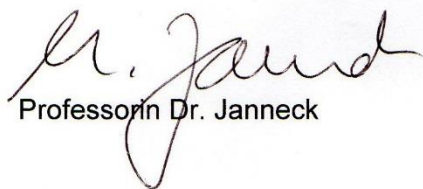


## Thema der Bachelorarbeit

für

Herrn Johann M a n t l e r


Entwicklung einer Web-App für Tagebuchstudien

  
Professorin Dr. Janneck

Ausgabedatum: 01. Juli 2014  
Abgabedatum: 01. Oktober 2014

gefördert durch:



  
(Professor Dr. rer. nat. habil. Schiffer)  
Vorsitzender des Prüfungsausschusses

## **Erklärung zur Bachelorarbeit**

Ich versichere, dass ich die Arbeit selbständig, ohne fremde Hilfe verfasst habe.

Bei der Abfassung der Arbeit sind nur die angegebenen Quellen benutzt worden. Wörtlich oder dem Sinne nach entnommene Stellen sind als solche gekennzeichnet.

Ich bin damit einverstanden, dass meine Arbeit veröffentlicht wird, insbesondere dass die Arbeit Dritten zur Einsichtnahme vorgelegt oder Kopien der Arbeit zur Weitergabe an Dritte angefertigt werden.

---

(Datum)

---

Unterschrift

# 1 Inhaltsverzeichnis

<b>Erklärung zur Bachelorarbeit.....</b>	<b>2</b>
<b>1 Inhaltsverzeichnis.....</b>	<b>3</b>
.....	5
<b>Aufgabenstellung.....</b>	<b>5</b>
<b>2 Einleitung .....</b>	<b>6</b>
<b>3 Anforderungen .....</b>	<b>7</b>
<b>4 Vorgehensmodell .....</b>	<b>9</b>
4.1 Umgebungsbedingungen .....	9
4.2 Agiles Vorgehen.....	9
4.2.1 Manifest für agile Softwareentwicklung .....	9
4.2.2 Wahl des agilen Prozessmodells .....	10
<b>5 Entwurf .....</b>	<b>13</b>
5.1 Client-Server-Modell.....	14
5.2 Client-Server-Kommunikation .....	15
5.3 Serverseitiges Modul .....	15
5.3.1 Datenhaltung .....	18
5.3.1.1 Standardisiertes Datenbankschema .....	18
5.3.1.2 Datenbankentwurf.....	20
5.3.1.3 Objektrelationale Abbildung.....	21
5.4 Clientseitiges Modul.....	22
<b>6 Technologien .....</b>	<b>23</b>
6.1 Serverseitiges Modul .....	23
6.1.1 Wahl der Programmiersprache .....	23

6.1.2	Wahl des Webservers .....	24
6.1.3	Wahl des Datenbankservers .....	25
<b>7</b>	<b>Anhang .....</b>	<b>26</b>
	<b>Literaturverzeichnis.....</b>	<b>27</b>



## Aufgabenstellung

Tagebuchstudien sind eine sinnvolle Methode, um kontinuierlich Nutzererfahrungen im realen Anwendungsalltag zu erfassen. Sie sind jedoch für die Testpersonen aufwändig durchzuführen, insbesondere wenn papierbasierte Tagebücher zum Einsatz kommen. Im Rahmen dieser Abschlussarbeit soll eine Web-App entwickelt werden, um Tagebuchstudien mittels mobiler Endgeräte durchzuführen und somit den Testpersonen die Dateneingabe zu erleichtern. Die App soll das Ausfüllen von Fragebögen ermöglichen und auf verschiedenen mobilen Endgeräten (Smartphones, Tablets) sowie unter verschiedenen Betriebssystemen einsetzbar sein. Als konkreter Anwendungsfall wird der Fragebogen zur Erfassung computerbezogener Attributionen umgesetzt.

Die Aufgaben dieser Arbeit umfassen im Einzelnen:

1. Umsetzung der Benutzerschnittstelle einer Webanwendung für mobile Endgeräte nach Entwurfsvorlage.
2. Entwicklung der zugehörigen Serverapplikation inklusive Datenhaltung.
3. Installation und Konfiguration des Webserver zur Vorbereitung auf die Veröffentlichung im Internet.

Alle Aufgaben wurden umgesetzt. Als Ergebnis wird bereitgestellt:

- Einsatzfähige Web-App gemäß der Anforderungsspezifikation mit Ausnahme der 7. User Story (siehe Kapitel 3).

## 2 Einleitung

Im Zusammenhang mit der Mensch-Computer-Interaktion soll in einer Studie nach computerbezogenen Attributionen geforscht werden. Eine Attribution ist ein kognitiver Prozess, bei dem eine Zuschreibung von Ursache und Wirkung einer Handlung eine Verhaltensänderung bewirkt [För01]. Das Ziel der Studie ist es, Informationen über Attributionsmuster bei der Computernutzung zu erlangen, um möglicherweise Computersysteme besser benutzergerecht zu gestalten. Damit kontinuierlich attributionsrelevante Daten der Probanden gesammelt werden, wird die Studie als Tagebuchuntersuchung konzipiert. Die Studie wird im Zuge der Promotion von Adelka Niels stattfinden, die auch Zweitbetreuerin dieser Bachelorarbeit ist.

In jeder Studie müssen Daten gesammelt, verwaltet und verarbeitet werden. Die Informatik als Wissenschaft der systematischen, automatisierten Verarbeitung von Informationen bietet Lösungen [Ges06] und vereinfacht den gesamten Verlauf der Studie. Insbesondere bei Tagebuchstudien, die kontinuierlich Daten von Studienteilnehmern erheben, stellen papierbasierte Fragebögen einen hohen Aufwand dar. Die Studie ist eine solche Tagebuchstudie. Aus diesem Grund soll ein Programm entwickelt werden, welches die Studie unterstützt. Das Programm soll:

- Auf möglichst allen Plattformen laufen, um die größtmögliche Anzahl an Studienteilnehmern zu erreichen.
- Den Aufwand der Studienteilnahme von Testpersonen möglichst gering halten. Dazu gehört insbesondere, den Studienteilnehmer nicht mit Installation von Programmen zu konfrontieren.

Diese beiden Anforderungen können nur durch ein Programm basierend auf Webtechnologien gelöst werden. Eine Webseite, umgesetzt mit Hypertext Markup Language (Abk. HTML), Cascading Style Sheets (Abk. CSS) und clientseitigen Skriptsprachen bietet die geforderte Plattformunabhängigkeit. Nahezu jeder Webbrowser „versteht“ diese Technologien und nahezu jedes Betriebssystem hat einen Webbrowser vorinstalliert. In Kombination mit serverseitigen Sprachen, welche die Datenverwaltung übernehmen, ist eine Webanwendung eine Programmlösung für die Studie.

Die Webanwendung soll primär für mobile Endgeräte optimiert sein. Die steigende Anzahl an mobilen Internetnutzern lässt darauf schließen, dass immer mehr Menschen ein mobiles Endgerät mit sich tragen [Car14]. Aufgrund dieser Tatsache ist es sinnvoll, die Webapplikation für mobile Nutzer zugänglich zu machen. Eine schnelle Erledigung des wiederkehrenden Fragebogens ist von großer Bedeutung, um der Vergesslichkeit des Geschehenen entgegen zu kommen.

### 3 Anforderungen

In diesem Kapitel wird eine Übersicht der Anforderungen an die HCI-Diary App, im Folgenden vereinfacht Web-App genannt, gegeben. Ein vollständiges Lastenheft, sowie späte Anforderungsänderungen sind auf der beigelegten DVD im Ordner „1.2 Anforderungen“ zu finden. Auch ist dort ein umfangreicher Mock-up hinterlegt, der die Navigation und den Entwurf der Benutzerschnittstelle eindeutig zeigt. Deshalb sind die folgenden Anforderungsbeschreibungen nur als Ergänzung zum Verständnis dieser Arbeit zu verstehen.

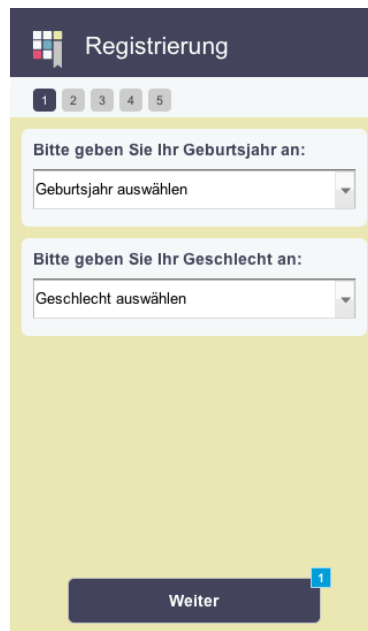
Die Web-App ist für mobile Endgeräte optimiert, soll aber auch auf großen Bildschirmen geeignet dargestellt werden. Studienbeginn (und damit Produktiveinsatz) ist im 4. Quartal 2014. Die Studie soll mindestens zwei Monate andauern, jedoch ist eine Untersuchung in einem längeren Zeitraum nicht ausgeschlossen (z.B. halbes Jahr oder länger) nicht ausgeschlossen. Es werden bis zu 20 Studienteilnehmer erwartet, wobei die Anzahl an Teilnehmern nicht beschränkt werden sollte. Die Web-App kann auch in weiteren Usability-Studien eingesetzt werden. Folgende User Stories können aus den Anforderungsdokumenten abgeleitet werden:

