

FERNUNIVERSITÄT HAGEN

# Thesis Title

by

Roman Sachse

A thesis submitted in partial fulfillment for the  
degree of Doctor of Philosophy

in the

Faculty Name

Department or School Name

19. Januar 2013

# Declaration of Authorship

I, AUTHOR NAME, declare that this thesis titled, 'THESIS TITLE' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

---

Date:

---

*“Write a funny quote here.”*

If the quote is taken from someone, their name goes here

FERNUNIVERSITÄT HAGEN

# *Abstract*

Faculty Name

Department or School Name

Doctor of Philosophy

by Roman Sachse

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

# *Acknowledgements*

The acknowledgements and the people to thank go here, don't forget to include your project advisor...

# Inhaltsverzeichnis

<b>Declaration of Authorship</b>	<b>i</b>
<b>Abstract</b>	<b>iii</b>
<b>Acknowledgements</b>	<b>iv</b>
<b>List of Figures</b>	<b>viii</b>
<b>List of Tables</b>	<b>ix</b>
<b>Abbreviations</b>	<b>x</b>
<b>Physical Constants</b>	<b>xi</b>
<b>Symbols</b>	<b>xii</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Einleitung . . . . .	1
<b>2 Hintergrund/Begriffsklärung</b>	<b>2</b>
2.1 Personal Learning Environments . . . . .	2
2.1.1 Computergestütztes Lernen . . . . .	2
2.1.2 Probleme mit Lern-Management-Systemen . . . . .	2
2.1.2.1 Lebenslanges Lernen . . . . .	3
2.1.2.2 Technologische Entwicklung . . . . .	3
2.1.3 Personal Learning Environments - Einführung . . . . .	4
<b>3 Anforderungsanalyse</b>	<b>5</b>
3.1 Use Case . . . . .	5
3.1.1 Akteure . . . . .	5
3.1.2 Ausgangssituation . . . . .	5
3.1.2.1 Einschränkungen/Umwstände . . . . .	6
3.1.3 Arbeitsabläufe . . . . .	7
3.1.3.1 Grundsätzlicher Arbeitsablauf für Studenten . . . . .	7
3.1.3.2 Arbeitsablauf Student 1 . . . . .	7

---

3.1.3.3	Arbeitsablauf Student 3	9
3.2	Anforderungen	10
<b>4</b>	<b>Übersicht Stand der Forschung/Technik</b>	<b>12</b>
4.1	Klassifizierungsmethoden für Personal Learning Environments	12
4.1.0.4	Dimensionen nach Palmér	12
4.1.0.5	Muster für das Design von Personal Learning Environments	14
4.2	Technologie	17
4.3	Widgets	17
4.3.1	CORS	17
4.3.2	HTML5	19
4.3.2.1	Appcache	19
4.3.2.2	Local Storage	19
4.3.3	REST	19
4.3.3.1	Widgetparameter	20
<b>5</b>	<b>Lösungsansatz</b>	<b>21</b>
5.1	Überblick	21
5.2	Design	21
5.3	Technisch	21
5.3.1	Unabhängigkeit Offline/Online	21
<b>6</b>	<b>Details zur Lösung</b>	<b>22</b>
6.1	Entwicklungsumgebung/Tools	22
6.1.1	Symfony2	22
6.1.2	eigene Bundles	23
6.1.3	Fremd-Bundles	23
6.1.4	Doctrine2	23
6.1.5	AngularJS	23
6.1.6	Jquery	23
6.1.7	Twitter Bootstrap	24
6.1.8	Apache Wookie	24
6.2	Implementierungsdetails	24
6.2.1	Ablauf holen der Daten	24
6.2.2	Umsetzung der Offline-Fähigkeiten	24
6.2.2.1	Local Storage	24
6.2.2.2	Einschränkungen im Offline-Betrieb	25
6.2.2.3	Preloads der Widgets	25
6.2.3	Widgets auf Hauptebene	25
6.2.4	Probleme bei multiple Instanzen des selben Widgets	27
6.2.5	Drag'n Drop	27
6.2.6	CORS	28
6.2.7	Kommunikation zwischen den IFrames	28
<b>7</b>	<b>Zusammenfassung/Ausblick</b>	<b>29</b>
7.0.8	Offene Fragen für weitere Forschungsarbeiten	29

<b>A</b>	<b>Installation/Deployment</b>	<b>30</b>
A.1	Installation . . . . .	30
A.1.1	Wookie . . . . .	30
A.1.2	Plesynd . . . . .	30
A.2	Deployment . . . . .	30
A.2.1	Plesynd . . . . .	30
	<b>Literaturverzeichnis</b>	<b>31</b>



# Abbildungsverzeichnis

# Tabellenverzeichnis

3.1 Anforderungen . . . . .	11
-----------------------------	----

# Abbreviations

DOM Document Object Model

# Physical Constants

$$\text{Speed of Light } c = 2.997\,924\,58 \times 10^8 \text{ ms}^{-\text{s}} \text{ (exact)}$$

# Symbols

$a$	distance	m
$P$	power	W ( $\text{Js}^{-1}$ )
$\omega$	angular frequency	$\text{rads}^{-1}$

*For/Dedicated to/To my...*

# Kapitel 1

## Einleitung

### 1.1 Einleitung

Das Ziel dieser Arbeit ist die Entwicklung eines Prototypen einer leichtgewichtigen Lernumgebung auf Basis der aktuellen HTML5-Technologien. Diese Umgebung soll als Personal Learning Environment (PLE) Verwendung finden. Möglich wird dies, indem die Umgebung als Dashboard fungiert, welches Programme und Services unterschiedlichster Quellen, sogenannte Widgets, auf einer oder mehrerer Seiten zusammenfasst und die Interaktion mit ihnen ermöglicht. Diese Widgets sollen Inhalte von Services wie Twitter, Facebook, Etherpad lite oder auch Todo Listen darstellen und bearbeitbar machen. Somit wird das Dashboard zu einem Einstieg oder Portal in die Arbeit mit unterschiedlichsten Services. Die Arbeit betrachtet primär die Screen- und die Temporal-Dimension nach Palmer, wird sich aber auch, zumindest teilweise, mit der Data-Dimension befassen. Im ersten Teil der Arbeit wird die Screen-Dimension betrachtet. Dies bedeutet, dass das User-Interface des Dashboards auf Grund vorheriger Überlegungen und Klassifizierungen designed und prototypisch umgesetzt wird. Der zweite Teil der Arbeit befasst sich mit der Temporal-Dimension. Hier geht es um die Planung und Implementierung unterschiedlichster Mechanismen, um das System auch ohne einen ständigen Internetzugang nutzbar zu machen. Das Ergebnis der Arbeit soll ein System liefern, welches durch Designüberlegungen, Apis und prototypische Implementierungen darauf aufbauenden Arbeiten das Fundament liefert es zu einer vollwertigen PLE auszubauen. In dieser finalen Version sollen dann alle 6 Dimensionen beachtet und implementiert werden.

## Kapitel 2

# Hintergrund/Begriffsklärung

### 2.1 Personal Learning Environments

#### 2.1.1 Computergestütztes Lernen

Computergestütztes Lernen (E-Learning) wird in Universitäten und anderen Bildungseinrichtungen zum großen Teil über zentralisierte Lern-Management-Systeme (LMS) durchgeführt. Diese ermöglichen es den Lehrenden Kurse zu Erstellen und Informationen, Zeitpläne, Dateien und andere Ressourcen zu diesen Kursen hinzuzufügen. Studenten können sich für anschließend für diese Kurse einschreiben. Dieses Schema ist momentan der Status Quo in den meisten höheren Bildungseinrichtungen [1].

#### 2.1.2 Probleme mit Lern-Management-Systemen

Die zentralisierte Art von LMS hat jedoch im Laufe der Zeit einige Schwächen offenbart. Nach Mott werden LMS zum Großteil für administrative Aufgaben eingesetzt (z.B. für Zeitplanung oder für das Hochladen von PDF-Dokumenten oder Vorlesungsfolien) und nicht für aktive oder interaktive Lernprozesse [1]. Des weiteren sind LMS sehr kurs- und lehrerzentriert. Der Lernende wird primär als Konsument aufgefasst. Er kann sich beispielsweise bei Kursen anmelden, Hausaufgaben einreichen und sich über Resultate informieren. Aktive Beteiligung ist ihm möglich, jedoch nur in dem von dem Lehrenden vorgegebenen Rahmen. Schließlich gibt es noch das Problem, dass der Zugriff auf die in den Systemen verteilten Daten auf mehrere Art und Weisen für den Nutzer beschränkt ist. Zum einen stehen die Kurse meist nur den Studenten während einer definierten Zeit (z.B. einem Semester) offen. Nach Ablauf des Zeitraums wird der Zugriff eingeschränkt oder komplett aufgehoben. Zum anderen ist es so, dass dem Nutzer nach



Verlassen der Bildungseinrichtung der Zutritt zu dem LMS oft komplett verweigert wird und er so keine Möglichkeit mehr hat auf die Daten und Werkzeuge innerhalb des LMS zuzugreifen [2].

