

# MASTERARBEIT

## Erfolgreiche Monetarisierung einer Open-Source Webanwendung durch die Integration eines innovativen Geschäftsmodells

zur Erlangung des akademischen Grades

MASTER OF SCIENCE

vorgelegt dem Fachbereich Mathematik, Naturwissenschaften und  
Informatik der Technischen Hochschule Mittelhessen

**Autor:** Marius Trautrim  
marius.trautrimsmni.thm.de  
MatNr. 5047709

**Version vom:** 5. Oktober 2016

**Referent:** Prof. Dr. Klaus Quibeldey-Cirkel  
**Korreferent:** Hr. Christoph Thelen (M.Sc.)





Dieses Werk ist lizenziert unter einer Creative Commons Namensnennung -  
Nicht-kommerziell - Keine Bearbeitung 3.0 Deutschland Lizenz.



---

## Kurzfassung

Die vorliegende Masterthesis gibt einen Überblick über die Monetarisierung einer Web-Anwendung am Beispiel der E-Learning-Plattform THMcards.

Dabei wurde chronologisch zuerst das theoretische Wissen über die verschiedenen Geschäftsmodelle, die Kunst der Preisgestaltung und den verschiedenen Zahlungsmethoden vermittelt. Anschließend wurde mit dem Erstellen eines passenden Geschäftsmodells für THMcards und der zugehörigen Preisgestaltung begonnen. Danach wurden die notwendigen Grundlagen innerhalb von THMcards geschaffen, um das Geschäftsmodell umzusetzen. Die Integration eines Zahlungssystem ermöglicht es schlussendlich Umsatz mit THMcards zu generieren.

Diese Masterthesis ist sowohl für Studierende als auch für Lehrende im Bereich Web-Entwicklung interessant.



# Inhaltsverzeichnis

Abbildungsverzeichnis . . . . .	viii
Tabellenverzeichnis . . . . .	x
Listingverzeichnis . . . . .	xii
Abkürzungsverzeichnis . . . . .	xiv
<b>I. Grundlagen</b>	<b>1</b>
<b>1. Einführung</b>	<b>3</b>
1.1. Motivation . . . . .	3
1.2. Problemstellung und Zielsetzung . . . . .	4
1.3. Gliederung und Vorgehensweise . . . . .	5
<b>2. Erfolg einer Web-App</b>	<b>7</b>
2.1. Was ist eine Web-App? . . . . .	7
2.2. Erfolgsfaktoren . . . . .	8
2.2.1. Idee . . . . .	8
2.2.2. Einzigartigkeit . . . . .	9
2.2.3. Umsetzung . . . . .	10
2.2.4. Kontext . . . . .	11
<b>II. Monetarisierung in der Theorie</b>	<b>13</b>
<b>3. Monetarisierung</b>	<b>15</b>
3.1. Begriffsdefinition . . . . .	15
3.2. Kommerzielle Open Source Software . . . . .	15
3.3. Kommerzialisierung von Bildung . . . . .	16
<b>4. Geschäftsmodelle</b>	<b>19</b>
4.1. Definition . . . . .	19
4.2. Aufgaben eines Geschäftsmodells . . . . .	20
4.3. Gratiskultur . . . . .	20
4.4. Arten . . . . .	21
4.4.1. Direkt . . . . .	21
4.4.2. Indirekt . . . . .	23
4.4.3. Lateral . . . . .	24
4.5. Auswahl eines geeigneten Geschäftsmodells . . . . .	25

<b>5. Preisgestaltung</b>	<b>27</b>
5.1. Art und Qualität des Produktes . . . . .	27
5.2. Laufende Kosten . . . . .	28
5.3. Konkurrenz . . . . .	29
5.4. Konsumentenbedürfnisse . . . . .	29
5.4.1. Zeit . . . . .	30
5.4.2. Rarität . . . . .	30
5.4.3. Komfort . . . . .	31
5.4.4. Ansehen . . . . .	31
5.4.5. Zugehörigkeit . . . . .	31
5.4.6. Körperliches Befinden . . . . .	32
5.4.7. Finanzielle Sicherheit . . . . .	32
5.4.8. Unterhaltung . . . . .	32
5.4.9. Herausforderung . . . . .	33
5.5. Die Nachfragekurve . . . . .	33
5.6. Preissegmentierung . . . . .	34
5.7. Die Psychologie des Preises . . . . .	35
5.7.1. Preisanker . . . . .	36
5.7.2. Preisanmutung . . . . .	36
5.7.3. Unsichtbare Preisänderung . . . . .	36
<b>6. Bezahlvorgang</b>	<b>37</b>
6.1. Problemstellung . . . . .	37
6.2. Zahlungsmethoden . . . . .	37
6.2.1. Gegenüberstellung . . . . .	38
6.2.2. Auswahl treffen . . . . .	41
6.3. Payment Service Provider . . . . .	43
6.3.1. Kriterien . . . . .	44
6.3.2. Gegenüberstellung . . . . .	45
<b>III. Monetarisierung am Beispiel von THMcards</b>	<b>49</b>
<b>7. Die Open-Source-Anwendung THMcards</b>	<b>51</b>
7.1. Zieldefinition . . . . .	51
7.2. Re-Implementierung von THMcards . . . . .	54
7.2.1. Evaluierung eines geeigneten Web-Frameworks . . . . .	54
7.2.2. Codestruktur . . . . .	60
7.3. Technische Aspekte von THMcards . . . . .	62
7.3.1. Meteor . . . . .	63
7.3.2. Node.js . . . . .	68
7.3.3. MongoDB . . . . .	69



<b>8. Anwendung und Implementierung</b>	<b>73</b>
8.1. Geschäftsmodell . . . . .	73
8.2. Preisgestaltung . . . . .	74
8.3. Grundlagen . . . . .	75
8.3.1. Kartensatzarten . . . . .	75
8.3.2. Rollen . . . . .	77
8.4. Bezahlvorgang . . . . .	78
8.4.1. Zahlungsverkehrsplattform . . . . .	79
8.4.2. Abonnement . . . . .	81
8.4.3. Einzelkauf . . . . .	82
8.4.4. Abrechnung . . . . .	83
<b>9. Resümee</b>	<b>87</b>
<b>Literaturverzeichnis</b>	<b>i</b>



# Abbildungsverzeichnis

2.1. Wahrnehmung einer Web-App gemessen an deren Einzigartigkeit .	10
5.1. Die Nachfragekurve einer Web-App . . . . .	33
5.2. Die monatlichen Einnahmen einer Web-App . . . . .	35
6.1. Genutzte Bezahlverfahren in 2016 beim Kauf im Online- und Versandhandel [Bon16] . . . . .	41
6.2. Verlauf der genutzten Bezahlverfahren beim Kauf im Online- und Versandhandel [Bon16] . . . . .	42
6.3. Checkout von Stripe . . . . .	45
6.4. Checkout von Braintree . . . . .	46
6.5. Wiederkehrende Zahlungsvorgänge in Braintree [Brab] . . . . .	47
7.1. Die Commits eines Web-Frameworks . . . . .	56
7.2. Dateistruktur von THMcards . . . . .	61
7.3. Aufbau von Client und Server in Meteor 1.0 [Mon15] . . . . .	64
7.4. Die Architektur von Node.js [Hec16, Seite 12] . . . . .	68
7.5. Das konzeptionelle Modell von Node.js [Nod] . . . . .	69
7.6. Popularität von dokumentenbasierten Datenbanksystemen [DE16] .	70
7.7. Aufbau der dokumentenbasierten Datenbank MongoDB [Sut12] .	71
8.1. Poolansicht in THMcards . . . . .	76
8.2. Eine Dozentenanfrage in THMcards ist nur mit vollständig angegebenen Kontaktdaten möglich . . . . .	78
8.3. Benachrichtigungen über Kartensatz-Freigaben . . . . .	79
8.4. Übersicht über alle Kartensatz-Freigabe-Anfragen . . . . .	79
8.5. Einen Kartensatz freigeben oder nicht freigeben . . . . .	80
8.6. Sicherheitsmechanismus von Braintree [Braa] . . . . .	80
8.7. Auswahl der Mitgliedschaft in THMcards . . . . .	81
8.8. Mitgliedschaftsarten von THMcards in Braintree . . . . .	82
8.9. Kaufen eines Kartensatzes in THMcards . . . . .	82
8.10. Übersicht über angelegte Zahlungsmethoden in THMcards . . . .	84
8.11. Dashboard im Braintree Control Panel . . . . .	85
8.12. Übersicht über alle Kunden von THMcards in Braintree . . . . .	85
8.13. Einzelansicht eines Kunden von THMcards in Braintree . . . . .	86



# Tabellenverzeichnis

7.1. Größe der Framework Community (Stand 12.09.2016) . . . . .	55
7.2. Größe der einzelnen Frameworks (Stand 12.09.2016) . . . . .	57



# Listingverzeichnis

7.1. Definition einer Collection . . . . .	65
7.2. Veröffentlichung der Daten im Server . . . . .	65
7.3. Abonnieren der Daten im Client . . . . .	65
7.4. Definition einer Funktion im Server . . . . .	66
7.5. Method Call im Client . . . . .	66
7.6. Veröffentlichung der Daten im Client . . . . .	66
7.7. Iteration von Collections in Meteor . . . . .	67
7.8. Bedingungen in Meteor-Templates . . . . .	67





# Abkürzungsverzeichnis

API .....	Application Programming Interface
BSON .....	Binary JSON
CSS .....	Cascading Style Sheets
GPL .....	GNU General Public License
GUI .....	Graphical User Interface
HTML .....	Hypertext Markup Language
JS .....	JavaScript
JSON .....	JavaScript Object Notation
LGPL .....	GNU Lesser General Public License
PSP .....	Payment Service Provider
SDK .....	Software Development Kit
SQL .....	Structured Query Language
VC .....	Venture-Capital



# **Teil I.**

## **Grundlagen**



# 1. Einführung

## 1.1. Motivation

Ideen für eine Web App sind leicht zu finden, diese allerdings in ein Produkt zu formen oder daraus ein Geschäft zu entwickeln ist der schwierige Teil. So beschreibt auch das Zitat vom Behance-Gründer Scott Belsky gut, dass eine Idee an sich lange nicht alles ist.

„It’s not about ideas. It’s about making ideas happen“

Wie ist es infolgedessen möglich, nicht an der Anzahl der vielen Ideen zu scheitern oder eine Anwendung bereitzustellen, wie man sie an jeder Ecke findet, sondern eine erfolgreiche Web App zu betreiben?

Es ist zudem allgemein bekannt, dass aus einer guten und erfolgreichen Internetseite ein beträchtlicher Gewinn erzielt werden kann. Nicht zuletzt aus diesem Grund haben es in der letzten Zeit schon viele Gründer versucht aus ihrer Idee im Internet Profit zu ziehen.

Die Welt ist im Wandel. Immer mehr Menschen haben durch PCs, Laptops, Tablets, Smartphones oder Smartwatches Zugriff zum Internet. Bereits heute nutzen mehr als 3,1 Milliarden Menschen das Internet. Diese Entwicklung betrifft jedoch nicht nur die Konsumenten, sondern auch Unternehmen, die auf veränderte Kundenbedürfnisse treffen und daher mit neuen Diensten an den Markt gehen müssen. Das Internet hat dementsprechend weitreichende Auswirkungen auf das gesamte Business-Umfeld und birgt die Möglichkeit für neue und innovative Geschäftsmodelle.

“Move fast and break things. Unless you are breaking stuff, you are not moving fast enough.”

Dieses Zitat von Mark Zuckerberg erläutert sehr anschaulich das dynamische Umfeld, in dem wir uns heutzutage befinden. Ohne Innovationen, die von Per-

sönlichkeiten und Unternehmen aus aller Welt ausgelöst wurden, hätten wir den heutigen technologischen Entwicklungsstand niemals erreichen können.

Warum demnach nicht selbst ebenso Teil dieses dynamischen Umfelds sein und mit einer E-Learning-Plattform Profit erzielen?

## 1.2. Problemstellung und Zielsetzung

Die Problemstellung, mit der sich diese Master-Thesis befasst, basiert auf der Einführung von kostenpflichtigen Diensten im Internet. Bei einem solchen kostenpflichtigen Dienst im Internet handelt es sich um eine über das Internet angebotene Leistung für die ein Nutzer bezahlt. In Abgrenzung zu einem Produkt, welches man kauft und anschließend besitzt, kann man einen Dienst nur so lange in Anspruch nehmen, wie man diesen von dem entsprechenden Anbieter bezieht.

Überwiegend Startups, aber auch etablierte Unternehmen, entwickeln immer wieder neue innovative Geschäftsmodelle, um mit ihren Diensten erfolgreich im Internet tätig zu sein. Dabei ist jedoch die Akzeptanz und somit die Nachfrage der potenziellen Nutzer auf ein solches Angebot nicht zu vernachlässigen. Auf Grund der immer noch vorhandenen Gratiskultur fehlt jedoch bekanntermaßen meist eine gewisse Zahlungsbereitschaft für kostenpflichtige Dienste im Internet. Dies lässt sich darauf zurückführen, dass früher vermehrt die Dienste kostenlos nutzbar waren, aber auch oftmals illegale Quellen genutzt wurden. Durch die steigende Quantität und Qualität der angebotenen Dienste im Internet, zeichnet sich mittlerweile jedoch ein Wandel hin zum Trend der steigenden Zahlungsbereitschaft ab.

Eine weitere Schwierigkeit beinhaltet die Art des in dieser Master-Thesis behandelten Dienstes. Durch die geplante Monetarisierung des momentan kostenlosen kollaborativen E-Learning-Dienstes, sollte die Lernqualität nicht beeinträchtigt, sondern vielmehr gesteigert werden. Ein kollaborativer Dienst ermöglicht den Nutzern Daten einzustellen, die wiederum von anderen Nutzern verwendet werden können. Die Richtigkeit der Daten ist bei einem solchen Ansatz allerdings nicht garantiert. Hierbei sollte die Wandlung hin zu einem kostenpflichtigen Dienst genutzt werden, um den Lernenden die Sicherheit zu geben keine falschen Lerninhalte zu erwerben.

Erklärtes Ziel dieser Arbeit ist es somit, ein passendes Geschäftsmodell für die E-Learning-Plattform zu entwickeln, um diesen als kostenpflichtigen Dienst im

Internet zu etablieren und dabei den Kundenbedürfnissen, insbesondere hinsichtlich der Lernqualität, gerecht wird.

## 1.3. Gliederung und Vorgehensweise

Die vorliegende Masterthesis ist grundlegend in drei Abschnitte aufgeteilt: Einem Grundlagenteil, einem theoretischen Teil und abschließend einem praktischen Teil.

Mit einem Überblick über den aktuellen Stand der Dinge im Web, bildet der einführende Teil die Grundlagen für die Entwicklung einer erfolgreichen Web App. Zudem werden in diesem Abschnitt die Erfolgs-Elemente einer Web App abgesteckt. In diesem Abschnitt wird demnach vor allem vermittelt was man wissen, tun und erwarten muss bevor man beginnt eine Web App zu entwickeln.

Der zweite Abschnitt widmet sich der Entwicklung einer Strategie für eine Webanwendung. Im ersten Kapitel dieses Abschnitts wird der Begriff Monetarisierung analysiert und definiert. Des weiteren wird dort auf die Nebenwirkungen der Kommerzialisierung von Bildung eingegangen. Mit der Definition des Begriffs Geschäftsmodell beginnt das nächste Kapitel. In diesem wird unter anderem auf den Begriff der Gratiskultur eingegangen, aber auch die verschiedenen Arten von Erlösmodelle und die Erstellung eines digitalen Geschäftsmodells mit Hilfe eines Frameworks aufgezeigt. In diesem Kapitel wird somit die Grundlage für eine langfristige Rentabilität gesetzt. Die Kunst der Preisgestaltung wird im darauf folgenden Kapitel behandelt. Hierbei werden die Konsumentenbedürfnisse und die Nachfrage berücksichtigt, um eine aussagekräftige Preisfindung zu ermöglichen. Ebenfalls wird aufgezeigt was die Psychologie eines Preises bei einer Preisbildung bedeutet. Damit die Transaktionen zwischen Kunde und Anbieter auch stattfinden können benötigt man so genannte Bezahlmethoden. Im letzten Kapitel dieses Abschnitts wird somit auf die verschiedenen Arten von Bezahlmethoden eingegangen und eine Auswahl von bekannten und potenziellen Bezahlssystemen vorgestellt und evaluiert.

Im dritten Abschnitt wird die Umsetzung der theoretischen Inhalte am Beispiel der E-Learning-Plattform THMcards aufgezeigt. Zuerst wird hierfür die E-Learning-Plattform THMcards im Allgemeinen vorgestellt. Dabei wird auf die Re-Implementierung von THMcards und allen zugehörigen Aktionen eingegangen. Anschließend werden die einzelnen technischen Aspekte von THMcards erläutert. Das nächste Kapitel befasst sich mit der eigentlichen Monetarisierung von

der E-Learning-Plattform THMcards. Dabei wird auf das gewählte Geschäftsmodell, die Preisgestaltung, die Neuerungen im System und dem Bezahlvorgang eingegangen. Mit einem abschließenden Resümee wird diese Masterthesis beendet.



## 2. Erfolg einer Web-App

Was ist die Motivation und was sind die Ziele hinter einer Web-App? Eine Umfrage aus dem Jahr 2011 zeigt deutlich, weshalb Entwickler sich dazu entschlossen haben eine Web-App zu entwickeln. Erstaunlicherweise sind die finanziellen Gründe erst auf dem zweiten Platz der Umfrage gelandet. Die meisten Entwickler betreiben die Web-App-Entwicklung als Freizeitbeschäftigung, sehen es aber als Chance, bei Erfolg damit Geld zu verdienen. Um den Zufall, ob man mit einer Web-App Geld verdienen kann, zu beseitigen, ist es notwendig zu wissen was eine solche Web-App ist und wie man diese erfolgreich betreiben kann. [Zam12a]

### 2.1. Was ist eine Web-App?

Kurzum ist der Begriff Web-App die Abkürzung für "Website Application", zu Deutsch Web-Anwendung. Eine Web-App ist somit ein Stück Software, dass auf einer Webseite, gewöhnlich innerhalb eines beliebigen Browsers läuft. Eine Web-App muss somit nicht zwangsweise eine eigenständige Webseite sein, sondern kann durchaus ein Teil einer solchen sein. Web-Apps verbessern vorrangig die Webseite selber und sind somit auch allgegenwärtig, werden jedoch nur selten als solche von den Nutzern erkannt.

Zugunsten einer genaueren Definition des Begriffes "Web-App" werden folgend Objekte erwähnt, die keine Web-App sind.

Zuallererst ist der Unterschied zwischen mobilen bzw. nativen Apps (Apps auf dem Smartphone) und einer Web-App darzulegen. Beide werden häufig nur als Apps bezeichnet, stehen auch in gewissen Maße im Zusammenhang, unterscheiden sich in ihrer Art der Nutzung jedoch grundlegend. Mobile Apps müssen vor der Nutzung auf das entsprechende Gerät geladen werden und funktionieren dann eigenständig. Web-Apps hingegen laufen quasi im Hintergrund einer Webseite und ermöglichen somit einen unbeschränkteren Zugriff und Nutzung. [Zam12b]

Der Unterschied zwischen einer Web-App und einer Webseite ist hingegen nicht so einfach zu erkennen. Hält man sich an die strikten Definitionen beider Begriffe, sind alle Webseiten eine Web-App. Mit dieser Feststellung lässt sich indes weiterhin keine klare Linie zwischen den zwei Begriffen ziehen. Betrachtet man nun die beiden Begriffe Web-Design und Web-Development, ist zu sehen, dass eine Webseite nur mittels Web-Design zu erstellen ist, eine Web-App allerdings beides benötigt. Somit ist die Wichtigkeit des Design und der Entwicklung gleichermaßen groß bei einer Web-App. [Zam12b]

## 2.2. Erfolgsfaktoren

„Das Geheimnis des Erfolgs ist, den Standpunkt des anderen zu verstehen.“(Henry Ford, Unternehmer, 1863-1947)

Wer kennt es nicht? Man selber hat eine gute Idee, die seit Monaten in Betracht gezogen wird und möchte nun endlich den Anfang machen. Meist gibt es zudem so viele Ideen, allerdings wurde nie ein Versuch gestartet diese umzusetzen, da schlichtweg keine Entscheidung getroffen werden konnte, mit welcher Idee angefangen werden sollte.

In diesem Kapitel werden die typischen Merkmale von erfolgreichen Web-Anwendungen betrachtet, mit denen Projektideen in geeigneter Weise bewertet und priorisiert werden können. Am Ende dieses Kapitels ist es möglich festzustellen, ob die gewählte App realisierbar ist.

Es gibt vier zusammenhängende Elemente die bei einer Web-Anwendung betrachtet werden müssen:

1. Die Idee: Was soll die Anwendung können?
2. Die Originalität der Anwendung, sowohl der Idee als auch der Umsetzung.
3. Die Qualität: Bezogen auf die Umsetzung.
4. Der Kontext: Wie passt die Anwendung in den weiteren Kontext der Web-Technologien?

### 2.2.1. Idee

Die Idee ist die Grundlage für die Aufgaben, die die Anwendung bearbeiten kann. Für Neueinsteiger sind einzelne Ideen zumeist irreführend, da Erfahrung, Persön-

lichkeit und eine entsprechende Umgebung fehlen oder sehr beschränkt vorhanden sind.

Es ist somit für viele Entwickler schwierig zu beurteilen, ob ihre Idee ausreichend Potenzial hat erfolgreich zu werden. Um diese Einschränkungen zu umgehen gibt es die Möglichkeit eine einfache vorläufige Analyse durchzuführen.

Erstens ist es notwendig sicherzustellen, was der eigentliche Zweck der App ist. Das Ganze sollte als Elevator Pitch<sup>1</sup> formuliert werden. Hierbei sind kurze Sätze wie "Es ist ein Lernkartensystem für Studenten" oder "Es macht lernen im Studium einfacher" völlig ausreichend.

Anschließend ist es wichtig über den möglichen Zielmarkt nachzudenken und welche Vorteile eine solche Anwendung dort bringen würde. Dabei sind folgende Fragen hilfreich:

- Gibt es einen passenden Zielmarkt?
- Wie groß ist dieser Markt und wie viele Personen davon sind potenzielle Kunden?
- Wie bewerkstelligen die potenziellen Kunden aktuell ihre Aufgaben?
- Wie viel Geld geben diese Online dafür aus?
- Wie wird sich der Markt in den nächsten Jahren weiterentwickeln?

In Kapitel 4 (Geschäftsmodelle) wird auf diese Thematik genauer eingegangen.

### 2.2.2. Einzigartigkeit

Die Einzigartigkeit oder Originalität einer Anwendung hängt grundlegend von der Idee selbst ab. Im einfachsten Szenario ist eine Anwendung dann einzigartig, wenn die Idee eine Anwendung hervorbringt die etwas komplett neues ermöglicht. Zudem gibt es allerdings auch die Möglichkeit eine Anwendung basierend auf einer bereits vorhandenen Anwendung einzigartig zu machen, indem für diese zum Beispiel eine einmalige Oberfläche oder ein neuartiger Algorithmus implementiert wird.

Die nachfolgende Grafik stellt den Zusammenhang zwischen Einzigartigkeit und der daraus resultierenden Wahrnehmung dar.

---

<sup>1</sup><http://www.startwerk.ch/2012/01/30/30-sekunden-die-zahlen-dein-elevator-pitch/>

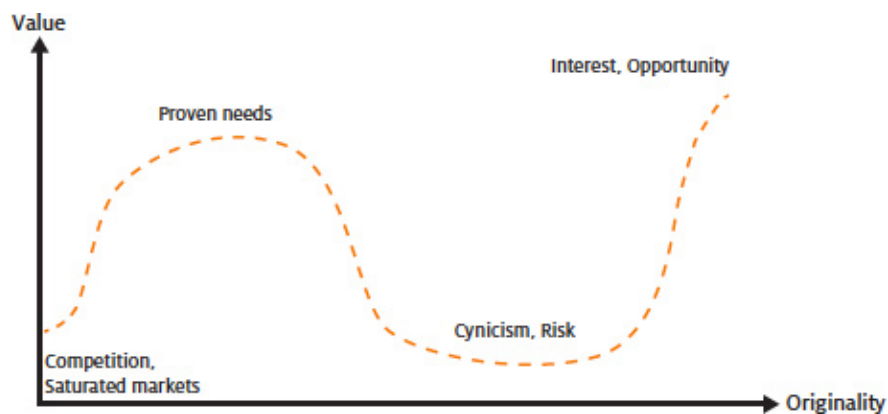


Abbildung 2.1.: Wahrnehmung einer Web-App gemessen an deren Einzigartigkeit  
Quelle: [http://webappsucccess.com/img-2\\_2.jpg](http://webappsucccess.com/img-2_2.jpg)

Anwendungen die wenig Einzigartigkeit bieten werden schlichtweg auch weniger wahrgenommen. Dies ist durch den großen Wettbewerb und den gesättigten Markt vorprogrammiert. Sobald die Anwendung etwas an Einzigartigkeit hinzugewinnt - sich somit von den Mitbewerbern abgrenzt, nimmt die Konkurrenz ab und die Anwendung wird somit durch ihre neuen Funktionen auch besser wahrgenommen. Dabei wird allerdings aus Nutzersicht weiterhin die Verbindung zu bekannten Diensten beibehalten um eine schnelle Eingewöhnung der Nutzer zu ermöglichen. Wenn die Einzigartigkeit nun allerdings noch weiter steigt entsteht das Risiko, dass potenzielle Nutzer mit dem neuen Dienst zuerst nichts anfangen können. Hierbei müssen vor allem die Vorteile der App gut vermarktet werden um sie erfolgreich zu machen. Sollte die geplante Anwendung ein noch größeres Alleinstellungsmerkmal vorweisen, so ist hierbei wiederum der Wahrnehmungsfaktor sehr groß. Problematisch ist in diesem Bereich jedoch die fehlende Markterfahrung und infolgedessen muss eine ausführliche Analyse des Nutzerbedarfs durchgeführt werden.