1. Als Benutzer muss ich mich registrieren, um an der Studie teilzunehmen.
2. Als Benutzer wähle ich ein Benutzernamen und Passwort, um mein Konto zu schützen.
3. Als Benutzer möchte ich meine E-Mail-Adresse angeben, wenn ich an der Verlosung (Gewinnspiel) teilnehmen will.
4. Als Benutzer möchte ich mich anmelden, um einen Erfolg einzutragen.
5. Als Benutzer möchte ich mich anmelden, um ein Problem einzutragen.
6. Als Benutzer möchte ich ein Beispiel-Problemeintrag sehen, damit ich weiß, wie ich Probleme und Erfolge eintragen kann.
7. Als Benutzer möchte ich den Fragebogen zum Selbstkonzept auch später ausfüllen.  
*Diese User Story wurde nicht umgesetzt.*
8. Als Benutzer möchte ich sehen, wie viele Erfolge ich schon eingetragen habe.
9. Als Benutzer möchte ich sehen, wie viele Probleme ich schon eingetragen habe.
10. Als Benutzer kann ich das HCI-Diary App Team kontaktieren, wenn ich etwas mitzuteilen habe.
11. Als Benutzer möchte ich, dass meine Anmeldedaten gespeichert werden, damit ich mich nicht jedes Mal anmelden muss.
12. Als Benutzer möchte ich mein Passwort zurücksetzen können.
13. Als Benutzer möchte ich mich abmelden können.
14. Als Benutzer möchte ich jederzeit die Teilnahmebedingungen einsehen können.
15. Als Benutzer möchte ich jederzeit die Datenschutzrichtlinien einsehen können.
16. Als Benutzer möchte ich jederzeit das Impressum einsehen können.
17. Als Administrator möchte ich mich anmelden, um alle bis dato gesammelten Studiendaten als CSV-Datei zu bekommen.
18. Als Administrator möchte ich mich abmelden können.

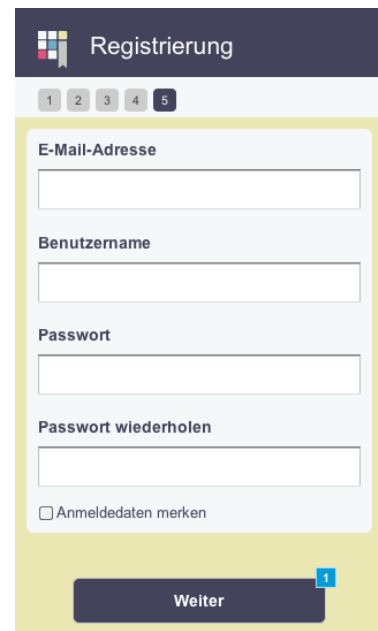
Die nachfolgenden Abbildungen zeigen Ausschnitte aus dem Mock-up.



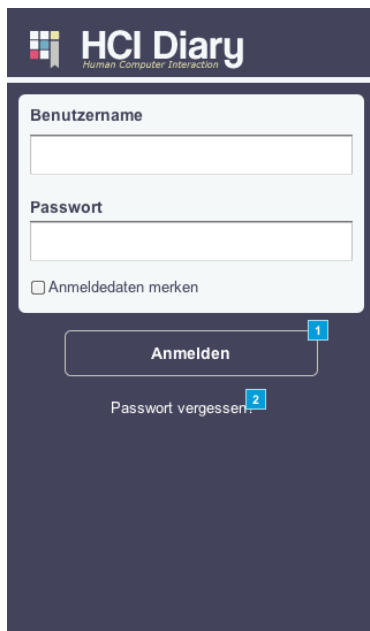
**Abbildung 1:** Der Benutzer kann auf der Startseite zwischen „Anmelden“ und „Registrieren“ wählen.



**Abbildung 2:** Die Registrierung besteht aus fünf Seiten mit persönlichen Fragen.



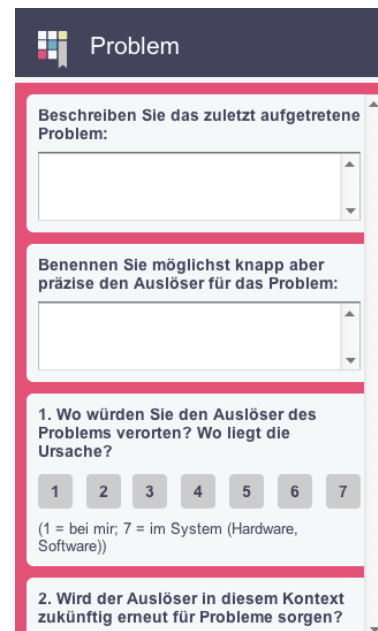
**Abbildung 3:** Der Benutzer kann auch auf „Anmeldedaten merken“ klicken.



**Abbildung 4:** Auch eine „Passwort vergessen“ - Funktion ist Teil des Anmeldeformulars.



**Abbildung 5:** Nach Anmeldung kann der Benutzer Erfolge und Probleme eintragen.



**Abbildung 6:** Zu einem Problem sind mehrere Fragen auszufüllen.



## 4 Vorgehensmodell

In diesem Kapitel wird das organisatorische Vorgehen zur Entwicklung der Web-App erklärt.

### 4.1 Umgebungsbedingungen

Im zeitlichen Rahmen einer Bachelorarbeit von drei Monaten soll die Web-App von einer Person, dem Autor, fertiggestellt werden. Fertiggestellt bedeutet, eine Version der Web-App abzuliefern, die den Anforderungen entspricht. Auftraggeber ist die Erst-, und Zweitbetreuerin dieser Arbeit. Anforderungen werden primär von der Zweitbetreuerin Adelka Niels gestellt.

### 4.2 Agiles Vorgehen

Unter Berücksichtigung der Umgebungsbedingungen in Abschnitt 4.1 wird ein Vorgehen gewählt, dass möglichst leichtgewichtig ist. Die Anforderungsbesprechungen finden nur zwischen maximal drei Personen statt. Wirtschaftlich sinnvoll ist es, auf Dokumentation zu verzichten, die keinen Wert für eine der drei Parteien liefert. Dazu gehört beispielsweise das Aktualisieren von detaillierten Anforderungsbeschreibungen, obwohl diese durch direkte Kommunikation mitgeteilt wurden. Das liefern des Programms in Versionen stellt in diesem Fall eine geeignete Methode dar, Anforderungen zu verifizieren oder Anforderungsänderungen zu stellen. Das Vorgehen orientiert sich am Manifest für agile Softwareentwicklung.

#### 4.2.1 Manifest für agile Softwareentwicklung

Das agile Manifest ist unter der Webseite [Bec01] erreichbar. Es beschreibt mit vier Werten und zwölf Prinzipien einen grundsätzlich anderen Weg des Entwicklungsprozesses, als es monumentale Softwareentwicklungsprozessmodelle tun. Monumentale Prozessmodelle definieren ein großes Regelwerk mit festgelegten Details und werden deshalb auch als schwergewichtig bezeichnet [Bal08]. Agile Prozessmodelle hingegen zeichnen sich durch einen geringen bürokratischen Aufwand und wenige, anpassbare Regeln aus [Bal08]. Diese wenigen Regeln werden auf Grundlage des Manifestes erstellt. Das Manifest ist unterteilt in vier Werten und davon zwölf abgeleiteten Prinzipien. Die Werte sind:

- Individuen und Interaktionen sind wichtiger als Prozesse und Werkzeuge
- Funktionierende Software ist wichtiger als umfassende Dokumentation
- Zusammenarbeit mit dem Kunden ist wichtiger als Vertragsverhandlung
- Reagieren auf Veränderung ist wichtiger als das Befolgen eines Plans

A ist wichtiger als B bedeutet, dass B zwar wichtig ist, aber A ein noch höherer Stellenwert zukommt [Bec01]. Beispielsweise ist das Befolgen eines Plans wichtig. Möchte der Kunde jedoch eine Veränderung, ist das Befolgen zweitrangig und auf die Veränderung wird

eingegangen. Die Prinzipien und deren Umsetzung werden im nachfolgenden Kapitel 4.2.2 genauer erklärt.

### **4.2.2 Wahl des agilen Prozessmodells**

Ein agiles Prozessmodell beschreibt konkret wie sich das Manifest in einem Projekt umsetzen lässt. Dazu definiert es einen Satz von Methoden und einige wenige, anpassbare Regeln. Das agile Prozessmodell unternimmt den Versuch, einen Kompromiss zwischen keinem Prozess und zu viel Prozess zu finden [See13b]. Der Kompromiss ist abhängig vom Umfang, sowie den Ressourcen des Projektes. Unter Beachtung der Umgebungsbedingungen 4.1 wird kein Prozessmodell gewählt. Die Entwicklung orientiert sich jedoch an den Werten des Manifestes durch Einhalten der dort beschriebenen Prinzipien. Alle Prinzipien können direkt der Webseite [Pri01] entnommen werden. Nachfolgend werden diese mit kurzer Stellungnahme zur Umsetzung aufgelistet:

„Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufrieden zu stellen.“ [Pri01] Dieses Prinzip beschreibt eine Methode um die Kundenzufriedenheit zu erhöhen. Durch wiederholte und möglichst frühe Auslieferung merkt der Kunde, dass die Anforderungen seinen Wünschen entsprechend umgesetzt werden. In der Bachelorarbeit erfolgt die Auslieferung in zwei Etappen. Nach der Umsetzung und Lieferung des clientseitigen Programmteils (Kapitel 5.4) wird zum Abschluss ergänzend der serverseitige Programmteil (Kapitel 5.3) geliefert.