#### **2.1.2.1 Lebenslanges Lernen**

Der zeitlich eingeschränkte Zugriff auf Informationen steht dem Konzept des lebenslangen Lernens gegenüber. Lebenslanges Lernen ist eine Sichtweise, die es dem Menschen ermöglichen soll sich während seines gesamten Lebens selbständig neue Fähigkeiten anzueignen, also zu Lernen. Nach dem Bundesministerium für Bildung und Forschung hört „Lernen [...] nach Schule, Ausbildung oder Studium nicht auf, denn Lernen ist das wesentliche Werkzeug zum Erlangen von Bildung und damit für die Gestaltung individueller Lebens- und Arbeitschancen. Lebenslanges Lernen heißt das Schlüsselwort, wenn man auf dem Arbeitsmarkt mithalten, einen Berufs- oder Schulabschluss nachholen oder sich einfach nur weiterbilden will“ [3]. Aus diesem Grund benötigen auch Arbeiter kontinuierliches Lernen während ihrer gesamten Arbeitslebens, um ihrer berufsbezogenen Fähigkeiten zu aktualisieren und sich neue benötigte Fähigkeiten anzueignen [4]. Lern-Management-Systeme, die den Nutzer nach einiger Zeit ausschließen oder ihm Informationen verwehren passen nicht in dieses Konzept.

#### **2.1.2.2 Technologische Entwicklung**

Die Technologien, die den am Lehrbetrieb Beteiligten (Lehrenden und Lernenden) für ihre Arbeitsabläufe zur Verfügung stehen in den letzten Jahren extrem geändert und weiterentwickelt. Das Aufkommen des Web 2.0 hat eine Vielzahl von Services und sozialen Netzwerken hervorgebracht, welche es den Nutzern erlauben aktiv in die Gestaltung von Inhalten und Ressourcen einzubringen. Hierzu gehören Wikis, Blogs, soziale Netzwerke wie Facebook, Webservices unterschiedlichster Arten (für Notizen, Todo-Listen, Kalender) etc.. Diese Entwicklung müsse sich nach Attwell auch im Bereich des E-Learnings widerspiegeln. Computergestütztes Lernen dürfe in Zeiten von Web 2.0 nicht mehr einfach nur die Formen des Lernens in Klassenräumen und Universitäten auf Software übertragen [4]. Es müsse dem Lernenden die Möglichkeit geben sich aktiv in den Lernprozess einzubringen und die von ihm genutzten Werkzeuge selber zu bestimmen. Der Lerner soll sich von einem reinen „Consumer“ zu einem „Prosumer“ entwickeln, also zu jemandem der gleichzeitig Konsument („Consumer“) und Produzent („Producer“) von Lernmaterialien ist [2].

### 2.1.3 Personal Learning Environments - Einführung

Personal Learnings Environments (PLEs) sind ein Konzept, um diese Forderungen umzusetzen und die oben beschriebenen Probleme von LMS zu vermeiden. [4]. PLEs nehmen Abstand von der kurs- oder institutszentrierten Sichtweise des computergestützten Lernens und propagieren eine nutzerzentrierte Sichtweise. Eine PLE soll dem Nutzer die Möglichkeit geben, dass System nach seinen Vorstellungen und Anforderungen zu personalisieren sich persönlich seine Werkzeuge, die er zum Lernen benötigt zusammenstellen und in einer Applikation zusammenfassen [5]. Dies erlaubt es ihm diese auch bei sich veränderten Lebensumständen oder Lerngewohnheiten (wechseln der Bildungseinrichtung, des Arbeitsplatzes) anzupassen und weiter zu benutzen. Der Nutzer kann unabhängig vom Lehrinstitut Daten und Inhalte erstellen, welcher auch nach dem Verlassen des Instituts weiterhin zur Verfügung stehen. Er kann Lern- und Interessengruppen auch über Institutsgrenzen hinweg bilden [2] und er kann Werkzeuge wie Kalender, To-do-Listen oder Notizen nutzen und diese nach seinen eigenen Vorstellungen klassifizieren und sortieren.

## Kapitel 3

# Anforderungsanalyse

Das Ziel der vorliegenden Arbeit ist die Planung und Implementierung eines Prototypen für eine leichtgewichtige Personal-Learning-Environment. Die Anforderungen an das finale System lassen sich aus dem in dem nächsten Abschnitt vorgestellten Use-Case ableiten, welcher die Arbeit mit dem System verdeutlichen soll.

### 3.1 Use Case

Der folgende Use-Case soll durch das finale System idealerweise abgedeckt werden:

#### 3.1.1 Akteure

- Dozent mit Sitz in Hagen
- Student 1 mit Sitz in Kamerun (Fernuni Hagen)
- Student 2 mit Sitz in Kamerun (Fernuni Hagen)
- Student 3 mit Sitz in Berlin (Fernuni Hagen)
- Student 4 mit Sitz in Osnabrück (Universität Osnabrück)

#### 3.1.2 Ausgangssituation

Der Dozent betreut an der Fernuni Hagen unter anderem den Kurs „E-Learning: A new approach“. Hierfür haben auch die Studenten 1 und 3 eingeschrieben. Für diesen Kurs hat der Dozent mehrere Kanäle zur Kommunikation angelegt. Die Studenten sollen die Kanäle wählen, die ihren Arbeitsgewohnheiten am meisten entsprechen. Am Ende des

Semesters soll es eine Auswertung geben, welche Kanäle am häufigsten genutzt und welche von den Studenten eher ignoriert wurden. Parallel dazu werden die Systeme der Fernuni zur Online Bearbeitung der Einsendeaufgaben genutzt. Die vom Dozenten angelegten Kommunikationskanäle sind:

- ein Twitter Channel
- eine Facebookgruppe
- einen eigenen Google-Calender
- einen Chat
- zur Terminabsprache und Abstimmung der nächsten Schritte soll Doodle verwendet werden
- ein System zur Hinterlegung von Todo-Listen
- einen eigenen Blog, welcher einen RSS Feed bereitstellt
- Etherpad Lite soll zur gemeinsamen Erstellung von Texten genutzt werden

Die Studenten sind angehalten sich regelmäßig über Aktualisierungen der Kanäle auf dem Laufenden zu halten.

Zusätzlich haben sich Student 1, 2 und 4 zu einer virtuellen Lerngruppe zum Thema Datenbanken zusammengeschlossen. Hierfür können nicht die Systeme der Fernuni Hagen genutzt werden, da nur Student 2 momentan in dem spezifischen Kurs eingeschrieben ist. Student 1 hat sich nicht für den Kurs angemeldet und Student 4 hat überhaupt keine Möglichkeit dazu, da er nicht an der Fernuni immatrikuliert ist. Sie haben sich dazu entschlossen primär einen Chat zur Kommunikation zu benutzen.

### **3.1.2.1 Einschränkungen/Umstände**

Student 1 und Student 2 leben in Kamerun. Beide haben dort das Problem, dass der Internetzugriff aus mehreren Gründen nicht immer gegeben ist. Student 1 hat zu Hause keinen Internetzugang. Er hat nur die Möglichkeit in der Universität oder in einem Internetcafe online zu gehen. Student 2 hat einen Internetzugang, allerdings ist dieser relativ langsam und wird nach Zeit abgerechnet, so dass es vorteilhaft für ihn ist, wenn er nur für kurze Zeit online ist. Aus diesem Grund benötigen die beiden idealerweise ein System, welches ihnen die Möglichkeit bietet die neuesten Informationen auch offline zu lesen und zumindest rudimentär auch offline kleine Aufgaben zu erledigen. Diese

sollten sich bei Wiederverbindung mit dem Internet mit den entsprechenden Services synchronisieren. Des Weiteren sollten sie in der Lage sein die Daten auch ohne Internetverbindung zwischen verschiedenen Rechnern auszutauschen. Insbesondere Student 1 sollte in der Lage seine Aktionen bei sich zu Hause vorzunehmen und die durchgeführten Änderungen dann an einem Rechner mit Internetanschluss zu synchronisieren. Die Anforderungen von Student 3 sind ähnlich gelagert. Er ist sehr viel unterwegs und erledigt daher viele kurze Aufgaben mit dem Smartphone. Auch hier ist eine Internetverbindung nicht immer gewährleistet ist oder sie wird temporär deaktiviert um die Akkulaufzeit zu verlängern. Durch die Arbeit an unterschiedlichen Rechnern mit potentiell unterschiedlichen Betriebssystemen, ist die Installation einer komplexen Software nicht ohne Weiteres möglich.