### 2.2.3. Umsetzung

Neben der Idee und der Originalität einer Web-Anwendung ist der dritte wichtige Teil die qualitative Umsetzung des Ganzen. Hierbei sind folgende vier Bereiche besonders wichtig bei der Durchführung:

- Entwicklung
- Design

- Preisgestaltung
- Marketing

Die Themen Entwicklung, Design und Marketing sind kein Teil dieser Arbeit und werden aus diesem Grund hier nicht genauer behandelt. Genauere Informationen zur Preisgestaltung wird es in Kapitel 5 geben.

### 2.2.4. Kontext

Der wichtigste, aber komplexeste Erfolgsfaktor ist der Kontext einer Web-Anwendung. Viele Umgebungsfaktoren beeinflussen den Erfolg ohne selbst Möglichkeiten zu haben, etwas dagegen zu unternehmen.

Eines dieser Umgebungsfaktoren ist die Geografie. Das Internet ist zwar global, allerdings gibt es reichlich lokale Unterschiede in dem entsprechenden Markt. Dazu zählen unter anderem die verschiedenen Sprachen, Kulturen und Vorlieben, allerdings auch der Wohlstand verbunden mit der unterschiedlichen Bereitschaft Geld auszugeben. Hierbei kann durch fehlende Sprachen oder Einschränkung auf bestimmte Orte der Markt ungewollt eingeschränkt werden.

Ein weiterer Faktor ist die Wirtschaftslage in verschiedenen Gebieten. Größere Wirtschaftskrisen können einer zuvor gut funktionierenden Web-Anwendung gehörige Umsatz-Probleme bereiten. Solche Szenarien müssen bei der Entwicklung gut eingeplant und passende Lösungen dazu bereit gestellt werden.

Nicht zu vernachlässigen ist zudem der Wettbewerb. Eine Kontrolle der Mitbewerber ist schlicht nicht möglich, allerdings ist es machbar sich strategisch gut in einem Markt zu positionieren um erfolgreich zu werden und zu bleiben.



## **Teil II.**

### **Monetarisierung in der Theorie**





## 3. Monetarisierung

### 3.1. Begriffsdefinition

Monetarisierung beschreibt den Vorgang, mit dem aus einem bestehenden Produkt ein finanzieller Nutzen gewonnen wird [Grüa].

Die wörtliche Bedeutung von Monetarisierung ist "Vergeldlichung". Die Monetarisierung richtet sich somit an bereits bestehende kostenfreie Produkte, die durch ein Geschäftsmodell den Inhabern finanzielle Einnahmen gewähren soll. Solche Geschäftsmodelle gibt es in den verschiedensten Arten. Genauer erklärt werden diese in Kapitel 4. Im Folgenden werden nun einige Grundsatzfragen zur Monetarisierung beantwortet.

### 3.2. Kommerzielle Open Source Software

Immer mehr Unternehmen setzen ihren Fokus darauf, ihre Software als Open Source anzubieten. Mittlerweile setzen laut einer Umfrage 78 Prozent aller Unternehmen zumindest zum Teil auf öffentlich zugänglichen Quellcode [Bla16]. Darüber hinaus werden Open Source Projekte an Universitäten und Hochschulen stark gefördert. Hierbei stellt sich die Frage, ob ein Programm, welches den Anschein erregt "Frei" zu sein, auch kommerziell nutzbar sein darf.

Hierbei sollte zuallererst das Missverständnis, eine Open Source Software sei das Gegenteil einer kommerziellen Software, aufgeklärt werden. Eine Software wird erst zu einer Open Source Software, wenn diese unter einer entsprechenden Lizenz veröffentlicht wird. Diese Lizenzen gewähren bestimmte Nutzungsfreiheiten und somit auch, je nach Auswahl dieser, den Verkauf oder die Vermietung der Software [ifr16]. In der Open Source Definition der Open Source Initiative werden 10 Kriterien aufgezählt, die eine solche Software erfüllen muss um als Open Source zu gelten. Eine dieser Kriterien ist es eine Diskriminierung von Einsatzbereichen nicht zu erlauben. Damit sind unter anderem Nutzungseinschränkungen hinsicht-

lich der Benutzung des Programms im kommerziellen Bereich nicht erlaubt. Ziel dieses Kriterium ist es, Nutzer einer kommerziellen Version in die Community aufzunehmen und diese nicht auszuschließen [Ini, Wikc].

Zusammenfassend lässt sich somit sagen, eine Open Source Software ist mit der entsprechenden Lizenz kommerziell nutzbar, kann aber auch frei verfügbar sein.

## 3.3. Kommerzialisierung von Bildung

Eine Kommerzialisierung bestimmter Bereiche birgt immer Risiken in sich. So ist es auch bei der Kommerzialisierung von Bildung. Wie kann garantiert werden, dass die Bildung eines Menschen durch die Kommerzialisierung nicht negativ beeinflusst, eher noch verbessert wird?

Bei der Ausbildung von Schülern und Studenten wird schon seit vielen Jahren große Mengen an Geld mit Unterrichtsmaterialien erwirtschaftet. Zudem ist bereits die Lehre selbst, mit privaten Schulen und Hochschulen zu einem gewissen Teil aus der Hand des entsprechenden Staates gegeben worden. Es ist somit nichts neues Bildung aus nichtstaatlichen Quellen zu erhalten. Allerdings hat der Staat bzw. die Länder auch weiterhin die Kontrolle über die Bildungsinhalte und bietet diese für Jedermann frei zugänglich an.

Nicht zwangsläufig ist eine Auslagerung der Lerninhalte zu Drittanbietern auch mit einer Verschlechterung der Lernqualität gleichzusetzen. Die Anbieter einer Lernplattform sind letzten Endes selbst darauf aus gute Inhalte in ihrem System anbieten zu können, da es ihrem Produkt zu gute kommt. Sollte sich jedoch eine solche Plattform eine quasi Monopolstellung erarbeitet haben, ist es für die Lernenden nicht mehr einfach Alternativen zu nutzen. Es gilt somit auch zukünftig zu beachten Lerninhalte weiterhin unabhängig anzubieten.

Ein weiteres Problem der Kommerzialisierung von Bildung ist, dass auch personenbezogenen Daten eines Lernenden genutzt werden können, um diese an potenzielle Arbeitgeber zu verkaufen. Hier ist es nötig einheitliche Datenschutzrichtlinien zu erarbeiten, um die Lernenden vor solchen Machenschaften zu schützen.

Eine Möglichkeit eine solche Plattform unabhängig von kommerziellen Unternehmen zu führen, besteht beispielshalber durch das Nutzen von Spenden oder mittels einer staatlichen Förderung. Solche Förderungen durch den Staat werden bereits in vielen Ländern umgesetzt. Besonders in Kalifornien finanziert der Bun-

desstaat mittlerweile kaum noch Schulbuchverläge und Lern-Software-Hersteller deren Produkte wegen Urheberrechtsschutz nur eingeschränkt zur Verfügung stehen. Vielmehr werden dort Bildungsmaterialien die unter den Creative-Commons-Lizenzen bereitgestellt werden staatlich gefördert. [Buc13]

Wenn man alles berücksichtigt ist es wichtig eine Unabhängigkeit der Lerninhalte zu fokussieren. Dies ist durch die freie Bereitstellung der Plattform in Verbindung mit der Förderungen durch den Staat am Ehesten umzusetzen.



## 4. Geschäftsmodelle

Ohne ein durchdachtes und funktionierendes Geschäftsmodell wird selbst die beste Geschäftsidee nicht funktionieren.

Mit der Definition des Begriffs Geschäftsmodell wird dieses Kapitel eingeleitet. Anschließend wird auf den Begriff der Gratiskultur eingegangen, aber auch die verschiedenen Arten von Erlösmodelle. In diesem Kapitel wird somit die Grundlage für eine langfristige Rentabilität gesetzt.

### 4.1. Definition

Eine einheitliche Definition des Begriffes Geschäftsmodell gibt es nicht. Zudem entstehen gerade seit Anbruch des Internet-Zeitalters viele verschiedene und neue Geschäftsmodelle, doch nicht alle sind auch gleichzeitig erfolgreiche Geschäftsmodelle.

Nach der Definition von Stähler [Stä01] besteht ein Geschäftsmodell aus den folgenden drei Hauptkomponenten:

1. Nutzenversprechen: Bezeichnet welchen Nutzen Kunden des Unternehmens aus der Verbindung mit diesem Unternehmen ziehen können.
2. Architektur der Wertschöpfung: Stellt dar, in welcher Konfiguration die Leistung angeboten wird.
3. Ertragsmodell: Beschreibt welche Erlöse das Unternehmen aus welchen Quellen generiert.

In dieser einfachen Definition gibt es somit die drei Komponenten "was", "wie" und "wodurch". Weitere Definitionen von Wirtz [Wir16] oder von Osterwalder und Pigneur [YP10] sind weitaus detaillierter aber auch komplexer. Dabei ist zu beachten, dass ein Geschäftsmodell immer nur eine Konvergenz der wirklichen

Struktur eines Unternehmens ist. Je nach Einsatzschwerpunkt des Geschäftsmodells reicht eine abstrakte Version oder aber es wird eine detaillierte Version benötigt. [Inn, Wikd, Stä01, Grüc]

## 4.2. Aufgaben eines Geschäftsmodells

Ein Geschäftsmodell ist keine Strategie an sich, sondern eine Beschreibung des Unternehmen. Die Entwicklung von Geschäftsmodellen in Unternehmen sind momentan primär für die strategische Analyse mit folgenden Zielen wichtig:

- Bestehende Unternehmen und deren Abläufe besser zu verstehen.
- Verbessern des aktuellen Unternehmen um sich gegenüber Wettbewerbern einen Vorteil herauszuarbeiten.
- Neue Geschäftsideen systematisch darstellen und evaluieren.
- Überprüfen eines neuen Geschäftsmodell auf seine Skalierbarkeit.

Da die ersten drei oben gelisteten Aufgaben eines Geschäftsmodells sich selbst erklären, wird nun noch einmal genauer auf die Skalierbarkeit von Geschäftsmodellen eingegangen. Ein Geschäftsmodell ist dann skalierbar, wenn es einem Unternehmen erlaubt wachsen zu können. Dies geschieht unter anderem wenn innerhalb kürzester Zeit durch eine gestiegene Nachfrage die Produktion oder das Kapital erhöht werden müssen. [Grüb]

## 4.3. Gratiskultur

Geschäftsmodelle die ein Angebot im Internet abdecken wollen, müssen die dort vorhandene Gratiskultur zusätzlich beachten. Die Gratiskultur selbst gibt es seit den Anfängen des Internets als Massenmedium. Sie nimmt immer mehr ab, ist allerdings immer noch bei vielen Internetnutzern stark vertreten. Gratiskultur heißt für diese Personen, dass im Internet angebotenen Inhalte, wie Musik, Film und Software kostenlos sein müssen. [Dar07]

Viele Inhalte wurden zu Beginn des Internets illegalerweise kostenlos angeboten. Diese Angebote wurden zudem gerne und auch häufig genutzt, da es keine wirklichen Alternativen gab. Diese illegalen Quellen gibt es heutzutage immer

noch, jedoch wurde mittlerweile festgestellt, dass wenn es ein passendes legales Angebot für Nutzer gibt, diese auch gerne dafür bezahlen. [Lob13]

Das Kaufen eines Produktes im Internet ist somit, auch wenn die Personenanzahl abnehmend, immer noch eine Hürde. Diese Tatsache muss somit bei der Erstellung und der Auswahl eines Geschäftsmodells im Hinterkopf behalten werden.

## 4.4. Arten

Es gibt viele verschiedene Arten von Geschäftsmodellen. Dabei ist es jedoch in den seltensten Fällen die richtige Entscheidung sich für nur eines dieser Modelle zu entscheiden. Für ein Unternehmen ist vielmehr eine Verknüpfung mehrerer Modelle zu einem Hybrid-Geschäftsmodell die richtige Wahl.

Geschäftsmodelle lassen sich grundsätzlich in folgende drei Kategorien aufteilen:

1. Direkt: Verkauf der Inhalte durch kostenpflichtige Apps, In-App-Käufe oder Paywalls.
2. Indirekt: Gutes Marketing, Services und Branding die das Kaufverhalten positiv beeinflussen.
3. Lateral: Werbeeinnahmen durch Dritte oder Verkauf von Nutzerdaten.

Da die Wahl des passenden Geschäftsmodells stark vom Angebot und der Zielgruppe der App abhängt, lohnt sich häufig eine Kombination der Modelle. Zudem ist die Auswahl ohne die vorherige Erforschung des Marktes und der Zielgruppe nicht möglich. Daher ist es notwendig eine Markt- und Kundenanalyse (siehe dazu [Aul16, Zam12b]) im Voraus durchzuführen.

In den folgenden Abschnitten wird auf die für eine Web-Anwendung relevanten Modelle und ihre Besonderheiten eingegangen.

### 4.4.1. Direkt

Direkte Geschäftsmodelle sind solche, die Erlöse durch den Verkauf von Inhalten oder Funktionen erwirtschaften.

**Abonnement** Bei dem Abonnement-Model müssen die Kunden für einen bestimmten Zeitraum wiederkehrend zahlen. Im Allgemeinen ist der Zeitraum meist auf monatlich festgelegt, kann jedoch auch beliebig gewählt werden. Dieses Modell erlaubt eine sehr fortdauernde Einnahmequelle und ist zudem sehr gut mit den jeweils monatlichen Gehältern der Kunden und der geschäftlichen Abrechnungen zu vereinen.

**Bezahlzeitraum:** Ein hoher Bezahlzeitraum ist oftmals eine zu hohe Hürde für einzelne Personen, da dadurch an einem Zeitpunkt höhere Kosten anfallen. Es sollte demgemäß monatliches Bezahlen angeboten werden. Sollte die Anwendung indes auch im Enterprise-Segment angeboten werden, ist es wichtig, zusätzlich zu den monatlichen Bezahlungen, auch jährliche Bezahlungen zu erlauben, da dies für die Abrechnung innerhalb eines Unternehmens deutlich einfacher ist.

**Abschlusszeitraum:** Grundsätzlich ist eine monatlich kündbare Vereinbarung die richtige Variante. Die Kunden empfinden diese Art des Angebots flexibler und schließen somit das Abonnement wahrscheinlicher ab. Im Vergleich zu einer jährlichen oder mehrjährigen Vereinbarung können mit einer monatlichen Vereinbarung somit auch höhere Beträge eingenommen werden.

**Variationen:**

- **Fester Preis:** Für jeden Kunden ist der Preis gleich
- **Variabler Preis:** Unterschiedliche Preise für verschiedene Angebote (Funktionsumfang, Nutzeranzahl, ...)
- **À la carte:** Funktionen sind mit Preisen belegt. Nutzer wählt die benötigten Funktionen aus.
- **Pay what you want:** Jeder Kunde bekommt das gleiche Angebot, wählt seinen Preis dafür selbst aus. Ein minimaler Preis ist hier notwendig.

**Freemium** Der Begriff setzt sich aus den Wörtern Free und Premium zusammen. Freemium bezeichnet ein Geschäftsmodell welches sowohl Frei zugänglich ist, zudem aber auch Geld für zusätzliche Funktionen verlangen kann. Prinzipiell ist es somit ein Abonnement-Modell mit variablem Preis, inklusive der Erweiterung einen Teil des Angebots frei zu erhalten. Dieses Modell muss wohl durchdacht angewendet werden um die Balance zwischen zahlen-



den und freien Nutzern in der Waage zu halten. Da das Anbieten der freien Zugänge auch gewissermaßen eine Marketingaktion ist, ist Freemium zudem auch eine Kombination aus einem direkten und indirekten Geschäftsmodell. Freemium sollte nur eingesetzt werden, wenn alle der folgenden Aspekte erfüllt sind:

- Das Produkt wird in einem stark umkämpften Markt freigegeben oder es ist ein Dienst den die Leute noch nicht als notwendig ansehen.
- Von dem Produkt wird zu erwarten sein, dass es die Nutzer langfristig hält.
- Das Produkt wird im Laufe der Zeit eine Wertsteigerung durch die Anwender erhalten.

**Ad-Hoc Bezahlung** Anwender können Zugriff auf bestimmte Teile der Anwendung mithilfe von Einzelzahlungen bekommen. Folgende vier Variationen sind dabei am bekanntesten:

- Bezahlen pro Benutzung: Der Nutzer kann die Anwendung nach Bezahlung einmalig oder für einen bestimmten Zeitraum nutzen.
- Virtuelle Produkte: Der Anwender kauft bei diesem Modell ein digitales Produkt. Digitale Produkte können vom Anbieter einfach und ohne wesentlichen Aufwand vervielfältigt werden. Auch hierbei können wiederum die Grundfunktionen der Anwendung kostenlos sein und der Nutzer kauft sich weitere Funktionen oder Inhalte durch eine einmalige Bezahlung hinzu.
- Spenden: Für alle Anwender sind die gleichen Funktionen verfügbar. Zusätzlich können Nutzer eine Spende tätigen und werden so an speziellen Stellen der Anwendung dafür gewürdigt.

#### 4.4.2. Indirekt

Geschäftsmodelle die mit Marketing, Services und Branding die Anwendung etablieren wollen sind den indirekten Modellen zuzuordnen.

**Anwenderdaten ausnutzen** Dieses Geschäftsmodell ist nur für Anwendungen, die große Mengen an nutzergenerierten Daten speichern, geeignet. Hierbei werden zu den gespeicherten Daten der Anwender passende Funktionen zur kostenpflichtigen Weiterverarbeitung angeboten.

**Plattform** Die Anwendung bietet eine Entwicklungsplattform an, die von unabhängigen Entwicklern frei genutzt werden kann um die angebotene Anwendung zu erweitern. Nur ab einer bestimmten Bekanntheit oder Beliebtheit werden Servicegebühren vom Entwickler erhoben.

**Branding** Bei diesem Geschäftsmodell wird die Person, die hinter der Anwendung oder Produkt steht, besonders hervorgehoben um eine Verbindung zwischen der Person und der Anwendung in der Öffentlichkeit herzustellen. Dadurch können mithilfe zum Produkt passenden Präsentationen, Workshops, Büchern und beratender Arbeit zusätzliches Geld verdient werden.

**Übernahme durch Verkauf** Auch der Verkauf der Anwendung und einer darauf folgenden Übernahme durch ein anderes Unternehmen kann ein eigenständiges Geschäftsmodell sein.

### 4.4.3. Lateral

Laterale Geschäftsmodelle erhalten ihren Erlös von einer dritten Partei. Hierzu zählen unter anderem das Schalten von Werbung und der Verkauf von personenbezogenen Daten.

**Werbung** Das Schalten von Werbung in einer Anwendung ist ein gängiges Geschäftsmodell. Die geschaltete Werbung wird in der Regel nach "Kosten-pro-Klick" (CPC), "Kosten-pro-Aktion" (CPA) oder "Kosten-pro-tausend-Impressionen" (CPM) abgerechnet. Bei diesem Geschäftsmodell ist jedoch schwierig voranzuplanen wie viel Erlös am Ende dabei herauskommt. Zudem kann nicht auf jeder Anwendung problemlos jegliche Art von Werbung eingebunden werden.

**Sponsoring** Sponsoren können eine Anwendung durch ihren bereitgestellten Beitrag unterstützen. Ein solches Sponsoring kann selbstverständlich zeitlich begrenzt ebenso wie auf Dauer laufen. Im Gegenzug können beispielsweise auch spezielle Werbeanzeigen eingeblendet werden.

**Paid Content** Mit Paid Content können außenstehende Unternehmen ihre Inhalte in eine Anwendung durch begleichen einer Zahlung einbringen. Im Vergleich zur normalen Werbung, wird der bezahlte Inhalte ähnlich eines in der Anwendung vorhanden Inhalts dargestellt. Ein solches Modell ist somit

vielmehr für ein datenreiche Anwendung, als für eine funktionsorientierte Anwendung geeignet.

**Lizensierter Inhalt** Wenn die Anwendung genügend Daten (keine personenbezogenen Daten) anbietet, die weiter verwendet werden können und sollen, ist dies über die Lizenzierung der Inhalte möglich.

**Personenbezogene Daten** Natürlich können auch die gesammelte Daten einer Person weiterverkauft oder einen zeitweiligen Zugriff darauf ermöglicht werden. Allerdings sind solche Praktiken nicht gerne bei den Konsumenten gesehen und sollten nicht eingesetzt werden.

## 4.5. Auswahl eines geeigneten Geschäftsmodells

Wie bereits in der Definition eines Geschäftsmodells erwähnt, ist die Auswahl eines Einzelnen dieser nun bekannten Modelle meist nicht ausreichend.

Des Weiteren sollte die Auswahl eines Geschäftsmodell nicht vernachlässigt werden, denn es ist wichtig um aus einer Anwendung eine erfolgreiche Anwendung zu machen. Ob das gewählte Geschäftsmodell eine gute Entscheidung für das Projekt war, kann mit folgenden zwei Kennzahlen überprüft werden:

- Customer Lifetime Value (CLV): Der Deckungsbeitrag den ein Kunde während seines gesamten Kundenlebens realisiert.
- Cost of Customer Acquisition (COCA): Bemisst die Kosten zur Kundengewinnung.

Ein bereits umgesetztes Geschäftsmodell kann jederzeit nachträglich verändert werden. Jedoch ist dieser Vorgang nicht einfach und sollte, wenn möglich, vermieden werden. Es sollte somit von vorne herein versucht werden ein Geschäftsmodell zu wählen, was sich von der Konkurrenz absetzt und der Anwendung einen Wettbewerbsvorteil verschafft. [Zam12b, Aul16]



## 5. Preisgestaltung

Die Kunst der Preisgestaltung wird in diesem Abschnitt behandelt. Hierbei werden die Konsumentenbedürfnisse und die Nachfrage berücksichtigt um eine aussagekräftige Preisfindung zu ermöglichen.

Nach der Auswahl eines Geschäftsmodells für eine Anwendung, folgt die zugehörige Preisbestimmung für die angebotenen Funktionen und Inhalte. Während ein Geschäftsmodell in der Regel nicht gewechselt wird, ist die Preisgestaltung durchaus ein dauerhafter Prozess. Jeder Unternehmer ist in seiner Preisgestaltung grundsätzlich frei. Die Preisgestaltung ist nur bei eventuellen staatlichen Vorschriften nicht vollends frei.

Folgende Kriterien müssen bei allerdings bei jeder Preisgestaltung berücksichtigt werden:

- Art und Qualität des Produkts
- Laufende Kosten
- Preise der Konkurrenzprodukte
- Preisvorstellungen und Kaufkraft der potenziellen Konsumenten
- Mögliche Rabatte

In den folgenden Unterabschnitten werden diese einzelnen Kriterien genauer betrachtet.

### 5.1. Art und Qualität des Produktes

Dieses Kriterium sollte relativ einfach verständlich sein. Eine Anwendung mit einem hohem Qualitätsanspruch und der zum Markt und Kundenbedürfnis passenden Art kann einen etwas höheren Preis verlangen, als eine Anwendung ohne solche Ansprüche.

## 5.2. Laufende Kosten

Der Preis einer Anwendung sollte niemals direkt durch die entstehenden Kosten ermittelt werden. Es sollte jedoch immer ein Mindestmaß an Absicherung durch den gewählten Preis vorhanden sein, um die Anwendung dauerhaft betreiben zu können.

Bereits angefallene Entwicklungskosten sind aus der Preisgestaltung raus zu halten. Die wenigsten Anwender werden Verständnis für das Beachten solcher Kosten aufbringen. Es ist das Ziel, diese Entwicklungskosten im Nachhinein durch einen eventuellen Profit wieder zu egalisieren.

Wesentlich sind jedoch die langfristigen Kosten einer Anwendung. Es ist essentiell den dauerhaften Kostenaufwand einer Anwendung mindestens zu decken um diese am Laufen zu behalten. Diese Unkosten teilen sich zum einen in feste Kosten und zum anderen in variable Kosten auf.

Zu den **festen Kosten** gehören all die, die über einen Zeitraum relativ konstant bleiben:

- Hosting, Backup und Bandbreite
- Support und weiterführende Entwicklungskosten
- Bürofläche und zugehörige Kosten
- Marketingkosten
- Gebühren für Bank- und Händlerkosten
- Rechts- und Versicherungskosten

**Variable Kosten** sind solche, die von den Kunden selbst beeinflusst werden:

- Hosting, Backup und Bandbreite
- Zahlungsbearbeitungsgebühren

Wie zu sehen ist, werden Hosting, Backup und Bandbreite sowohl bei den festen als auch bei den variablen Kosten aufgeführt. Einerseits können diese Kostenfaktoren je nach Auswahl des Anbieters und Vertrags über eine gewisse Zeit die gleichen Kosten aufweisen. Andererseits ist vor allem auch zu Beginn einer Anwendung zu empfehlen auf Cloud-Computing und somit auf variable Verträge

zu setzen. Zudem ist oftmals auch bei ressourcenintensiven Anwendungen mit Mehrkosten im Bereich des Hosting und Backups zu rechnen, da dort jeder neue Nutzer erheblich mehr Datenvolumen zusätzlich benötigen wird.

Mit der Auflistung aller Kosten und der geschätzten Nutzeranzahl über ein Jahr, kann der minimale Preis erhoben werden, den ein Nutzer mindestens Zahlen muss, damit die Anwendung keine roten Zahlen schreibt.

## 5.3. Konkurrenz

Instinktiv wird sicherlich jeder einmal ein Auge auf die Konkurrenz-Produkte und deren Preise werfen. Dieser Vorgang sollte auch ab und zu wiederholt werden, jedoch ist ein solcher Preisabgleich mit den Wettbewerben wenig aussagekräftig.

Sollte der eigene Preis deutlich unter dem des Mitbewerbers liegen kann es bei den Kunden dazu führen, die Anwendung als billig anzusehen. Zudem wird es wahrscheinlich zu einem Preiskampf mit den Mitbewerbern kommen. Dies ist für niemanden förderlich und sollte vermieden werden.

Andererseits werden die Kunden jedoch auch von einem deutlich zu hohen Preis im Vergleich zum Konkurrenzprodukt abgeschreckt. Hier muss dann schon dem Kunden der deutliche Vorteil der eigenen Anwendung in Gegenüberstellung zu denen der Konkurrenten sichtbar sein.

Den Preis des Konkurrenten zu übernehmen ist allerdings auch nicht der richtige Weg. Bestandskunden der Wettbewerber werden dann sicherlich nicht wechseln und Neukunden werden auch keinen Mehrwert darin sehen bei einem gleichen Preis zu einer unbekannten Anwendung zu greifen.

