

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

11. 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

Declaration of Authorship	i
Abstract	iii
Acknowledgements	iv
List of Figures	vii
List of Tables	viii
Abbreviations	ix
Physical Constants	x
Symbols	xi
1 Einleitung	1
1.1 Einleitung	1
2 Hintergrund/Begriffsklärung	2
3 Anforderungsanalyse	3
3.1 Use Case	3
3.1.1 Ziel	3
3.1.2 Voraussetzungen	3
3.1.3 Workflow	4
3.2 Anforderungen	4
4 Übersicht Stand der Forschung/Technik	6
4.1 PLE	6
4.1.1 Definition einer PLE	6
4.1.2 Klassifizierungsmethoden	7
4.1.2.1 Dimensionen nach Palmér	7
4.1.2.2 Wilson Patterns	8
4.2 Widgets	8
4.2.1 CORS	9

4.2.2	HTML5	11
4.2.2.1	Appcache	11
4.2.2.2	Local Storage	11
4.2.3	REST	11
4.2.3.1	Widgetparameter	11
5	Lösungsansatz	12
5.1	Überblick	12
5.2	Design	12
5.3	Technisch	12
5.3.1	Unabhängigkeit Offline/Online	12
6	Details zur Lösung	13
6.1	Entwicklungsumgebung/Tools	13
6.1.1	Symfony2	13
6.1.2	eigene Bundles	14
6.1.3	Fremd-Bundles	14
6.1.4	Doctrine2	14
6.1.5	AngularJS	14
6.1.6	Jquery	14
6.1.7	Twitter Bootstrap	15
6.1.8	Apache Wookie	15
6.2	Implementierungsdetails	15
6.2.1	Ablauf holen der Daten	15
6.2.2	Umsetzung der Offline-Fähigkeiten	15
6.2.2.1	Local Storage	15
6.2.2.2	Einschränkungen im Offline-Betrieb	16
6.2.2.3	Preloads der Widgets	16
6.2.3	Widgets auf Hauptebene	16
6.2.4	Probleme bei multiple Instanzen des selben Widgets	18
6.2.5	Drag'n Drop	18
6.2.6	CORS	19
6.2.7	Kommunikation zwischen den IFrames	19
7	Zusammenfassung/Ausblick	20
7.0.8	Offene Fragen für weitere Forschungsarbeiten	20
A	Installation/Deployment	21
A.1	Installation	21
A.1.1	Wookie	21
A.1.2	Plesynd	21
A.2	Deployment	21
A.2.1	Plesynd	21

Literaturverzeichnis	22
-----------------------------	-----------

Abbildungsverzeichnis

Tabellenverzeichnis

3.1 Anforderungen	5
-----------------------------	---

Abbreviations

DOM Document Object Model

Physical Constants

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

Symbols

a	distance	m
P	power	W (Js^{-1})
ω	angular frequency	rads^{-1}

For/Dedicated to/To my...

Kapitel 1

Einleitung

1.1 Einleitung

Das Ziel der Masterarbeit ist die Entwicklung 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 unterschiedlicher 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

eventuell raus (Kapitel 4 erweitern)

Kapitel 3

Anforderungsanalyse

Das Ziel der vorliegenden Arbeit ist die Planung und Implementierung einer leichtgewichtigen Personal-Learning-Environment. Die Anforderungen an das 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 zu entwickelnde System idealerweise abgedeckt werden:

3.1.1 Ziel

Das Ziel ist die Möglichkeit der Betreuung eines Studienkurses über das Internet. Es soll hierbei irrelevant sein, ob Studenten und Kursbetreuer sich in dem gleichen geografischen Gebiet aufhalten oder zur gleichen Zeit Zugriff auf das System haben.