„Heisse [sic!] Anforderungsänderungen selbst spät in der Entwicklung willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.“ [Pri01] Während monumentale Prozessmodelle alle Anforderungen zu Beginn eines Projektes möglichst vollständig erfassen, wird im agilen Modell auf eine umfassende Anforderungsanalyse verzichtet. Stattdessen beschränkt sich die Anforderungsermittlung nur auf die notwendigen Angaben zum Erstellen der ersten lieferfähigen Version. Durch ständige Kommunikation und kontinuierliche Lieferung werden neue Anforderungen ermittelt und durchgeführt. Das Softwareprodukt wird von Anfang an auf spätere Änderungen vorbereitet, weshalb auch Änderungen spät in der Entwicklung willkommen sind. In der Bachelorarbeit reicht ein Lastenheft um die erste lieferfähige Version der Web-App zu erstellen. Die Erstellung eines Pflichtenheftes wird durch Kommunikation mit den Auftraggeberinnen vermieden.

„Liefere funktionierende Software regelmäßig innerhalb weniger Wochen oder Monate und bevorzuge dabei die kürzere Zeitspanne.“ [Pri01] Ergänzend zu dem Prinzip mit der frühen, kontinuierlichen Auslieferung, wird eine möglichst kurze Zeitspanne der Lieferung gefordert. In der Bachelorarbeit ist die Zeitspanne zwischen den Lieferungen abhängig von der Entwicklungszeit.

„Fachexperten und Entwickler müssen während des Projektes täglich zusammenarbeiten.“ [Pri01] Entwickelte Software ist immer ein Werkzeug für eine Anwendungsdomäne. Bei Fachfragen zu der Domäne sollten schnell Antworten von Fachexperten bereitstehen, um Leerlaufzeiten zu vermeiden. Die Anwendungsdomäne in der Bachelorarbeit sind Tagebuchstudien und die Fachexpertin Adelka Niels. Die tägliche Zusammenarbeit wird größtenteils durch E-Mail Kommunikation gelöst.

„Errichte Projekte rund um motivierte Individuen. Gib ihnen das Umfeld und die Unterstützung, die sie benötigen und vertraue darauf, dass sie die Aufgabe erledigen.“ [Pri01] Die Autoren gehen davon aus, dass Motivation ein wesentlicher Faktor für eine erfolgreiche Aufgabenerledigung ist. Das Arbeitsumfeld und die Unterstützung von Individuen beeinflusst die Motivation. Durch die Tatsache das Bachelorarbeiten betreut werden und das Umfeld das gewohnte Umfeld der Hochschule ist, ist dieses Prinzip erfüllt.

„Die effizienteste und effektivste Methode, Informationen an und innerhalb eines Entwicklungsteams zu übermitteln, ist im Gespräch von Angesicht zu Angesicht.“ [Pri01] Direkte Kommunikation ohne Zwischenmedium verringert die Wahrscheinlichkeit von Missverständnissen und ist deshalb vorzuziehen. In der Kommunikationswissenschaft sind mögliche Erklärungen dazu zu finden. So beschreibt Friedemann Schulz von Thun anhand seinem Vier-Seiten-Modell wie Missverständnisse entstehen können [Sch10]. In der Bachelorarbeit wird deshalb ein möglichst häufiges Treffen mit Adelka Niels angestrebt, insbesondere wenn mehrere Anliegen vorliegen.

„Funktionierende Software ist das wichtigste Fortschrittsmaß.“ [Pri01] Fortschritte werden weder an Dokumenten noch an nicht lauffähiger Software gemessen. Maßstab ist ausschließlich funktionierende Software, weil nur dies dem Kunden wichtig ist. Zum Abschluss der Bachelorarbeit soll eine funktionierende Software geliefert werden. Je mehr Anforderungen erfüllt werden, desto fortschrittlicher ist das Ergebnis.

„Agile Prozesse fördern nachhaltige Entwicklung. Die Auftraggeber, Entwickler und Benutzer sollten ein gleichmäßiges Tempo auf unbegrenzte Zeit halten können.“ [Pri01] Das Prinzip fordert eine stets gleichmäßige Arbeitsgeschwindigkeit. In der Praxis wird dies durch Vermeidung von Überstunden erreicht. Vermutlich hängt diese Forderung mit der Kreativität zusammen die ein Entwickler benötigt. Leistungsdruck ist eine Kreativitätsblockade. Übermüdung kann zu Fehlerzuständen im Code führen und beeinträchtigt die Effektivität der Entwicklung. In der Bachelorarbeit ist durch die freie Arbeitszeiteinteilung dieses Prinzip nicht relevant.

„Ständiges Augenmerk auf technische Exzellenz und gutes Design fördert Agilität.“ [Pri01] Um agil selbst auf späte Änderungswünsche eingehen zu können, muss die Softwarequalität entsprechend angemessen sein. Eine passende Verwendung von Entwurfsmustern und

Entwurfsprinzipien erleichtert die Änderbarkeit. Entwurfsmuster sind generische Lösungen von wiederkehrenden Entwurfsproblemen im Software-Entwurf [Gol13]. Ein Entwurfsmuster beschreibt wie ein Entwurfsproblem den Entwurfsprinzipien folgend, ideal gelöst werden kann. Entwurfsprinzipien können als Richtlinien im Software-Entwurf angesehen werden. In der Bachelorarbeit werden Entwurfsmuster und Entwurfsprinzipien an geeigneten Stellen gewinnbringend genutzt.

„Einfachheit -- die Kunst, die Menge nicht getaner Arbeit zu maximieren -- ist essenziell.“ [Pri01] Neben technischer Exzellenz und gutem Design sollte auf Einfachheit Wert gelegt werden. Dieses Prinzip plädiert zur Effizienz bezüglich der Zielerreichung. Effizienz durch Einfachheit gibt es auf zwei Weisen.

- Ersteres ist Einfachheit bezogen auf die Entwicklung. Dies bedeutet, Technologie nicht auf Vorrat zu bauen [Wol11]. Beispielsweise sollte ein Entwurfsmuster nur dann eingesetzt werden, wenn es der Problemstellung wirklich dienlich ist. Es sollte nicht eingesetzt werden um auf möglicherweise zukünftige Erweiterungen zu reagieren. Darüber hinaus wird Einfachheit in der Entwicklung gefördert durch Nutzung bestehender Frameworks<sup>1</sup>, um die Eigenentwicklung möglichst einzuschränken.
- Zweitens wird das Prinzip der Einfachheit auf den Entwicklungsprozess selbst bezogen [Wol11]. Dokumentation und Planung sind für den Kunden keine wertbringenden Tätigkeiten und sollten auf das nötigste reduziert werden. Beispielsweise wird in der Bachelorarbeit auf ein Pflichtenheft verzichtet. Werden dennoch die Wünsche vom Auftraggeber erfüllt, wurde effizienter gearbeitet als mit Pflichtenheft.

„Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams.“ [Pri01] Teams die sich in der Regel innerhalb eines vom Management festgelegten Rahmens bezüglich Rollen-, Aufgaben- und Zeitverteilung selbst organisieren, sind (so behauptet das Prinzip) produktiver. Vermutlich hängt dies mit der Selbstbestimmungstheorie der Motivation zusammen [sel14]. In der Bachelorarbeit ist durch das Nichtvorhanden sein eines Teams dieses Prinzip nicht relevant.

„In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann und passt sein Verhalten entsprechend an.“ [Pri01] Kontinuierliche Verbesserung in der Effektivität ist durch reflektieren des Prozessvorgangs und entsprechenden Anpassungen an diesem Vorgang möglich. In der Bachelorarbeit wird das reflektieren aufgrund des Fehlens eines Teams

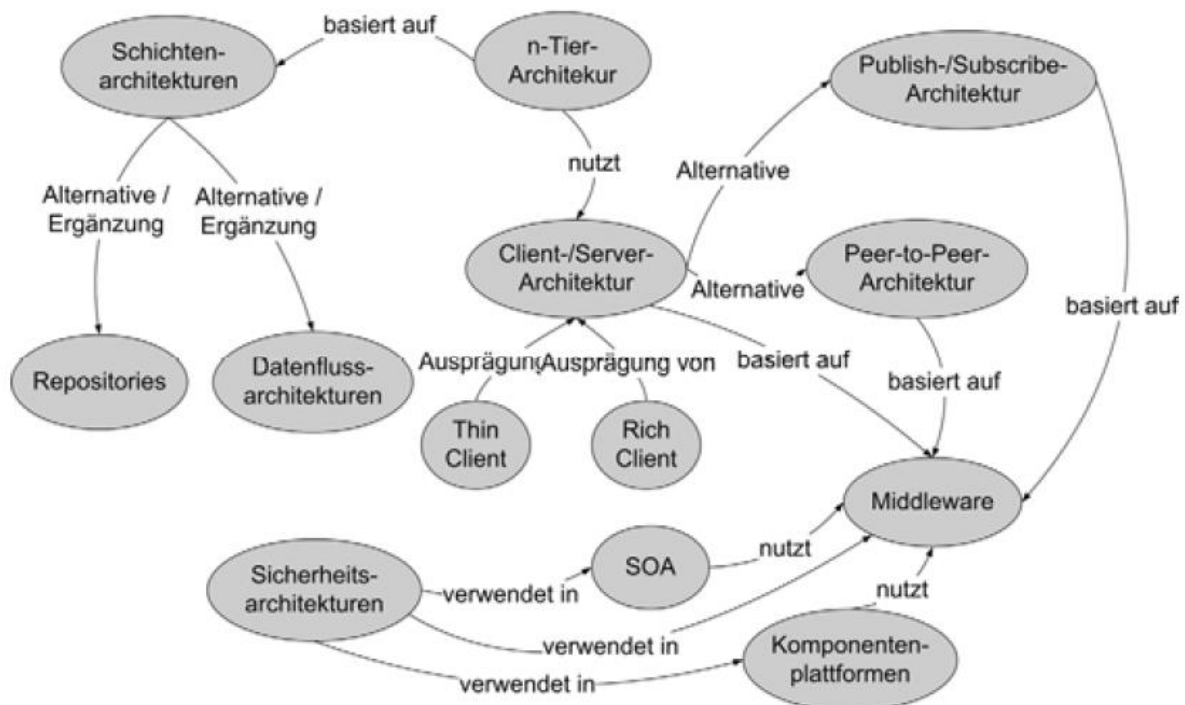
---

<sup>1</sup> Ein Framework ist ein Programmiergerüst, welches wiederverwendbare Teile eines Anwendungsbereichs implementiert [Cwa08]. Ein Entwickler kann das Programmiergerüst nutzen, um die anwendungsspezifischen Teile darin auszuprogrammieren.

erschwert. Es wird allein auf Grundlage von Rückmeldungen seitens der Betreuer, sowie dem eigenem Zeitmanagement reflektiert.