Idealerweise nutzen alle Studenten das selbe Basis-System und können sich hier die benötigten Services und Applikationen so zusammenstellen wie es ihren Ansprüchen entspricht.

### 3.1.3 Arbeitsabläufe

Im Folgenden werden die unterschiedlichen Arbeitsabläufe mit dem System exemplarisch an Student 1 und an Student 3 dargelegt.

#### 3.1.3.1 Grundsätzlicher Arbeitsablauf für Studenten

Um mit der PLE arbeiten zu können müssen die Studenten als erstes online eine Account in dem PLE-System erstellen. Anschließend melden sie sich mit ihren gewählten Login-Daten an. Bei erster Anmeldung hat das System bereits zwei Workspaces erstellt und mit Standard-Widgets gefüllt. Die PLE führt den User mit Hilfe von Tooltips durch die ersten Schritte im System (Tooltips, Workspace umbenennen, Einstellungen, offline gehen, Synchronisierung etc).

#### 3.1.3.2 Arbeitsablauf Student 1

- *Tag 1:* Student 1 befindet sich in der Universität und verbindet seinen USB-Stick mit einem PC. Auf diesem Stick befindet sich die ausführbare mobiler Version eines aktuellen Browsers. Er öffnet die Applikation online in diesem Browse (alle folgenden Aktionen werden mit dem selben Browser durchgeführt). Student 1 löscht den nicht benötigten Workspace und benennt den anderen in “PLE” um. Anschließend sucht er die Widgets für den PLE Workspace aus der Widget-Datenbank heraus. Das System zeigt ihm dabei an, für welche Widgets eine Offline-Fähigkeit zur

Verfügung steht. Darauf löscht er nicht benötigte Widgets und organisiert die Anordnung der Widgets nach seinen Vorstellungen per Drag and Drop. Damit er mit den einzelnen Widgets auch arbeiten kann meldet sich Student 1 schließlich bei den jeweiligen Services mit seinen Account-Daten an. Bevor er den Browser schließt, klickt das erste mal auf “Offline gehen”. Das System sorgt dafür, dass sich alle Widgets noch einmal mit dem Internet synchronisieren.

- *Tag 2:* Student 1 loggt sich an der Uni in das PLE-System ein. Das System zeigt ihm auf der Startseite (dem Dashboard) an wie viele neue Items es auf seinem PLE Workspace gibt. Ein direkter Link führt ihn zum Workspace. Er sieht, dass momentan mehrere Leute im Chat sind und unterhält sich über das Widget mit ihnen. Der Dozent hat ein paar globale Todos angelegt und die ersten Nachrichten kommen über den Twitter Channel herein. Nach einiger Zeit klickt Student 1 auf “jetzt offline gehen” wodurch alle Widgets seines Workspaces auf den neuesten Stand gebracht werden. Er sieht, dass Student 3 einen längeren Text im Chat geschrieben hat, beschließt diesen jedoch später zu lesen. Gleiches gilt für den Einführungsartikel des Dozenten im Kursblog. Hierfür erstellt er sich Items in seiner Todo-Liste. Abschließend schließt Student 1 den Browser und entfernt den USB-Stick.
- *Tag 3:* Student 1 öffnet die Applikation in seinem mobilen Browser zu Hause. Das System erkennt, dass es sich im Offline-Modus befindet und versucht keine Synchronisierung mit dem Internet herzustellen. Student 1 liest den Text, den Student 3 im Chat hinterlegt hat und beantwortet die darin gestellten Fragen ebenfalls im Chatfenster. Anschließend teilt er dies über Twitter mit und erledigt sein Todo-Item.
- *Tag 4:* Student 1 loggt sich im Internet-Cafe mit seinem mobilen Browser in das System ein. Das System erkennt, dass es online ist, lädt die neuesten Items der Services herunter und synchronisiert die nur lokal vorgenommenen Aktionen (Twitter, Chat, Todo). Per Mail hat Student 1 den Vorschlag von Student 2 bekommen gemeinsam mit Student 4 eine Lerngruppe zum Thema Datenbanken zu gründen. Hierzu wollen sie unter anderem ein Chat-System benutzen. Student 1 schlägt per Mail das ihm bekannte Chat-Widget für die PLE-Applikation vor. Er erstellt einen neuen Workspace, nennt ihn “Lerngruppe DB” und fügt das Chat-Widget mit den passenden Einstellungen hinzu.
- *Tag 5:* Student 1 sieht in seinem Dashboard, dass es für den Workspace “Lerngruppe DB” 12 neue Items gibt. Er geht direkt zu dem Workspace und stellt fest, dass die beiden anderen Studenten sich ebenfalls in dem Chat angemeldet haben. Da

momentan alle online verfügbar sind, beginnen sie ihre erste Gruppenunterhaltung und planen die weitere Vorgehensweise.

### 3.1.3.3 Arbeitsablauf Student 3

Student 3 nutzt das System hauptsächlich mit seinem Tablet, welches in der Lage ist sich über Wlan, sowie UMTS mit dem Internet zu verbinden.

- *Tag 1:* Student 3 meldet sich ähnlich wie Student 1 in der PLE an und erstellt für seine Uni-Kurse jeweils einen Workspace. Außerdem legt er sich einen Workspace an, in dem sich nicht kursspezifische Widgets finden. Hierzu gehören sein Google-Kalender, sein RSS-Reader, eine Todo Liste, sowie ein News-Widget
- *Tag 2:* Student 3 verbindet sich zu Hause mit seinem Wlan und loggt sich in der PLE ein. Nach Prüfung des Dashboards erstellt er sich in der allgemeinen Todo Liste für den Tag. Er sieht, dass es in seinem RSS-Reader 9 neue Artikel gibt. Diese möchte er auf seinem Weg zur Uni in der U-Bahn lesen. Da dort der Empfang eher schlecht ist, synchronisiert er die PLE noch einmal und bemerkt, dass das System standardmäßig nur die letzten 5 Einträge eines Widgets zwischenspeichert. Aus diesem Grund geht er in die Einstellungen des RSS-Widgets und stellt dort ein, dass nicht nur 5, sondern 10 Items lokal gecached werden sollen. Anschließend betätigt er nochmal den “offline gehen” Button und sieht, dass nun alle neuen Artikel zwischengespeichert sind. Auf dem Weg zu Uni ruft Student 3 das System auf seinem Tablet auf. Da kein Internetzugang besteht greift das System nur auf die lokalen Daten zurück. Das News-Widget in seinem globalen Workspace ist nicht offline-fähig. Aus diesem Grund wird es auch nur ausgegraut und nicht-funktional angezeigt. Der Student ist jedoch in der Lage seine lokal gecachten RSS-Artikel zu lesen. Bei Lektüre eines Artikels über Javascript fällt ihm ein, dass er sein Uni-Projekt noch mit der neuesten JQuery Version updaten wollte. Hierzu erstellt er sich ein weiteres Todo auf seiner Liste. In der Uni verbindet er sein Tablet mit dem Uni-Wlan. Der RSS-Reader markiert die 6 Artikel die, Student 3 in der U-Bahn gelesen hat als gelesen und synchronisiert das neueste Todo-Item mit dem Service im Internet.

## 3.2 Anforderungen

Zusammenfassend kann also gesagt werden, dass Betreuer und Studenten stehen über unterschiedliche Kanäle in Kommunikation miteinander stehen. Es müssen Termine geplant oder auch Notizen und Nachrichten hin und hergeschickt werden. Diese Kanäle sollen auf einem zentralen Zugang so zusammengefasst sein, dass es den Teilnehmern des Kurse alle relevanten Informationen an einer aufrufen und bearbeiten zu können. Dabei sind die Teilnehmer zu unterschiedlichen Zeiten online. Die Teilnehmer sollen das System offline Nutzen können, um einfache Arbeiten wie das Schreiben von Twitter-Nachrichten, Notizen und Instant-Messaging Nachrichten oder eine Terminabsprache über einen Kalender erledigen können. Bei dem Wechsel zwischen Online und Offline müssen die Daten synchronisiert werden. Idealerweise haben die Nutzer alle Daten auf einem USB-Stick bei sich und können so von unterschiedlichsten Rechnern, wie beispielsweise in der Universität, im Internetcafe oder zu von Hause aus, arbeiten.