Es ist somit in jeder Hinsicht nicht sonderlich von Vorteil sich an dem Mitbewerber zu orientieren. Die Preise der Konkurrenz in bestimmten Intervallen zu überprüfen schadet sicherlich nicht, dennoch sollten dies nicht als Vorlage für die eigene Preisgestaltung dienen.

## 5.4. Konsumentenbedürfnisse

Jeder Mensch hat eine begrenzte Anzahl an Bedürfnissen, die er dauerhaft versucht zufrieden zu stellen. Meist werden die vorhandenen Wünsche durch den Erwerb von materiellen Gütern befriedigt. Bedürfnisse die sich durchaus auf die Preisgestaltung auswirken sind die Rarität von Dingen, die finanzielle Sicher-

heit, sowie die Zeit, der Komfort, das Ansehen, die Zugehörigkeit, das körperliche Befinden, die Unterhaltung und die Herausforderung eines Menschen beziehungsweise Konsumenten.

### 5.4.1. Zeit

Eines der größten Bedürfnisse des Menschen ist es mehr Zeit zu haben. Deshalb sind auch Anwendungen, die durch Einfachheit und Effizienz die Arbeit erleichtern, stark gefragt. Bei solchen Angeboten sind potenzielle Kunden bereit, für einen Dienst der ihnen eine gewisse Zeitersparnis einbringt, auch dementsprechend mehr zu bezahlen.

Als anschauliches Beispiel dient hier eine Zug-Verbindung zwischen zwei Städten. Mit dem Regionalexpress dauert die Verbindung um einiges länger als mit dem ICE. Allerdings wird dem Kunden für die Fahrt mit dem ICE dementsprechend auch mehr berechnet. Als Berechnungsgrundlage für den Mehrpreis, kann einem Kunden, für eine Funktion oder einen Service, der ihm die Hälfte der Zeit erspart, durchaus der doppelte Preis in Rechnung gestellt werden.

### 5.4.2. Rarität

Viele Dinge im alltäglichen Leben sind durch ihre Knappheit nur teuer und oft auch nur schwer zu bekommen. Oftmals haben die Gegenstände selbst keinen großen Materialwert und Nutzen, sind jedoch durch ihre Seltenheit alles andere als preiswert. Beispiele solcher Gegenstände sind vor allem Kunst, Antiquitäten und andere Einzelstücke. Daneben gibt es allerdings auch Dinge, die selten sind aber in der großen Bevölkerung durchaus nützlich und oftmals auch unabdingbar sind. Hierzu zählt als Beispiel das Öl, welches knapp ist, aber dennoch von fast jedermann benötigt wird.

In der Informatik kann diese Rarität allen voran bei Domains des World Wide Web beobachtet werden. Gefühlt gibt es unendlich an verfügbaren Domain-Namen. Eine beliebige Domain kann für wenig Geld registriert werden. Jedoch gibt es im gleichen System bei kurzen und aussagekräftigen Domain-Namen oftmals Übernahmen in Millionenhöhen.

Eine direkte Auswirkung auf die Preisgestaltung einer Anwendung lässt sich bei der Rarität nicht herstellen. Zu beachten ist jedoch, dass eine begrenzte Nut-



zeranzahl und eine eingeschränkte Registrierung über Einladungen den Preis einer Anwendung nach oben treiben kann.

### **5.4.3. Komfort**

Die Anzahl an im täglichen Leben zu erwerbenden Komfortfunktionen nimmt stetig zu. Diese Angebote werden auch von vielen Menschen genutzt und geschätzt, da sie ihrem Körper etwas gutes tun möchten. Dies wird vorrangig durch den Erwerb einer Komfortfunktion und einer damit einhergehenden Verbesserung der Benutzerfreundlichkeit erreicht.

Ein gute Beispiel für dieses Thema ist etwa die Nutzung von Spotify. Üblicherweise melden sich dort Nutzer in der freien Version an, erhalten dort fast uneingeschränkten Zugriff auf die Inhalte, werden alledrings durch Pop-up-Werbung und Ankündigungen um ihren Komfort gebracht. Viele Nutzer entscheiden sich deshalb zur kostenpflichtigen Mitgliedschaft.

### **5.4.4. Ansehen**

Eine Mitgliedschaft als Statussymbol. Jeder Mensch will, sei es bewusst oder unterbewusst, sein Selbstwertgefühl stärken. Dies geschieht heutzutage zu Meist über den Erwerb neuer Markenklamotten, Technik oder auch Autos.

Wie bei der Rarität ist möglich hohe Preise zu fordern, jedoch nicht durch eine künstliche Verknappung, sondern durch einen direkt höheren Preis um ein Statussymbol zu erschaffen.

Eine solche Preisgestaltung wird allerdings zu den wenigsten Anwendungsfällen passen.

### **5.4.5. Zugehörigkeit**

Ein weiteres Bedürfnis des Menschen ist die Zugehörigkeit. Damit sind Beziehungen und Zuneigungen zu anderen Menschen gemeint. Beziehungen zwischen Menschen werden grundsätzlich innerhalb von Sozialen-Netzwerken oder Dating-Plattformen erstellt.

Dabei sind die Sozialen-Netzwerke meistens frei zugänglich, die Dating-Seiten erheben jedoch einen Beitrag. Sobald eine Anwendung eine höhere Wahrscheinlichkeit hat, dass Personen miteinander intimer werden, ist es möglich und angebracht dafür einen höheren Preis zu verlangen.

### 5.4.6. Körperliches Befinden

Auch bei diesem Bedürfnis ist es nicht auf Anhieb ersichtlich, wie zu einem geplanten Angebot ein passender Preis gefunden werden kann. Gesundheit, Sicherheit und Wohlbefinden sind die grundlegenden Elemente dieses Bedürfnisses. Hier lässt sich zur Preisgestaltung einzig erwähnen, dass je effektiver die Anwendung ist, desto mehr Geld kann dafür verlangt werden.

### 5.4.7. Finanzielle Sicherheit

Zur finanziellen Sicherheit gehören unter anderem Begriffe wie Wohlhaben, Erfolg, Karriere oder Status. Ein Mensch sieht die finanzielle Sicherheit ironischerweise selbst als Freiheit an, wird dafür jedoch die meiste Zeit seines Lebens eingebunden.

Anwendungen die in den Bereich der finanziellen Sicherheit fallen, sind sowohl offensichtliche wie Banking und Job-Suche. Zu diesem Bedürfnis gehören jedoch auch Dienstleister, die über kurz oder lang den Wohlstand erhöhen können. Solche sind beispielsweise Gutscheine-Sammlungen, Schulungen und Glücksspiele.

Festgestellt wurde, dass mehr in Anwendungen investiert wird, die sich auf persönliches Wohlhaben und Erfolg beziehen. Dabei zählt nicht nur die Wahrscheinlichkeit, dass eine solche Anwendung erfolgreich ist, sondern ob sie auch den Investierenden genügend Kontrolle über ihre Investitionen bietet. Somit lässt sich sagen, je höher die Erfolgchancen sind und je größer die Kontrolle darüber ist, desto mehr Zahlungsbereitschaft ist vorhanden.

### 5.4.8. Unterhaltung

Das Unterhaltungsbedürfnis ist breit gefächert. Von vermindern der Langeweile bis hin zur Sehnsucht nach Fröhlichkeit kann dabei alles abgedeckt werden.

Video- und Audio-Angebote, sowie Online-Games gehören beispielsweise zu den Anwendungen die sich mit diesem Bedürfnis beschäftigen.

Auch hier gilt wieder, je niedriger der Preis für eine angebotene Anwendung, desto weniger Inhalt wird erwartet. Das heißt aber auch, dass wenn etwas in der Unterhaltungsbranche für wenig Geld angeboten wird, die Hemmschwelle es zu kaufen sinkt, da es den Preis auch bei einem schlechten Inhalt rechtfertigen würde.

### 5.4.9. Herausforderung

Auch anspruchsvolle Aufgaben, wie das Ausleben von Kreativität und das Verlangen nach Wissen, sind ein Bedürfnis des Menschen.

Zu diesem Bedürfnis werden Kunst- und Fotografie-Anwendungen, Blogging und Lernsysteme gezählt. Jedoch wird es kaum eine Anwendung geben die ausschließlich dieses Bedürfnis abdeckt. Oftmals umfassen solche Anwendungen auch die Bedürfnissen der Zugehörigkeit und Unterhaltung.

Bei diesem Bedürfnis ist es unmöglich eine geeignete Preisgestaltung mittels Formeln oder einfachen Hilfsmitteln zu erstellen, vielmehr muss hier eine geeignete Strategie gefunden werden.

## 5.5. Die Nachfragekurve

Die Preisgestaltung auf Basis der verschiedenen Kundenbedürfnisse ist ein guter Einstieg und steckt bei den meisten Bedürfnissen den groben Rahmen ab. Allerdings wird dabei meist nur ein Preis für ein entsprechendes Angebot eruiert. Nicht jeder mögliche Kunde ist jedoch bereit genau diesen Preis zu bezahlen. Um nun herauszufinden, wie viele Kunden für ein bestimmtes Angebot in Frage kommen, ist das Erstellen einer Nachfragekurve, wie in Abbildung 5.1 zu sehen, essentiell.

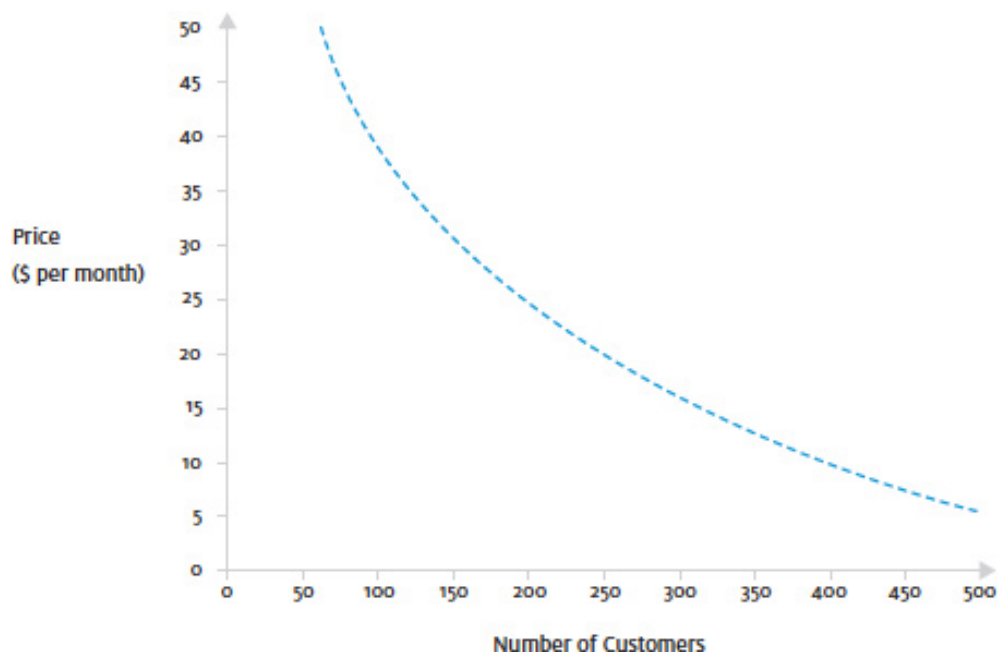


Abbildung 5.1.: Die Nachfragekurve einer Web-App  
Quelle: [http://webappsucces.com/img-10\\_7.jpg](http://webappsucces.com/img-10_7.jpg)

Sofern nur ein Angebot den Kunden zur Verfügung gestellt wird, ist eine Nachfragekurve nicht leicht zu erstellen. Es besteht nur die Möglichkeit den Preis zu erhöhen oder zu senken, um anschließend zu sehen, wie viele Kunden das jeweilige Angebot anspricht. Diese Art der Untersuchung bringt allerdings folgende Nachteile mit sich:

Senkt man den Preis, sind nicht nur die Bestandskunden verärgert, sondern es kommt auch zu einem unnatürlichen Anstieg an Anmeldungen, da viele erwarten, dass das Angebot nur kurzzeitig gilt. Das ist zwar kurzfristig vorteilhaft für den Umsatz, lässt jedoch keine aussagekräftige Datenerhebung für eine Nachfragekurve zu.

Ebenfalls ist eine Erhöhung des Preises nicht sonderlich nützlich. Sollte man nämlich merken, dass es zu wenig Nachfrage gibt, muss man den Preis wieder senken, was auch dazu führt, dass viele Kunden eine Rückzahlung fordern.

Eine einfache Änderung der Preise ist somit keine Option. Vielmehr ist es besser dem Kunden von vorne herein mehrere Auswahlmöglichkeiten zu bieten. Mehr dazu im folgenden Abschnitt.

## 5.6. Preissegmentierung

Mittels Preissegmentierung, also dem Kunden mehrere Preise für unterschiedliche Versionen einer Anwendung anbieten, ist es möglich den besten Preis zu finden. Eine größere Auswahl erhöht immer die Chance, dass ein Kunde die passende Option für sich findet. Dennoch ist eine zu große Auswahl wiederum kontraproduktiv, da es zu unübersichtlich für den Kunden wird.

Die Preissegmentierung benötigt als Basis den Preis, der aus den vorherigen Kriterien ermittelt wurde. Dieser wird zuerst als Standardauswahl gesetzt. Hinzu kommt ein teureres Angebot mit einem größeren Funktionsumfang und ein günstigeres Angebot mit weniger Funktionen innerhalb der Anwendung. Drei Auswahlmöglichkeiten sind für einen neuen Kunden als Optimum anzusehen. Es können in der Anfangszeit jedoch auch bis zu fünf verschiedene Versionen angeboten werden. Diese sollten allerdings nach erfolgreicher Evaluierung der einzelnen Preise, auf die drei besten Angebote reduziert werden.

Eine solche Evaluierung ist mithilfe der Nachfragekurve möglich. Allerdings ist die Nachfragekurve alleine nicht aussagekräftig, denn nicht immer ist der Preis, der den meisten Kunden zusagt, auch der wirtschaftlich am Besten geeignete. Um

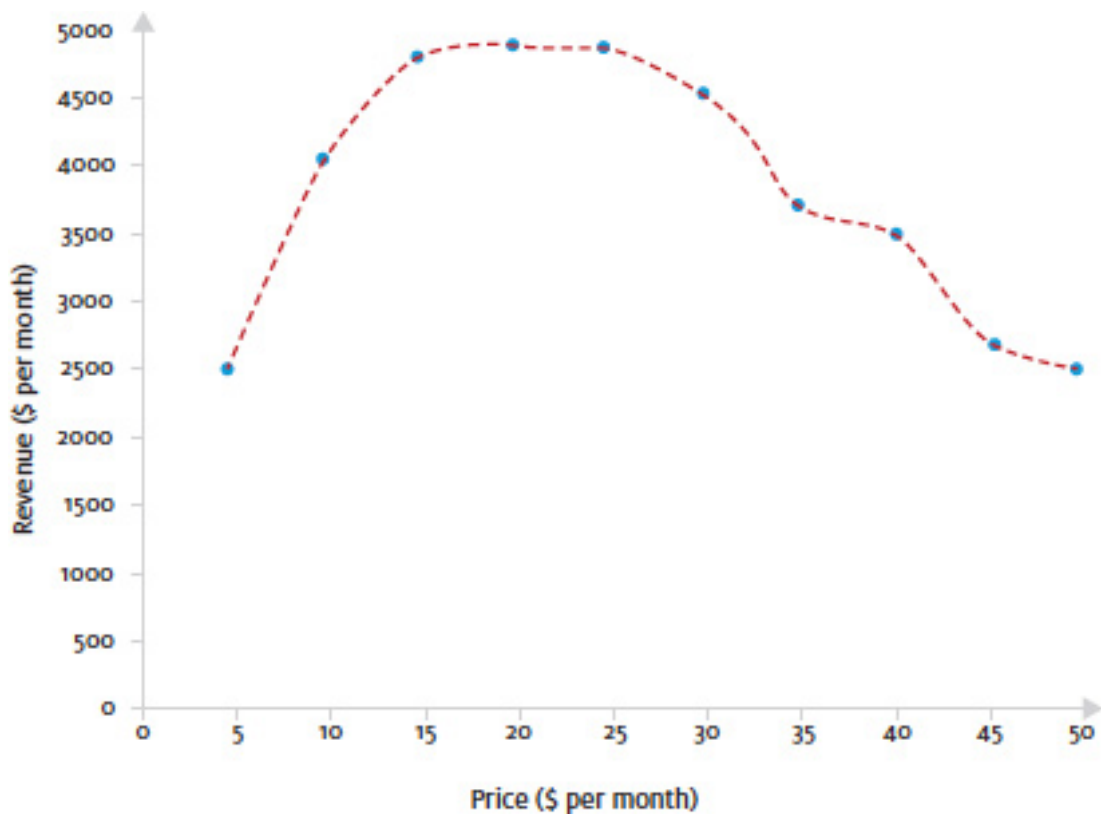


Abbildung 5.2.: Die monatlichen Einnahmen einer Web-App  
Quelle: [http://webappsuccess.com/img-10\\_8.jpg](http://webappsuccess.com/img-10_8.jpg)

dies zu erkennen ist eine Verrechnung (Monatlicher Preis multipliziert mit der Kundenanzahl) der Nachfragekurve notwendig. Diese Ergebnisse sind am Besten wiederum in ein Diagramm zu setzen.

Eine solche Übersicht, die die monatlichen Einnahmen abhängig vom Angebot darstellt, ist in Abbildung 5.2 zu sehen. Bei diesem Beispiel sind somit für die Anwendung die Variation von 15, 20 und 25\$ am geeignetsten.

## 5.7. Die Psychologie des Preises

Der grobe Rahmen der Preisgestaltung wird insofern auf Basis des Geschäftsmodells und mit Hilfe der verschiedenen Kriterien und der Preissegmentierung festgelegt. Zusätzlich zu beachten ist jedoch auch die Psychologie eines Preises, sprich wie eine Person den angebotenen Preis wahr nimmt. [Mac16, Zam12b, Pur11]

### 5.7.1. Preisanker

Stehen einem potenziellen Kunden unterschiedliche Versionen mit unterschiedlichen Preisen zur Auswahl, wählt der Nutzer meist den Mittleren, da dies als sicherste Version wahrgenommen wird. Wenn man demgemäß mehrere Preise für eine Anwendung anbietet, sollte man den Preis, den man aus Unternehmersicht bevorzugt, zur mittleren Version machen. [Wira, Yu13]

### 5.7.2. Preisanmutung

Der Fokus eines jeden Kunden liegt zuerst immer auf der ersten Ziffer des Preises [MS09]. Aus diesem Grund werden die meisten Preise mit Hilfe des 99-Cent-Tricks, also dem Absenken des Preises um einen Cent auf einen \*,99 Betrag, verschönert. Auch wenn dieser Trick bekannt sein sollte, fühlt sich ein Preis von 1,99€ eher wie ein 1€ Kauf, anstatt wie eines 2€ Kaufes an. Da der Fokus so stark auf der ersten Ziffer einer Zahl liegt, hat zudem auch eine Preiserhöhung von 24€ auf 29€ keine wirklichen Auswirkungen auf die Nachfrage der Kunden. [Wirb]

Andererseits wirkt sich auch die letzte Ziffer eines Preises auf die Preisvorstellung aus. Eine 9 am Ende steht für ein preiswertes Angebot, eine 0 repräsentiert Premium-Produkte und eine 4 oder 7 steht für einen präzise berechneten Preis. [Yu13]

### 5.7.3. Unsichtbare Preisänderung

Eine unsichtbare Preisänderung ist nach dem Weber-Fechnersches Gesetz möglich. Dieses Gesetz sagt aus, dass alle Änderungen, somit auch Preisänderungen, relativ zu ihrem Startpunkt sind. [Dic09]

Studien haben ergeben, dass Preiserhöhungen unter 10% von dem Kunden selbst meist nicht wahrgenommen werden. Dies gilt jedoch andersherum auch bei Preissenkungen. Eine Preisreduzierung um weniger als 10% wird schlicht nicht wahrgenommen, da die Differenz zum Ausgangspreis zu niedrig ist. [Sex13]

## **6. Bezahlvorgang**

Damit eine Transaktion zwischen Kunden und Anbieter stattfinden kann, werden so genannte Bezahlmethoden benötigt. Jedoch nicht jedes dieser Systeme ist für jeden Kunden die richtige Wahl. Es ist somit wichtig dem Kunden ein einfaches und kostenfreies Bezahlsystem anzubieten um keine Bezahlhürde zu erzeugen. Eine Bezahlhürde wird oftmals durch eine zu geringe oder falsche Auswahl der Bezahlmethoden oder wird durch einen zu langwierigen Kauf-Prozess geschaffen. Da es dies zu vermeiden gilt, werden in diesem Kapitel daher verschiedene Arten von Bezahlmethoden dargeboten und eine Auswahl von bekannten und potenziellen Bezahlssystemen vorgestellt und evaluiert.

### **6.1. Problemstellung**

Sei es beim Online-Shopping, dem Kaufen einer digitalen Ware oder dem Abschließen eines Online-Abonnements, es gibt bei all diesen Transaktionen ein Problem: Händler und Käufer begegnen sich nicht persönlich und haben deshalb keinen Grund einander zu vertrauen. Ein weiterer Umstand der nicht zu vernachlässigen ist, ist oftmals die Langwierigkeit einer einzelnen Transaktion. Dies ist auch die Ursache warum es verschiedene Arten und Anbieter von Zahlungssystemen gibt.

### **6.2. Zahlungsmethoden**

Die verfügbaren Zahlungsmethoden lassen sich in zwei grundlegende Kategorien einteilen: Zahlungen, die über traditionelle Systeme getätigt werden oder die Zahlung über Online-Dienste. In den nachfolgenden Abschnitten werden die gängigsten Zahlungsmethoden für eine Web-Anwendung aufgezählt und erläutert.

### 6.2.1. Gegenüberstellung

#### **Vorkasse**

Irrtümlicherweise wird der Kauf per Vorkasse häufig als eine Zahlungsmethode angesehen, die für den Händler kostenlos ist. Allerdings entstehen bei der zeitaufwändigen Zuordnung von Zahlungen hohe Personal-Kosten. Dies ist unter anderem mit der Nachforschung bei nicht zuordnungsbaaren Zahlungen, sowie für diverse Nebenkosten wie Telefonkosten zu begründen.

Für einen Händler oder Anbieter hat die Vorkasse auf der anderen Seite jedoch auch seine Vorzüge. Der Händler kann mit dieser Methode sichergehen, seine Leistung auch bezahlt zu bekommen. Kunden hingegen haben keinerlei Sicherheit vom Händler, sie sind somit vor allem bei unbekannten Anbietern sehr vorsichtig.

Sollte die Vorkasse als einzigste Zahlungsart angeboten werden, ist dies für den Umsatz nicht sonderlich förderlich. Eine Studie von ibi Research [res13] belegt, dass bei einem solchen Fall fast neun von zehn Kunden den Kauf direkt abbrechen. Die Vorkasse sollte somit niemals als alleinige Zahlungsmethode angeboten werden.

#### **Nachnahme**

Die Nachnahme ist mittlerweile eine der unattraktivsten Zahlungsmethoden, da sie für Anbieter und Kunde gleichermaßen teuer ist. Die höhere Sicherheit auf Seiten des Kunden ist zudem auch nicht voll durchdacht. Der Kunde bezahlt zwar erst nach Erhalt einer Ware, diese Ware kann jedoch bei ihrer Übergabe nicht auf ihre Vollständigkeit überprüft werden. Die Bezahlung per Nachnahme ist mittlerweile eine fast gar nicht mehr genutzte Art. Ausschließlich teure Produkte werden noch gerne mit dieser Variante gekauft. Anbieter solcher Produkte sollten dennoch zusätzliche Alternativen zur Nachnahme anbieten. Die Nachnahme ist zudem nicht für digitale Güter und Abonnements geeignet.

#### **Rechnung**

Der Kauf auf Rechnung ist immer noch eine der beliebtesten Zahlweisen. Kunden bevorzugen diese, da diese Zahlweise eine große Sicherheit bietet und ebenso, da die Zahlung selbst hinausgezögert werden kann um eine kurzfristige Zahlungsunfähigkeit zu überbrücken. Diese Zahlungsart kann somit einerseits überaus um-



satzfördernd wirken, andererseits jedoch ist sie für Händler gleichzeitig risikoreich, da Rechnungen verspätet oder gar nicht bezahlt werden.

Diese Methode sollte allerdings, da viele ältere Onlinekunden nur so einkaufen, unbedingt angeboten werden. Das bestehende Risiko ist mithilfe eines Dienstleisters für externe Zahlungsabsicherung wie Klarna, BillPay, BillSafe oder PayRate zu minimieren, da Statistisch betrachtet 95 Prozent aller Händler mit dieser Zahlungsweise Probleme haben.

### **Lastschrift**

Die Lastschrift ist in Deutschland ein weit verbreitetes und beliebtes Zahlungsverfahren. Der Händler erhält hierbei von dem Kunden eine Ermächtigung einen festgeschriebenen Betrag von dem Konto des Kunden abzugheben. Hierbei muss sich der Händler, wie beim Kauf auf Rechnung, gegen Zahlungsausfälle oder Rückbuchungen absichern. Auf Kundenseite sind es vor allem die hohe Akzeptanz, der Bekanntheitsgrad, sowie die schnelle und einfache Abwicklung, die den Kauf per Rechnung oder Lastschrift attraktiv machen.

Im Gegensatz zu den drei vorigen Zahlungsweisen eignet sich das Lastschriftverfahren nicht nur für den Normalfall einer einmaligen Einzelzahlung und für physikalische Güter, sondern ebenso für digitale Güter oder Abonnements.

### **Kreditkarte**

Eine weitere bei Kunden besonders akzeptierte Zahlungsmethode, die auch internationale Transaktionen erleichtert, ist das Zahlen per Kreditkarte. Für Online-Händler, die auch international verkaufen möchten, ist die Kreditkarte somit ein Pflicht-Zahlungsmittel. Das Risiko für einen Zahlungsausfall ist für den Händler minimal, da der Kaufbetrag zunächst vom Kreditkartenanbieter autorisiert werden muss, bevor die Bestellung abgeschlossen werden kann. Der Nachteil für einen Anbieter sind jedoch die relativ hohen Gebühren der Kreditkartenanbieter bei einer Transaktion.