## 5 Entwurf

Zur technischen Umsetzung wurde das Client-Server-Modell als Basisarchitektur gewählt. Da es sich um eine Webanwendung zur Erhebung von Daten handelt und diese Daten einheitlich gesammelt werden sollen, wird ein Server als zentrales und immer präsenten Programm im Internet benötigt. Des Weiteren wird wegen der Forderung, die Webanwendung ohne Installationsaufwand im Browser laufen zu lassen, ein Webserver benötigt. Ein Peer-To-Peer-Modell, bei dem jede beteiligte Anwendung im Netzwerk gleichberechtigt ist, ist aus diesem Grund nicht sinnvoll. Auch ein Vermittler wie in der Publish-/Subscribe-Architektur beschrieben, der Nachrichten als Ereignisse interpretiert und weiterleitet, bringt keine Vorteile.



**Abbildung 7:** Überblick Basisarchitekturen, aus [See13]

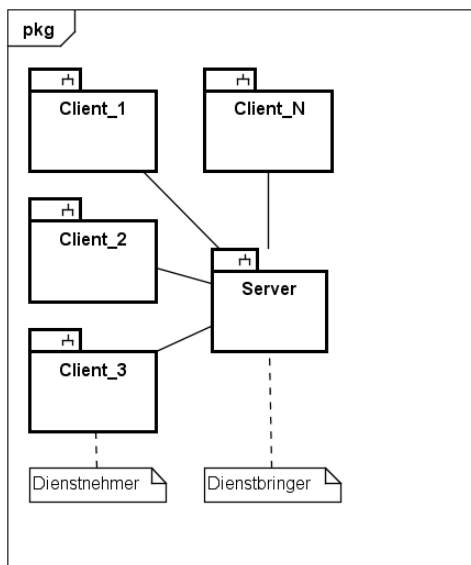
In den nachfolgenden Kapiteln wird auf das Client-Server-Modell detaillierter eingegangen. Es wird beschrieben, wie Client und Server miteinander kommunizieren und wie sich die Programmteile der Web-App auf Client und Server verteilen. Die Programmteile selbst sind gemäß dem Prinzip Trennung der Anliegen modular strukturiert [Lah09]. Ein Modul kann als ein für sich abgeschlossener Programmbaustein verstanden werden. Es übernimmt eine genau abgegrenzte Aufgabe des Gesamtsystems und bietet eine Schnittstelle zur Kommunikation mit anderen Modulen im Gesamtsystem an [Lah09].

Die Entwurfsbeschreibungen entsprechen dem aktuellen Stand. Sie sind entsprechend dem Vorgehensmodell evolutionär entstanden.

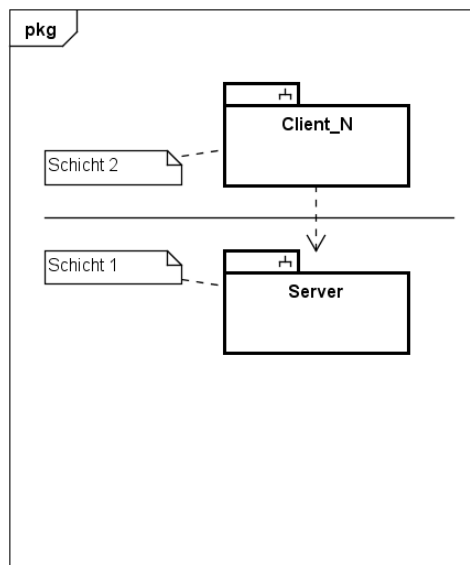
## 5.1 Client-Server-Modell

Das klassische Client-Server-Modell basiert auf einer 2-Schichten-Architektur. Eine Schicht ist ein Modul mit speziellen Eigenschaften, die im Architekturmuster Schichten beschrieben ist. Das Muster zeigt auf, wie eine hierarchische Aufteilung des Programmcodes in logische Ebenen (Schichten) die Abhängigkeitsbeziehungen einschränkt. Kombiniert mit dem Prinzip Trennung der Anliegen (engl. Separation of Concerns) wird der Code in logisch zusammengehörige Module aufgeteilt [Lah09] und so angeordnet das eine Hierarchie zwischen ihnen entsteht. Die Hierarchie zeigt sich dadurch, dass ein Modul in der höheren Hierarchieebene abhängig von dem Modul in der unmittelbar darunter liegende Ebene ist, während der umgekehrte Fall nicht gilt. Das Modul in der unteren Ebene fungiert als Dienstbringer für die höherliegende Ebene und ist deshalb unabhängig von der höheren Ebene.

In der Regel hat der Server die Rolle des Dienstbringers, während der Client der Dienstnehmer ist und die Kommunikation initiiert. Ohne Dienstbringer bekommt der Dienstnehmer nicht den gewünschten Dienst, deshalb ist er von ihm abhängig.



**Abbildung 8:** Während der Client der Dienstnehmer ist, ist der Server der Dienstbringer.



**Abbildung 9:** Der Client ist abhängig vom Server, der Server aber nicht vom Client.

Der Serverdienst: „Web-App für Tagebuchstudien“ ist der zu entwickelnde Teil dieser Bachelorarbeit. Die Clients sind die Webbrowser auf den Endgeräten der Anwender.

Der Dienst ist aufgeteilt in zwei Module: Die serverseitige Modul und das clientseitige Modul. Wenn der Client den Dienst anfordert, bekommt er zuerst nur das clientseitige Modul geliefert. Dieses Modul ist eine Webseite, bestehend aus HTML, CSS und JavaScript. Im weiteren Verlauf kommuniziert der Client (Webbrowser) durch das clientseitige Modul (Webseite) mit dem Modul, das auf dem Server läuft. Somit ist der Server zwar abhängig von der Kommunikation mit der Webseite, aber nicht abhängig vom Client selbst. An dieser Stelle sei gesagt, dass die Abhängigkeit durch Verwendung von standardisierten Kommunikationsmechanismen wie z.B. HTTP erheblich reduziert werden kann.

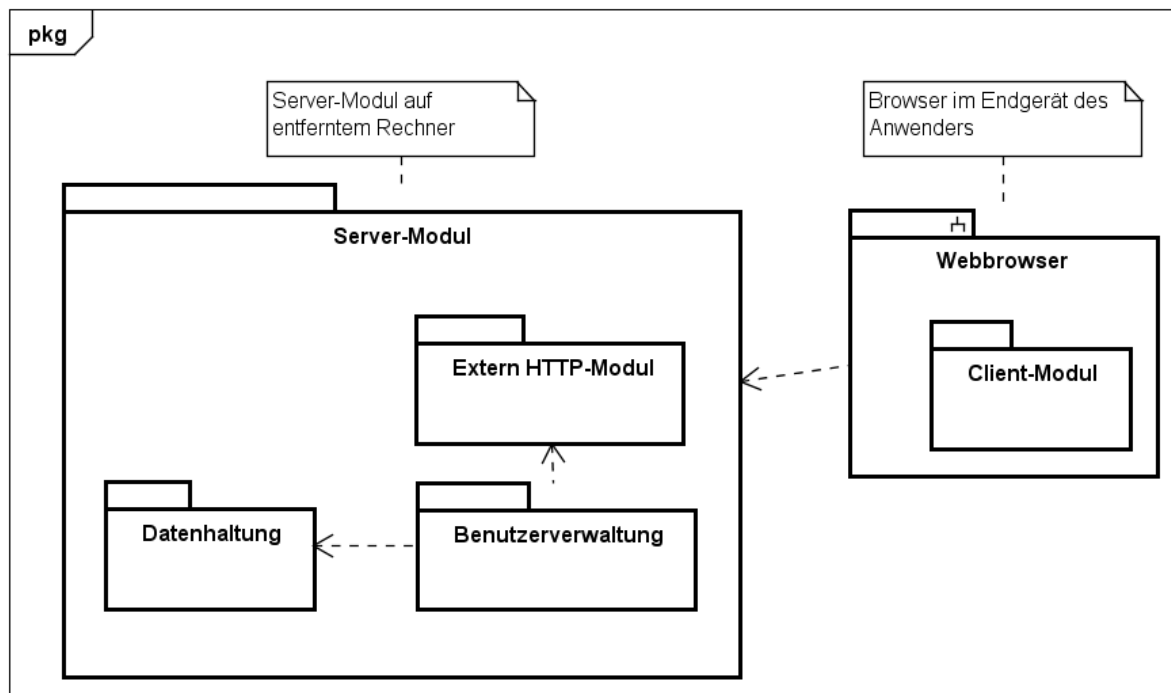
## **5.2 Client-Server-Kommunikation**

Aus der Anforderung geht plausibel hervor, dass es sich um ein verteiltes System handelt. Client und Server laufen nicht bloß in verschiedenen Prozessen, sondern auf verschiedenen Computern. Demnach muss ein Netzwerkprotokoll die Kommunikation regeln. Bei einer Webanwendung erübrigt sich die Wahl eines solchen Protokolls. Ein WWW-Client (World Wide Web - Client, kurz Web-Client) und ein WWW-Server kommunizieren über HTTP (Hypertext Transfer Protocol) seit 1990 [Fie99]. Die Hauptaufgabe eines Webbrowsers ist es, Webseiten darzustellen. Daher ist anzunehmen, dass jeder Webbrowser HTTP implementiert hat. Soll die Webanwendung ohne Installationen auskommen, ist es nicht nur unnötig, sondern sogar unmöglich ein eigenes Protokoll für diese Webanwendung zu entwickeln. Jeder Client (Smartphone, Tablet, Desktop-Computer), welcher mit dem Server kommunizieren will muss dieses Protokoll bereits vorher installiert haben. Für die Entwicklung wird deshalb HTTP in der neusten Version 1.1 verwendet. Ein Vergleich zu Version 1.0 findet sich unter [Fie99] Kapitel 19.6.1.