Daraus lassen sich mehrere Anforderungen an das zu entwickelnde System ableiten. Es soll in der Lage sein als Aggregator für die unterschiedlichsten Services und Kanäle zu dienen. Es muss möglich sein auf die wichtigsten Informationen an einem zentralen Platz zuzugreifen und diese auch zu bearbeiten (1). Dadurch, dass die Teilnehmer an unterschiedlichen Rechnern arbeiten, welche zum Teil nicht in ihrem persönlichen Besitz sind, ist es für sie nicht oder nur sehr schwer möglich eine neue Software zu installieren. Aus diesem Grund soll das System mit nativen Browsertechnologien ohne weitere Installation nutzbar sein (2). Das System muss in der Lage die wichtigsten Funktionalitäten auch dann zu Verfügung zu stellen, wenn es keinen Kontakt zu dem Internet hat (3). Des Weiteren soll es in der Lage sein bei einer Wiederaufnahme der Verbindung die durchgeführten Aktionen mit dem jeweiligen Service zu synchronisieren (4). Dies gilt insbesondere für die Arbeit mit unterschiedlichen Rechnern. Der Nutzer soll sein System an einem Computer mit dem Internet synchronisieren können und dann an einem anderen Rechner offline weiterarbeiten können. Es ist also notwendig, dass es dem Nutzer ermöglicht wird die Daten mitzunehmen (beispielsweise per USB-Stick) und an anderer Stelle weiterzuverwenden (5).

Neben diesen funktionalen Anforderungen gibt es noch weitere Anforderungen, welche das System erfüllen soll. In dieser Arbeit kann nur eine prototypische Implementierung der Anforderungen erfolgen. Das System soll also als Basis für weitere Entwicklungen und Forschungsarbeiten dienen und einfach erweitert und verändert werden können (6). Es soll auch möglich sein auf Basis einer vorgegebenen Implementierung oder API weitere Services oder Kanäle in das System zu laden und es so beständig in seiner Funktionalität zu erweitern (7). Schließlich wird die Software in unterschi Bereichen, insbesondere



in einem universitären Umfeld eingesetzt. Dies verlangt eine Nutzbarkeit ohne Lizenzgebühren für die genutzten Technologien. Daraus folgt, dass für die Umsetzung keine proprietären, sondern nur freie Technologien verwendet werden dürfen (8).

TABELLE 3.1: Anforderungen

1	Informationsaggregator
2	ohne Installationsaufwand lauffähig in aktuellen Browsern
3	Möglichkeit des (zumindest rudimentären) Weiterarbeitens, wenn offline
4	Synchronisierung der vorgenommenen Änderungen, wenn wieder online
5	Daten können zwischen unterschiedlichen Rechnern offline ausgetauscht werden
6	kann als Basis für weitere Entwicklungen dienen
7	neue Services und Kanäle sollen sich einfach in das System integrieren lassen
8	ausschließliche Verwendung freier Technologien

Die in Abschnitt 3.1.2 beschriebenen Kanäle und Services müssen für eine vollständige PLE implementiert werden. Da dies aber den Rahmen dieser Arbeit sprengen würde und das Ziel ist einen Prototypen für die Grundlage einer solchen PLE zu schaffen, werden diese nicht in die Anforderungen aufgenommen.

## Kapitel 4

# Übersicht Stand der Forschung/Technik

### 4.1 Klassifizierungsmethoden für Personal Learning Environments

Es existieren zwei wichtige Ansätze um Funktionalitäten von PLEs auf unterschiedliche Klassen abzubilden. Zum einen gibt es den Versuch von Palmér Dimensionen zu definieren und die Funktionalitäten von PLEs diesen Dimensionen zuzuordnen. Palmér schafft damit ein System, um unterschiedliche PLE-Systeme kategorisieren zu können. Zum anderen definiert Wilson Muster, die helfen sollen eine PLE zu entwickeln und bestehende Systeme auf wichtige Funktionalitäten zu überprüfen. Diese beiden Ansätze werden im Folgenden vorgestellt.

#### 4.1.0.4 Dimensionen nach Palmér

Palmér definiert sechs Dimensionen mit denen er so viele relevanten Funktionalitäten von PLEs wie möglich erfassen möchte [6]. Trotz dessen sollen diese Dimensionen relativ unabhängig voneinander sein, so dass es möglich ist das unterschiedliche Plattformen einige Dimensionen mehr und andere weniger berücksichtigen und implementieren. (TODO Grafik der Dimensionen). Eine PLE kann dann Anhand des Grades ihrer Implementierung der einzelnen Dimensionen kategorisiert und bewertet werden.

1. *Screen-Dimension*: Die Screen-Dimension befasst sich mit Aspekten, welche die Darstellungsebene von PLEs definieren. Hierzu zählt Palmér insbesondere das

User-Interface und die Usability des PLE-Containers (wie sind die Widgets angeordnet, wie können neue Widgets gesucht und hinzugefügt werden, wie einfach kann sich der Nutzer im System bewegen etc.), welcher als Einstiegspunkt in die Systembedienung dient und das User-Interface der einzelnen Widgets. Es gehören aber auch Funktionalitäten wie die Möglichkeit Inhalte und Ressourcen mit anderen Nutzern zu teilen und die Integration der selben Widgets in unterschiedlichen PLE-Containern zu der Screen-Dimension.

2. *Data-Dimension*: Mit der Data-Dimension beschreibt Palmér Funktionalitäten, die für die Portabilität der verwendeten Daten innerhalb einer PLE notwendig sind. Idealerweise sollen Widgets in der Lage sein untereinander und mit dem PLE-Container zu kommunizieren. Sie sollen Daten austauschen können und sich so weit wie möglich über ihre Zustände informieren. Des Weiteren soll es möglich sein die Daten der Widgets zu exportieren und sie an anderer Stelle oder in einem anderen PLE-Container wieder zu importieren und weiterzuverwenden. Mit der fortschreitenden Mobilität der Nutzer wird es immer wichtiger, dass der Zugriff auf die PLE auch dann möglich ist, wenn kein Zugriff auf das Internet besteht. Somit ist es nicht nur notwendig Daten zu importieren und zu exportieren, sondern auch zwischen einem Offline und einem Online-Speicher zu synchronisieren, sobald eine entsprechende Zustandsänderung eintritt.
3. *Social-Dimension*: Ein wesentlicher Bestandteil des Web 2.0 ist die Vernetzung von Freunden und Menschen mit ähnlichen Interessen untereinander. Dem trägt Palmér mit der Social-Dimension Rechnung. Diese Dimension gibt an, wie sehr eine PLE Funktionalitäten sozialer Netzwerke wie Freundeslisten integriert und Möglichkeiten bietet den Zugriff auf geteilte Ressourcen auf bestimmte Typen von Freunden einzuschränken. Palmér zählt aber auch die Möglichkeit zur Erstellung von eigenen offenen oder geschlossenen Lerngruppen zu Funktionalitäten, die in diese Dimension fallen.
4. *Temporal-Dimension*: Die in der Screen- und der Social-Dimension beschriebene Kollaboration zwischen Nutzern der PLE neue Anforderungen mit sich. So ist es beispielsweise notwendig oder zumindest wünschenswert, dass geänderte Inhalte sich in Echtzeit in den Instanzen der Widgets manifestieren, die ebenfalls auf diese Inhalte zugreifen. Hierbei sollten auch Probleme, wie auftretende Konflikte bei gleichzeitigem Bearbeiten der selben Ressource in Betracht gezogen werden.
5. *Activity-Dimension*: Die Activity-Dimension beschreibt die Möglichkeit Abläufe und Workflows innerhalb einer PLE aktiv zu gestalten. Hierzu gehören unter anderem einfache Dinge wie Anleitungen als Hilfestellung für den Nutzer zur Bewegung innerhalb der PLE. Besonderen Wert legt Palmér aber auf die Abbildung

von Lernsequenzen innerhalb der PLE. So können bestimmte Widgets auf bestimmte Ereignisse reagieren oder sich selbst aktivieren oder deaktivieren. Des Weiteren können unterschiedlichste Konzepte aus dem Bereich des E-Learnings (IMS Learning Design (TODO cite) oder Learning Object Metadata (LOM)) in den Widgets oder dem PLE-Container selber implementiert werden.