Dieses Zahlungsverfahren ist sowohl für Einmalzahlungen als auch für Abonnements oder für den Einzug von Kleinstbeträgen (Micropayment) geeignet.

### **E-Wallet**

Immer mehr Kunden vertrauen auf die Bezahlung über externe Zahlungsdienstleister. Diese ermöglichen den Händlern einen besseren Service, da diese die Wa-

re gleich nach Zahlungsbestätigung durch den Dienstleister verschicken können. Zudem haben die Händler eine gewisse Sicherheit gegen Zahlungsausfälle, da ein nicht gedecktes Konto des Kunden ein Problem des Dienstleisters wird. Allerdings sind, um diese Zahlungsmethode mit all ihrer Sicherheit anbieten zu können, von der Händlerseite die anfallenden Kosten zu tragen. Diese Art des Bezahlen ist dennoch definitiv anzubieten. Jedoch ist darauf zu achten, welcher Anbieter in den entsprechenden Ländern am Meisten vertreten ist. Die wenigsten Kunden werden sich für einen Kauf schnell noch ein zusätzliches E-Wallet-Konto bei einem anderen Anbieter erstellen, wenn ihr eigentlicher E-Wallet-Anbieter nicht unterstützt wird.

Die E-Wallet-Zahlungsverfahren sind unter anderem besonders nützlich für Abonnement-Abwicklungen und Micropayment, werden allerdings auch oft für Einmalzahlungen eingesetzt.

### **Direkt Überweisung**

Die direkte Überweisung ist eine Erweiterung der Vorkasse. Bei dieser Art der Überweisung sitzt zwischen den Banken der beiden Parteien ein Zahlungsanbieter. Sobald die Zahlung aus Kundensicht erfolgt ist, teilt dieser dem Händler dies sofort mit. Dadurch ist die Zahlung garantiert und die Lieferung der Kundenbestellung kann unmittelbar erfolgen ohne auf die ausstehende Überweisung warten zu müssen. Eine Händlersicherheit ist somit gegeben, eine Kundensicherheit jedoch weiterhin nicht.

Dieses Zahlungsverfahren ist ebenfalls, wie die Vorkasse, nur für Einmalzahlungen geeignet.

### **Dezentrale Zahlungen**

Bei der dezentralen Zahlung werden Überweisungen mittels eines Verbunds von Rechnern über das Internet mithilfe einer speziellen Peer-to-Peer-Anwendung abgewickelt. Bei dieser Art der Zahlung wird keine zentrale Abwicklungsstelle wie im herkömmlichen Bankverkehr benötigt. Da diese Zahlungsart noch relativ neu ist, fehlt dementsprechend auch noch die Akzeptanz in der Bevölkerung. Einzig bei anonymen Zahlungen, ist diese Zahlungsmethode relativ erfolgreich.

Dezentrale Zahlungen sind für alle Arten der Zahlung einsetzbar.

### 6.2.2. Auswahl treffen

Es gibt keine passende Zahlungsart, die für jeden Kunden, jede Zahlungsart und jeden Markt passt. Deshalb ist eine Kombination aus mehrere Zahlungsmethoden, am Besten zusammengestellt aus den meistverwendeten Zahlungsarten, zu bevorzugen. Nach einer Studie [res13] sinkt die Kaufabbruchquote fast auf null, sobald sechs der beliebtesten Zahlungsmethoden angeboten werden. Allerdings sollte der Integrations-, Verwaltungs- und Kostenaufwand bei einer solch hohen Anzahl an Diensten nicht vernachlässigt werden. Unter anderem für kleine bis mittelgroße Unternehmen ist es daher ratsam zuerst nur zwei Bezahldienste zu integrieren. Für eine gewissenhafte Entscheidung welche Bezahldienste genutzt werden sollen, gibt es einige Anhaltspunkte die im folgenden behandelt werden.

#### Popularität

Die in 2016, wie in Abbildung 6.1 zu sehen, beliebtesten Bezahlverfahren der Deutschen sind diverse online Bezahlssysteme gefolgt vom Kauf auf Rechnung. Solche Online-Bezahlssysteme sind unter anderem PayPal, sofortüberweisung.de, ClickandBuy oder giropay. Traditionelle Bezahlssysteme kommen im Jahr 2016 auf einen Nutzer-Anteil von 57%. Sie werden somit immer noch häufig von Kunden genutzt.

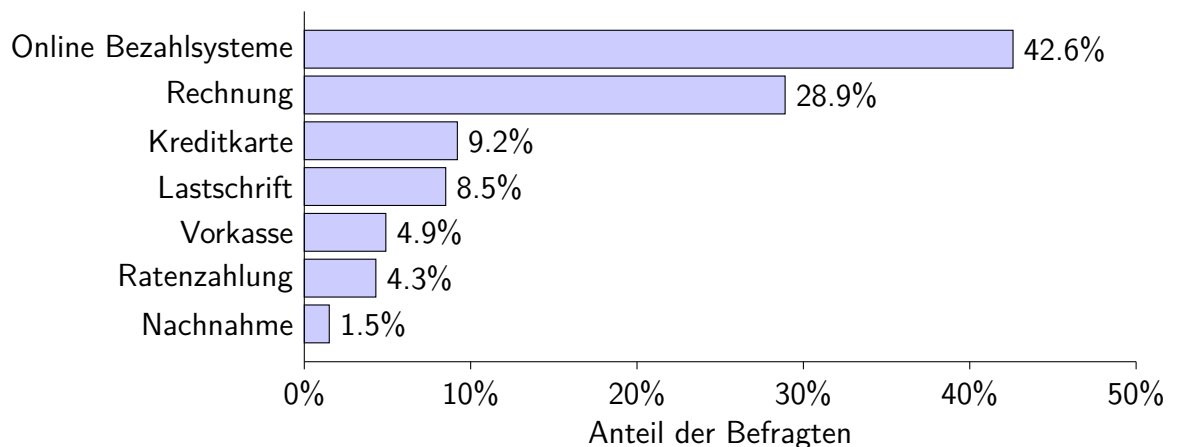


Abbildung 6.1.: Genutzte Bezahlverfahren in 2016 beim Kauf im Online- und Versandhandel [Bon16]

Es gilt jedoch zu beachten, dass althergebrachte Zahlungsarten wie der Kauf auf Rechnung oder mittels Lastschrift vor allem ihre Beliebtheit mit Personen über

50 Jahren erzielen. Diese Zahlungsmittel werden somit über kurz oder lang an Popularität verlieren, wohingegen moderne Zahlungsmittel immer häufiger genutzt werden. In Abbildung 6.2 ist die Popularität der verschiedenen Bezahlverfahren als Verlauf dargestellt. Diesem Diagramm ist zu entnehmen, dass eine Veränderung nur sehr langsam von staten geht. Zudem ist ein Zuwachs an Popularität nur bei den online Bezahlssystemen zu erkennen. Der Kauf auf Rechnung sowie die Kreditkartenzahlung halten ihre Beliebtheit von Jahr zu Jahr auf einem in etwa gleichbleibenden Niveau.

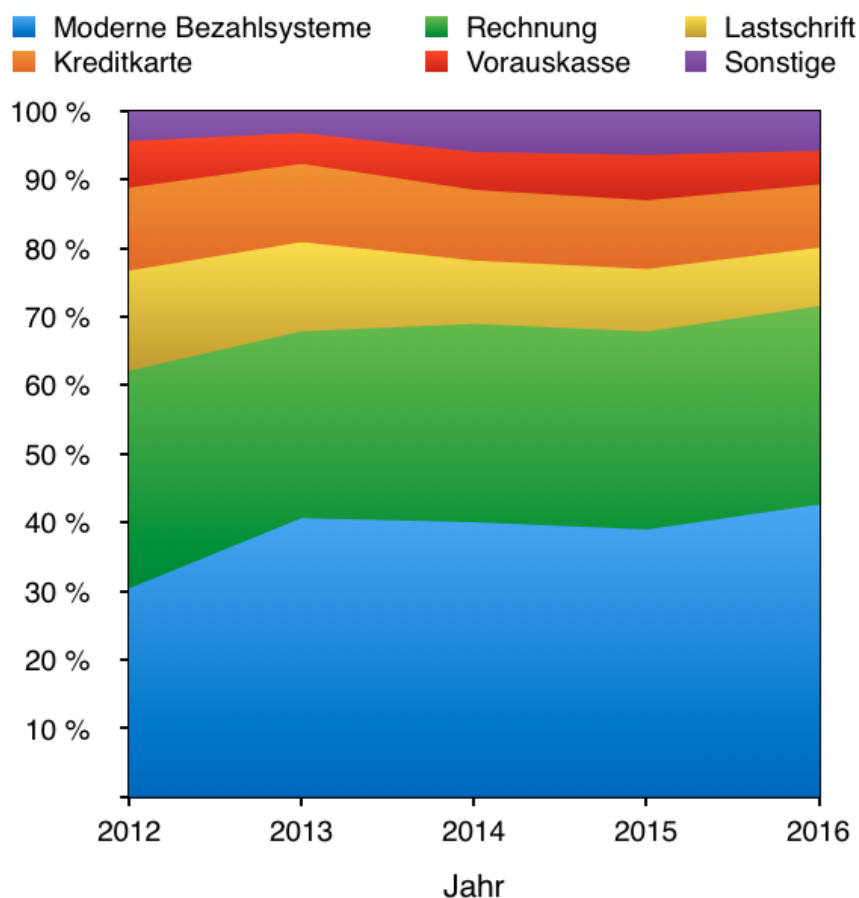


Abbildung 6.2.: Verlauf der genutzten Bezahlverfahren beim Kauf im Online- und Versandhandel [Bon16]

### Zielmarkt

Die reine Popularität eines Bezahlsystems reicht nicht aus um eine angemessene Entscheidung treffen zu können. Vielmehr muss die Popularität eines Bezahlsystems im entsprechenden Zielmarkt betrachtet werden. Somit gehören das gewähl-

te Geschäftsmodell und die entsprechende Zielgruppe ebenfalls zu den zentralen Aspekten bei der Wahl der geeigneten Bezahlmethoden.

Sollte das Produkt zum Beispiel als Abonnement angeboten werden, ist auch ein Einsatz von Bezahlssystem, die dies unterstützen, nötig. International agierende Projekte müssen die in einem jeweiligen Land präferierten Bezahlmethoden anbieten. Ein Kunde wird eben nur dort kaufen, wo er seine favorisierte Payment-Methode wiederfindet und einen möglichst einfachen Checkout-Prozess genießt.

Die Kreditkarte ist als international akzeptiertes Zahlungsmittel bei weltweit gültigen Angeboten einer Anwendungen immer einzusetzen.

#### **Sicherheit**

Bei der Auswahl der Bezahlmethoden ist auch auf die Sicherheit in zweierlei Hinsicht zu achten. Einerseits muss das Bezahlssystem dem Kunden eine gewisse Sicherheit bieten, muss andererseits jedoch auch den Händler schützen.

Es muss somit auf der einen Seite ein Mechanismus zum Schutz der Kundendaten eingesetzt werden (unter anderem durch eine Zwei-Faktor-Authentifizierung oder 3-D Secure). Um Kaufabbrüche zu vermeiden, sollte dieser Mechanismus allerdings den Bezahlvorgang selbst, nicht zu komplex werden lassen.

Dahingegen ist es parallel für den Händler wichtig, dass die Auszahlung der Umsätze zuverlässig und sicher stattfindet. Zudem ist die Haftung des Händlers bei Missbrauch von Daten zu beachten. Sollte etwa bei der Kreditkartenbezahlung auf das 3-D Secure-Verfahren verzichtet werden, haftet immer der Betreiber eines Projektes für missbräuchlich eingesetzte Kreditkarten. Bei anderen Bezahlverfahren, wie zum Beispiel der Kauf auf Rechnung oder dem Lastschriftverfahren, ist es für den Händler zudem notwendig sich bei seiner Bank gegen eventuelle Zahlungsausfälle der Kunden abzusichern.

### **6.3. Payment Service Provider**

Um dem Händler mehr Sicherheiten zu bieten gibt es sogenannte Payment Service Provider (PSP). Diese bieten verschiedene Zahlungsmethoden in einem System an und erleichtern somit die Integration und erhöhen die Sicherheit.

### 6.3.1. Kriterien

Bei der Auswahl eines solchen Anbieters stellen sich grundlegend drei Fragen. Erstens, ob der Anbieter die bevorzugten Zahlungsweisen der Zielgruppe unterstützt, sowie wie hoch die Kosten sind und zu letzt, wie einfach die Integration möglich ist.

#### Unterstützte Zahlungsmethoden

Unterstützt der Anbieter die bevorzugten Zahlungsweisen der Zielgruppe? Ein PSP ohne die passenden Zahlungsmethoden in einem Gebiet und Markt ist schlicht keine Option. Daher muss zuerst überprüft werden ob die gängigsten Zahlungsmethoden unterstützt werden.

#### Integrationsmöglichkeiten

Wie sieht es mit der Schnittstelle des Providers aus? Dabei gilt herauszufinden, ob diese einfach zu nutzen und auch für die entsprechenden Programmiersprachen verfügbar ist. Oftmals spielt auch die Frage ob ein Mobile-SDK angeboten wird eine wichtige Rolle.

#### Kosten

Gibt es einmalige Implementierungskosten? Wie hoch sind die monatlichen Kosten und die Transaktionsgebühren? Bei Kreditkarten ist auch die Frage nach dem Disagio, also der Service-Gebühr des Kartenherausgebers, entscheidend.

Ein ausgewogenes Preis-Leistungsverhältnis ist elementar für den Händler, da die Kosten der Zahlungsmethode natürlich auch Einfluss auf die Einnahmen haben. Daneben sollten Bezahlmethoden gewählt werden, die für den Kunden keine Zusatzkosten verursachen, solange es mit dem Service vereinbar und sinnvoll ist. Auch Zahlungsausfälle verursachen Kosten. Daher ist es wichtig, entsprechende Betrugspräventionsmaßnahmen zu ergreifen und gegebenenfalls eine Bonitätsprüfung im Vorfeld durchzuführen.

### 6.3.2. Gegenüberstellung

#### Stripe

Stripe<sup>1</sup> wurde im Jahre 2010 in San Francisco von den Brüdern John und Patrick Collison gegründet [Wik16]. Dieser Dienst hat den Zahlungsmarkt gewissermaßen revolutioniert. Viele namhafte Investoren und VCs haben über 300 Millionen US-Dollar in das Startup investiert. Im Sommer 2015 hatte das Unternehmen einen geschätzten Marktwert von fünf Milliarden US-Dollar erreicht. [Rao15]

Zu den bekanntesten Nutzern der Plattform zählen Foursquare, Kickstarter und Reddit. Stripe stellt als Full-Service-Anbieter alle notwendigen Werkzeuge bereit um Kreditkartenzahlungen online zu ermöglichen und Abonnements professionell zu verwalten. Dabei werden keine Einrichtungsgebühren oder monatliche Grundgebühren erhoben, sondern pro Transaktion ein bestimmter Betrag vom Dienst eingezogen. Aktuell wird für eine Transaktion mit einer europäischen Kreditkarte ein prozentualer Betrag von 1,4% des Preises plus einem Festbetrag von 0,25 € einbehalten [Str].

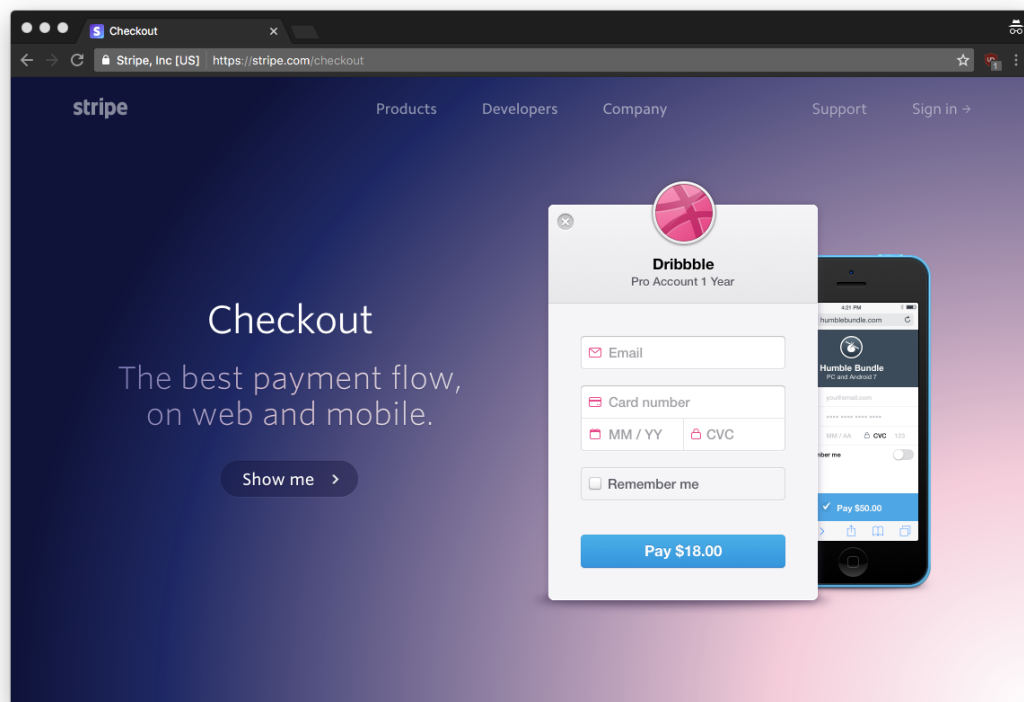


Abbildung 6.3.: Checkout von Stripe

<sup>1</sup><https://stripe.com/de>

## 6. Bezahlvorgang

Die Stripe-Integration in eine Anwendung ist durch die mächtige, aber gleichzeitig einfach zu handhabende API schnell durchgeführt. Die Dokumentation ist umfangreich und es stehen für die verschiedensten Programmiersprachen entsprechende Client-Bibliotheken zur Verfügung. Im Online-Dashboard können Pläne, Kunden, Rechnungen, Auszahlungen und Rabatt-Coupons erstellt, eingesehen und editiert werden.

Derzeit wird die stabile und produktionssichere Version von Stripe nur in den USA und in Teilen von Europa angeboten. Deutschland zählt momentan bedauerlicherweise nicht dazu. Hierzulande befindet sich Stripe in der Beta-Phase. Es ist somit noch nicht möglich diesen Dienst innerhalb von Deutschland produktiv einzusetzen.

### Braintree

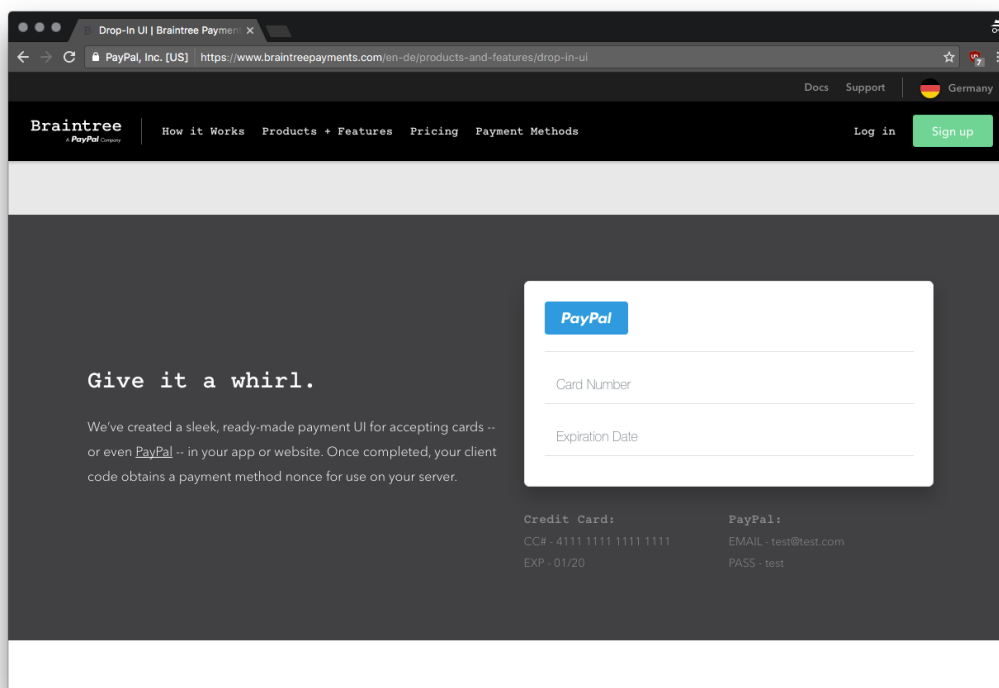


Abbildung 6.4.: Checkout von Braintree

Eine Alternative zu Stripe ist Braintree<sup>2</sup>. Braintree ist seit dem Jahr 2013 ein Teil von den Unternehmen PayPal. Dementsprechend werden neben Zahlungen

<sup>2</sup><https://www.braintreepayments.com/en-de>



über die Kreditkarte auch Zahlungen über PayPal akzeptiert. Zusätzlich wird in naher Zukunft die in vielen Teilen der Welt bereits angebotene Bezahlung über Apple Pay, Android Pay, Venmo oder Bitcoin auch in Deutschland möglich sein. Namhafte Unternehmen die Braintree nutzen, sind unter anderem Uber, Airbnb und GitHub.

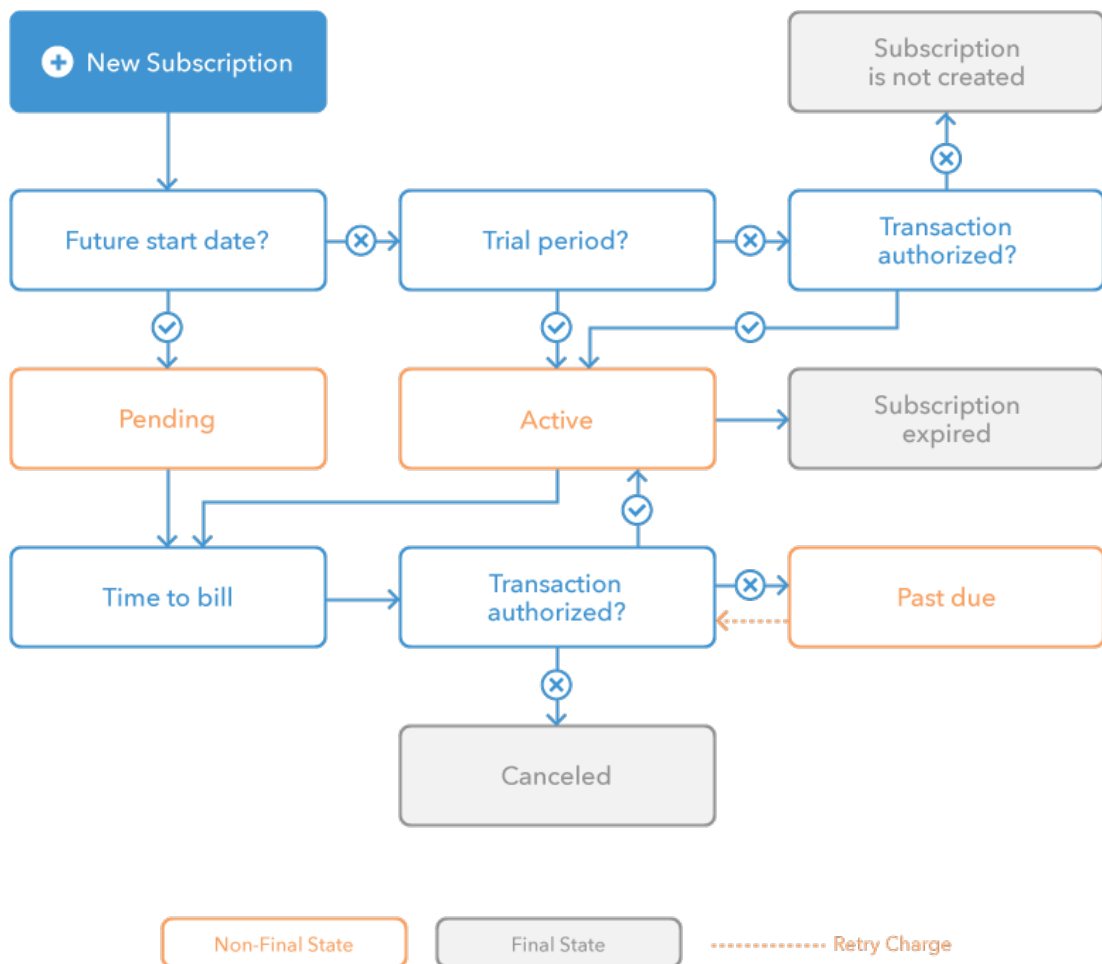


Abbildung 6.5.: Wiederkehrende Zahlungsvorgänge in Braintree [Brab]

Braintree zeichnet sich durch den Einsatz von leistungsstarken Anti-Fraud-Tools und der Unterstützung von 3D Secure aus. Zudem bietet Braintree bemerkenswert flexible wiederkehrende Zahlungsabläufe. Die Abbildung 6.5 stellt den Ablauf eines wiederkehrenden Zahlungsvorgang dar.

Da Braintree selbst alle Zahlungsdaten eines Käufers speichert und zu jedem Eintrag einen einzigartigen Token generiert ist es für den Entwickler nur notwendig diesen Token auf dem eigenen Server zu speichern. Dadurch sind die

Kundendaten besser gesichert und dennoch können wiederkehrende Zahlungen reibungslos durchgeführt werden.

Transaktionsgebühren werden bei Braintree erst erhoben wenn mehr als 50.000 € Umsatz generiert wurde. Anschließend wird pro Transaktion 1,9% + 0,30 € einbehalten. Eine Einrichtungs- oder monatliche Grundgebühr, sowie einen Mindestumsatz gibt es nicht.

Die API von Braintree ist wie die von Stripe einfach zu benutzen. Es werden alle gängigen Sprachen unterstützt und angeboten.

### **Zusammenfassung**

Der Zahlungsprozess ist eines der grundlegenden Teile einer Anwendung und sollte deshalb wohl bedacht integriert werden. Es gibt viele Anbieter die in etwa die gleichen Funktionen bieten, jedoch unterscheiden sie sich oftmals erheblich in ihrer Art und Einfachheit der Integration in ein genutztes System. Vor allem die Integration in ein System ist nicht zu vernachlässigen, da dort erheblich Zeit und Aufwand gespart werden kann.