## **5.3 Serverseitiges Modul**

Das serverseitige Modul ist der Teil der Webanwendung, der ausschließlich auf dem Server läuft. Auslieferung des clientseitigen Moduls (Webseite), Datenhaltung und Kommunikation zwischen Webseite und Datenhaltung sind die wesentlichen Aufgaben des serverseitigen Moduls. Des Weiteren kommt noch eine Benutzerverwaltung hinzu, die die Daten validiert und den einzelnen Benutzern zuordnet. Vorteilhaft ist, dass die Kommunikation auf einem Standard basiert (HTTP, siehe Abschnitt 5.2). Es existieren diverse Softwaremodule, die die Kommunikationsaufgabe inklusive Auslieferung der Webseite basierend auf HTTP bereits

implementiert haben<sup>2</sup>. Eine Eigenentwicklung des HTTP-Kommunikationsmodul entfällt somit.



**Abbildung 10:** Illustriert den Aufbau des Server-Moduls.

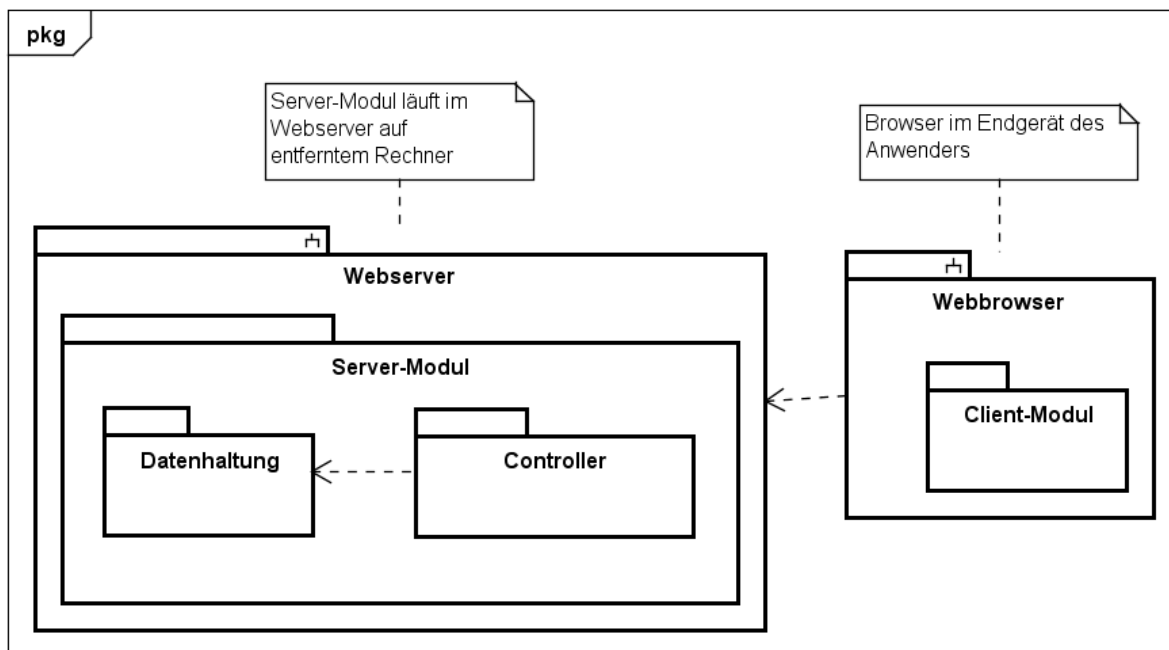
Darüber hinaus stehen fertige Systeme zu verschiedenen Lizenzen zur Verfügung, die mehr können als bloß in Webanwendungen integriert zu werden<sup>3</sup>. Vielmehr werden die Webanwendungen als Module in ein solches System integriert. Diese werden meist in Form eines Servers geliefert, um direkt als Dienstbringer für Clients zu fungieren<sup>4</sup>. Die Kommunikation mit dem Client wird vollständig abstrahiert. Sitzungsmechanismen für die Benutzerverwaltung, Verschlüsselung und diverse andere Funktionserweiterungen werden angeboten. Die zu schreibende Codemenge reduziert sich, falls auf solche Erweiterungen Wert gelegt werden muss. Zudem konzentriert sich die Entwicklung dann auf die wesentlichen, spezifischen Aufgaben der Webanwendung. Aus diesen Gründen wird die Verwendung eines Webservers, in dem Webanwendungen als Module integriert werden, Vorzug gegeben.

<sup>2</sup> Beispiele für solche Module sind GNU Libmicrohttpd, Monkey HTTP Server oder NanoHttpd.

<sup>3</sup> Beispiele für solche Systeme sind Apache HTTP Server, Jetty oder Microsoft Internet Information Services (IIS).

<sup>4</sup> Server die Webseiten im WWW anbieten werden auch als Webserver bezeichnet.





**Abbildung 11:** Ein Modul namens Controller validiert die Anfragen.

Wie in Abbildung 11 gezeigt, fällt das Modul Benutzerverwaltung durch Nutzung eines Webserver weg. Stattdessen wird ein Modul namens Controller modelliert. Dieses ersetzt keineswegs die Benutzerverwaltung. Es enthält nur den Teil, den der Webserver nicht übernommen hat oder übernehmen kann, weil dieser zu anwendungsspezifisch ist. Jedenfalls hängt dieser Teil ab von den funktionalen Erweiterungen, die dem Webserver mitgegeben werden. In der Web-App hat das Modul Controller die Aufgabe eingehende Client-Anfragen zu validieren, um diese anschließend an die Datenhaltung weiterzugeben. Die Zuordnung der Anfragen zu Benutzern anhand von Sitzungen übernimmt der Webserver.

Während Module eines Systems meist auf eine Datenhaltung angewiesen sind, ist die Datenhaltung in der Regel nicht auf die Module angewiesen. **Abbildung 11** verdeutlicht dies am Entwurf der Web-App. Der Controller (ein Modul) ist abhängig von der Datenhaltung, aber die Datenhaltung nicht vom Controller. Insgesamt ist so eine 3-Schichten-Architektur im Client-Server-Modell entstanden.

Die Namen „Client-Modul“ und „Server-Modul“ wurden bisher zum besseren Verständnis der Verteilung des Programms auf Webserver und Webbrowser gewählt. Um den Inhalt der einzelnen Module genauer zu beschreiben, werden geeignetere Namen bestimmt. Die Namenswahl nach dem Inhalt stellt die einzige Verantwortung des Moduls besser dar (Prinzip der einzigen Verantwortung [Lah09]). Das Client-Modul enthält die Webseite, umgesetzt mit vom Browser interpretierenden Sprachen (HTML, CSS, JavaScript). Das Modul wird in „WebContent“ umbenannt. Das Server-Modul enthält die Applikation zu der Webseite und wird deshalb kurz „App“ genannt. Der Name Controller wurde in Anlehnung an das

Entwurfsmuster Model-View-Controller gewählt [Gol13]. Das Modul Datenhaltung wird zur Vereinheitlichung in „Model“ umbenannt.