6. *Runtime-Dimension*: Die Runtime-Dimension befasst sich mit Funktionalitäten, die die Interoperabilität zwischen PLE-Systemen und Komponenten. Nach Palmér werden Nutzer in der Zukunft nicht nur eine PLE benutzen, sondern je nach Bedürfnis oder Anwendungsfall zwischen ihnen hin und herwechseln. Hierfür sollte es möglich sein Importe und Exporte für Inhalte und Einstellungen sowohl der einzelnen Widgets als auch des PLE-Containers vorzunehmen. Damit eine Interoperabilität zwischen PLEs möglich wird, ist es notwendig, dass Standards geschaffen werden, welche von den unterschiedlichen PLEs anerkannt und implementiert werden. Zusätzlich gehört für Palmér auch die Möglichkeit der Einbettung und Kommunikation von PLEs in und mit größeren Systemen zu dieser Dimension.

#### 4.1.0.5 Muster für das Design von Personal Learning Environments

Wilson stellt unterschiedliche Muster vor, die bei der Entwicklung einer PLE helfen sollen [7]. Hierbei legt er seinen Fokus auf die dezentralisierte nutzerzentrierte Natur von PLEs. Als Motivation dafür gibt er an, dass PLEs nicht ein einfaches Stück Software seien. Vielmehr stellen sie eine Umgebung dar, in der Menschen mit Werkzeugen Ressourcen und Communities in einer lockeren und nicht vorher strikt fixierten Art und Weise kommunizieren. Dieser lockere Aufbau macht es für den Entwickler jedoch nicht leichter für den Nutzer gut nutzbares System zu entwerfen. Wilson leitet seine Muster von denen ab, die von der Universität von Bolton während des „Personal Learning Environments Reference Model Project“ Projektes entwickelt wurden. Er versucht die Anzahl der Muster jedoch von ursprünglich 77 deutlich zu reduzieren, indem er Muster, die sich auf einfache Funktionen beziehen zu größeren Gruppen zusammenfasst und so eher allgemeine Charakteristika von PLEs beschreibt.

Wilson unterscheidet zwischen zwei verschiedenen Arten von Mustern für PLEs: Muster für persönliche Anwendungen (Personal Tools) und Muster für Lernnetzwerke (Learning Networks). Personal Tools stellen hierbei die Werkzeuge dar, die ein Nutzer direkt für seine Lernaktivitäten nutzt. Er interagiert mit unterschiedlichen sozialen Netzen (zum Beispiel Lernnetzwerken) oder verwendet die Tools anderweitig für seine persönliche Art des Lernens. Wilson definiert Learning Networks als die Infrastruktur, welche notwendig ist um soziale Netze oder Communities aufzubauen. Des weiteren zählt er auch die Menge

der Online Services hinzu, welche von einem Lehrinstitut den Lernenden zur Verfügung gestellt wird.

Diese Arbeit beschäftigt sich mit dem Aufbau einer PLE, welche direkt vom Nutzer für seine Lernaktivitäten verwendet wird. Learning Networks spielen hierfür keine oder eine nur sehr untergeordnete Rolle, so dass die Muster für das Erstellen eben dieser hier nicht weiter beleuchtet werden. Im Folgenden werden die Muster für Personal Tools vorgestellt. Diese ähneln meist einer Empfehlung, wie eine bestimmte Funktion umgesetzt werden sollte oder welche Funktionalitäten vorhanden sein sollten, um die Erfahrung des Nutzers im Umgang mit dem System zu verbessern.

1. *Discourse Monitor*: In einer PLE werden Informationen aus potentiell sehr vielen und unterschiedlichen Quellen verarbeitet. Um dem Nutzer die Möglichkeit zu geben wichtige Informationen schneller herauszufiltern oder zu priorisieren, sollte ein Discourse Monitor implementiert werden. Dieser fasst die wichtigsten Informationen aus den unterschiedlichen Quellen zusammen und bereitet sie in übersichtlicher Art und Weise auf. Dem Nutzer soll es dann möglich sein, die dargestellten Daten zu Filtern, zu priorisieren, seine Favoriten zu kennzeichnen oder neu eingetroffene Daten einfach und zeitnah zu erkennen.
2. *Connection Hub*: Der Nutzer einer PLE ist bei verschiedensten Netzwerken angemeldet und hat dort höchstwahrscheinlich unterschiedliche Informationen hinterlegt. Es ist auch möglich, dass eine Kommunikation über Netzwerkgrenzen hinweg stattgefunden hat. Diese Daten sind meist nicht in einem einzigen Netzwerk darstellbar. Ein Connection Hub soll in der Lage sein oder den Nutzer in die Lage versetzen diese Verbindungen darzustellen und aufzubereiten. Es wäre beispielsweise vorstellbar, dass es dem Nutzer ermöglicht wird Informationen und Daten verschiedenster Netzwerke zu kombinieren ohne, dass diese Netzwerke direkt in Kontakt zueinander stehen.
3. *Create and Mix Media*: Es ist unabdingbar das während des computergestützten Lernens sowohl von dem Lernenden, als auch von dem Lehrenden Ressourcen erstellt und diese auch anderen zur Verfügung gestellt werden. Hierzu gehören alle Arten von Ressourcen, also Textdateien, Notizen, Quellcode, Videos, Bilder, Präsentationen etc.. Die PLE sollte es dem Nutzer ermöglichen diese Ressourcen zu erstellen (auch unter Zuhilfenahme externer Services) und diese dann an unterschiedliche Netzwerke zu verteilen.
4. *Integrate Identities*: Durch das Nutzen unterschiedlicher Services hat der Nutzer höchstwahrscheinlich auch Useraccounts bei all diesen Services erstellt. Wilson

schlägt für die Vereinfachung des Umgangs mit diesen Accounts die Implementierung von Mechanismen zur Vereinfachung vor. Hierzu gehören die Nutzung von Systemen, die in der Lage sind Zugangsdaten für unterschiedliche Accounts zu verwalten und zu speichern. Es können aber auch Konzepte wie das zentrale Hinterlegen von Profildaten, zum Beispiel bei einem OpenID Server-Anbieter <sup>1</sup>, verwendet werden. Die Registrierung und der Login bei unterschiedlichen Services erfolgt dann über den Umweg über den zentralen Service, welcher alle notwendigen Daten zu dem Service übermittelt, bei dem sich der Nutzer einloggen möchte.

5. *Manage Time and Effort*: Benutzt man ein System wie eine PLE zur Organisation des persönlichen Lernens, so sollte einem dieses System Werkzeuge an die Hand geben, um das Lernen und die Arbeit zu organisieren. Hierzu gehören Zeitplaner, Kalender, Todo-Listen oder auch die Möglichkeit zur Erstellung von Notizen innerhalb des Systems.
6. *Navigation Layer*: Bei einer PLE hat sich der Nutzer idealerweise seine Lernumgebung aus unterschiedlichen Quellen und Werkzeugen zusammengestellt. Die Navigationsebene fasst diese in einem System zusammen und ermöglicht dem User einen einfachen Zugriff auf seine Werkzeuge. Wilson schlägt vor die Services als Widgets in der PLE einzubinden. Somit wird die PLE zu einem zentralen Zugriffspunkt oder zu einem Dashboard von dem aus der Nutzer alle seinen Aktionen ausführen kann.
7. *Multi-platform/Multimode*: Die Nutzergewohnheiten bezüglich des Gebrauchs des Internets haben sich in den letzten Jahren deutlich gewandelt (TODO: schlaues cite). Der Zugriff erfolgt nicht mehr primär über den eigenen (Heim-)Rechner, sondern über die verschiedensten Zugriffspunkte. Hierzu gehören Computer in der Universität, am Arbeitsplatz, im Internetcafé und in letzter Zeit verstärkt auch mobile Geräte wie Smartphones oder Tablets. Aus diesem Grunde ist es notwendig, dass der Zugriff auf Lernnetzwerke von all diesen Geräten aus möglich ist und die Systeme auch gut von den unterschiedlichen Geräten aus bedienbar sind. Des weiteren sollte es ein System geben, welches die Daten auf den unterschiedlichen Geräten miteinander synchronisiert, so dass der Nutzer von überall auf die aktuellsten Daten zugreifen kann.
8. *Choose, Change, Discard*: Dieses Pattern steht in engem Bezug zu der nutzerzentrierten Herangehensweise in dem Aufbau von PLEs. User sollen in der Lage sein, sich ihre Lernumgebung nach eigenen Vorlieben und Anforderungen einzurichten. Außerdem ist es sehr gut möglich, dass sich die Anforderungen im Laufe der Zeit ändern. Aus diesen Gründen ist es notwendig, dass es dem Nutzer frei steht Inhalte

---

<sup>1</sup><http://openid.net/>

und Services innerhalb der PLE zu verschieben und anzupassen und neue Werkzeuge hinzuzufügen und nicht mehr benötigte Werkzeuge wieder zu entfernen. Die PLE sollte hierbei nicht zu viele vom Nutzer nicht änderbare Bedienungsvorgaben und Konfigurationseinstellungen machen, sondern dem Nutzer so viele Freiheiten wie möglich bei Einrichten der eigenen Lernumgebung zu geben.