## **Teil III.**

# **Monetarisierung am Beispiel von THMcards**



# 7. Die Open-Source-Anwendung THMcards

Dieses Kapitel ist in Zusammenarbeit mit Matthias Zimny entstanden. Nachfolgend wird auf die Re-Implementierung von THMcards eingegangen und die technischen Aspekte von THMcards erläutert. In der Re-Implementierung findet die Evaluierung eines geeigneten Web-Frameworks statt. Zudem werden die Anpassungen in der Codestruktur dargestellt. Die in THMcards verwendeten Technologien werden im Kapitel 7.3 *Technische Aspekte* beschrieben.

## 7.1. Zieldefinition

Aus den folgenden Ideen und Anforderungen von Prof. Dr. Klaus Quibeldey-Cirkel [QC13] ist die E-Learning Plattform THMcards entstanden:

„Gegen das Aufschieben des Lernens bis zum Klausurtermin und das Vergessen kurz danach soll Leitners Lernsystem in digitalisierter Form in die Vorlesungen gebracht werden. Leitners Lernkartei ist so organisiert, dass sie dem Anspruch der Lernmethode ‚Distributed Practice‘ bestens genügt und effektiv und nachhaltig dem Vergessen des Gelernten entgegenwirkt.

Der Fünf-Fächer-Algorithmus nach Leitner sieht vor, gelernte Inhalte in Form von Lernkarten (Vorderseite: Lernfrage, Rückseite: richtige Antwort) in bestimmten Zeitintervallen zu wiederholen: Neue Lernkarten sind im vordersten Fach des Karteikastens und wandern bei richtiger Beantwortung ins zweite Fach. Die Lernkarten dort werden alle zwei bis drei Tage zum wiederholten Lernen vorgelegt. Bei richtiger Beantwortung wandert die Lernkarte ins dritte Fach, dessen Karten alle zehn Tage wiederholt werden. Lernfragen im vierten Fach werden monatlich wiederholt und im fünften Fach nach drei Monaten. Da-

nach bleiben sie nachhaltig im Langzeitgedächtnis haften. Bei falscher Beantwortung wandert jede Lernkarte, egal aus welchem Fach, zurück ins erste. Der Fünf-Fächer-Algorithmus kompensiert quasi den Verlauf der Vergessenskurve.

Eine digitale Neuauflage der Lernkartei bietet viele weitere Vorteile: multimediale Lernkarten mit eingebetteten Video- und Audiodateien, komplexe Formeln mit TeX-Formatierung, Lernstand-Berechnung und mehr. [...]

Im Wintersemester 2012/13 habe ich die Online-Lernkarten-Software CoboCards in zwei Kursen mit jeweils ca. 75 Studierenden parallel zum Vorlesungsbetrieb eingesetzt und von den Teilnehmenden evaluieren lassen. Die Quintessenz: Das Online-Lernen mit Lernkarten wurde allgemein begrüßt und als effektiv eingeschätzt; [...]. Allerdings wurden die Kollaborationsfunktionen von CoboCards, das Erstellen und Abfragen von Lernkarten online im Team, und auch die sozialen Funktionen, wie Team-Pinnwand und Diskussionsforum, kaum genutzt. Außerdem fehlte die intrinsische Motivation, Lernkarten von sich aus zu nutzen. Nur eine extrinsische Motivation in Form von Bonuspunkten auf die Klausurnote für besonders gut formulierte Antworten auf vom Dozenten gestellte Lernfragen erhöhte die Nutzung von CoboCards. Einstimmig gewünscht waren offizielle, das heißt vom Lehrenden erstellte Lernkarten. Hier setzt das Neuartige meiner geplanten Lehrinnovation an:

Es soll eine Online-Plattform THMcards entwickelt werden, auf der Lehrende Lernkarten zu ihren Modulen erstellen und aktualisieren können. Studierende können auf die Lernkarten ihrer Kurse per Smartphone, Tablet und Laptop zugreifen. Der individuelle Lernstand wird automatisch ermittelt und angezeigt. Die Lernkarten-Funktionalität der Plattform wird in das mobile TED-System ARSnova integriert; [...]. Vorbereitungs-, Vorlesungs- und Konzeptfragen, die mit ARSnova in verschiedenen Frageformaten (Single/Multiple Choice, Likert- und Noten-Skalen, Freitext) erstellt wurden, können als Lernkarten nach THMcards exportiert werden.

Meine geplante Lehrinnovation will die mediengestützten Lehr- und Lernmethoden der Hörsaal-Didaktik um das Leitnersche Lernsystem

erweitern: Neben ‚Peer Instruction‘ und ‚Just-in-Time Teaching‘, die mithilfe des TED-Systems ARSnova in den letzten beiden Semestern in mehreren Vorlesungen erfolgreich praktiziert wurden, soll in den kommenden Semestern die Lernkartei-Plattform THMcards eingesetzt werden. Ich verspreche mir nicht nur eine messbare Verbesserung der Behaltensquote über einen langen Zeitraum. Neben dem Lernen im Hörsaal soll auch ein verteiltes Lernen der Vorlesungsinhalte per Lernkarten in informellen Lernsituationen überall und jederzeit ermöglicht werden. ‚Mobile Learning‘ postuliert das Lernen in kleinen Portionen (‚Lernhäppchen‘); Lernkarten erfüllen genau diese Anforderung.

Es bleibt das Risiko der Motivation zur Nutzung der Lernkarten. Hier kommt die Gamifizierung des Lehrens und Lernens ins Spiel. Laut dem ‚NMC Horizon Report: 2013 Higher Education Edition‘ zählt Gamifizierung zu den Schlüsseltechnologien im Bildungsbereich und wird in den kommenden zwei bis drei Jahren bedeutende Auswirkungen auf die Hochschullehre haben. [...]

Zusammengefasst soll mit THMcards eine Lernplattform geschaffen werden, die ihrem Namen gerecht wird. Im Gegensatz zu Moodle, ILIAS, OLAT oder Stud.IP, die in aller Regel nicht als Plattform zur Unterstützung von Lehr- und Lernprozessen auf dem Campus und unterwegs eingesetzt werden, sondern nur für die Organisation der Kurse und Verteilung der Vorlesungsmaterialien, soll das Lehren und Lernen tatsächlich auf der Plattform THMcards und mit ARSnova auch im Hörsaal und unterwegs unterstützt werden.“

Aufbauend auf diesem Zitat ist die erste Version von THMcards durch die Studenten Jan Christopher Kammer und Daniel Knapp entstanden. Dazu können auch die Masterarbeiten *Implementierung von Spaced Repetition Algorithmen zur effektiven Abfrage von Lernkarten innerhalb der eLearning Plattform THMcards* [Kam14] und *Implementierung von Spielmechaniken zur Steigerung der Lernmotivation von Studierenden am Beispiel der Lernkarten Plattform THMcards* [Kna13] eingesehen werden.

Im Rahmen des Entwicklungsprojekts galt es die erste Version von THMcards in ihrem Aufbau zu optimieren. Hierzu sollte ein Redesign und eine Re-

Implementierung der bestehenden Use Cases mit einem für THMcards geeigneterem Web-Framework realisiert werden.

### 7.2. Re-Implementierung von THMcards

Die Gründe für die Re-Implementierung von THMcards sind hauptsächlich auf eine mangelhafte Softwarequalität zurückzuführen. Sowohl auf die Verständlichkeit als auch auf die Wartbarkeit der Software wurde hierbei keine Rücksicht genommen. Durch die Nichteinhaltung vorgegebener Programmierparadigmen bei der Erstellung von Programmcode und der daraus resultierenden unübersichtlichen Codestruktur, war die Verständlichkeit der Anwendung nicht mehr gegeben. Weiterhin wurde mit Backbone ein JavaScript-Framework gewählt, das an Bekanntheit und somit an Support verloren hat. Eine dauerhafte Wartung von THMcards wäre aufgrund des zunehmenden Projektumfangs durch die anstehenden Erweiterungen nicht ohne größeren Energieaufwand möglich. Die vorhandene Grundlage war somit für eine erfolgreiche Weiterentwicklung von THMcards nicht geeignet.

In den folgenden Abschnitten werden die getroffenen Entscheidungen zur Re-Implementierung von THMcards erläutert. Diesbezüglich wird mit Hilfe einer Evaluierung ein geeignetes Web-Framework für das Projekt THMcards ermittelt. Darüber hinaus wird auf die vorgenommenen Änderungen der Codestruktur eingegangen.

#### 7.2.1. Evaluierung eines geeigneten Web-Frameworks

Um ein geeignetes Web-Framework für das Projekt THMcards zu finden, wurde im Rahmen des Entwicklungsprojekts die Evaluierung der populärsten MV\* Frameworks für Web-Anwendungen durchgeführt. Zu diesen gehören Sencha Touch bzw. Ext JS, Angular, Backbone, Ember und Meteor.

Bevor der eigentliche Vergleich durchgeführt wurde, konnten zwei dieser Frameworks bereits ausgeschlossen werden. Da die Web-Applikation THMcards primär als Desktop-Anwendung genutzt werden soll, ist das Framework Sencha Touch, das speziell für Mobile-Web entwickelt wurde, für das Projekt nicht geeignet. Darüber hinaus war es in diesem Vergleich nicht notwendig auf das proprietäre Ext JS einzugehen, da bei der Entwicklung von THMcards ausschließlich Open-Source Produkte zum Einsatz kommen sollten.



Bei der Evaluierung spielten folgende Kriterien eine wichtige Rolle: Community, Framework Size und Templating.

### Community

Die Community ist eine der wichtigsten Faktoren bei der Auswahl eines geeigneten Frameworks. Bei einer großen Community stehen dem Benutzer mehr Tutorials, Third-Party Module sowie generelle Hilfestellungen zur Verfügung. In der folgenden Tabelle wird die Größe der Community der einzelnen Frameworks gegenüber gestellt:

Metrik	Angular	Backbone	Ember	Meteor
GitHub Stars	52k	25,5k	16,8k	35,3k
GitHub Contributors	1,5k	290	614	310
GitHub Commits	8k	3,3k	13,3k	17,5k
Third-Party Module	2k	275	2,8k	11,4k
Stack Overflow Fragen	196,7k	20k	19,7k	23,3k
YouTube Ergebnisse	181k	29,8k	28,1k	33,9k

Tabelle 7.1.: Größe der Framework Community (Stand 12.09.2016)

In Tabelle 7.1 ist zu erkennen das Angular die größte Community hat. Während die Frameworks Ember und Backbone an Popularität verlieren, konnte Meteor und Angular ein starkes Wachstum verzeichnen. Außerdem ist der Abbildung 7.1 zu entnehmen, dass hinter Meteor eine große Community steht, die stetig damit bemüht ist das Framework weiter zu entwickeln.

### Framework Size

Die Ladezeiten bzw. der Seitenaufbau sind Fundamental für den Erfolg einer Webseite. Für den im Allgemeinen ungeduldigen Benutzer ist es wichtig, im Web schnellstmöglich an Informationen zu gelangen. Aus diesem Grund sind lange Ladezeiten auf Webseiten zu verhindern. Diesbezüglich sind bei der Wahl des richtigen Frameworks zwei ausschlaggebende Faktoren zu beachten: Die Frameworkgröße und die Zeit, die benötigt wird, um das Framework zu laden.

Im Allgemeinen werden JavaScript-Dokumente minifiziert und komprimiert dem Benutzer bereitgestellt. Aus diesem Grund wird in dieser Gegenüberstellung die Dateigröße der minifizierten und komprimierten Versionen verglichen. Da einige Frameworks nicht eigenständig nutzbar sind, sollte die Dateigröße der benötigten Dependencies ebenfalls bei diesem Vergleich berücksichtigt werden.

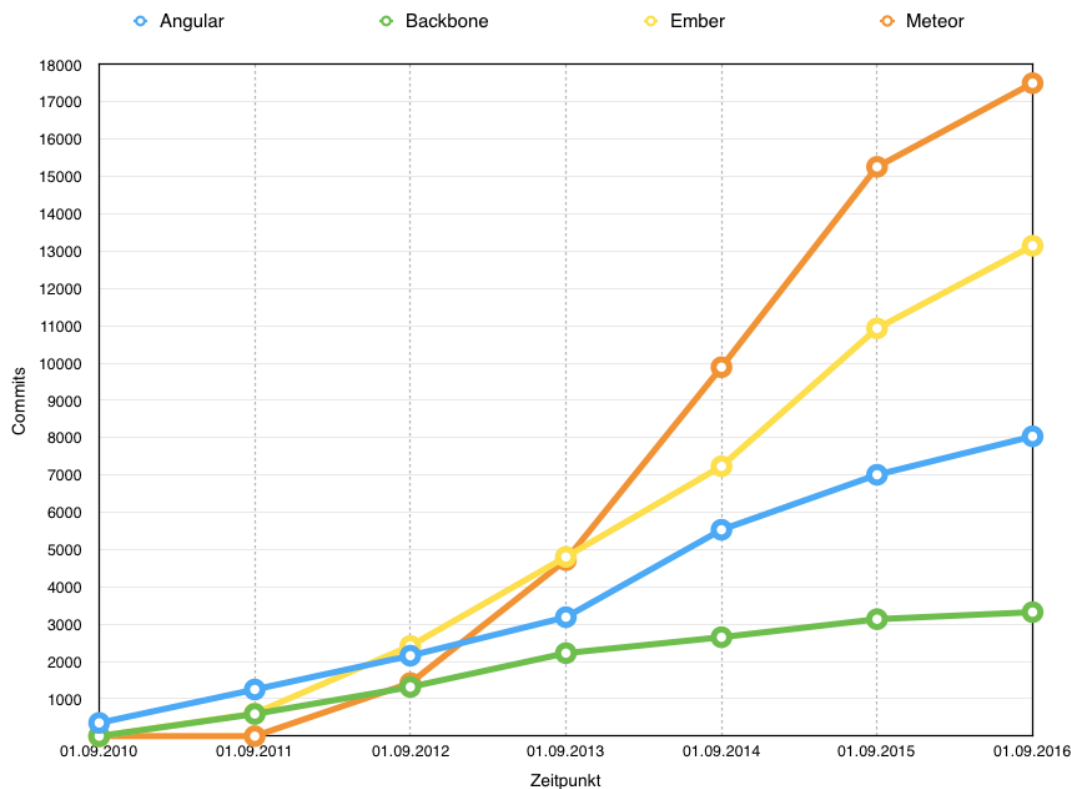


Abbildung 7.1.: Die Commits eines Web-Frameworks

Der Tabelle 7.2 ist zu entnehmen, dass es sich bei Backbone um das Framework mit der kleinsten Dateigröße handelt. Berücksichtigt man jedoch alle benötigten Dependencies, wird der Abstand zwischen Backbone und Angular deutlich kleiner, da Angular eigenständig nutzbar ist.

### Templating

Angular und Ember sind bereits mit ihrer eigenen Template Engine ausgestattet. Backbone und Meteor hingegen ermöglichen die Auswahl unterschiedlicher Template Engines. Die Template Engine von Angular setzt sich aus einfachem HTML und Binding Expressions zusammen, während bei Ember Handlebars zum Einsatz kommt. In Backbone wird meistens Underscore genutzt, da dieses zu den benötigten Dependencies von Backbone gehört. Durch den einfachen Aufbau von Underscore hat der Nutzer die Möglichkeit JavaScript in den HTML Code einzubinden. Meteor hingegen nutzt üblicherweise Spacebars, eine an Handlebars angelehnte Template Engine, ermöglicht jedoch auch die Verwendung von Jade.

Framework	Version	Größe	Dependencies	Gesamtgröße
Angular	1.5.8	55kb	-	55kb
Backbone	1.3.3	7,4kb	jQuery, Underscore	42,1kb
Ember	2.8.0	108kb	jQuery, Handlebars	156kb
Meteor	1.4	-	Mongo, Blaze, jQuery	135kb

Tabelle 7.2.: Größe der einzelnen Frameworks (Stand 12.09.2016)

## Angular

*Vorteile:* Angular bietet zahlreiche innovative Konzepte für Webentwickler. Unter anderem lässt sich durch *two-way data binding* ein großer Teil des Boilerplate-Code einsparen. Der erforderliche JavaScript-Code muss nicht selbst implementiert werden, sondern lässt sich auf einfachste Weise im HTML-Template durch spezifische Attribute und Expressions einbinden.

Von den Frameworks bietet Angular die größte Community und am meisten Online-Content. Darüber hinaus wird es von Google unterstützt und gefördert. Bei der Weiterentwicklung von Angular spielt die Community eine wichtige Rolle. Design-Entscheidungen werden von dem Angular-Team gemeinsam mit der Community getroffen, indem Anmerkungen direkt im veröffentlichten Design-Dokument geäußert werden können.

Des Weiteren ermöglicht Angular die eigene Applikation in Kategorien einzuteilen. Dazu werden verschiedene Typen zur Verfügung gestellt: Controllers, Directives, Filters, Services und Views (Templates).

Angular setzt viel Wert auf die Möglichkeit die Anwendung zu testen. Dies wird durch die Bereitstellung von Mock-Objekten, Unit Isolation und die Vorteile des Dependency Injection Mechanismus ermöglicht.

*Nachteile:* Für seine Komplexität der Directives API ist Angular oft kritisiert worden. Es nimmt einige Zeit in Anspruch die zur Verfügung gestellten Directives zu beherrschen.

Angular Expressions werden oftmals zu exzessiv im View-Template einer Anwendung verwendet. Die daraus resultierende komplexe Logik führt dazu, dass sich die Anwendung schwer bis gar nicht testen lässt.

### Backbone

*Vorteile:* Backbone ist leichtgewichtig, schnell und beansprucht den Speicher nur gering. Die Lernkurve ist sehr linear, da es nur wenige Konzepte (Models/Collections, Views, Routes) zu verstehen gibt. Die Dokumentation des Codes ist einfach gehalten und ausführlich beschrieben.

Durch den einfachen Aufbau können eigene Frameworks mit Backbone kombiniert werden. Außerdem besteht die Möglichkeit bereits vorhandene Frameworks, wie z.B. Marionette oder Backbone UI zu verwenden.

*Nachteile:* Das größte Problem von Backbone ist, dass es keine Struktur vorschreibt. Jedem Entwickler ist freigestellt, wie die Struktur der Applikation aufgebaut werden soll. Dadurch kann es für externe Entwickler viel Zeit in Anspruch nehmen den Aufbau einer bestehenden Anwendung zu verstehen.

Der einfache Aufbau von Backbone bringt auch negative Aspekte mit sich. Um komplexe Anwendungen mit Backbone zu erstellen ist man gezwungen weitere Plugins einzubinden. Viele Funktionalitäten werden vom Backbone-Framework allein nicht abgedeckt.

Zudem unterstützt das Backbone-Framework kein *two-way data binding*, so dass Entwickler gezwungen sind viele Standardformulierungen im Code selbst zu implementieren, um bei Änderungen im Model den View bzw. bei Änderungen im View das Model zu aktualisieren.

Ein weiteres Problem von Backbone ist, dass es das Document Object Model (DOM) direkt manipuliert und somit die Durchführung der Unit-Tests erschwert.

### Ember

*Vorteile:* Genau wie in Angular kann bei Ember ein großer Teil des Boilerplate-Codes eingespart werden, da viele Funktionalitäten durch die Konfiguration von Ember abgedeckt werden.

Im Gegensatz zu den anderen beiden Frameworks, die mit einem minimalen Data-Layer ausgestattet sind, bietet Ember ein voll entwickeltes Daten-Modul welches hervorragend mit RESTful JSON APIs harmoniert.

Darüber hinaus spielt bei Ember die Performance eine wichtige Rolle. Konzepte wie *The Run Loop* helfen dabei schnelle Ladezeiten der Applikation zu ermöglichen.

*Nachteile:* Aufgrund vieler grundlegender Änderungen in der Ember-API gibt es viele veraltete Informationen und Beispiele die nicht mehr mit aktuellen Versionen des Frameworks kompatibel sind. Das führt bei Anfängern zu einem erschwerten Einstieg.

Die aktuell noch oft verwendete Handlebars Template Engine, bei der im DOM viele `<script>`-Tags eingebunden werden, führt schnell zu einem unübersichtlichen DOM-Tree. Darüber hinaus kann es vorkommen, dass das CSS-Styling nicht mehr richtig dargestellt wird oder andere Frameworks nicht mehr funktionieren.

### **Meteor**

*Vorteile:* Der wesentliche Vorteil von Meteor ist, dass man nur eine Applikation für Client und Server entwickeln muss. Damit eignet sich Meteor vor allem für Einsteiger, da man sich keine Gedanken über die Kommunikation zwischen Client und Server machen muss.

Ein weiteres nützliches Feature ist die sofortige Aktualisierung der Anwendung im Browser, nachdem Änderungen im Quellcode oder der Datenbank vorgenommen wurden.

Mit der Unterstützung von *Hard Code Push* garantiert Meteor Continuous Deployment. Dadurch ist es möglich JavaScript im Quellcode zu ändern ohne den Benutzer bei seinen Aktionen auf der Webseite zu unterbrechen.

Ein weiterer Vorteil ist, dass neben der vorgegebenen Blaze Bibliothek auch Angular oder React als User Interface Rendering Library genutzt werden können. Dies ermöglicht je nach Art der Anwendung die geeignetste Bibliothek für Projekt auszuwählen.

*Nachteile:* Meteor eignet sich nicht als Framework, wenn hohe Ansprüche an die Performance gestellt werden. So sollten Spiele-Applikationen mit hohen Grafikanforderungen nicht mit Meteor, sondern besser als native App programmiert werden.

Darüber hinaus ist ein Großteil der für Meteor angebotenen Plugins veraltet, da sie nicht von Meteor selbst, sondern von Community-Mitgliedern erstellt worden sind. Oftmals handelt es sich bei dem Angebot um npm-Module, die mittels eines Wrappers zu einem Meteor-Plugin umfunktioniert wurden. Während die npm-Module kontinuierlich weiterentwickelt werden, ist eine Aktualisierung der Meteor-Plugins durch die Community meist nicht gegeben. Jedoch wurde die-

ses Problem durch die direkte Integration von npm in die Meteor-Anwendung relativiert.

### Fazit

Aufgrund des minimalistischen Aufbaus von Backbone und dem daraus resultierenden Einsatz von weiteren Frameworks, ist die Struktur jeder Backbone-Anwendung verschieden. Das gestaltet die fortlaufende Entwicklung der Anwendung schwieriger, da eine große Einarbeitungszeit für neue Entwickler in ein Projekt notwendig ist. Da es sich bei THMcards um ein fortlaufendes Projekt, bei dem zukünftig weitere Studenten beteiligt sind, handelt, ist das Backbone-Framework für dieses Projekt nicht geeignet.

Ember und Angular führen beide für das Projekt ausreichend Vorteile mit sich. Was jedoch eher für die Entscheidung zugunsten von Angular spricht, ist die ausführliche Dokumentation und die mit Abstand größte Community, sowie die Unterstützung von Google.

Aufgrund der hohen Komplexität des Framework Angular und der damit resultierenden langsamen Lernkurve, fällt die Entscheidung jedoch zugunsten des in der Evaluierung verbliebenen Frameworks Meteor. Meteor bietet ein gutes Gesamtpaket, eine einfache Handhabung und lässt sich ohne großen Aufwand mit weiteren Plugins erweitern. Es ist somit für Projekte wie THMcards bestens geeignet.

### 7.2.2. Codestruktur

Bei der ersten Version von THMcards der Masterstudenten Jan Christopher Kammer und Daniel Knapp handelte es sich um eine Anwendung die mit den Frameworks Backbone, Marionette und Node.js, sowie der Datenbank CouchDB erstellt wurde. Für das ausschließlich in JavaScript geschriebene THMcards wurde Node.js im Zusammenspiel mit der CouchDB für die serverseitigen Aufgaben verwendet. Im Front-End kamen die Frameworks Backbone und Marionette zum Einsatz. Da Backbone standardmäßig für häufig auftretende Szenarien keine Funktionalitäten bereithält und Boilerplate-Code vermieden werden sollte, ist der Einsatz von Marionette notwendig gewesen.

Im Laufe der Projektphase gab es immer häufiger auftretende Verständnisprobleme aufgrund essentieller Designfehler in der Codebasis. Der größte Teil der

Anwendungslogik wurde innerhalb von zwei Dateien realisiert, wodurch die Einarbeitung erschwert wurde und die Weiterentwicklung von THMcards unmöglich machte. Ebenfalls zu kritisieren waren die vielen Redundanzen in den ohnehin schon unübersichtlichen Dateien. Zudem ist nicht auf die Trennung von Template und Styles geachtet worden. Viele der CSS-Deklarationen sind direkt innerhalb der HTML-Dokumente vorgenommen worden. Des weiteren wurden große Teile der responsive Funktionen von Bootstrap fehlerhaft eingesetzt oder diese sogar überschrieben.

Um in der Re-Implementierung von THMcards solche Fehler zukünftig zu vermeiden, wird unter anderem die Dateistruktur an die vorgegebenen Richtlinien von Meteor angepasst. Das Beachten der Meteor-Richtlinien ermöglicht nicht nur eine einfache Einarbeitung weiterer Entwickler, sondern ist auch für die Weiterentwicklung der Anwendung hilfreich. Ein weiterer Vorteil dieser Modularisierung ist eine verbesserte Performance der Anwendung, da nur benötigte Module anstatt der gesamten Anwendung geladen werden. Ein Überblick über diese Dateistruktur von THMcards ist in Abbildung 7.2 zu sehen.