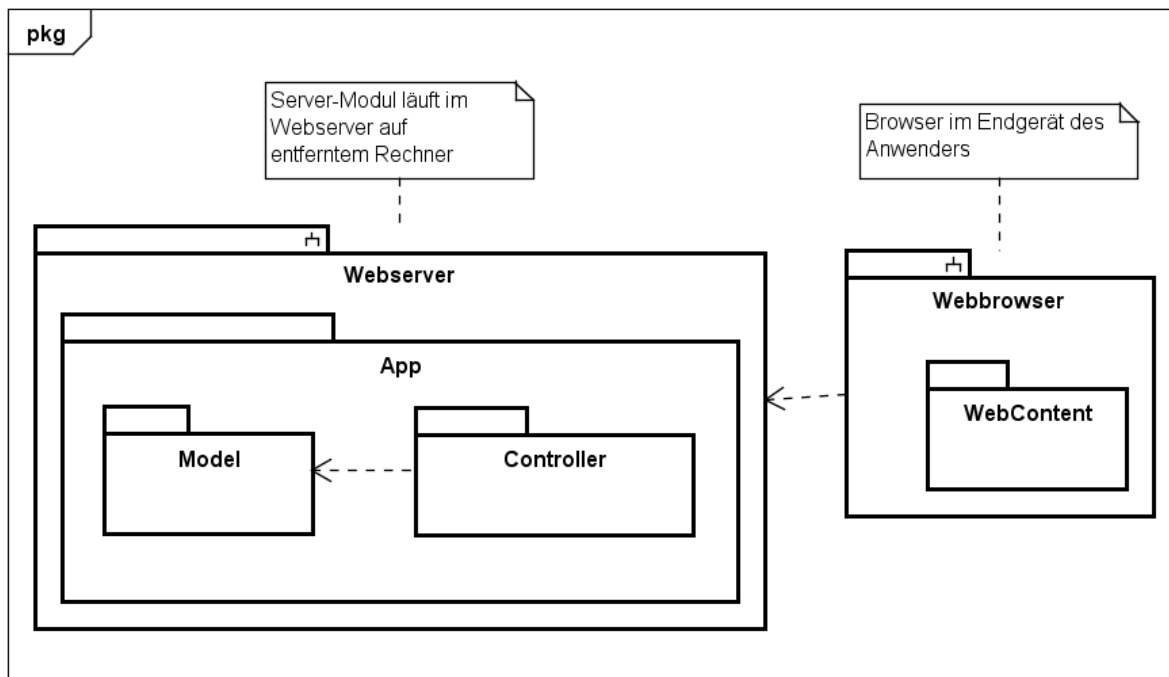


Abbildung 12: Verdeutlicht die endgültige Namensgebung der Module

### 5.3.1 Datenhaltung

Die Datenhaltung ist ein wichtiger Teil der Web-App. Alle Antworten der Probanden der Studie müssen persistent gespeichert werden. Ein Abgleich der Anforderung liefert die zu speichernden Studiendaten. Darüber hinaus müssen Daten für die Benutzererkennung gehalten werden.

#### 5.3.1.1 Standardisiertes Datenbankschema

Jede Anwendung kann ihre Daten direkt auf Dateien des Dateisystems speichern. Beispielsweise könnte eine einfache Textdatei als Datenbank benutzt werden. Damit die einzelnen Datenobjekte voneinander unterschieden werden können, müssen Metadaten hinzugefügt werden. Metadaten sind Daten für die Strukturierung von Nutzdaten [Gei09]. Beispielsweise könnten die Datenobjekte mit Leerzeichen als Trennzeichen gespeichert werden. Die Trennzeichen sind dann die Metadaten. Die Metadaten zusammengekommen werden auch als Datenbankschema bezeichnet [Kem09]. Die Nutzung eines selbst entworfenen Datenbankschemas konfrontiert den Programmierer mit der Aufgabe, einen für diese Datenbank spezifischen Parser zu schreiben der das Schema versteht. Als Parser wird ein Programm bezeichnet das Eingaben einliest und diese für eine Weiterverarbeitung vorbereitet. In diesem Kontext liegt die Vorbereitung darin, die gelesenen Nutzdaten aus den gelesenen Metadaten zu filtern und nur die Nutzdaten zu präsentieren. Der Anforderung zufolge sind nicht nur Schreiboperationen nötig. Es wird eine Benutzerverwaltung gefordert, woraus resultiert,

dass nach den Daten Benutzername, E-Mail-Adresse, Anzahl der Erfolgs- und Problemeinträge, Passwort und Daten zum Passwort zurücksetzen gesucht werden muss. Einen eigenen Parser zu schreiben, der diesen Anforderungen gerecht wird, nimmt viel Entwicklungszeit in Anspruch.

Abgesehen von dem Arbeitsaufwand, geht damit eine Abhängigkeit der Anwendung von dem individuellen Schema einher. Es existieren diverse standardisierte Datenbankschemata mit zugehörigen Parsern, die dieses Problem lösen. Wird ein standardisiertes Datenbankschema, wie beispielsweise das relationale Schema oder die Extensible Markup Language (Abk. XML) zur Speicherung herangezogen, kann die Anwendungsentwicklung durch Einsetzen bereits existierender Parser wesentlich verkürzt werden. Der Parser selbst, kann auch bei Bedarf durch einen anderen ersetzt werden. Generell ist das Verwenden von Standards in der Softwareentwicklung der Austauschbarkeit und damit der Wartbarkeit förderlich.

Der Anforderung kann entnommen werden, dass die Web-App auch für andere Studien wiederverwendet werden könnte. Bei Anpassung einer neuen Studie ist davon auszugehen, dass sich die zu speichernde Datenstruktur ändern wird. Demzufolge muss dann auch der Parser angepasst werden (falls dieser nicht aufwendig austauschbar programmiert wurde).

Neben dem Parser wird weitere Funktionalität benötigt, welche die Datenkonsistenz (Korrektheit der Daten) auch im Mehrbenutzerbetrieb gewährleistet. Die Web-App ist auf Mehrbenutzerbetrieb angewiesen. Wenn zwei Clients dieselbe Datei gleichzeitig verändern wollen, darf der Datenbestand nicht in einen inkonsistenten Zustand geraten. Die Web-App muss zu jedem Zeitpunkt bereit sein, mehrere Datenabfragen von unterschiedlichen Clients zu verarbeiten.

Neben der Mehrbenutzerkontrolle müssen auch Transaktionsmechanismen implementiert werden. Eine Transaktion ist eine Folge von Operationen, die entweder ganz oder gar nicht ausgeführt wird [Kem09]. Bricht die Abarbeitungsfolge vor Ende der letzten Operation durch Aufkommen einer Fehlerwirkung (Auswirkungen eines Fehlerzustands) oder Programmabsturz ab, folgt ein Rollback (der Datenbestand nimmt wieder die ursprüngliche Ausprägung an). Transaktionsmechanismen sichern die Datenkonsistenz und Datenkonsistenz hat einen hohen Stellenwert bei Studiendaten.

Aus diesen Gründen ist einer Eigenentwicklung mit einhergehender Abhängigkeit von anwendungsspezifischen Schemata abzuraten. Stattdessen wird ein standardisiertes Datenbankschema mit einem Datenbankverwaltungssystem (aus Fremdentwicklung) genutzt.

Eine Datenbank hat immer ein Datenbankverwaltungssystem (engl. database management system, abgekürzt DBMS). DBMS ist ein Programm (oder sind mehrere Programme [Kem09]), welches die Verwaltung einer oder mehrerer Datenbanken übernimmt. In der Regel kann es als

ein Modul in das Softwareprodukt integriert werden oder als eigenständiges System (Subsystem) mit dem Softwareprodukt kommunizieren. Im letzteren Fall läuft das DBMS in einem eigenen Prozess. Dann wird vom DBMS ein Protokoll bereitgestellt, welches die Interprozesskommunikation zwischen dem Softwareprodukt und dem DBMS regelt. DBMS implementieren in der Regel Mehrbenutzerkontrolle und Transaktionsmechanismen. Zudem bieten sie eine standardisierte Programmierschnittstelle in Form einer Datenbanksprache [Kem09].

### 5.3.1.2 Datenbankentwurf

Als Datenbankschema wird das weit verbreitete relationale Schema gewählt. Es ist klar zu erkennen, dass fast alle Relationen (Tabellen) von der Relation Benutzer ausgehen. Die Relationen und ihre Beziehungen zueinander veranschaulicht die Abbildung 13: *Illustriert den Datenbankentwurf in der Web-App.*



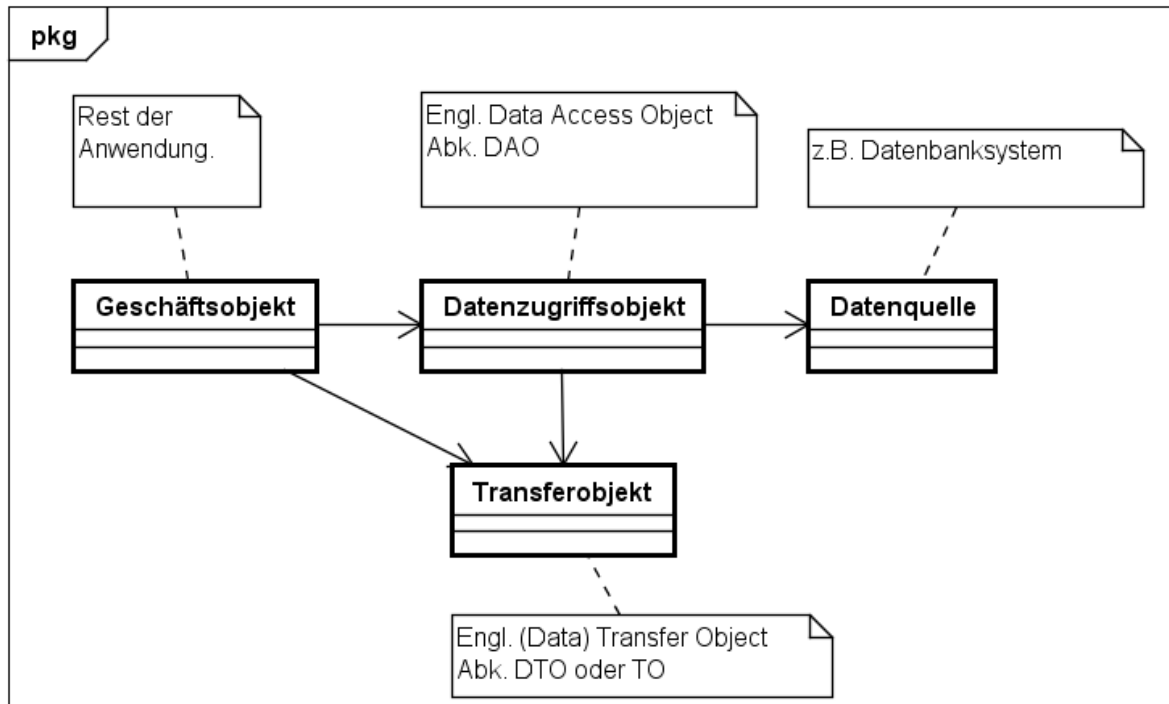
**Abbildung 13:** *Illustriert den Datenbankentwurf in der Web-App*