Kritik an den Patterns?

## 4.2 Technologie

Im Folgenden werden Technologien und Konzepte vorgestellt, die den Stand der Technik im Bereich der PLE-Entwicklung darlegen oder notwendig sind, um die in Kapitel 3 entwickelnden Anforderungen an eine Web basierte und offline fähige PLE zu erfüllen.

## 4.3 Widgets

Widgets Opensocial vs W3C warum w3c widgets, opensocial wird nicht weiterentwickelt etc, Apache Shindig Problematische W3C Widgets haben keine gesonderte Definition für HTML5 Appcache, unterschiedliche Formate

Wookie während der Entwicklung vom Incubator zum vollwertigen Apache Projekt

### 4.3.1 CORS

Moderne Browser benutzen als Teil ihres Sicherheitskonzeptes die Same-Origin-Policy. Diese bewirkt, dass Sprachen, die auf Clientseite ausgeführt werden (wie JavaScript), nicht die Möglichkeit haben, Request an einen anderen Zielpunkt als ihrem Ursprung zu starten[8]. Diese Policy wird also lediglich bei Zugriff auf URLs mit der selben Domain und dem selben Port, wie die URL von der die Seite geladen wurde, erfüllt. Das bedeutet, dass ein Skript auf `http://sop.example.com/directory1` Requests an `http://sop.example.com/directory2` starten kann, nicht jedoch an `http://example.com/directory` (unterschiedliche Domain) oder an `http://sop.example.com:8080/directory2` (unterschiedlicher Port). Ausgenommen ist hierbei das in eine Seite eingebettete Laden von Ressourcen. Hierzu gehören externe Inhalte, die über iFrames geladen werden aber auch externe JavaScript-Dateien (über `<script src=...>` Tags) und Medienressourcen wie Bilder und Videos. Des Weiteren ist es auch möglich Formulare an andere Zielpunkte als

den Ursprung abzuschicken. Diese Einschränkung hat also primär Auswirkungen auf das Absenden von XMLHttpRequest, also auf normale Ajax-Requests.

Die Same-Origin-Policy ist sehr sinnvoll um beispielsweise das Ausspähen privater Daten zu verhindern. Sie erschwert jedoch die Entwicklung moderner Ajax-Anwendungen und insbesondere die Entwicklung von Mashup-Applikationen wie PLEs, welche prinzipiell schon so aufgebaut sind, dass sie ihre Inhalte und Ressourcen aus unterschiedlichen Quellen beziehen. In einer PLE wie in dieser Arbeit beschrieben, ist es beispielsweise so, dass die Widgets (siehe ??) selber von einem Widget-Container wie Wookie (siehe 6.1.8) ausgeliefert werden und dadurch auch die Domain des Widget Containers als Origin besitzen. Arbeiten diese Widgets nun aber nicht nur lokal beim Client, sondern benötigen für ihre Funktionalität auch externe Server, so müssen sie in der Lage sein XMLHttpRequests an diese zu senden. Aus diesem Grund wurde der Mechanismus des Cross-Origin Resource Sharing (CORS)[9] eingeführt. Dieser erlaubt es unter bestimmten Bedingungen und Einschränkungen die Same-Origin-Policy zum umgehen.

Ein einfacher CORS-Request vom Client zum Server sieht wie folgt aus (Workflow analog zu [10]):

Der Client sendet eine Cross-Origin-Anfrage mit einem Origin Header an den Server:

---

```
GET /cors HTTP/1.1
Origin: http://api.bob.com
Host: api.alice.com
Accept-Language: en-US
Connection: keep-alive
User-Agent: Mozilla/5.0...
```

---

Anschließend antwortet der Server mit:

---

```
Access-Control-Allow-Origin: http://api.bob.com
Access-Control-Allow-Credentials: true
Access-Control-Expose-Headers: FooBar
Content-Type: text/html; charset=utf-8
```

---

Alle für den CORS-Request relevanten Header beginnen mit Access-Control. **Access-Control-Allow-Origin** bedeutet, dass der Server eine Cross-Origin-Anfrage von dem angegebenen Origin erlaubt, **Access-Control-Allow-Credentials: true** besagt, dass in diesem Request auch Cookies erlaubt sind. Möchte der Client Zugriff auf Nicht-Standard-Header aus der Antwort des Servers, müssen diese in **Access-Control-Expose-Headers** angegeben werden.

Sollte der Client einen Request mit einer anderen Methode als GET oder POST (siehe ??) senden, reicht dieser einfache Workflow nicht aus. In diesem Fall muss vor der eigentlichen Anfrage ein so genannter „Preflight-Request“ ablaufen, welcher verifiziert, dass der Server diese Methode als CORS-Request erlaubt.



Zuerst wird vom Client eine Anfrage mit der `OPTIONS`-Methode durchgeführt, welche den folgenden Request authentifizieren soll:

---

```
OPTIONS /cors HTTP/1.1
Origin: http://api.bob.com
Access-Control-Request-Method: PUT
Access-Control-Request-Headers: X-Custom-Header
Host: api.alice.com
Accept-Language: en-US
Connection: keep-alive
User-Agent: Mozilla/5.0...
```

---

`Access-Control-Request-Method` gibt hierbei an, welche Methode genutzt werden soll, `Access-Control-Request-Headers` informiert den Server über zusätzlich zu erwartende Header.

Der Server antwortet beispielsweise mit:

---

```
Access-Control-Allow-Origin: http://api.bob.com
Access-Control-Allow-Methods: GET, POST, PUT
Access-Control-Allow-Headers: X-Custom-Header
Content-Type: text/html; charset=utf-8
```

---

Der Preflight-Request ist nur erfolgreich, wenn die Methode aus `Access-Control-Request-Method` in `Access-Control-Allow-Methods` und alle Header aus `Access-Control-Request-Headers` in `Access-Control-Allow-Headers` vorhanden sind.

### 4.3.2 HTML5

neuerungen HTML5, nicht nur eine Technologie, Zusammenfassung mehrerer Technologien, CSS 3 unterschiedliche JavaScript Apis (local storage, appcache) drag and Drop

#### 4.3.2.1 Appcache

Eine wichtige Neuerung in Html

#### 4.3.2.2 Local Storage

Local Storage

### 4.3.3 REST

Rest

#### 4.3.3.1 Widgetparameter

Die Parameter für Widgets werden in der Wookie DB hinterlegt. Die Widget Implementation wird über `http://localhost:8080/wookie/shared/js/wookie-wrapper.js` ausgeliefert. Hier ist `setItem/getItem` so implementiert, dass es einen Request an den Server sendet und die Einstellungen speichert.

Die Standardeinstellungen aus der `config.xml` werden nur beim ersten deploy ausgelesen, anschließend nicht! mehr

Jedes Widget bekommt eine eigene id! Diese kann dann für die Identifizierung genutzt werden.

Die Werte werden NICHT im local storage hinterlegt./

# Kapitel 5

## Lösungsansatz

Beschreibung, wie die eigene Lösung die Anforderungen erfüllt Konzeptioneller (holistischer) Überblick über die eigene Lösung

### 5.1 Überblick

Konzentration auf die die Screen und Temporal Dimension

erweiterung des Multimode patterns auf offline/online

### 5.2 Design

wie ist das Design aufgebaut, welche Patterns könnte man nehmen Screen Dimension

### 5.3 Technisch

Temporal Dimension

#### 5.3.1 Unabhängigkeit Offline/Online

Local Storage, Appcache, Minification, warum kein Indexed DB

Drop in Rest oder Local

# Kapitel 6

## Details zur Lösung

Implementierungsdetails (nur die interessantesten) Erfahrungen / Evaluation

### 6.1 Entwicklungsumgebung/Tools

Für die Entwicklung von Plesynd wurden unterschiedliche Tools und Frameworks genutzt. Entwickelt wurde das System auf Serverseite mit [PHP](http://php.net)<sup>1</sup>. Der clientseitige Code wurde mit [JavaScript](http://de.wikipedia.org/wiki/JavaScript)<sup>2</sup> implementiert.

Die Frameworks im Folgenden kurz vorgestellt.