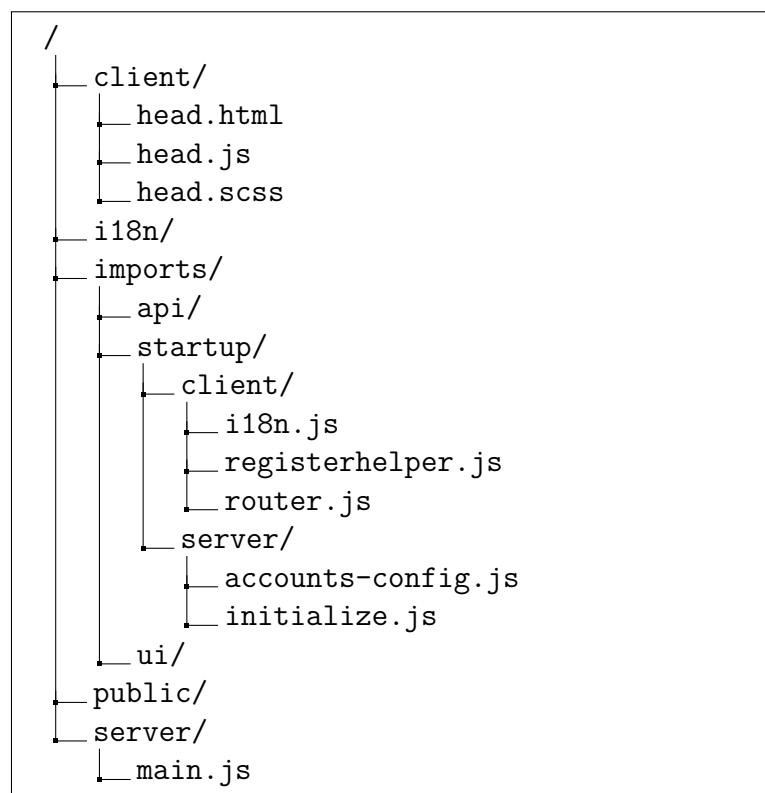


Abbildung 7.2.: Dateistruktur von THMcards

Module die beim Aufruf der Anwendung nicht erforderlich sind, werden im Verzeichnis `imports` abgelegt. Dateien aus `imports` werden nur geladen, sobald diese in einer anderen Datei referenziert wurden. Alle weiteren Dateien außerhalb dieses Verzeichnisses werden nach der von Meteor vorgegeben Reihenfolge aufgerufen. Die Eingangspunkte der Anwendung werden durch `client/head.js` für die Client-Funktionen und `server/main.js` für die Server-Funktionen definiert. Diese Einstiegsdateien sorgen dafür, dass beim Start der Anwendung Startup-Module aus `imports` aufgerufen werden. Da sich alle wichtigen Dateien in `imports` befinden, bietet es sich außerdem an die Anwendungslogik und die grafische Benutzeroberfläche voneinander zu trennen. Hierfür werden die Verzeichnisse `imports/api` und `imports/ui` verwendet.

### 7.3. Technische Aspekte von THMcards

THMcards wurde mithilfe des Open-Source Web-Frameworks Meteor als Single-Page Web Application, oder auch kurz SPA, umgesetzt. Die Besonderheit einer SPA liegt darin, dass die gesamte Anwendung aus einem einzigen HTML-Dokument besteht, dessen benötigte Inhalte dynamisch nachgeladen werden [Bru15]. Dies ermöglicht gerade bei komplexen Anwendungen eine verbesserte User Experience, da im Gegensatz zu klassischen Webanwendungen, die aus mehreren untereinander verlinkten HTML-Dokumenten bestehen, auf das erneute Laden von Unterseiten verzichtet wird. Daraus resultiert bei SPA's eine reaktionsschnelle Benutzeroberfläche, sowie eine erhebliche Kostenersparnis, da nur minimale Datenmengen über das Web transferiert werden. [2BI]

Bei Single-Page Web Applications wird eine klare Schnittstelle zwischen Front-End und Back-End definiert. Alle relevanten Daten einer Webanwendung werden durch klassische Programmiersprachen und Datenbanken verarbeitet und gespeichert. Das Front-End beruht auf HTML, CSS, sowie sehr stark auf JavaScript und dem damit verbundenem JavaScript-Framework. [Sti14] Durch JavaScript stellt das Front-End Anfragen an das Back-End und erhält dadurch die benötigten Daten im JSON-Format zurück. Diese werden an die entsprechenden Stellen gesetzt, sodass ein weiterer Seitenaufruf ausbleibt. [2BI]

Durch den Einsatz des Web-Frameworks Meteor ist THMcards einheitlichen in JavaScript für Client und Server umgesetzt. Hierbei sind alle Funktionalitäten von der Datenbank über den Server bis hin zum Client von Meteor abgedeckt.



Für die Datenbankanbindung wird die in Meteor integrierte MongoDB verwendet, während auf dem Server Node.js zum Einsatz kommt.

### 7.3.1. Meteor

Meteor ist ein Open-Source Web-Framework zur Entwicklung moderner Mobile- und Webanwendungen [Metd]. Diese Anwendungen werden mit Meteor vollständig in JavaScript entwickelt. Folglich wird die Anbindung zur Datenbank, der Server und der Client mit einer einzigen Programmiersprache abgedeckt. Meteor bündelt verschiedene Frameworks zu einer Plattform und integriert diese nahtlos, um dem Benutzer eine einfache Bedienung zu ermöglichen. [Tut15]

Was Meteor so besonders macht und von anderen Web-Frameworks unterscheidet, beschreiben die Entwickler der Plattform mit folgenden Grundsätzen:

- **Data on the Wire:** Meteor sendet kein HTML über das Netzwerk. Es werden lediglich Daten an den Client gesendet, die auf diesem wiederum verarbeitet und dargestellt werden. [NK13]
- **One Language:** Für den Code der Client- und Server-Anwendung wird als einzige Programmiersprache JavaScript verwendet [NK13]. Das ermöglicht nicht nur die schnelle Entwicklung von Applikationen, sondern erfordert auch weniger Umdenken, wenn in verschiedenen Bereichen der Anwendung gearbeitet werden muss [Tur14]. Da es sich bei JavaScript um eine der populärsten Programmiersprachen handelt, gibt es viele Anwendungsfälle und viele Entwickler sind bereits mit der Sprache vertraut [Owe15].
- **Database Everywhere:** Über eine gemeinsame API können sowohl der Client als auch der Server auf die Datenbank zugreifen. [NK13]
- **Latency Compensation:** Durch Vorauswahlen und Modellsimulationen auf dem Client kann eine latenzfreie Datenbankverbindung simuliert werden. [NK13]
- **Full Stack Reactivity:** Meteor erlaubt reaktive Programmierung: Anwendungen werden also in Echtzeit entwickelt. Alle möglichen Schnittstellen einer Anwendung, von der Datenbank bis zu den Templates werden automatisch aktualisiert. Seiten müssen nicht neu geladen werden, um Updates

zu sehen und Änderungen an Dokumenten können ohne Verzögerung gespeichert werden. Das ermöglicht die einfache Zusammenarbeit an einer Anwendung in Echtzeit. [Owe15]

- **Embrace the Ecosystem:** Meteor ist Open-Source und integriert andere Open-Source Frameworks, statt diese zu ersetzen oder nachzubilden. [NK13]
- **Simplicity equals Productivity:** Einfachheit steht bei Meteor an erster Stelle. Mithilfe eines großen Angebots an Packages der Node.js und JavaScript Community wird in Meteor weniger Code benötigt als in anderen gängigen Frameworks wie Angular oder Backbone, um die gleichen Aufgaben zu realisieren [Tut15]. Durch JavaScript in Front-End und Back-End erlaubt Meteor schneller zu programmieren und mit wenig Aufwand ein fertiges Produkt zu entwickeln [Owe15].

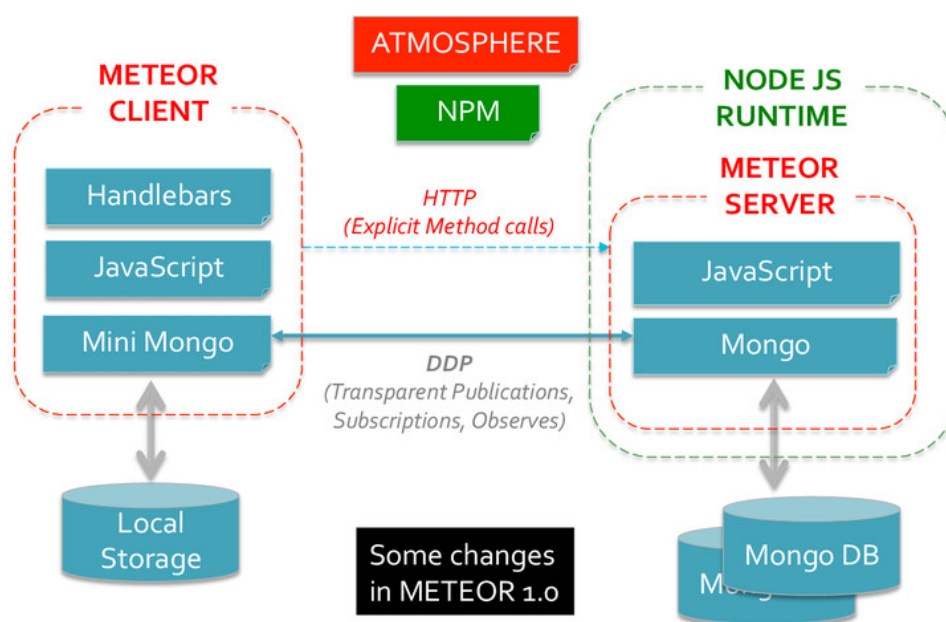


Abbildung 7.3.: Aufbau von Client und Server in Meteor 1.0 [Mon15]

Um Anwendungsdaten über einen langen Zeitraum bereit zu halten verwendet Meteor die Datenbank MongoDB [NK13]. Durch die Implementierung einer an der MongoDB API angelehnten Minimongo Datenbank werden auf dem Client vergleichbare Befehle wie die auf dem Server bereitgestellt. Die Kommunikation und Synchronisation der Daten zwischen Client und Server wird hierbei gänzlich

von Meteor übernommen, sollten Ereignisse über zu lesende oder zu schreibende Änderungen eintreten. Das Zusammenspiel zwischen Client und Server wird nochmal in Abbildung 7.3 veranschaulicht. [Mon15]

Die Daten werden wie in MongoDB üblich in Collections abgebildet, deren Definition einfach gehalten wird. Collections stehen dem Client als auch dem Server zur Verfügung und werden von Meteor automatisch synchronisiert. Jede Collections erhält einen eindeutigen Namen als Argument und wird wie folgt definiert: [NK13]

```
1 export const Cardsets = new Mongo.Collection("cardsets");
```

Listing 7.1: Definition einer Collection

Mit dem Meteor Kommandozeilen-Tool wird eine neue Meteor-Applikation standardmäßig mit einem Autopublish-Verhalten versehen. Das erlaubt jeden Client (auch anonyme, nicht authentifizierte und nicht autorisierte) auf die Collections zuzugreifen und jede beliebige Zugriffsoperation wie das Speichern, Modifizieren, Suchen oder Aggregieren von Daten, auszuführen. Dieses Standardverhalten lässt sich jedoch abschalten, sodass sensible oder rechenintensive Zugriffsoperationen auf dem Server durchgeführt und die aktiven Clients über das Ergebnis benachrichtigt werden. Durch dieses sogenannte *Publish/Subscribe*-Verfahren können mit **publish** die Ergebnisse einer Datenbankabfrage auf dem Server veröffentlicht und mithilfe von **subscribe** die Clients abonniert werden. Auf diese Weise lässt sich verhindern, dass Änderungen einer Collection Einfluss auf alle Clients hat und alle Felder öffentlich sind. [Lie13][NK13]

```
1 Meteor.publish("cardsets", function() {  
2   return Cardsets.find({});  
3 });
```

Listing 7.2: Veröffentlichung der Daten im Server

In der Server-Datei werden die Collections mit **Meteor.publish**, wie in Listing 7.2 dargestellt, veröffentlicht. Der Client abonniert diese mit dem Bezeichner der Collection:

```
1 Meteor.subscribe("cardsets");
```

Listing 7.3: Abonnieren der Daten im Client

Eine weitere Möglichkeit die Logik vom Client auf den Server zu verlagern sind die in Meteor sogenannten *Method Calls*. Mit ihnen lassen sich Funktionen auf

dem Server definieren und auf dem Client aufrufen. Ein Beispiel einer solchen Funktion ist das Entfernen eines Kartensatzes: [NK13]

```
1 Meteor.methods({
2   deleteCardset: function(id) {
3     ...
4     Cardsets.remove(id);
5     ...
6   }
7 });
```

Listing 7.4: Definition einer Funktion im Server

Über `Meteor.call()` können die Server-Funktionen vom Client ohne weitere Konfigurationen aufgerufen werden. Eine solche Definition veranschaulicht Listing 7.5.

```
1 Meteor.call("deleteCardset", id);
```

Listing 7.5: Method Call im Client

Die bereitgestellten Modelle und Daten werden mit Templates in Verbindung mit dem MVVM-Entwurfsmuster (Model, View, ViewModel) in die Views implementiert. Hier kommt in Meteor die Template-Engine Handlebars zum Einsatz, jedoch besteht auch die Möglichkeit andere Template-Engines einzubinden. Handlebar-Templates werden mit den Tags `<template name="...">...</template>` innerhalb eines HTML-Dokuments definiert. Diese Templates lassen sich mit der Anweisung `{{> name}}` mit ihren Namen an verschiedenen Stellen einbinden. [NK13]

Zur Laufzeit werden Template-Variablen an Funktionen gebunden, die beispielsweise Daten aus den Collections abrufen oder manipulieren [NK13]. Wie das Zusammenspiel zwischen Server, Client und der Template-Engine genau funktioniert, soll das nächste Beispiel erläutern. Die durch `Meteor.subscribe` zur Verfügung gestellte Collection `Cardsets` lässt sich mit der Anweisung in 7.6 unter dem `Created`-Template mit der Funktion `cardsetList` verwenden.

```
1 Template.created.helpers({
2   cardsetList: function() {
3     return Cardsets.find({...});
4   };
5 });
```

Listing 7.6: Veröffentlichung der Daten im Client

Ein weiterer wesentlicher Bestandteil unter Meteor ist die `each`-Anweisung mit dessen Hilfe über eine Collection iteriert werden kann. In Listing 7.7 wird die Tabelle mit Inhalten gefüllt und für jeden Kartensatz eine Zeile mit Namen und der zugehörigen Beschreibung erstellt.

```
1 {{#each cardsetList}}
2   <tr>
3     <td>{{name}}</td>
4     <td>{{description}}</td>
5   </tr>
6 {{/each}}
```

Listing 7.7: Iteration von Collections in Meteor

Neben Schleifen sind auch Bedingungen sinnvoll, die Bereiche des Templates je nach Zustand anzeigen oder ausblenden. In Beispiel 7.8 wird die Tabelle nur angezeigt, wenn Einträge in der Collection vorhanden sind.

```
1 {{#if cardsetList.count}}
2   <!-- Inhalt der Tabelle -->
3 {{else}}
4   <> cardsetEmpty<>
5 {{/if}}
```

Listing 7.8: Bedingungen in Meteor-Templates

Zu einer Plattform wie Meteor gehört auch ein Build- und Deployment-System, das die Veröffentlichung einer Anwendung erlaubt. Hierzu wird von Meteor eine eigene Infrastruktur unter dem Namen *Galaxy* zur Verfügung gestellt. [NK13]

Soll auf die Nutzung der Meteor-Infrastruktur verzichtet und eine gänzlich eigene Anwendung betrieben werden, lassen sich mit dem `build`-Kommando von Meteor alle notwendigen Ressourcen, die im Betrieb einer Meteor-Anwendung erforderlich sind, verpacken und in einer eigenen Infrastruktur installieren [Metb]. Dazu wird neben dem regulären Node.js-Server die Datenbank MongoDB benötigt [NK13].

Meteor bietet zudem ein ständig wachsendes und von der Community unterstütztes Verzeichnis unter dem Namen *Atmosphere* an, dass zahlreiche nützliche Erweiterungen bereithält. Einzelne Packages von *Atmosphere* sind sogar fest im Core von Meteor integriert. Darüber hinaus gibt es zahlreiche Tipps und Tricks rund um das Meteor-Framework. Die Dokumentation von Meteor ist sehr umfangreich und man erhält in Google Docs oder Stackoverflow Antworten aus der Meteor-Entwicklergemeinschaft. [NK13]

### 7.3.2. Node.js

Node.js ist eine plattformübergreifende Laufzeitumgebung für JavaScript-Anwendungen. Sie ermöglicht Entwicklern die leichte Entwicklung von schnellen und skalierbaren JavaScript-Applikationen. Alle Node.js Anwendungen werden typischerweise auf dem Server ausgeführt. [Wikb]

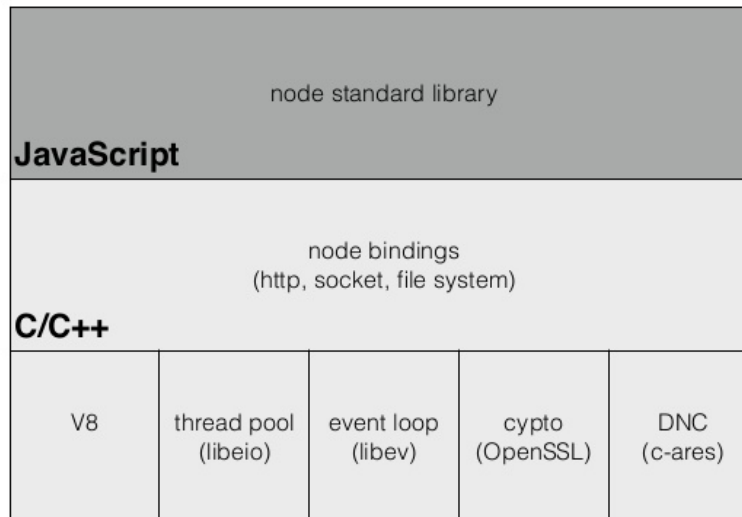


Abbildung 7.4.: Die Architektur von Node.js [Hec16, Seite 12]

Vorgestellt wurde Node.js im Jahr 2009 von Ryan Dahl. Die Node.js-Plattform besteht im Kern aus einer Ansammlung von verschiedensten Bibliotheken. Dieser Aufbau ermöglicht es die Vorteile der einzelnen Bibliotheken innerhalb einer Plattform zu nutzen und gleichzeitig eine unabhängige Weiterentwicklung zu ermöglichen, da die Bibliotheken getrennt voneinander gepflegt werden. In der Architektur von Node.js, wie in Abbildung 7.4 dargestellt, ist die Basis der Plattform in C und C++ programmiert. Um zusätzliche Module ohne große Schwierigkeiten zu erstellen, werden diese in JavaScript entwickelt. Das Herzstück von Node.js ist die V8-Engine von Google. Das primäre Einsatzgebiet der Engine ist der Chrome-Browser, dennoch kann sie auch unabhängig für die Interpretation und Ausführung von JavaScript-Code genutzt werden. [Spr13a, Seite 19-25 und 71-72]

Da Node.js durch den Single-Threaded-Ansatz nur eine Operation gleichzeitig ausführen kann, gibt es den so genannten Event Loop (siehe Abbildung 7.5). Dieser verhindert das Blockieren des Threads, indem alle zeitintensiven Operationen, wie lesende und schreibende Zugriffe, an das Betriebssystem ausgelagert werden.

Nach Abschluss der Aufgaben wird das Ergebnis mittels eines Callbacks an den Eventloop zurückübermittelt. [Spr13b]



Abbildung 7.5.: Das konzeptionelle Modell von Node.js [Nod]

Da es zwischen Joyent, dem Hauptsponsor von Node.js, und der Community zu immer größeren Differenzen bei der Weiterentwicklung kam, entschied sich Ende 2014 die Community dazu einen Fork (Javascript I/O: <https://iojs.org/de/>) von Node.js zu erstellen [Grü14]. Jedoch sind beide Projekte mittlerweile wieder unter der neu gegründeten Node.js-Foundation vereint. Mit Version 4.0.0 wurde der Zusammenschluss im August 2015 komplettiert [Nod15]. Aktuell liegt Node.js in der Version 6.6.0 vor. Die neuste Meteor-Version 1.4.1 verwendet allerdings die Long-Term-Support-Version 4.5.0 von Node.js [New16].

Seit der Meteor-Version 1.3 werden npm<sup>1</sup>-Module vollends unterstützt. Zukünftig sollen alle Atmosphere-Plugins durch npm-Module ersetzt werden.<sup>8</sup> [Meta]

### 7.3.3. MongoDB

MongoDB ist eine dokumentenorientierte und somit eine schemafreie Datenbank. Dadurch haben Datensätze keine einheitliche Struktur und können auf Grund dessen Einträge mit unterschiedlichen Typen enthalten. Darüber hinaus ist die Erfassung mehrerer Werte durch das Einbinden von Arrays, sowie eine verschachtelte Struktur möglich. [Mon13, Wika]

Nach Aussage von Merriman, einem der ursprünglichen Entwickler von MongoDB, leitet sich der Name der Datenbank vom englischen Wort *humongous* (zu

<sup>1</sup>Node Package Manager

Deutsch: riesig, gigantisch) ab, um zu verdeutlichen, dass das Programm den Umgang mit umfangreichen Datenmengen unterstützt. [Wika]

Die Datenbank erschien im Jahr 2009 als Open-Source-Software unter der kommerziellen Lizenz GNU AGPL Version 3.0 der Free Software Foundation. [Wika]

### Popularität

Seit einigen Jahren ist MongoDB die populärste dokumentenbasierte Datenbank. Dies ist im Trend-Diagramm (Abbildung 7.6) dargestellt. Hierbei wird ersichtlich, dass ein erheblicher Vorsprung von MongoDB zu anderen Datenbanksystemen besteht. Betrachtet man alle verfügbaren Datenbanksysteme, liegt MongoDB auch hier auf einem guten fünften Platz. Es wird lediglich von relationalen Datenbanksystemen wie MySQL deutlich geschlagen, die jedoch für große Datenmengen unbrauchbar sind. [DE16]

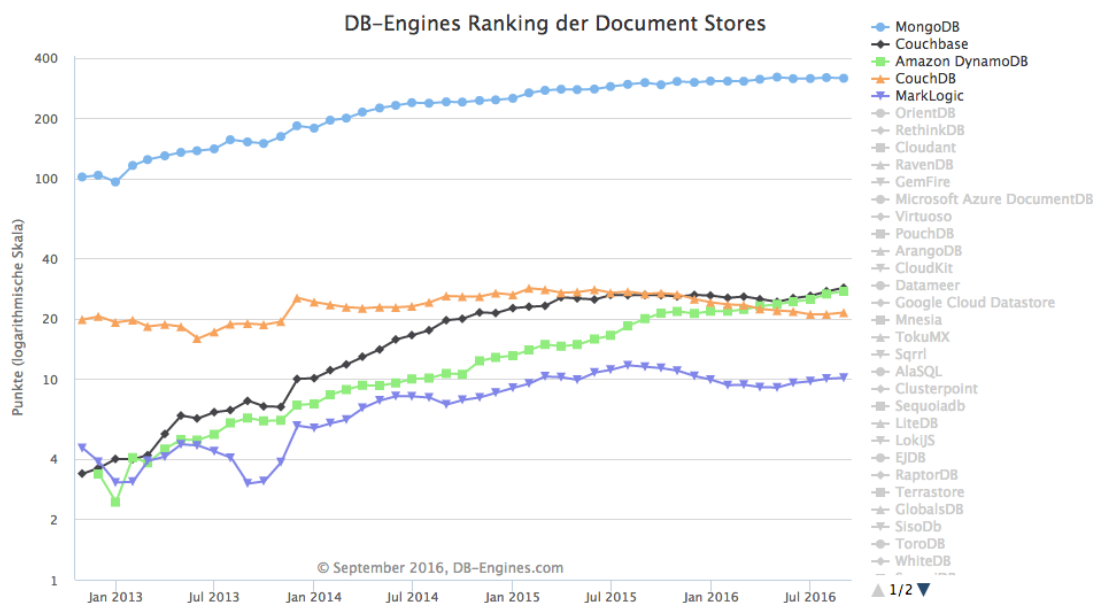


Abbildung 7.6.: Popularität von dokumentenbasierten Datenbanksystemen [DE16]

### Aufbau

Eine MongoDB kann, wie in Abbildung 7.7 dargestellt, mehrere Datenbanken enthalten, die wiederum aus mehreren Collections bestehen. Dabei kann jede Collection ein oder mehrere Dokumente umfassen. Ein Dokument kann Auf-



grund der schemafreien Implementierung von MongoDB unterschiedliche Objekte, auch Key-Value-Paare genannt, beinhalten. Ein Objekt selbst besteht neben dem Schlüsselwort entweder aus einem einfachen Wert, einem Array oder einer Liste von Key-Value-Paaren. [Sut12]

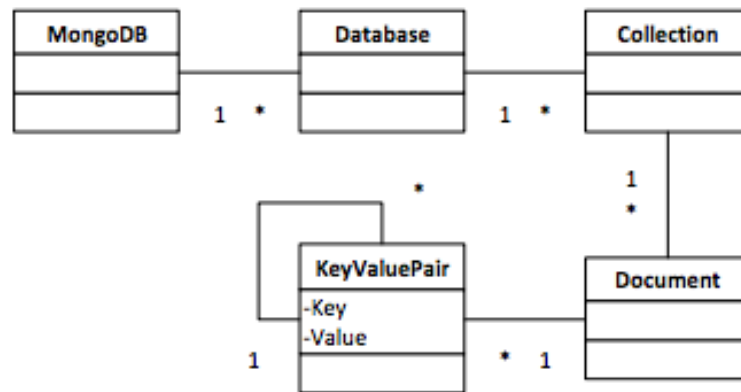


Abbildung 7.7.: Aufbau der dokumentenbasierten Datenbank MongoDB [Sut12]

Für die Datenspeicherung und den Datenaustausch verwendet die Datenbank das sogenannte BSON-Format (kurz für Binary JSON). Dieses Format bietet eine binäre Darstellung von JSON-ähnlichen Dokumenten, die so entwickelt wurde, dass folgende Charakteristiken gewährleistet werden: Leichtgewichtigkeit (Lightweight), eine schnelle Durchsuchung der Inhalte (Traversable), sowie die Effizienz bei der En- und Decodierung des Formates (Efficient). [bso]

### Integration in Meteor

Wie in Abbildung 7.3 dargestellt, läuft Meteor serverseitig innerhalb einer Node.js-Runtime. Dabei wird innerhalb des Meteor-Servers mittels der Mongo-API auf die MongoDB zugegriffen. Auf dem Client wird mithilfe von Minimongo ein lokales Abbild der MongoDB erstellt. Unter Zuhilfenahme von DDP (Distributed Data Protocol) werden alle Daten, die für den Benutzer freigeschaltet sind, lokal auf dem neusten Stand gehalten. Das Laden der Daten findet standardmäßig etwa alle 10 Sekunden oder sobald diese serverseitig durch einen anderen Client modifiziert wurden, statt. Ändert ein Benutzer lokal die Daten, müssen diese durch einen Methoden-Aufruf serverseitig aktualisiert werden, um in die MongoDB übernommen zu werden. [Mar16]

Um eine Validierung der eingefügten Werte zu ermöglichen, ist der Einsatz zusätzlicher Plugins notwendig. Innerhalb von Atmosphere gibt es unter anderem

die Erweiterung *SimpleSchema*, die es ermöglicht, Regeln für Key-Value-Paare einzelner Collection einzuhalten. Zu den Regeln gehören beispielsweise das Festlegen des erlaubten Typs oder die Angabe einer maximalen Länge eines Wertes. Durch die Einhaltung der definierten Vorgaben wird bei einer grundlegend schemafreien Datenbank eine gewisse Ordnung geschaffen. [Metc]

## 8. Anwendung und Implementierung

Im Rahmen einer Implementierung wurden die im zweiten Teil der Masterarbeit präsentierten Methoden und Informationen genutzt, um THMcards zu einer kommerzialisierten Lernkartenplattform auszubauen. Gegenstand dieses Kapitels wird es sein, das Geschäftsmodell von THMcards und die zugehörige Preisgestaltung vorzustellen, sowie die grundlegenden Aspekte der Umsetzung darzulegen.