Die Relation Benutzer wurde aus Effizienzgründen von den Relationen Registrierung und Fragebogen getrennt. Diese Relation ist hochfrequentiert mit zugriffen von Clients. Denn bei jeder Anmeldung oder Registrierung wird auf diese zugegriffen. Hingegen werden die Relationen Registrierung und Fragebogen nur wenige Male angefordert. Ein anderer Grund ist die Einfachheit. Während die Relation Benutzer alle Aspekte einer Benutzerverwaltung

beinhaltet, existieren die anderen Relationen nur zum Speichern von Studiendaten. Diese Aufteilung zeigt sich auch in einer Vereinfachung des Programmcodes. Zu jeder Relation existiert eine Programmeinheit (z.B. Klasse, Modul) die die Daten des Clients zu genau dieser Relation speichert, nur zur Relation Benutzer gibt es keine solche Programmeinheit. Die Relationen Erfolg und Problem haben nach dem aktuellen Stand identische Attribute. Ursprünglich waren diese unterschiedlich, weshalb für Erfolg- und Problemeinträge jeweils eigene Relationen entworfen wurden. Würden beide Relationen nun zu einer modelliert werden, könnte auf späte Änderungen bezüglich der Problemrelevanten Attribute oder Erfolgsrelevanten Attribute nicht schnell reagiert werden.

### **5.3.1.3 Objektrelationale Abbildung**

Falls ein in objektorientierter Sprache geschriebenes System eine relationale Datenbank nutzt, müssen Objektattribute in Relationen gespeichert werden. Bei der Entwicklung der Web-App trifft genau dieser Fall zu. Kapitel 6.1.1 beschreibt Gründe für die Wahl einer objektorientierten Sprache. Die Entscheidung, ob hierfür ein Framework hinzugezogen oder die objektrelationale Abbildung selbst programmiert wird, hängt von der Komplexität der Aufgabenstellung ab. Der Arbeitsaufwand ein Rahmenwerk einzubinden sollte nicht höher sein, als die native Implementierung. Eine Ausnahme bildet das Effizienzkriterium, welches jedoch für kleine Projekte irrelevant ist. Der Arbeitsaufwand hängt unmittelbar von der Anzahl der Relationen und deren Beziehungen, sowie den Fähigkeiten des Entwicklers ab. Die Abbildung 13 zeigt klar, dass nur wenige Relationen nötig sind. Deshalb wird auf ein Framework verzichtet. Die native Implementierung wird auf Grundlage des Entwurfsmusters Data Access Objekt (Abk. DAO) realisiert. Das Entwurfsmuster eignet sich ideal für Datenbankentwürfe mit wenigen Relationen [Ora141].



**Abbildung 14:** Aufbau des Entwurfsmusters DAO

## 5.4 Clientseitiges Modul

Das clientseitige Modul ist der Teil der Webanwendung, der im Webbrowser des Anwenders läuft. Aufgabe des Moduls ist es, die Benutzerschnittstelle in Form von Webseiten bereitzustellen. Eine Webseite wird in einer separaten HTML-Datei beschrieben [W3C14]. Jede einzelne Datei wird über eine HTTP-Anfrage seitens des Browsers angefordert [Fie99] und anschließend im Browser interpretiert. Falls die gesamte Benutzerschnittstelle in mehrere Dateien gegliedert ist, muss jede von ihnen übers Netzwerk transportiert werden. Die Effizienz der Webanwendung hängt somit unmittelbar von der Anzahl der Dateien der Benutzerschnittstelle zusammen. Deshalb ist es wichtig, die Implementation des Moduls so zu gestalten, dass eine minimale Anzahl von Dateien benötigt wird. Clientseitigen Skriptsprachen bieten diese Möglichkeit. Es wird nur eine HTML-Datei geschrieben, in der mehrere Webseiten definiert sind. Ein zusätzliches Skript sorgt für die richtige Interpretation und Darstellung der einzelnen Seiten im Browser. Solch eine Webseite nennt man auch Single Page Application (Abk. SPA) [Mik14]. Die Benutzerschnittstelle der Web-App ist in mehrere SPAs aufgeteilt. Teile der Benutzerschnittstelle die selten oder nur einmalig aufgerufen werden, sollten insbesondere für mobilen Endgeräte, als separate SPA realisiert sein. Konkret für die Web-App heißt das, dass die Seiten für die Registrierung und die Seiten des Fragebogens (Selbstkonzept) als eine in sich geschlossene SPA realisiert werden. Auch die Webseite des Administrators braucht nicht in den Browser eines Probanden geladen werden.

## 6 Technologien

In diesem Kapitel wird die Wahl von Programmiersprachen, Frameworks und Systemen erklärt. Grundsätzlich gibt es mehrere Technologien, mit denen sich das gleiche Ergebnis realisieren lässt. Unter Berücksichtigung bestimmter Kriterien lässt sich die Wahl einschränken. Im weiteren Verlauf wird anschließend ein pragmatischer Ansatz gewählt.

Ein wesentliches Kriterium ist die Einfachheit. Die Entwicklungskosten dürfen nur so hoch sein, dass das Projektergebnis in Anbetracht der Umgebungsbedingungen die Anforderung erfüllt. Die Entwicklungskosten sind hoch, wenn der Entwickler eine Technologie erst erlernen muss. Zukünftige Weiterentwicklungen, bei denen der Erstautor nicht mehr involviert ist, müssen auch berücksichtigt werden. Einer modernen, geläufigen Technologie wird Vorzug gegeben.

Finanzieller Aufwand ist ein weiteres Kriterium. Finanzielle Aufwände sind im Rahmen dieser Bachelorarbeit zu vermeiden. Während beispielsweise Technologien von Microsoft oft nur auf kommerziellen Webservern von Microsoft laufen, sind kostenfreie Webserver für Java (z.B. [Apa141]) oder PHP (z.B. [Apa14]) im Umlauf. Mittlerweile gibt es auch freie Webserver für die Microsoft Technologie ASP.NET [Ult14]. Im Vergleich zu den freien Webservern für Java oder PHP sind diese jedoch wenig im Produktiveinsatz vertreten [QSu14].

### 6.1 Serverseitiges Modul

Für das serverseitige Modul müssen Programmiersprache, Webserver und Datenbankserver gewählt werden.

#### 6.1.1 Wahl der Programmiersprache

Es muss eine Programmiersprache für die Realisierung des serverseitigen Moduls gewählt werden. Zusätzlich zu den Kriterien Einfachheit und finanziellen Aufwand wird noch Objektorientierung hinzugezogen. Objektorientierung fördert die Wartbar- und Erweiterbarkeit. Während Objektorientierte Sprachen Daten und deren Verarbeitung zu Klassen gruppieren, werden im prozeduralen Programmierparadigma<sup>5</sup> Daten und deren Verarbeitung getrennt. Die Trennung bewirkt, dass es keinen Schutz vor unberechtigtem Datenzugriff durch Prozeduren gibt [Lah09]. Folglich ist eine Datenkonsistenz nicht durchweg gewährleistet. Neben der Datenkapselung in Klassen, bieten objektorientierte Sprachen noch Vererbung und Polymorphie an [Lah09]. Werden diese Techniken gemäß den

---

<sup>5</sup> Ein Programmierparadigma beschreibt die grundlegende Art und Weise, also unter welchen Regeln (Syntax) programmiert werden kann. Es gibt dem Programmierer dadurch eine Sichtweise zur Lösung eines Problems vor [Gra03].

objektorientierten Entwurfsprinzipien eingesetzt, werden Wartbar- und Erweiterbarkeit erhöht. Das Kriterium Effizienz ist bei kleinen Projekten nicht relevant.

Aktuell genutzte Programmiersprachen, bei denen wahrscheinlicher ist, dass Nachfolgeautoren diese beherrschen, können aktuellen Studien entnommen werden. Die Studie [hei14] zeigt auf, dass .NET, Java, ASP und PHP mit jeweils 28, 25, 16 und 11 Prozent die vier meist genutzten Programmiersprachen im Webbereich sind.

Dem Kriterium des finanziellen Aufwands entsprechend, werden Technologien des Unternehmens Microsoft ausgeschlossen. Zur Auswahl stehen folglich PHP und Java. Während Java eine rein objektorientierte Sprache ist, kann mit PHP auch nicht objektorientiert programmiert werden. Das Einhalten des objektorientierten Paradigmas könnte verletzt werden und die Softwarequalität beeinträchtigen. Weiterhin sind die Kenntnisse und Fähigkeiten des Autors vielmehr im Java-Umfeld vertreten. Aus diesen Gründen fällt die Wahl auf Java.

Java bietet mit der Servlet Programmierschnittstelle (engl. application programming interface, Abk. API) eine Möglichkeit, HTTP-Anfragen von Clients entgegenzunehmen und HTTP-Antworten zu verschicken. Die Kommunikation mit dem Client erfolgt indirekt durch einen Servlet-Container. Der Servlet-Container ist ein Softwaremodul und Teil der Servlet API. Während der Servlet-Container vom Webserver bereitgestellt wird, enthalten die Servlets die anwendungsspezifischen Details und müssen vom Entwickler programmiert werden. Servlets sind als Programmelemente (Klassen) im Modul Controller zu finden.