#### 6.1.1 Symfony2

[Symfony2](http://symfony.com)<sup>3</sup> ist ein in PHP implementiertes komponenten-basiertes full-stack Framework zur Entwicklung von MVC(Model-View-Controller) Anwendungen. Zu den bereitgestellten Komponenten gehört beispielsweise ein [Service Container](http://symfony.com/doc/current/book/service_container.html)<sup>4</sup>. Dieser erlaubt es die von den genutzten Klassen benötigten Abhängigkeiten zu definieren. Der Container sorgt dann für ein Instantiieren der Abhängigkeiten und ein Injizieren dieser zur Laufzeit (TODO: Dependency-Injection erklären?). Des weiteren bietet Symfony2 unter anderem eine eigene Template-Engine (TODO: Twig Link), sowie ein Event-Listener System, welches ermöglicht auf bestimmte Events (TODO: Request Beispiele) zu reagieren. Das Model wird in Symfony2 standardmäßig über Doctrine2 (siehe [6.1.4](#)) abgebildet.

---

<sup>1</sup><http://php.net>

<sup>2</sup><http://de.wikipedia.org/wiki/JavaScript>

<sup>3</sup><http://symfony.com>

<sup>4</sup>[http://symfony.com/doc/current/book/service\\_container.html](http://symfony.com/doc/current/book/service_container.html)

Die Komponenten werden in Symfony2 über sogenannte Bundles implementiert (TODO: Bundle Link). Bundles sind von dem System unabhängige Plugins, mit einer festen Struktur, welche einfach in eine Applikation eingebunden werden können. Das besondere an Symfony2 ist, dass das gesamte System aus Bundles aufgebaut ist. Hierzu gehören also auch das Kern-Framework, alle Komponenten, zusätzlicher Third-Party-Code, sowie die eigene Applikation selbst. Die für die Applikation entwickelten Bundles werden in 6.1.2, die Fremd-Bundles in 6.1.3 beschrieben

### 6.1.2 eigene Bundles

Wichtig: Bundles, was sind Bundles, System komplett aus Bundles aufgebaut. einfaches einbinden von Third Party Bundles. Plesynd besteht aus 3 Bundles: Plesynd, Wookie-Connector, User (welches wiederum vom FOSUserbundle ableitet. Benutzt noch weitere Bundles: FOSRest (mit Verweis zu Rest Erklärung), NelmioCorsBundle

### 6.1.3 Fremd-Bundles

### 6.1.4 Doctrine2

Orm, DataMapper Pattern, kein Active Record, Arbeit mit einfachen Objekten, Entity Manager Unit of Work

### 6.1.5 AngularJS

### 6.1.6 JQuery

[Jquery](#)<sup>5</sup> ist eine JavaScript-Bibliothek zur einfachen Manipulation des DOM. Desweiteren stellt sie erweiterte JavaScript-Funktionalitäten zur Verfügung und vereinfacht die Arbeit mit den browserbasierten Event-System(TODO: ref <http://de.wikipedia.org/wiki/JQuery>) AngularJS verwendet JQuery, wenn vorhanden, insbesondere zur Manipulation des DOM. In Plesynd basiert unter anderem noch das Postmessage(TODO: ref [postmessage](#)) auf JQuery.

---

<sup>5</sup><http://jquery.com/>

### 6.1.7 Twitter Bootstrap

[Twitter Bootstrap](#)<sup>6</sup> ist ein von Twitter entwickeltes Framework zur schnellen und einfachen Entwicklung von Frontends. Es stellt CSS-Vorlagen und JavaScript-Komponenten zur Verfügung, welche es dem Entwickler ermöglichen sollen in kurzer Zeit ein User-Interface zu entwerfen und umzusetzen. Bootstrap beinhaltet CSS-Vorlagen für Grids, Tabellen, Buttons etc. . In Plesynd wird von den von Bootstrap zur Verfügung gestellten Javascript-Komponenten momentan nur die [Modal-Komponente](#)<sup>7</sup> verwendet.

### 6.1.8 Apache Wookie

## 6.2 Implementierungsdetails

### 6.2.1 Ablauf holen der Daten

WookieConnectorBundle Der Loginname `connection.getUser()` `setLoginName(demo2);` ist dafür verantwortlich, dass Wookie eine neue Widgetinstanz erstellt oder eine bestehende zurück liefert.

Man muss sich jetzt fragen, ob es möglich sein soll, dass jeder Workspace die selben Instanzen von Widgets hat oder nicht. Wenn ja, kann einfach die User Id aus Plesynd als identifier genommen werden, wenn nicht, sollte zum Beispiel eine Kombination aus UserId und WorkspaceId als Identifier benutzt werden.

==> eigener unique Identifier beim Widget jedes Widgets bekommt seinen eigenen localStorage über `window.name` prefix

### 6.2.2 Umsetzung der Offline-Fähigkeiten

#### 6.2.2.1 Local Storage

Drop In Rest Benutzen von Services, Controller bekommen das nicht mit

---

<sup>6</sup><http://twitter.github.com/bootstrap/>

<sup>7</sup><http://twitter.github.com/bootstrap/javascript.html#modals>

### 6.2.2.2 Einschränkungen im Offline-Betrieb

Trotz der Offline Fähigkeiten des System stehen im Offline-Betrieb nicht alle Funktionalitäten zur Verfügung. Hauptaugenmerk wurde auf die Weiterbenutzbarkeit der Widgets im Offline-Modus gelegt. Im Folgenden sind einige Funktionen aufgelistet, welche während des Offline-Betriebes deaktiviert werden.

- Hinzufügen, Bearbeiten und Löschen von Widgets
- Hinzufügen, Bearbeiten und Löschen von Workspaces
- Drag and Drop von Widgets
- Anlegen und Bearbeiten von Todo-Listen im Todo-Widget

Die Funktionalitäten werden meist einfach ausgeblendet, sobald das System feststellt, dass es sich im Offline-Modus befindet. (TODO: Codebeispiel)

### 6.2.2.3 Preloads der Widgets

Plesynd muss den User in die Lage versetzen nach einmaligem Laden des Systems offline weiterzuarbeiten. Dies ist nur möglich, wenn alle Daten beim erstmaligen Aufruf geladen werden. Insbesondere gilt dies für die IFrames der Widgets. Es müssen die Appcache-Dateien der einzelnen Widgets geladen werden und das Dashboard benötigt die Infos der Widgets zur Ausgabe der zusammenfassenden Informationen direkt bei Systemstart. Der Nutzer soll sich nicht erst durch alle vorhandenen Workspaces bewegen müssen, um die Daten aller Widgets zu laden. Aus diesem Grund müssen die IFrames aller Widgets aller Workspaces bei dem ersten Aufruf im DOM vorhanden sein. Des weiteren dürfen die IFrames bei Wechsel zwischen den Workspaces nicht wieder aus dem DOM entfernt. Der Grund hierfür ist, dass bei Wechsel des Online-Status alle IFrames aktualisiert werden müssen und nicht nur diejenigen des aktuellen Workspaces.

Aus diesen Gründen wurde entschieden alle IFrames bei Systemstart direkt zu laden und im DOM vorzuhalten. Es findet lediglich eine Filterung auf Basis des anzuzeigenden Workspace statt, welche der IFrames ausgegeben werden und welche ausgeblendet werden (TODO: Codebeispiel)

### 6.2.3 Widgets auf Hauptebene

Die ursprüngliche Architektur von Plesynd sah vor, dass der Local Storage nur direkten Zugriff auf die Hauptdatensätze liefert. Dies hätte zur Folge, dass nur Workspaces

direkt referenziert werden könnten. Der Zugriff auf die Widgets würde dann über ihre Workspaces erfolgen. Das Storage/Resource System kann jedoch nur mit Hauptsatzen arbeiten. Dieses Vorgehen hat jedoch einige Probleme und Fragen nach sich gezogen:

- Wie kann man lokal mit Subdatensätzen arbeiten, wenn man keinen direkten Zugriff auf sie hat? Es ist nicht ohne weiteres möglich aus dem Local Storage eines Workspaces ein Widget zu löschen oder zu bearbeiten. Der Workspace müsste geholt, das Widget in dem Workspace gelöscht/bearbeitet und der Workspace wieder geschrieben werden.
- Als Rest Service wäre es kein Problem direkt Widgets zu löschen oder zu bearbeiten, aber dies geht nicht ohne weiteres im Local Storage. Der Local Storage muss aber geupdatet werden, damit das System auch offline mit den korrekten Daten arbeitet.

