.xlsx</b> Datei und hinzufügen zum Server	43
4.6	Publish Beispiel . . . . .	48
4.7	Subscribe Beispiel . . . . .	49
4.8	Subscribe der Dozenten . . . . .	49
4.9	Subscribe eines einzelnen Kurses . . . . .	49
4.10	Subscribe bestimmter Kursprojekte . . . . .	49

# Abbildungsverzeichnis

2.1	Meteor Development Group Gründer . . . . .	6
2.2	Fullstack Reactivity [ <a href="#">FUL</a> ] . . . . .	9
2.3	MongoDB Architektur . . . . .	10
2.4	Distributed Data Protocol [ <a href="#">AMA</a> ] . . . . .	12
3.1	Ergebnis der Frage 1 . . . . .	16
3.2	Ergebnisse der Fragen 3 und 4 . . . . .	17
3.3	Ergebnisse der Fragen 5 und 6 . . . . .	17
3.4	Ergebnisse der Fragen 7 und 8 . . . . .	18
3.5	Ergebnisse der Frage 9 . . . . .	18
3.6	Antworten der Frage 10 . . . . .	19
3.7	Antworten der Frage 11 . . . . .	19
3.8	Ergebnisse der Fragen 12 und 13 . . . . .	20
3.9	Antworten auf die Frage 14 . . . . .	20
3.10	Ausschnitt der Auswertung von survio . . . . .	21
3.11	Startseite des Projektors . . . . .	24
4.1	Differenzierung auf der ersten Kursebene . . . . .	29
4.2	<i>Modal</i> zum Beitreten eines Kurses . . . . .	29
4.3	Ansicht eines Kurses aus Sicht eines Dozenten . . . . .	30
4.4	Massenerstellung im Kurs . . . . .	31
4.5	Hinzufügen eines Dozenten . . . . .	32
4.6	Selbsteinschreibung . . . . .	32
4.7	Verteilung der Berechtigung . . . . .	35
4.8	Die Memberliste aus Sicht eines Dozenten . . . . .	37
4.9	ProjectFiles und XlsFiles Collection . . . . .	44
4.10	Courses Collection . . . . .	44
4.11	Projects Collection . . . . .	45
4.12	Kommunikation zwischen Server und Client (ohne Meteor) . . . . .	46
4.13	Kommunikation zwischen Server und Client(mit Meteor) . . . . .	47
4.14	Beispiel: Subscribe von bestimmten <i>Documents</i> . . . . .	50

# Tabellenverzeichnis

4.1	Bewertungen an der HAW <a href="#">[DEP]</a> . . . . .	38
4.2	Beispielansicht einer Heliostabelle . . . . .	40
4.3	Beispielansicht der Mitglieder als Excel Tabelle . . . . .	41



# Literaturverzeichnis

- [AMA] amazonaws.com *Distributed Data Protocol*, [http://s3.amazonaws.com/info-mongodb-com/\\_com\\_assets/blog/meteor/image05.png](http://s3.amazonaws.com/info-mongodb-com/_com_assets/blog/meteor/image05.png), letzter Zugriff: 25. 08. 2017
- [Ana] Anand, Raj: *Watch out for Meteor...!*, <http://hacktivist.in/articles/Watchout-for-meteor>, 2014, letzter Zugriff: 08. 08. 2017
- [BAN] Bannister, Mike *Seven Principles of Meteor*, <https://gist.github.com/possibilities/4685332>, 2017, letzter Zugriff: 15. 08. 2017
- [BET] Bettzüge, Björn: *EMILHAW FÜR LEHRENDE*, [https://www.elearning.haw-hamburg.de/pluginfile.php/79939/mod\\_book/chapter/255/EMIL%20f%C3%BCr%20Lehrende.pdf](https://www.elearning.haw-hamburg.de/pluginfile.php/79939/mod_book/chapter/255/EMIL%20f%C3%BCr%20Lehrende.pdf), 2014, letzter Zugriff: 09. 08. 2017
- [BLO] Bloch, Joshua: *How to Design a Good API and Why it Matters*, <http://static.scribd.com/docs/908bil5xonqxe.pdf>, Unbekannt, letzter Zugriff: 25. 08. 2017
- [BOO] bootstrapworld.de: *Modal in Bootstrap - Eine Anleitung*, <https://www.bootstrapworld.de/modal-tutorial.html>, 2017, letzter Zugriff: 30. 07. 2017
- [BVJ] B V, Jebin: *Mastering MeteorJS Application Development*, Packt Publishing Ltd. Verlag, 2015
- [CHR] Dingenotto, Christian *Zielgruppen definieren und wählen*, [http://www.cultural-business.com/media/070402\\_cubus\\_Zielgruppen.pdf](http://www.cultural-business.com/media/070402_cubus_Zielgruppen.pdf), 2007, letzter Zugriff: 25. 08. 2017
- [COL] Coleman T., Greif S: *Discover Meteor Building Real-Time JavaScript Web Apps*, 2016
- [DEP] Department Medientechnik: *Studienführer Modulhandbücher der Bachelor-Studiengänge Media Systems und Medientechnik Sommersemester 2017*, [https://www.haw-hamburg.de/fileadmin/user\\_upload/DMI-Mt/Studium/MT\\_StudfuehrModulhandbuch\\_web\\_\\_1\\_.pdf](https://www.haw-hamburg.de/fileadmin/user_upload/DMI-Mt/Studium/MT_StudfuehrModulhandbuch_web__1_.pdf), 2017, letzter Zugriff: 01. 08. 2017
- [DIM] dr.dimitru: *Meteor-Files: VeliovGroup/Meteor-Files*, <https://github.com/VeliovGroup/Meteor-Files>, 2017, letzter Zugriff: 10. 07. 2017