### 8.1. Geschäftsmodell

Das für THMcards ausgearbeitete Geschäftsmodell besteht, wie in Kapitel 4 Geschäftsmodelle empfohlen, aus einer Verknüpfung verschiedener Arten. Einerseits wird THMcards nach dem Freemium-Modell betrieben, andererseits soll es aber auch Ad-Hoc Bezahlungen unterstützen. Nachfolgend wird im Detail auf beide Modelle und ihr Zusammenwirken eingegangen.

Da THMcards seit seiner Veröffentlichung im Jahr 2011 für alle Benutzer frei zugänglich ist, sollte dies auch weiterhin so beibehalten werden. Ein einfaches Abonnement-Modell ist somit nicht möglich. Aus diesem Grund ist die Entscheidung zugunsten des Freemium-Modells gefallen.

Alle bestehenden und neuen Benutzer können weiterhin die bekannten Funktionen kostenfrei nutzen. Zusätzlich soll ein Abonnement angeboten werden, welches den Funktionsumfang von THMcards erweitert.

Das zur Verfügung gestellte Abonnement ermöglicht es den Benutzern ihre selbst erstellten Kartensätze zu verkaufen. Zudem werden diese Kartensätze vor der Veröffentlichung auf ihre inhaltliche Korrektheit überprüft. Um ein hohes Kartensatz-Niveau zu erhalten, ist diese Überprüfung ausschließlich Dozenten einer Universität oder Hochschule vorbehalten. Die prüfenden Dozenten erhalten eine Gewinnbeteiligung vom entsprechenden Kartensatz. Alle Abonnenten

erlangen darüber hinaus kostenfreien Zugriff auf normalerweise kostenpflichtige Kartensätze.

Wie bereits zuvor erwähnt, ist es möglich virtuelle Güter zu erstehen. Dies dient dazu, den Benutzern einen zweiten Weg des Erwerbs zu bieten. Nicht alle Benutzer sind möglicherweise bereit jeden Monat einen bestimmten Betrag zu bezahlen um Kartensätze zu lernen.

Des weiteren wird somit die Bezahl-Hürde gesenkt und THMcards hat dadurch im Endeffekt insgesamt mehr Umsatz erzielt. Ein Kunde, der einen Kartensatz einmalig erwirbt, ist deutlich besser als einer, der normalerweise THMcards nicht weiter genutzt hätte, da ihm der monatlich Beitrag zu hoch wäre. Der Kunde wird somit dauerhaft gebunden und zusätzlich erhöht sich die Möglichkeit, dass dieser im Laufe der Zeit ein Abonnement in THMcards abschließen wird.

### 8.2. Preisgestaltung

Da die angebotenen kostenpflichtigen Kartensätze durch die zusätzliche Prüfung qualitativ hochwertig sind, ist ein etwas höherer Preis angemessen.

Laufende Kosten gibt es aktuell keine bis geringe, da es ein Projekt des Fachbereichs Mathematik, Naturwissenschaften und Informatik der Technischen Hochschule Mittelhessen ist. Sollte das Projekt in der nächsten Zeit stark wachsen, sind die damit anfallenden zusätzliche Kosten dementsprechend einzuplanen.

Direkte Konkurrenz gibt es durchaus (beispielsweise CoboCards<sup>1</sup> oder Repetico<sup>2</sup>), jedoch nutzen diese alle ein Geschäftsmodell welches sich schlecht bis gar nicht mit dem von THMcards vergleichen lässt. Da eine Angleichung der Preise an die Preise der Konkurrenz sowieso nicht empfohlen wird, sondern nur als groben Rahmen gesehen wird, ist es somit zudem nicht sonderlich essentiell einen direkten Vergleich zu finden. Repetico beispielsweise bietet ähnlich wie THMcards ein Abonnement für zusätzliche Funktionen und Inhalte an. Dabei liegt der Fokus jedoch hauptsächlich auf der Funktionserweiterung. Dieses Abonnement lässt sich in unterschiedlichen Laufzeiten abschließen und kostet zwischen 1,50 € und 2,00 € monatlich. In einem Shop können Kartensätze von Verlägen erworben werden. Diese haben eine Preisspanne von etwa 0,50 € bis hin zu 299,00 €. Ho-

---

<sup>1</sup><http://www.cobocards.com/de/>

<sup>2</sup><https://www.repetico.de/>

he Preise werden jedoch nicht so häufig wie niedrige verlangt. Meist sind Preise zwischen 0,50 € und 20,00 € gefordert.

THMcards beschäftigt sich unter anderem mit den Kundenbedürfnissen der Zeitersparnis und der Herausforderung. Durch die Möglichkeit vorgefertigte Karten mit bekannten Algorithmen zu lernen, reduziert sich der Zeitaufwand im Vergleich zu herkömmlichen Lern-Varianten erheblich. Daraus lässt sich jedoch genauso wenig, wie aus der Herausforderung einen Kartensatz komplett zu können, ein Preis bilden.

Eine Preissegmentierung ist aufgrund des gewählten Geschäftsmodells aktuell ebenfalls nicht durchführbar, da es nicht möglich ist ein weiteres sinnvolles Abonnement anbieten zu können.

Da es keine passende Strategie gibt, richtet sich der Preis grundlegend nach den Preisen der Konkurrenz, bedient sich jedoch auch der Psychologie eines Preises.

Für das angebotene Abonnement wird ein Betrag von 6,99 € erhoben. Da THMcards Abonnenten den etwa gleichen Funktionsumfang aber jedoch inhaltlich deutlich mehr geboten bekommen, als es etwa bei Repetico der Fall ist, ist ein höherer Preis durchaus angemessen.

In THMcards werden die möglichen Preise eines Kartensätze eingeschränkt. Auf Grundlagen der Kartensatzpreise von Repetico werden nur die Kartensatzpreise von 0,99 €, 1,99 €, ... bis 9,99 € erlaubt. Der Grund dafür ist einerseits eine einheitlichere Preisanmutung mit Nutzung des 99-Cent-Tricks und andererseits sollte Bildung immer noch für jedermann bezahlbar bleiben.

## 8.3. Grundlagen

Bei der Umsetzung in THMcards galt es zunächst die Kartensätze zu erweitern um diese in verschiedenen Arten anbieten zu können. Anschließend wurde ein Rollensystem für die unterschiedlichen Berechtigungen in der Anwendung implementiert. Nachfolgend werden die getroffenen Entscheidungen aufgeführt und vorgestellt.

### 8.3.1. Kartensatzarten

Kartensätze sind initial nur für den Ersteller selbst sichtbar und somit nicht veröffentlicht. Sind dem angelegten Kartensatz mindestens fünf Lernkarten hinzugefügt worden, ist es für den Autor möglich, diesen zu veröffentlichen. THMcards

bietet drei verschiedene Veröffentlichungsarten, dabei sind jedoch nicht alle frei zugänglich. Im Folgenden werden die einzelnen Kartensatzarten in ihrem Umfang und Verfügbarkeit vollständig definiert.

Aktuell werden folgende Kartensätze angeboten:						
Thema		Autor/in	Hochschule	Studiengang	Modul	Verfügbarkeit
.git Branches	5	Niklas Veit	THM	Informatik	SWT	Free 0
.git Grundlagen	5	Niklas Veit	THM	Informatik	SWT	Free 0
Allgemeine Fragen zu IBS	5	Nihad Alomerovic	THM	Informatik	IBS	Pro 0.99€ 0
Algorithmen	5	Maximilian Müller	THM	Informatik	AUD	Edu 0.99€ 0
Allgemeinbildung	5	Dominik Maximilian Krion	Zulu-Universität	A-Studiengang	AB	Edu 0.99€ 0

Abbildung 8.1.: Poolansicht in THMcards

### Private

Private Kartensätze sind nur für den Eigentümer sichtbar. Initial werden alle neu angelegte Kartensätze dieser Art zugewiesen. Alle Standardfunktionen, wie beispielsweise Karten erstellen und bearbeiten oder Lernen nach Leitner und Super Memo, sind nutzbar.

### Free

Kartensätze dieser Art werden in der Pool-Ansicht (siehe Abbildung 8.1) von THMcards jedem angemeldeten Benutzer zur Verfügung gestellt. Auch bei dieser Kartensatzart sind alle Standardfunktionalitäten gegeben.

### Edu

*Edu*-Kartensätze sind ebenfalls öffentlich, können allerdings nur von Dozenten angelegt werden. Zudem können nur Mitglieder einer Universität oder Premium-Kunden diesen auch kostenfrei lernen. Alle anderen Anwender werden aufgefordert den vom Autor angegebenen Preis zu zahlen, um diesen Kartensatz zu nutzen.

### Pro

Zu guter Letzt gibt es noch die *Pro*-Kartensätze. Ähnlich den *Edu*-Kartensätzen können solche Kartensätze nur von Premium-Kunden angelegt werden und auch

nur von diesen kostenfrei genutzt werden. Alle anderen Benutzer können diese Kartensätze einmalig erwerben. Zu beachten ist, dass *Pro*-Kartensätze nicht sofort veröffentlicht werden. Ein solcher Kartensatz wird erst dann im Pool angezeigt, sobald dieser von einer berechtigten Person überprüft und freigegeben worden ist. Einschränkend gilt festzuhalten, dass eine darauf folgende Rückstufung des Kartensatzes durch den Autor nicht mehr möglich ist.

### 8.3.2. Rollen

Eine grobe Rollenunterscheidung ist bereits im vorherigen Abschnitt zu erkennen gewesen. Nun folgend wird jede einzelne Rolle nochmals aufgeführt und ihre jeweiligen Berechtigungen erklärt.

#### **Standard**

Ein Standard-Nutzer hat keine besonderen Rechte. Er kann auf alle grundlegenden Funktionen zugreifen, so wie es auch in der ersten Version von THMcards möglich war. Dazu zählt unter anderem private oder freie Kartensätze zu erstellen und seine eigenen oder die als *Free* gekennzeichnete Kartensätze zu lernen.

#### **Pro**

Entscheidet sich ein Standard-Nutzer sein Konto zu erweitern wird er ein Premium-Nutzer. Ein solcher hat alle Rechte eines Standard-Nutzers und kann zusätzlich unentgeltlich auf alle öffentlichen Kartensätze im System zugreifen und auch selbst *Pro*-Kartensätze anlegen. Ein Pro-Mitglied kann somit Geld mit dem Verkauf von Kartensätzen verdienen.

#### **University**

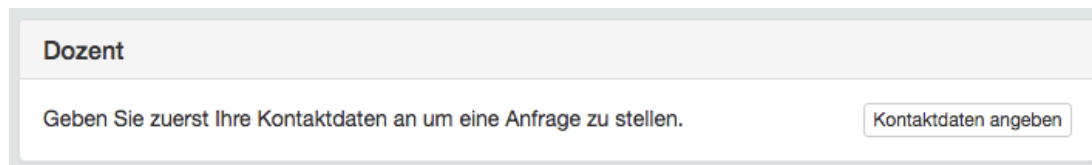
Die Rolle des Mitglieds einer Universität wird jedem Benutzer bei einer Anmeldung über CAS automatisch zugeteilt. Besitzt eine Person einen CAS-Login können somit alle *Edu*-Kartensätze unentgeltlich genutzt werden.

#### **Lecturer**

Dieser Benutzer hat die Rechte und Funktionen eines Dozenten. Er bekommt die zusätzliche Möglichkeit *Edu*-Kartensätze zu erstellen und kann *Pro*-Kartensätze

auf ihre Richtigkeit überprüfen und diese anschließend freischalten oder zurückstellen.

**Dozenten-Anfrage:** Momentan ist eine automatische Unterscheidung zwischen Studenten und Dozenten über den Login-Service CAS nicht möglich. Um in THMcards gegenwärtig die Rolle des Dozenten zu erhalten, ist es notwendig den Administratoren eine Anfrage zu stellen.



The image shows a web form titled "Dozent". Below the title, there is a text instruction: "Geben Sie zuerst Ihre Kontaktdaten an um eine Anfrage zu stellen." To the right of this text is a button with the label "Kontaktdaten angeben".

Abbildung 8.2.: Eine Dozenten-anfrage in THMcards ist nur mit vollständig angegebenen Kontaktdaten möglich

Diese Anfrage ist aus Gründen einer korrekten Personen-Validierung nur möglich, wenn zuvor alle nötigen personenbezogenen Daten im Profil angegeben worden sind (siehe Abbildung 8.2).

**Kartensatz-Freigabe:** Stellt ein Pro-Nutzer einen Antrag auf die Veröffentlichung eines Kartensatzes, erhalten alle Dozenten, wie in Abbildung 8.3 dargestellt, eine Benachrichtigung darüber. Da es für jeden Dozenten möglich ist einen beliebigen Kartensatz freizuschalten und um deshalb Verwirrungen zu vermeiden, wird in den Benachrichtigung zusätzlich der Status der Freigabe angezeigt.

Aufgrund dessen, dass sich die Benachrichtigungen löschen lassen, ist eine zusätzliche Ansicht über alle ausstehenden Anfragen, wie in Abbildung 8.4 zu sehen, implementiert worden.

Eine Kartensatzfreigabe direkt von dieser Ansicht aus ist nicht vorgesehen. Jeder Kartensatz sollte zuvor inhaltlich gründlich überprüft werden, daher ist die Möglichkeit diesen freizuschalten oder abzulehnen auch direkt auf der Kartensatz-Ansicht zu finden (siehe Abbildung 8.5). Das Ablehnen einer Kartensatz-Veröffentlichung ist nur mit einer aussagekräftigen Begründung möglich.

### 8.4. Bezahlvorgang

Neben der Implementierung der Rollen und Kartensatzarten galt es THMcards so zu erweitern, dass Abonnements ermöglicht werden und angebotene Inhalte



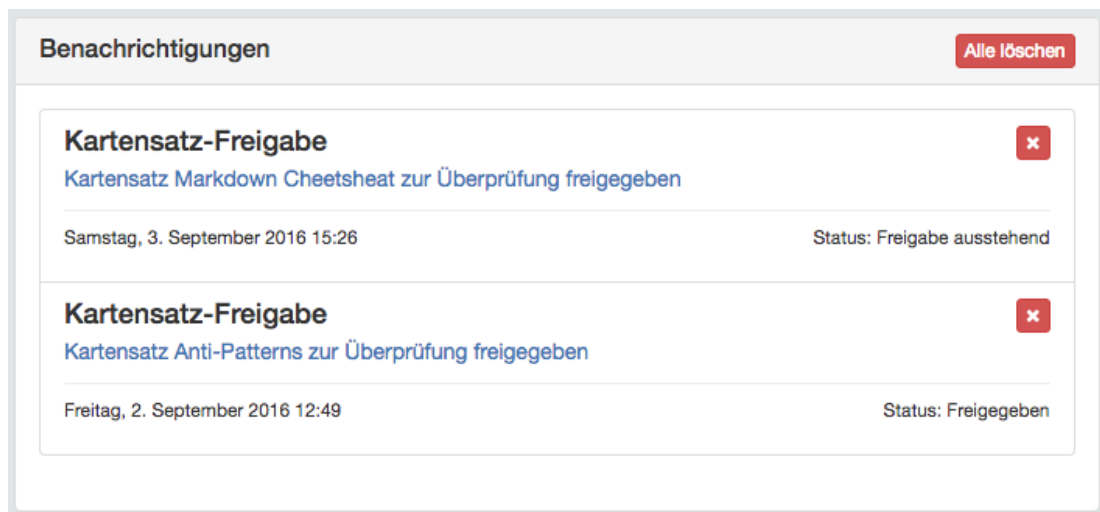


Abbildung 8.3.: Benachrichtigungen über Kartensatz-Freigaben

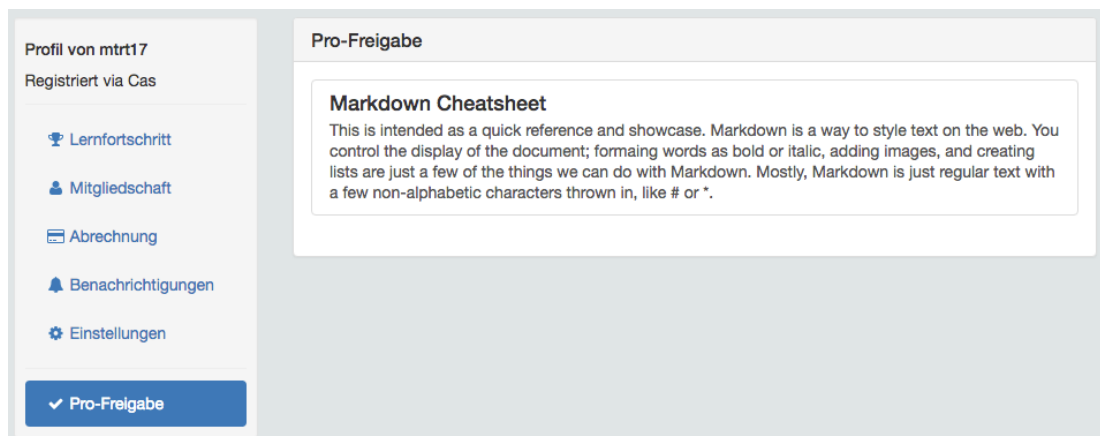


Abbildung 8.4.: Übersicht über alle Kartensatz-Freigabe-Anfragen

zum Kauf angeboten werden können. Hierfür ist die Implementierung eines Bezahlsystems notwendig gewesen. In den folgenden Kapiteln wird dies erläutert.

### 8.4.1. Zahlungsverkehrsplattform

Wenn THMcards für Einkäufe oder finanzielle Transaktionen genutzt wird (beispielsweise wenn ein Kartensatz gekauft wird oder ein Abonnement abgeschlossen wird), werden von THMcards keine Zahlungsinformationen gespeichert. Alle Transaktionen werden über den Dienst Braintree abgewickelt. Dort werden alle Konto- und Authentifizierungsinformationen und Angaben zur Abrechnung gesammelt.

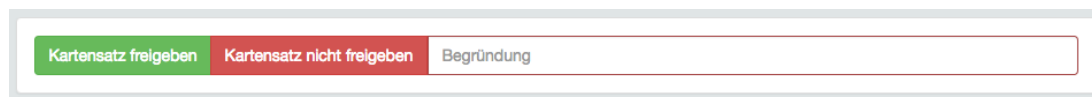


Abbildung 8.5.: Einen Kartensatz freigeben oder nicht freigeben

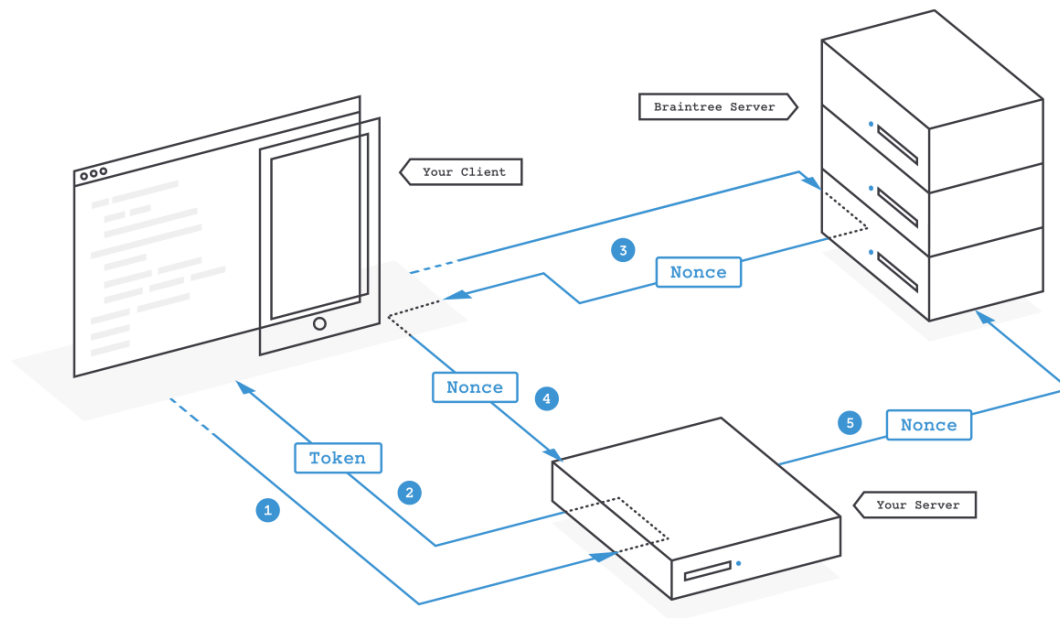


Abbildung 8.6.: Sicherheitsmechanismus von Braintree [Braa]

Die Kommunikation zwischen den einzelnen Instanzen läuft, wie in Abbildung 8.6 dargestellt, ab. Möchte ein Benutzer eine Transaktion tätigen oder ein Abonnement abschließen, fordert zuallererst sein Client vom THMcards-Server einen Client-Token. Daraufhin generiert der Server einen solchen und schickt ihn wieder zurück an den Client. Damit ist der Client für die folgende Aktion initialisiert. Dieser Token ist grundsätzlich nur für eine Aktion zu nutzen und verliert aus diesem Grund bei gehäufter Verwendung seine Gültigkeit. Nachdem der Client demgemäß den Token erhalten hat und die erforderlichen Informationen zur Zahlung durch den Nutzer vorliegen, werden diese Informationen an den Braintree-Server weitergegeben. Dieser erstellt aus den vorliegenden Daten eine Zahlungsmethoden-Nonce und sendet diese an den Client zurück. Eine Zahlungsmethoden-Nonce ist ein einfacher Verweis auf eine beim Benutzer hinterlegte Zahlungsmethode. Aus Sicherheitsmaßnahmen verfällt dieser nach spätestens 3 Stunden. Der Client übermittelt wiederum die Nonce an den THMcards-Server. Dieser kann nun

mittels der Zahlungsmethoden-Nonce eine Transaktion an den Braintree-Server autorisieren. [Braa]

### 8.4.2. Abonnement

Jedes neue Mitglied ist von Beginn an ein Standard-Nutzer. Möchte dieser jedoch die Vorteile des Pro-Nutzers nutzen, so muss er ein Abonnement abschließen. Hierfür ist in THMcards die Profilansicht um den Menüpunkt *Mitgliedschaft* erweitert worden. Dort kann ein Nutzer zwischen den beiden vorhandenen Mitgliedschaftsarten nach belieben wählen (siehe Abbildung 8.7).

Abbildung 8.7.: Auswahl der Mitgliedschaft in THMcards

Ein abgeschlossenes Abonnement hat grundsätzlich eine Dauer von einem Monat und verlängert sich grundsätzlich jeden Monat automatisch um einen weiteren. Eine Kündigung des Pro-Status ist jederzeit mit Wirkung zum Ende des Abrechnungszeitraum möglich. Alle Pro-Funktionen sind somit bis zum Auslaufen des Abonnements für den Anwender weiterhin nutzbar.

Die Nutzungsbedingungen des Abonnements sind in der Masterthesis *Ausarbeitung eines Konzepts für Datenschutz und Datensicherheit in E-Learning Systemen anhand des Beispiels der Lernkarten Plattform THMcards* von Matthias Zimny zu finden [Zim16].

Plan ID	Name	Price	Trial	Billing Cycle	Subscriptions	Actions
pro	pro	6,99 € EUR	none	Every 1 Month(s)	Active: 31 Canceled: 7 Past Due: 0 Pending: 0 Expired: 0	<a href="#">Edit</a>

Abbildung 8.8.: Mitgliedschaftsarten von THMcards in Braintree

Innerhalb von Braintree (siehe Abbildung 8.8) ist es notwendig die in THMcards angebotene kostenpflichtige Mitgliedschaft anzulegen. Dort ist auch eine Übersicht über alle aktuellen Abonnements zu sehen.

### 8.4.3. Einzelkauf

Da einem Standard-Nutzer die Pro-Kartensätze nicht unentgeltlich zur Verfügung gestellt werden, kann dieser einen Pro-Kartensatz einmalig erwerben und anschließend dauerhaft nutzen. Neben dem Titel und einer kurzen Beschreibung des Kartensatzes wird den Kaufinteressenten auch eine Auswahl an Karten als Vorschau angezeigt. Dies erhöht das Interesse an dem Kartensatz und lässt die Kaufschwelle sinken, da der Nutzer vor den Kauf sicher gehen kann ob der Kartensatz seinen Vorstellungen entspricht (siehe Abbildung 8.9).