### **6.1.2 Wahl des Webservers**

Die Wahl des Webservers ist abhängig von der Wahl der Programmiersprache und umgekehrt. Beispielsweise akzeptiert Microsoft IIS ursprünglich nur Webanwendungen geschrieben in PHP oder auf .NET-Basis [Mic13]. Sicherlich gibt es durch Umwege wie Übersetzerprogramme andere Möglichkeiten. Auf der Firmen-Webseite von Oracle steht, dass Java Servlets mit einem geeigneten Servlet-Container von Drittanbietern im Microsoft IIS laufen können [Ora14]. Legacy Systeme<sup>6</sup> oder generell Systeme die auf die Konstellation von Microsoft IIS und Java Servlets angewiesen sind, finden so eine Lösung. Empfehlenswert ist dies bei neu zu entwickelten Webanwendungen wegen dem zusätzlichen Arbeitsschritt jedoch nicht.

---

<sup>6</sup> Legacy Systeme werden in diesem Zusammenhang als Softwaresysteme mit viel Legacy Code bezeichnet. Legacy Code ist unstrukturierter, undurchdringlicher Programmcode [Fea11].



Die Programmiersprachenwahl fällt auf Java mit der Servlet API und folglich wird ein Webserver gesucht, der einen Servlet-Container implementiert. Pragmatisch wird der Open Source Webserver Apache Tomcat (zu finden unter [Apa141]) gewählt. Andere Webserver wie beispielsweise Jetty (zu finden unter [The14]) könnten genauso eingesetzt werden. An dieser Stelle sei erwähnt, dass durch Nutzung der standardisierten Java Servlet API jeder beliebiger Webserver mit einem Servlet-Container verwendet werden kann, ohne eine einzige Zeile Code zu schreiben.

### **6.1.3 Wahl des Datenbankservers**

Die Wahl des Datenbankservers orientiert sich in erster Linie an der Leichtgewichtigkeit. Entsprechend dem Umfang der zu speichernden Daten wird ein Datenbankserver gewählt der nur die notwendigsten Eigenschaften aufweist. Notwendige Eigenschaften beschränken sich auf Transaktions- und Mehrbenutzerkontrolle sowie CSV-Exportierungsfunktionen. Ein leichtgewichtiges Open Source Datenbanksystem, welches sogar eingebettet in Java-Anwendungen läuft, ist die HyperSQL Database (Abk. HSQLDB) [The142]. Die Wahl fällt pragmatischer Weise auf HSQLDB, da der Autor bereits umfangreiche Kenntnisse dieses Systems mitbringt.

## 7 Anhang

Beiliegend ist eine DVD mit folgenden Inhalten:

Ordner	Inhalt
1.1 Dokumentation	Enthält dieses Dokument als Word- und PDF-Datei, sowie alle verwendeten Abbildungen inklusive der Programmdateien (mit denen die Abbildungen erstellt wurden). Darüber hinaus sind alle Quellen, mit Ausnahme der kommerziellen Literatur, in diesem Ordner zu finden.
1.2 Anforderungen	Enthält alle Anforderungsdokumente.
1.3 Server Installation	<p>Enthält die virtuelle Maschine, verpackt in komprimierten Dateien. Diese ist schon mit dem Web- und Datenbankserver fertig installiert und einsatzfähig.</p> <p>Auch die Installationsdateien vom Debian OS 6.0.10 sind hier zu finden. Auf diesem Betriebssystem laufen letztendlich der Webserver mit der Web-App und der Datenbankserver.</p> <p>Wie die Installation und Konfiguration des Webserver auf Debian durchgeführt wird, ist in einem beiliegendem Dokument beschrieben.</p>
1.4 Software	Enthält den Programmcode der Web-App, welcher nach dem Selbstdokumentationsprinzip gestaltet ist. Gemäß diesem Prinzip wird die Dokumentation des Moduls direkt dem Code des Moduls hinzugefügt, damit die Dokumentationsänderungen bei Programmänderungen nicht versehentlich vergessen werden [Vog08]. Dokumentationswerkzeug ist Javadoc. In einem Unterordner findet sich die Javadoc-Dokumentation zusätzlich separat (automatisch generiert) als HTML-Version.

## Literaturverzeichnis

- [Apa14] The Apache Software Foundation. (2014) Apache HTTP Server Project. [Online]. <http://httpd.apache.org/>
- [Bal08] Helmut Balzert, *Lehrbuch der Softwaretechnik. Softwaremanagement*, 2nd ed.: Spektrum Akademischer Verlag Heidelberg, 2008.
- [Bec01] Kent Beck et al. (2001) Manifest für Agile Softwareentwicklung. [Online]. <http://agilemanifesto.org/iso/de/>
- [Car14] Carola Lopez. (2014, Mai) BVDW. [Online]. <http://www.bvdw.org/medien/online-nutzung-durch-mobile-endgeraete-deutlich-gestiegen?media=5728>
- [Cwa08] Krzysztof Cwalina and Brad Abrams, *Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .NET Libraries (Microsoft .Net Development)*, 2nd ed.: Addison Wesley, 2008.
- [See13b] Silke Seehusen, "Agile Softwareentwicklung," Fachhochschule Lübeck, Vorlesungsskript WS 2012/13.
- [Fea11] Michael Feathers, *Effektives Arbeiten mit Legacy Code. Refactoring und Testen bestehender Software*, 1st ed.: mitp, 2011.
- [Fie99] R. Fielding et al. (1999, Juni) Hypertext Transfer Protocol -- HTTP/1.1. [Online]. <https://www.ietf.org/rfc/rfc2616.txt>
- [För01] Friedrich Försterling, *Attribution. An introduction to theories, research and applications.*: Psychology Press, 2001.
- [Gei09] Frank Geisler, *Datenbanken. Grundlagen und Design*, 3rd ed.: mitp, 2009.
- [Ges06] (2006, Juli) Gesellschaft für Informatik. [Online]. <https://www.gi.de/themen/was-ist-informatik.html>
- [Gol13] Joachim Goll and Manfred Dausmann, *Architektur- und Entwurfsmuster der Softwaretechnik. Mit lauffähigen Beispielen in Java.*: Springer Vieweg, 2013.
- [Gra03] Martin Grabmüller, "Multiparadigmen-Programmiersprachen," Technische Universität Berlin, Forschungsbericht 2003.

- [The142] The hsql Development Group. (2014) HyperSQL. [Online]. <http://hsqldb.org/>
- [hei14] heise Developer. (2014, April) heise online. [Online]. <http://www.heise.de/developer/meldung/Studie-Gewaelte-Web-Programmiersprache-sagt-nicht-viel-ueber-Sicherheit-aus-2172516.html>
- [Kem09] Alfons Kemper and Andre Eickler, *Datenbanksysteme. Eine Einführung*, 7th ed.: Oldenburg Verlag München, 2009.
- [Lah09] Bernhard Lahres and Gregor Rayman, *Objektorientierte Programmierung. Das umfassende Handbuch*, 2nd ed.: Galileo Computing, 2009.
- [Mic13] Microsoft Corporation. (2013) microsoft.com. [Online]. <http://www.microsoft.com/web/platform/server.aspx?templang=de-de>
- [Mik14] Michael Mikowski and Josh Powell, *Single Page Web Applications.*: Manning Publications Co., 2014.
- [Ora14] Oracle. (2014, August) Java Servlet Technology Overview. [Online]. <http://www.oracle.com/technetwork/java/overview-137084.html>
- [Apa141] The Apache Software Foundation. Apache Tomcat. [Online]. <http://tomcat.apache.org/>
- [Pri01] Kent Beck et al. (2001) Prinzipien hinter dem Agilen Manifest. [Online]. <http://agilemanifesto.org/iso/de/principles.html>
- [QSu14] Q-Success. (2014, September) W3Techs. Web Technology Surveys. [Online]. [http://w3techs.com/technologies/overview/web\\_server/all](http://w3techs.com/technologies/overview/web_server/all)
- [Ora141] Oracle. Core J2EE Patterns - Data Access Object. [Online]. <http://www.oracle.com/technetwork/java/dataaccessobject-138824.html>
- [Sch10] Friedemann Schulz von Thun, *Miteinander reden 1: Störungen und Klärungen. Allgemeine Psychologie der Kommunikation*, 48th ed.: rororo, 2010.
- [See13] Silke Seehusen, "Software-Architekturen," Fachhochschule Lübeck, Vorlesungsskript WS 2012/13.
- [sel14] (2014, September) selfdeterminationtheory. [Online]. <http://www.selfdeterminationtheory.org/theory/>

- [The14] The Eclipse Foundation. (2014) Jetty. [Online]. <http://www.eclipse.org/jetty/>
- [Ult14] UltiDev LLC. UltiDev Cassini Web Server for ASP.NET Applications. [Online]. <http://www.ultidev.com/products/Cassini/>
- [Vog08] Oliver Vogel et al., *Software-Architektur: Grundlagen - Konzepte - Praxis*, 2nd ed.: Spektrum Akademischer Verlag, 2008.
- [W3C14] W3C. (2014, Juni) HTML 5.1: A vocabulary and associated APIs for HTML and XHTML. [Online]. <http://www.w3.org/TR/html51/dom.html>
- [Wol11] Henning Wolf and Wolf-Gideon Bleek, *Agile Softwareentwicklung. Werte, Konzepte und Methoden*, 2nd ed.: dpunkt.verlag, 2011.