- [DOC] Docker *What is a Container?*, <https://www.docker.com/what-docker>, 2017, letzter Zugriff: 25. 08. 2017
- [EMI] E-Learning-Team: *Das Buch EMIL E-Learning an der HAW Hamburg*, <https://www.elearning.haw-hamburg.de/mod/book/tool/print/index.php?id=38825&chapterid=132>, 2013, letzter Zugriff: 09. 08. 2017
- [FUL] *Fullstack Reactivity*, <https://image.slidesharecdn.com/understandingmeteor-150216072648-conversion-gate01/95/understanding-meteor-16-638.jpg?cb=1424093270>, letzter Zugriff: 25. 08. 2017
- [GAN] Ganev, Dobrin (): *Build Applications with Meteor*, Packt Verlag 2017
- [GOY] Goyvaerts, Jan: *Regular Expressions The Complete Tutorial 2007*, <https://www.princeton.edu/~mlovett/reference/Regular-Expressions.pdf>, 2007, letzter Zugriff: 01. 08. 2017
- [GUN] Dr.-Ing. Gunter Bolch, Dr.-Ing. Ulrich Zahner: *Echtzeit-Programmierung*, [https://www4.cs.fau.de/Lehre/WS03/V\\_STP1/Skript/stp1-pa-ws03-kapitel4.pdf](https://www4.cs.fau.de/Lehre/WS03/V_STP1/Skript/stp1-pa-ws03-kapitel4.pdf), 2003, letzter Zugriff: 28. 07. 2017
- [HEN] Henriksen, Nørgaard *raix/Meteor-handlebar-helpers*, <https://github.com/raix/Meteor-handlebar-helpers>, 2015, letzter Zugriff: 22. 07. 2017
- [HOC] Hochhaus S., Schoebel M.: *Meteor in Action*, Manning Publications Co. Verlag, 2016
- [HOS] Hochschule Ostwestfalen-Lippe *Handbuch Projektmanagement, Projektmanagement an der Hochschule Ostwestfalen-Lippe*, <https://www.hs-owl.de/fileadmin/aktuelles/rektoratsmitteilungen/Handbuch-Projektmanagement-Web.pdf>, 2014, letzter Zugriff: 14. 08. 2017
- [LI1] Life Sciences, Fakultätsservicebüro *Prüfdatenverwaltung in HELIOS*, [http://www.ls.haw-hamburg.de/~abc480/faqpavma/lib/exe/fetch.php?media=deutsch:lehrende:handout\\_pruefdatenverwaltunghelios.pdf](http://www.ls.haw-hamburg.de/~abc480/faqpavma/lib/exe/fetch.php?media=deutsch:lehrende:handout_pruefdatenverwaltunghelios.pdf), Unbekannt, letzter Zugriff: 08. 08. 2017
- [LI2] Life Sciences, Fakultätsservicebüro Helios für studierende *Helios für studierende*, [https://www.haw-hamburg.de/fileadmin/user\\_upload/FakLS/09Service/FSB-LS/Allgemein/Handout\\_HELIOS\\_fuer\\_StudierendeLS\\_20172.pdf](https://www.haw-hamburg.de/fileadmin/user_upload/FakLS/09Service/FSB-LS/Allgemein/Handout_HELIOS_fuer_StudierendeLS_20172.pdf), 2017, letzter Zugriff: 08. 08. 2017
- [LIE] Lienau, Matthias *Meteor: Einblick in die Full-Stack-JavaScript-Plattform für das Echtzeit-Web*, <http://t3n.de/magazin/meteor-full-stack-javascript-plattform-fuer-echtzeit-web-234154/>, 2013, letzter Zugriff: 25. 08. 2017