3.1.2 Voraussetzungen

Der Kursbetreuer oder Mentor des Kurses hat seinen Sitz in Deutschland. Die Studenten befinden sich größtenteils in Kamerun. Dort steht nicht zu jeder Zeit ein Internetzugang zur Verfügung. Des weiteren rechnen die Internetprovider oft nach Zeit und nicht nach Volumen ab, so dass der Nutzer auch bei dem Vorhandensein eines Internetzugangs nicht zwingend online sein muss. Für viele Studenten ist es die Regel, dass sie ihre Kursarbeit an unterschiedlichen Computern verrichten. Bei einem Teil der Rechner steht ihnen ein

Internetzugang zur Verfügung (beispielsweise in der Universität oder in einem Internet-café) und für den restlichen Teil ist dies nicht der Fall (beispielsweise zu Hause). Durch die Arbeit an unterschiedlichen Rechnern mit potentiell unterschiedlichen Betriebssystemen, ist die Installation einer komplexen Software nicht ohne Weiteres möglich.

3.1.3 Workflow

Betreuer und Studenten stehen über unterschiedliche Kanäle in Kommunikation miteinander. 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 Internetcafé oder zu von Hause aus, arbeiten.

3.2 Anforderungen

Aus dem beschriebenen Use-Case 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 Browser-technologien 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 unterschiedlichen 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 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

Kapitel 4

Übersicht Stand der Forschung/Technik

Darstellung existierender Ansätze (ggf. Klassifikation dieser Ansätze) Analyse der Ansätze in Bezug auf die Anforderungen Zusammenfassung der Defizite des Stands der Forschung

E-Learning E-Learning 1.0 - 2.0 von kurs zentriert zu user zentriert

E-learning is a technological infrastructure with applications and software that manage courses and users. The software that facilitates e-learning may be called a Learning Management System (LMS) and supports course creation, content delivery, user registration, monitoring, and certification.

Gonella and Panto (2008), in their paper on didactic architectures, have traced the following four stages in the evolution of e-learning: 1. Web-based Training 2. E-learning 1.0 3. Online Education 4. E-learning 2.0

4.1 PLE

4.1.1 Definition einer PLE

Was ist eine PLE? Worin unterscheidet sie sich von anderen Lernumgebungen? Welche Ansätze gibt es?

4.1.2 Klassifizierungsmethoden

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 (TODO cite) und zum anderen gibt es Wilson. Diese beiden Ansätze werden im Folgenden vorgestellt .

4.1.2.1 Dimensionen nach Palmér

Palmér definiert sechs Dimensionen mit denen er so viele relevanten Funktionalitäten von PLEs wie möglich erfassen möchte. 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.
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:*

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.

4.1.2.2 Wilson Patterns

4.2 Widgets

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

Wookie während der Entwicklung vom Incubator zum vollwertigen Apache Projekt

4.2.1 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[1]. 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 `scripti...i/scripti` 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)[2] 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 [3]):

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.

4.2.2 HTML5

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

4.2.2.1 Appcache

Eine wichtige Neuerung in Html

4.2.2.2 Local Storage

Local Storage

4.2.3 REST

Rest

4.2.3.1 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

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)¹. Der clientseitige Code wurde mit [JavaScript](http://de.wikipedia.org/wiki/JavaScript)² implementiert.

Die Frameworks im Folgenden kurz vorgestellt.

6.1.1 Symfony2

[Symfony2](http://symfony.com)³ 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)⁴. 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.

¹<http://php.net>

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

³<http://symfony.com>

⁴http://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](#)⁵ 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.

⁵<http://jquery.com/>

6.1.7 Twitter Bootstrap

[Twitter Bootstrap](#)⁶ 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](#)⁷ 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

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

⁷<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 4.2.3). 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/)⁸ der [JqueryUi-Bibliothek](http://jqueryui.com)⁹ 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

⁸<http://jqueryui.com/sortable/>

⁹<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] Jesse Ruderman. Same-Origin-Policy. https://developer.mozilla.org/en-US/docs/JavaScript/Same_origin_policy_for_JavaScript Abrufdatum: 08.01.2013, 2012.
- [2] Anne van Kesteren. Cross-Origin Resource Sharing. CTAN: <http://www.w3.org/TR/cors/> Datum: 08.01.2013, April 2012.
- [3] Monsur Hossain. Using CORS. <http://www.html5rocks.com/en/tutorials/cors/> Abrufdatum: 08.01.2013, September 2012.