Es gibt mehrere Möglichkeiten dieses Problem zu lösen:

1. Umstellung auf eine andere Lösung zur lokalen Speicherung der Daten (z.B. IndexedDb)
2. Erweiterung der Implementierung zur Speicherung der Daten im Local Storage, so dass auch Subeinträge gefunden und bearbeitet werden können.
3. Speicherung der Subdatensätze auf der Hauptebene. Alle direkt über die REST-Schnittstelle ansprechbaren Ressourcen werden auf der Hauptebene im Local Storage hinterlegt

Punkt ?? sollte vermieden werden, da er eine grundsätzliche Änderung der Systemarchitektur nach sich gezogen hätte (TODO: ref auf Entscheidung für Local Storage). Punkt 2 kam in die engere Betrachtung wurde jedoch verworfen, da er die Komplexität der Speicherung in den Local Storage beträchtlich erhöht hätte. Wie in 5.3.1 beschrieben, sollte es für die Implementierung der Logik unerheblich sein, ob das System zum Zeitpunkt einer Nutzeraktion online oder offline ist. Um dies mit Punkt 2 zu erreichen, hätte die in 6.2.2.1 vorgestellte Vorgehensweise grundlegend geändert werden müssen. Aus diesen Gründen wurde Punkt 3 umgesetzt. Jede REST-Ressource wird direkt auf der Hauptebene abgelegt. Somit ergibt sich eine einfache 1:1 Abbildung der Ressourcen auf den Local Storage.



#### 6.2.4 Probleme bei multiple Instanzen des selben Widgets

Es ist möglich, dass das selbe Widget mehrfach in einer Plesynd-Instanz verwendet wird. Beispielsweise könnte das TodoList-Widget auf mehreren Workspaces verwendet werden, um je nach Kontext unterschiedliche Todo-Listen zu verwalten. Es können hierbei jedoch Synchronisationsprobleme auftreten, wenn beide Widget-Instanzen offline auf der selben Datenbasis operieren. Der Grund dafür liegt darin, dass die POST-Methode nicht idempotent ist (siehe 4.3.3). Da die unterschiedlichen Widget-Instanzen nicht in Kommunikation miteinander stehen bedeutet dies im Falle einer Synchronisierung, dass beide Instanzen lokal hinzugefügte Datensätze per POST an den Server schicken. Somit entstehen nicht gewollte Dopplungen der Datensätze. DELETE und PUT Aufrufe stellen für diesen Fall keine Probleme dar, da diese Methoden idempotent sind und ein doppelter DELETE Request an eine Resource keine negativen Auswirkungen hat. Dieses Problem wurde gelöst, in dem jede Widget-Instanz seinen eigenen Local-Storage zur Speicherung seiner Daten erhält. Der Name des Local Storage ergibt sich aus dem internen Namen des Widgets (beispielsweise todo) konkateniert mit dem Namen des IFrames in dem das Widge aufgerufen wird. Der Name des IFrames ergibt sich aus der ID, welche Wookie für jedes Widgets erstellt. Diese ID ist einzigartig und verändert sich auch bei wiederholtem Aufrufen nicht. Somit ist sichergestellt, dass jedes Widget einen festen Local Storage erhält.

Es ist natürlich möglich, dass unterschiedliche Widget-Instanzen zwar unterschiedlichen Local Storage benutzen, jedoch auf der selben Datenbasis arbeiten (beispielsweise die selben Todo-Listen verwenden). Da momentan noch keine Inter-Widget Kommunikation stattfindet kann es hierbei zu Problemen kommen, wenn im Offline Betrieb beispielsweise ein Datensatz in einer Instanz gelöscht wird, während er in einer anderen bearbeitet wurde. Dieses Problem wurde im Rahmen dieser Arbeit nicht bearbeitet und sollte eventuell in Folgearbeiten bearbeitet werden.

#### 6.2.5 Drag'n Drop

Zur Umsetzung des Drag and Drop Funktionalität zum Sortieren der Widgets auf einem Workspace wurde das [Sortable Widget](http://jqueryui.com/sortable/)<sup>8</sup> der [JqueryUi-Bibliothek](http://jqueryui.com)<sup>9</sup> verwendet. Diese wurde in einer eigenen AngularJS-Direktive gekapselt, so dass das JqueryUi-Widget in der bestehenden Architektur verwendet werden konnte. Die Direktive verwendet den Widget-Service(TODO: Ref), um die neue Position der Widgets dem Server mitzuteilen

---

<sup>8</sup><http://jqueryui.com/sortable/>

<sup>9</sup><http://jqueryui.com>

und den OnlineStatus-Service (TODO:ref), um die Drag and Drop Funktionalität im Offline-Modus zu deaktivieren

(TODO: Codebeispiel)

### **6.2.6 CORS**

Beschreiben wie Cors umgesetzt, Das NelmioCorsBundle6.1.3 wird auf Serverseite eingesetzt um die Arbeit mit Cors zu erleichtern. Es ermöglicht die erlaubten Header über Config Einstellungen (TODO: beispiel) zu definieren und

### **6.2.7 Kommunikation zwischen den IFrames**

(TODO: Bib: <https://developer.mozilla.org/en-US/docs/DOM/window.postMessage>) Post-message:

## Kapitel 7

# Zusammenfassung/Ausblick

Kurzfassung des Problems und wie es gelöst wurde Usb-Stick, mobile Version der Browser Tabelle der Anforderungen , was geschafft

### 7.0.8 Offene Fragen für weitere Forschungsarbeiten

Änderung von HttpAuth zu Token Basierter Authentifizierung Widget Authentifizierung Stabile Wookieversion benutzen Websockets Serverpush bessere Möglichkeit zu Zusammenarbeit Änderung der Click Events auf Touch Events

# Anhang A

## Installation/Deployment

Write your Appendix content here.

### A.1 Installation

#### A.1.1 Wookie

Die Datei `local.widgetserver.properties` muss direkt im Wookie Dir angelegt werden. Achtung, hier kann nichts hinterlegt werden, auf das schon der Apache hört (also port 80) Install etc

#### A.1.2 Plesynd

### A.2 Deployment

#### A.2.1 Plesynd

# Literaturverzeichnis

- [1] Jonathan Mott. Envisioning the Post-LMS Era : The Open Learning Network The PLE and Personal Learning Networks. *EDUCAUSE Quarterly*, 33(1):1–8, 2010. URL <http://www.educause.edu/EDUCAUSE+Quarterly/EDUCAUSEQuarterlyMagazineVolum/EnvisioningthePostLMSEraTheOpe/199389>.
- [2] Sandra Schaffert and Wolf Hilzensauer. On the way towards Personal Learning Environments : Seven crucial aspects. *eLearning Papers*, 9(July):1–11, 2008. ISSN 18871542. URL <http://www.elearningeuropa.info/files/media/media15971.pdf>.
- [3] Bundesministeriums für Bildung und Forschung. Lebenslanges Lernen. in der Internet Archive-Version vom 22. August 2008 der Seiten des Bundesministeriums für Bildung und Forschung <http://web.archive.org/web/20080822031749/http://www.bmbf.de/de/411.php> Abrufdatum: 10.01.2013, August 2008.
- [4] Graham Attwell. Personal Learning Environments - the future of eLearning? *eLearning Papers*, 2(1):1–8, 2007. ISSN 18871542. URL <http://66.102.1.104/scholar?hl=en&lr=&q=cache:1EYPuXfu8kAJ:dialnet.unirioja.es/servlet/articulo?codigo=2202159&orden=99556&info=link>.
- [5] M. van Harmelen. Personal Learning Environments. *Sixth IEEE International Conference on Advanced Learning Technologies (ICALT'06)*, pages 815–816. doi: 10.1109/ICALT.2006.1652565. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1652565>.
- [6] Matthias Palmér, Stéphane Sire, Evgeny Bogdanov, Denis Gillet, and Fridolin Wild. Mapping Web Personal Learning Environments. *Proceedings of the First International Workshop on Mashup Personal Learning Environments MUPPLE08*, 16(1): 1–2, 2009.
- [7] Scott Wilson. Patterns of Personal Learning Environments. *Interactive Learning Environments*, 16(1):17–34, April 2008. ISSN 1049-4820. doi: 10.1080/

10494820701772660. URL <http://www.tandfonline.com/doi/abs/10.1080/10494820701772660>.
- [8] Jesse Ruderman. Same-Origin-Policy. [https://developer.mozilla.org/en-US/docs/JavaScript/Same\\_origin\\_policy\\_for\\_JavaScript](https://developer.mozilla.org/en-US/docs/JavaScript/Same_origin_policy_for_JavaScript) Abrufdatum: 08.01.2013, 2012.
- [9] Anne van Kesteren. Cross-Origin Resource Sharing. CTAN: <http://www.w3.org/TR/cors/> Datum: 08.01.2013, April 2012.
- [10] Monsur Hossain. Using CORS. <http://www.html5rocks.com/en/tutorials/cors/> Abrufdatum: 08.01.2013, September 2012.