- [LÜC] Lückenbach, Florian: *Hochschulentwicklung und Qualitätssicherung*, Projektmanagement Hochschule Koblenz, [https://www.hs-koblenz.de/fileadmin/media/hochschule/hochschulentwicklung\\_qm/pdf/kick-off-camp/ss14\\_projektmanagement\\_lueckenbach.pdf](https://www.hs-koblenz.de/fileadmin/media/hochschule/hochschulentwicklung_qm/pdf/kick-off-camp/ss14_projektmanagement_lueckenbach.pdf), Unbekannt, letzter Zugriff: 14. 08. 2017
- [NIC] Nicolas *mongo-xlsx: Moblox/mongo-xlsx*, <https://github.com/Moblox/mongo-xlsx>, 2016, letzter Zugriff: 10. 07. 2017
- [ORA] ORACLE: *Websockets*, <https://community.oracle.com/docs/D0C-982926>, 2017, letzter Zugriff: 15. 08. 2017
- [RIS] Rischpater, Ray: *Discover Meteor*, 2015
- [ROB] Robinson J., Gray A., Titarenco D.: *Introducing Meteor*, Apress Verlag, 2016
- [SCH] Scharfenorth, Kari: *Projektmanagement - Projekte entwickeln und planen (APOLLON Hochschule der Gesundheitswirtschaft)*, [https://www.apollon-hochschule.de/fileadmin/content/pdf/HZK/Probelektionen/Probekapitel\\_neu\\_PRMAH01\\_1215K02.pdf](https://www.apollon-hochschule.de/fileadmin/content/pdf/HZK/Probelektionen/Probekapitel_neu_PRMAH01_1215K02.pdf), Unbekannt, letzter Zugriff: 14. 08. 2017
- [SCR] Schröder, Finn: *Erweiterbare Webanwendungen unter Verwendung des Meteor-Frameworks. Implementierung eines Plugin-Systems am Praxis-Beispiel der Teamfindungs-Plattform Projektor.*, 2017, letzter Zugriff: 28. 08. 2017
- [STR] Strack, Isaac: *Getting Started with Meteor.js JavaScript Framework 2nd. Edition*, Packt Publishing Ltd. Verlag, 2015
- [TUR] Turnbull, David: *Your First Meteor Application*, 2015
- [WEI] Weisser P., Geibel J., Dembski N: *Zukunftsfähige Hochschulen gestalten. Beispiele des Gelingens aus Lehre, Governance, Betrieb und Forschung*, [https://www.fona.de/mediathek/pdf/Zukunftsfaehige\\_Hochschulen\\_Gestalten\\_netzwerk\\_n\\_VA\\_online.pdf](https://www.fona.de/mediathek/pdf/Zukunftsfaehige_Hochschulen_Gestalten_netzwerk_n_VA_online.pdf), Unbekannt, letzter Zugriff: 14. 08. 2017
- [WI1] Wikipedia *Bidirektional*, <https://de.wikipedia.org/wiki/Bidirektional>, 2017, letzter Zugriff: 25. 08. 2017
- [ZOD] zodern: *meteor-up*, <http://zodern.github.io/meteor-up/>, 2017, letzter Zugriff: 25. 08. 2017

Ich versichere, die vorliegende Arbeit selbstständig ohne fremde Hilfe verfasst und keine anderen Quellen und Hilfsmittel als die angegebenen benutzt zu haben. Die aus anderen Werken wörtlich entnommenen Stellen oder dem Sinn nach entlehnten Passagen sind durch Quellenangaben eindeutig kenntlich gemacht.

Ort, Datum

Adrian Abbassian