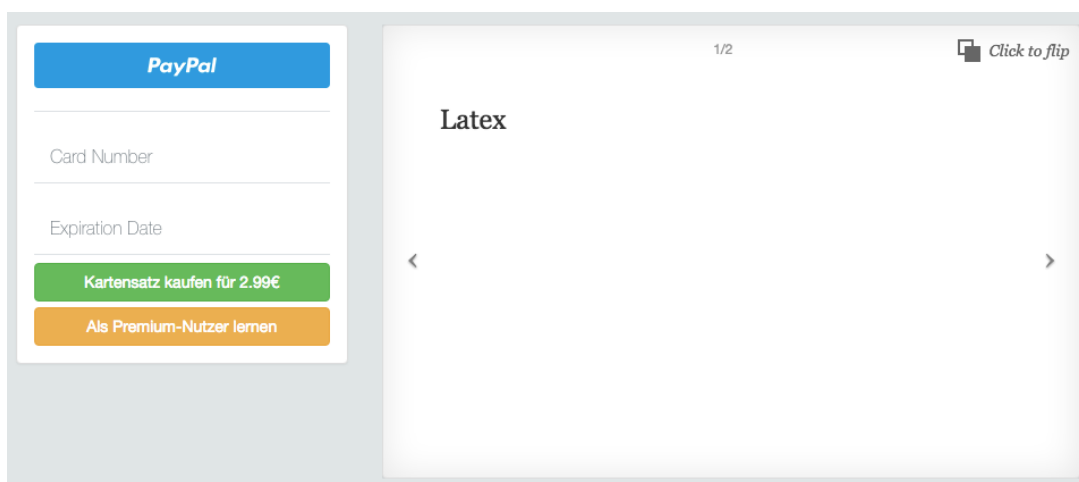


Abbildung 8.9.: Kaufen eines Kartensatzes in THMcards

Um eine einheitliche Preisgestaltung zu ermöglichen, sind die bei der Veröffentlichung eines Kartensatzes auswählbaren Preise fest vorgeschrieben. Die vorgegeben Verkaufspreise nutzen den 99-Cent-Trick und richten sich von der Höhe an den Preis einer kostenpflichtigen Smartphone-App. Verfügbar sind somit

die Beträge 0,99 €, 1,99 €, ... bis 9,99 €. Da bei Online Diensten die Höhe der Mehrwertsteuer abhängig vom Herkunftsland des Kunden ist, kann hier keine allgemeine Einnahmenverteilung dargelegt werden. Die Einnahmen pro verkauftem Kartensatz werden beispielsweise in Deutschland folgendermaßen aufgeteilt:

- 50% gehen an den Autor
- 26% verbleiben bei THMcards
- 19% Mehrwertsteuer
- 5% erhält der Prüfer

Der Autor erhält grundsätzlich 50% der Einnahmen die mit seinen Kartensätzen generiert werden. 45% des Ertrags werden der Anwendung THMcards zugeschrieben, wobei noch die jeweilige länderspezifische Umsatzsteuer und eventuelle Kosten für die Transaktion der Zahlung abgezogen werden müssen. Der freigebende Dozent der Lernkarten erhält pro Kauf des Kartensatzes einen Anteil von 5%.

Für den Erwerb eines Kartensatzes gilt das Widerrufsrecht, wie in den Nutzungsbedingungen von THMcards festgehalten [Zim16].

#### **8.4.4. Abrechnung**

Dieses Kapitel umfasst die Abrechnungs-Verwaltung einerseits aus Benutzersicht und andererseits vom Standpunkt der Administratoren.

##### **Benutzer**

Der Benutzer erhält in seinem Profil unter dem Menüpunkt *Abrechnung*, wie in Abbildung 8.10 dargestellt, einen Überblick über alle seine aktuell verknüpften Zahlungsmethoden. Ebenfalls ist es ihm dort möglich weitere Zahlungsmethoden hinzuzufügen und seine Standard-Zahlungsmethode auszuwählen. Braintree unterstützt in Deutschland aktuell eine Bezahlung mit allen gängigen Kreditkarten und über Paypal.

Hat ein Benutzer erfolgreich einen oder mehrere Kartensätze verkauft, werden ihm die dabei erzielten Gewinne auf sein Konto gutgeschrieben. Die aktuellen Einnahmen werden unter dem Reiter *Kontostand* angezeigt. Auf dieser Ansicht ist dem Nutzer auch eine Auszahlung seiner Umsätze möglich.

The screenshot shows the 'Abrechnung' (Billing) section of the THMcards interface. At the top, there are four tabs: 'Zahlungsmethode' (selected), 'Kontostand', 'Umsatzhistorie', and 'Zahlungshistorie'. Below the tabs, the heading 'Ihre Zahlungsmethoden:' is followed by a list of payment methods. The first method is 'Visa' with the card number '400011\*\*\*\*\*1115 - 12/2021'. The second method is 'PayPal' with the email 'bt\_buyer\_us@paypal.com'. Below this list, there is a section 'Zahlungsmethoden hinzufügen oder auswählen:' which shows a 'Visa ending in 15' and a link 'Change payment method'. At the bottom, there is a large blue button labeled 'Zahlungsmethode speichern'.

Abbildung 8.10.: Übersicht über angelegte Zahlungsmethoden in THMcards

Die beiden Reiter *Umsatzhistorie* und *Zahlungshistorie* listen alle bereits getätigten Transaktionen auf. Dort kann ein Benutzer somit sehen, woher seine Einnahmen stammen und welche Kartensätze er eventuell selbst erworben hat.

### Administrator

In der Datenbank von THMcards sind keinerlei Informationen zur Zahlungsmethode hinterlegt. Alle notwendigen Daten um eine Transaktion durchführen zu können liegen auf dem Braintree-Server. Einzig und alleine eine einzigartige Kundennummer zur Verknüpfung mit den Braintree-Daten wird auf dem THMcards-Server hinterlegt.

Die Verwaltung der Kunden findet von daher komplett über das Braintree Control Panel<sup>3</sup> statt. Dieses Verwaltungstool bietet ein Dashboard (siehe Abbildung 8.11) mit einer grafischen Aufarbeitung der Einnahmen und Transaktionen der vergangenen letzten 30 Tage an. Das Dashboard dient insofern als guter Überblick über die Akzeptanz des Angebots und lässt auch eventuelle Rückschlüsse auf bestimmte Ereignisse, wie etwa das Schalten von Werbung zu einem bestimmten Zeitraum, zu.

---

<sup>3</sup><https://www.braintreegateway.com/login>

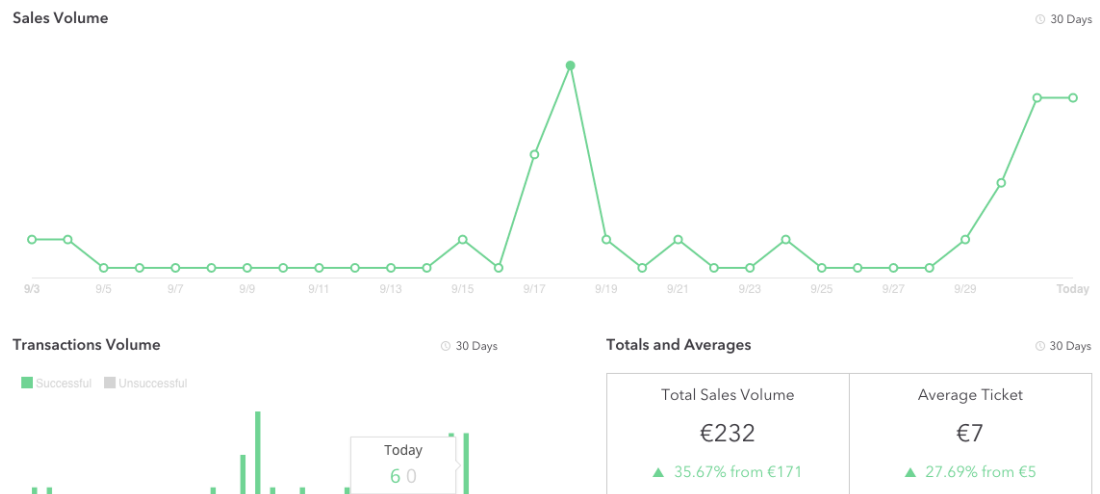


Abbildung 8.11.: Dashbord im Braintree Control Panel

ID	Name	Added	Payment Method	Token	
66145620		09/24/2016	bt_buyer_us@paypal.com	jwhm8r	Charge
16781778		09/19/2016	bt_buyer_us@paypal.com 400011*****1115	8z9q4r 3m7hyr	Charge
51341308		09/18/2016	bt_buyer_us@paypal.com	chbd32	Charge



Abbildung 8.12.: Übersicht über alle Kunden von THMcards in Braintree

Die Administratoren von THMcards erhalten innerhalb des Braintree Control Panels unter anderem, wie in Abbildung 8.12 dargestellt, einen Überblick über alle Kunden von THMcards.

Des Weiteren werden in der kundenspezifischen Ansicht deren Zahlungsmethoden und zugehörige laufende oder beendete Abonnements sowie Transaktionen angezeigt (siehe Abbildung 8.13).

### Payment Methods

[Add Payment Method](#)

Token	Payment Method	Last Used	#Transactions	Actions
3m7hyr	 400011*****1115		0	<a href="#">Edit</a> <a href="#">New Transaction</a> <a href="#">New Subscription</a> <a href="#">Delete</a>
8z9q4r Default	 bt_buyer_us@paypal.com		1	<a href="#">Edit</a> <a href="#">New Transaction</a> <a href="#">New Subscription</a> <a href="#">Delete</a>

### Subscriptions

Subscription ID	Plan Name	Price	Trial Period	Billing Cycle	Customer Name	Status	Actions
fvh5w2	pro	6,99 € EUR	none	Every 1 Month(s)		Active	<a href="#">Edit</a> <a href="#">Cancel</a>

### Addresses

[Add address](#)

This customer does not have saved addresses.

### Last 10 Transactions

[See All Transactions](#)

ID	Type	Status	Amount	Payment Method Token	Created
dc7cqr1k	sale	Settled	6,99 € EUR	8z9q4r	09/19/2016 02:29:05 AM CDT

Abbildung 8.13.: Einzelansicht eines Kunden von THMcards in Braintree



## 9. Resümee

Intention der vorliegenden Arbeit war es, die Geschäftsidee von THMcards in ein passendes Geschäftsmodell zu übertragen, um diesen als kostenpflichtigen Dienst im Internet zu etablieren und dabei den Kundenbedürfnissen, insbesondere hinsichtlich der Lernqualität, gerecht wird.

Nach der theoretischen Wissensaneignung über die verschiedenen Arten eines Geschäftsmodells und der Erkenntnis, dass eine Kombination aus mehreren Arten und somit ein Hybrid-Geschäftsmodell deutlich zu bevorzugen ist, konnten die einzelnen Kriterien einer Preisgestaltung eruiert werden. Darauf aufbauend sind die Grundlagen für eine Integration von Zahlungssystemen in eine Web-Anwendung geschaffen worden.

Die darauf folgende praktische Umsetzung mit der Wahl des Geschäftsmodells für THMcards war keine leichte, da es viele verschiedene Herangehensweisen gibt, um aus THMcards eine erfolgreiche Web-Anwendung zu machen. Ob es im Endeffekt die richtige Wahl war, lässt sich erst erkennen, sobald THMcards der breiten Masse angeboten wird. Dazu wird es nötig sein Statistiken über die Nutzungshäufigkeit und Akzeptanz von THMcards zu führen. Eine große Hilfe wird hierbei das Control Panel von Braintree bieten.

Mittels weiterer Projekte rund um THMcards kann das Geschäftsmodell sicherlich weiter verbessert und ausgebaut werden. Hierbei sollten unter anderem bei der Einführung neuer Funktionen mithilfe der Preissegmentierung weitere Abonnement-Versionen angeboten werden.

Die Qualität des Lernen wurde mit dem gewählten Geschäftsmodell sicherlich deutlich gehoben. Nun sind neben den normalen Kartensätzen, die durch jedermann veröffentlicht werden können, auch inhaltlich geprüfte Kartensätze vorhanden. Eine selbstständige Prüfung der Kartensätze, bevor diese gelernt werden können, entfällt somit völlig und es wird dementsprechend beim Lernen deutlich Zeit eingespart.

Für die Zukunft von THMcards lassen sich weitere interessante Ansatzweisen betrachten, die eine Verbesserung der Lernqualität ermöglichen würden. Dazu

zählt unter anderem die Möglichkeit zukünftig Kartensätze gemeinsam mit der Community von THMcards zu erstellen oder eine Überprüfung der Kartensätze durch die Community selbst zu ermöglichen.

Diese Masterthesis hat den Grundstein für eine spannende berufliche Perspektive gelegt, die nach dem Abschluss des Studiums sicherlich weiter verfolgt wird. Besonders der kreative Prozess der Geschäftsmodellierung und der zugehörigen Preisgestaltung war einerseits herausfordernd, aber andererseits auch sehr lohnenswert. Für zukünftige Projekte wird es jedoch als sinnvoll erachtet, solch einen Prozess in seinem vollen Umfang mit einem Gründungsteam zu bestreiten.

# Literaturverzeichnis

- [2BI] 2BIT: *SPA (Single Page Application) – der perfekte Begleiter für klassische Softwareanwendungen*. <http://www.2bit.ch/single-page-application/>, Abruf: 20.09.2016
- [Aul16] AULET, Bill: *Startup mit System*. O'Reilly, 2016
- [Bla16] BLACKDUCK: *The Tenth Annual Future of Open Source Survey*. <https://www.blackducksoftware.com/2016-future-of-open-source>. Version: 2016, Abruf: 06.09.2016
- [Bon16] *Winter-Umfrage 2015/16: Genutzte Bezahlverfahren von Verbrauchern im Online- und Versandhandel*. Boniversum und bevh. [http://www.boniversum.de/wp-content/uploads/2016/07/Bezahlverfahren\\_im\\_Online-\\_und\\_Versandhandel\\_2015-16\\_TAB.pdf](http://www.boniversum.de/wp-content/uploads/2016/07/Bezahlverfahren_im_Online-_und_Versandhandel_2015-16_TAB.pdf). Version: 2016, Abruf: 29.09.2016
- [Braa] *Get Started*. Braintree. <https://developers.braintreepayments.com/start/overview>, Abruf: 03.10.2016
- [Brab] *Recurring Billing*. Braintree. <https://developers.braintreepayments.com/guides/recurring-billing/overview>, Abruf: 30.09.2016
- [Bru15] BRUENTJE, Marlon: *Single-Page-Application - Individuelle Web-Anwendung für Unternehmen*. <http://www.flyacts.com/blog/single-page-application-individuelle-web-anwendung-fuer-unternehmen/>. Version: 2015, Abruf: 20.09.2016
- [bso] *BSON Description*. <http://bsonspec.org/>, Abruf: 19.09.2016
- [Buc13] BUCHMANN, Dennis: *Digitales Lernen*. Landesanstalt für Medien (Norrhein-Westfalen). [https://www.lfm-nrw.de/fileadmin/lfm-nrw/nrw\\_digital/Publikationen/DK\\_Digitales\\_Lernen.pdf](https://www.lfm-nrw.de/fileadmin/lfm-nrw/nrw_digital/Publikationen/DK_Digitales_Lernen.pdf). Version: 2013, Abruf: 06.09.2016
- [Dar07] DARNSTÄDT, Thomas: *Die Gratis-Kultur*. Spiegel. <http://www.spiegel.de/spiegel/spiegelspecial/d-52058065.html>. Version: 2007, Abruf: 06.09.2016

- [DE16] DB-ENGINES: *DB-Engines Ranking - Trend der Document Stores Popularität*. [http://db-engines.com/de/ranking\\_trend/document+store](http://db-engines.com/de/ranking_trend/document+store). Version: 2016, Abruf: 29.09.2016
- [Dic09] DICTIONARY, Farlex F.: *Weber's Law*. <http://financial-dictionary.thefreedictionary.com/Weber+Law>. Version: 2009, Abruf: 14.09.2016
- [Grüa] GRÜNDERSZENE: *Monetarisierung*. <http://www.gruenderszene.de/lexikon/begriffe/monetarisierung>, Abruf: 06.09.2016
- [Grüb] GRÜNDERSZENE: *Skalieren*. <http://www.gruenderszene.de/lexikon/begriffe/skalieren>, Abruf: 06.09.2016
- [Grüc] GRÜNDESZENE: *Business-Model*. <http://www.gruenderszene.de/lexikon/begriffe/business-model>, Abruf: 06.09.2016
- [Grü14] GRÜNER, Sebastian: *Node.js-Fork für mehr Freiheiten*. Golem. <http://www.golem.de/news/io-js-node-js-fork-fuer-mehr-freiheiten-1412-110994.html>. Version: 2014, Abruf: 21.09.2016
- [Hec16] HECKER, Peter: *JavaScript goes Enterprise - Node.js-Anwendungen mit Visual Studio und den Node.js-Tools entwickeln*. SlideShare. <http://de.slideshare.net/phecker65/javascript-goes-enterprise-nodejsanwendungen-mit-visual-studio-und-den-nodejstools-entwickeln>. Version: 2016, Abruf: 29.09.2016
- [ifr16] IFROSS: *Darf Open Source Software kommerziell sein?* <http://www.ifross.org/darf-open-source-software-kommerziell-sein>. Version: 2016, Abruf: 06.09.2016
- [Ini] INITIATIVE, Open S.: *The Open Source Definition (Annotated)*. <https://opensource.org/osd-annotated>, Abruf: 06.09.2016
- [Inn] INNOVATION, Business M.: *Definition Geschäftsmodell*. <http://www.business-model-innovation.com/definitionen/geschaeftsmodell.htm>, Abruf: 06.09.2016
- [Kam14] KAMMER, Jan C.: *Implementierung von Spaced Repetition Algorithmen zur effektiven Abfrage von Lernkaten innerhalb der eLearning Plattform THMcards*, Technischer Hochschule Mittelhessen, Diplomarbeit, 2014
- [Kna13] KNAPP, Daniel: *Implementierung von Spielmechaniken zur Steigerung der Lernmotivation von Studierenden am Beispiel der Lernkarten Plattform THMcards*, Technischer Hochschule Mittelhessen, Diplomarbeit, 2013

- [Lie13] 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/>. Version: 2013, Abruf: 26.09.2016
- [Lob13] LOBO, Sascha: *Warum der Begriff "Gratismentalität" Unsinn ist*. Spiegel. <http://www.spiegel.de/netzwelt/web/sascha-lobo-der-begriff-gratismentalitaet-ist-unsinn-a-873520.html>. Version: 2013, Abruf: 06.09.2016
- [Mac16] MACKAY, Jory: *How to price anything: The psychology of why we'll pay what we pay*. <https://crew.co/backstage/blog/the-psychology-of-pricing>. Version: 2016, Abruf: 14.09.2016
- [Mar16] MARTIN, Nick: *Tuning Meteor Mongo Livedata for Scalability*. Meteor. <http://info.meteor.com/blog/tuning-meteor-mongo-livedata-for-scalability>. Version: 2016, Abruf: 30.09.2016
- [Meta] *Atmosphere vs. npm*. Meteor. <https://guide.meteor.com/atmosphere-vs-npm.html>, Abruf: 30.09.2016
- [Metb] *Custom Deployment*. Meteor Developers. <https://guide.meteor.com/deployment.html#custom-deployment>, Abruf: 30.09.2016
- [Metc] *Defining a schema*. Meteor. <https://guide.meteor.com/collections.html#schemas>, Abruf: 30.09.2016
- [Metd] *METEOR: Introducing Meteor API Docs*. <http://docs.meteor.com/#what-is-meteor>, Abruf: 26.09.2016
- [Mon13] *MongoDB – Die dokumentorientierte JSON-Datenbank*. NodeCode. <http://nodecode.de/mongodb>. Version: 2013, Abruf: 30.09.2016
- [Mon15] MONGODB: *Meteor: Build iOS and Android Apps that are a Delight to Use*. <https://www.mongodb.com/blog/post/meteor-build-ios-and-android-apps-are-delight-use>. Version: 2015, Abruf: 26.09.2016
- [MS09] MANNING, Kenneth C. ; SPROTT, David E.: *Price Endings, Left-Digit Effects, and Choice*. Journal of Consumer Research, 2009
- [New16] NEWMAN, Ben: *Announcing Meteor 1.4.1: Node 4.5.0, Faster Package Downloads, + More*. Meteor Blog. <http://info.meteor.com/blog/announcing-meteor-1.4.1>. Version: 2016, Abruf: 21.09.2016
- [NK13] NIKO KOEBLER, Heiko S.: *Einstieg in die Entwicklung von Web-Apps mit Meteor*. <http://www.heise.de/developer/artikel/Einstieg-in-die-Entwicklung-von-Web-Apps-mit-Meteor-1949891.html>. Version: 2013, Abruf: 26.09.2016

- [Nod] [https://d262ilb51hltx0.cloudfront.net/max/800/1\\*Te1HNFdDpgCn9ImQivtwRA.jpeg](https://d262ilb51hltx0.cloudfront.net/max/800/1*Te1HNFdDpgCn9ImQivtwRA.jpeg)
- [Nod15] *Node v4.0.0 (Current)*. <https://nodejs.org/en/blog/release/v4.0.0/>. Version: 2015, Abruf: 21.09.2016
- [Owe15] OWENS, Josh: *Why Meteor?* <http://whymeteor.com/>. Version: 2015, Abruf: 26.09.2016
- [Pur11] PURDY, Kevin: *How Number Psychology Impacts the Prices You'll Pay*. <http://lifehacker.com/5794319/how-number-psychology-impacts-the-prices-youll-pay>. Version: 2011, Abruf: 14.09.2016
- [QC13] QUIBLEDEY-CIRKEL, Prof. Dr. K.: Leitners Lernkartei re-visited: Gegen das Aufschieben und Vergessen im Studium. (2013). <https://dl.dropboxusercontent.com/u/87040050/Fellowship/Leitners%20Lernkartei%20revisited-%20Gegen%20das%20Aufschieben%20und%20Vergessen%20im%20Studium.pdf>
- [Rao15] RAO, Leena: *Stripe's new funding makes it a 5 billion US-Dollar company*. Fortune. <http://fortune.com/2015/07/28/stripe-visa/>. Version: 2015, Abruf: 22.09.2016
- [res13] RESEARCH ibi: *Erfolgsfaktor Payment*. eCommerce Leitfaden. [http://www.ecommerce-leitfaden.de/download/studien/Studie\\_Erfolgsfaktor\\_Payment\\_2013.pdf](http://www.ecommerce-leitfaden.de/download/studien/Studie_Erfolgsfaktor_Payment_2013.pdf). Version: 2013, Abruf: 15.09.2016
- [Sex13] <http://www.slideshare.net/NakulPatel/webers-law-and-why-big-business-believes-you-wont-notice-a-price-increase-under-10-20029922>
- [Spr13a] SPRINGER, Sebastian: *Node.js - Das umfassende Handbuch*. Galileo Computing, 2013
- [Spr13b] SPRINGER, Sebastian: *Node.js: Das JavaScript-Framework im Überblick*. t3n. <http://t3n.de/magazin/serverseitige-javascript-entwicklung-nodejs-einsatz-231152/>. Version: 2013, Abruf: 30.09.2016
- [Stä01] STÄHLER, Patrick: *Geschäftsmodelle in der digitalen Ökonomie*. Eul Verlag, Köln-Lohmar, 2001
- [Sti14] STICHTENOTH, Olaf: *Single Page Applications: die Lösung für alle Probleme?* <https://blog.secu-ring.de/software/single-page-applications-loesung-fuer-probleme/>. Version: 2014, Abruf: 20.09.2016

- [Str] *Pricing*. Stripe. <https://stripe.com/de/pricing>, Abruf: 22.09.2016
- [Sut12] SUTER, Rico: *MongoDB - An introduction and performance analysis*. Hochschule für Technik Rapperswil. <http://wiki.hsr.ch/Datenbanken/files/MongoDB.pdf>. Version: 2012, Abruf: 29.09.2016
- [Tur14] TURNBULL, David: *7 Reasons to Develop Your Next Web App with Meteor*. <https://www.sitepoint.com/7-reasons-develop-next-web-app-meteor/>. Version: 2014, Abruf: 26.09.2016
- [Tut15] TUTORIALS, Bootstrap: *Meteor JS: Was ist das und wie kann ich es nutzen?* <http://bootstrapaholic.de/tutorials/meteor-js-tutorial/>. Version: 2015, Abruf: 26.09.2016
- [Wika] *MongoDB*. Wikipedia. [https://de.wikipedia.org/wiki/MongoDB#cite\\_note-mongowebseite-2](https://de.wikipedia.org/wiki/MongoDB#cite_note-mongowebseite-2), Abruf: 30.09.2016
- [Wikb] *Node.js*. Wikipedia. <https://de.wikipedia.org/wiki/Node.js>, Abruf: 30.09.2016
- [Wikc] WIKIPEDIA: *Definition von Open Source*. [https://de.wikipedia.org/wiki/Open\\_Source\\_Initiative#Definition\\_von\\_Open\\_Source](https://de.wikipedia.org/wiki/Open_Source_Initiative#Definition_von_Open_Source), Abruf: 06.09.2016
- [Wikd] WIKIPEDIA: *Geschäftsmodell*. <https://de.wikipedia.org/wiki/Gesch%C3%A4ftsmodell>, Abruf: 06.09.2016
- [Wik16] *Stripe (company)*. Wikipedia. [https://en.wikipedia.org/wiki/Stripe\\_\(company\)](https://en.wikipedia.org/wiki/Stripe_(company)). Version: 2016, Abruf: 22.09.2016
- [Wira] WIRTSCHAFTSLEXIKON24: *Preisanker*. <http://www.wirtschaftslexikon24.com/d/preisanker/preisanker.htm>, Abruf: 14.09.2016
- [Wirb] WIRTSCHAFTSLEXIKON24: *Preiswahrnehmung*. <http://www.wirtschaftslexikon24.com/d/preiswahrnehmung/preiswahrnehmung.htm>, Abruf: 14.09.2016
- [Wir16] WIRTZ, Bernd W.: *Electronic Business*. Springer, 2016
- [YP10] YVES PIGNEUR, Alexander O.: *Business Modell Generation*. John Wiley + Sons, 2010
- [Yu13] YU, Eric: *Price Anchoring to Optimize Your Pricing Strategy*. Price Intelligently. <http://www.priceintelligently.com/blog/bid/181199/Price-Anchoring-to-Optimize-Your-Pricing-Strategy>. Version: 2013, Abruf: 30.09.2016

- [Zam12a] ZAMBONINI, Dan: *Poll: Why are you building a web app?* Hacker News. <https://news.ycombinator.com/item?id=2210150>. Version: 2012, Abruf: 12.06.2016
- [Zam12b] ZAMBONINI, Dan: *A Practical Guide to Web App Success*. Five Simple Steps, 2012
- [Zim16] ZIMNY, Matthias: *Ausarbeitung eines Konzepts für Datenschutz und Datensicherheit in E-Learning Systemen anhand des Beispiels der Lernkarten Plattform THMcards*, Technischer Hochschule Mittelhessen, Diplomarbeit, 2016



## 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.

Petersberg-Steinau, den 5. Oktober 2016

---

MARIUS TRAUTRIMS

