Projektarbeit

Mashup-Plattform

Version 1.0

Projekttitel **Mashup-Plattform**	Projektbeginn **19.08.2011**	Projektende **07.01.2012**
Projektteam **Alexander Matt, Joël Poyet, Jan Wild, Malte Wintner**		

Inhaltsverzeichnis

1 Projektauftrag

1.1 Formale Angaben

1.1.1 Zweck dieses Dokuments

In diesem Dokument werden die Projektziele, Rahmenbedingungen und Anforderungen beschrieben.

1.1.2 Ablage

Dieses Dokument befindet sich in seiner einzig gültigen Fassung im teaminternen Git-Repository.
Die Zugangsdaten und Instruktionen zur Nutzung des Git-Repository haben alle Projektteilnehmer per Email erhalten.

1.2 Ausgangslage

Das Internet besteht aus einer riesigen Flut von uneinheitlich strukturierten Informationen. Diese Heterogenität verhindert die effiziente Aufnahme und Verarbeitung der Daten durch Mensch und Maschine.
Die Grundidee dieser Projektarbeit ist die Verknüpfung und Strukturierung unterschiedlicher Datenquellen, um sie in standardisierter Form den Anwendern zur Verfügung zu stellen.
Sogenannte Web-Mashups verfolgen dieses Ziel.

Die Vorteile für die Anwender sind unter anderem:

- bessere Übersicht von Informationen
- Informationsgewinn durch logische Verknüpfungen
- Individualisierung von Informationen

Web-Mashups sind im Allgemeinen noch relativ unbekannt und gewinnen langsam an Bedeutung. Plattformen für die Erstellung und Nutzung von Web-Mashups gibt es bis jetzt nur wenige und sind in erster Linie für die Bereitstellung von unternehmensinternen Informationen (z.B. in Form von „Dashboards“) konzipiert worden.

1.3 Projektziele

1.3.1 Funktionsziele

1.3.1.1 Gesamtziel

Das Ziel dieser Projektarbeit ist die Entwicklung einer Mashup-Plattform, welche den Anwendern das Betrachten, Erstellen und Ausführen von (Web-)Mashups ermöglicht.

1.3.1.2 Detailziele

Zielart	Ziel
MUSS-Ziele	Die Mashup-Plattform stellt die notwendigen Funktionen und Mechanismen für das Definieren, Hinzufügen und Ausführen eigener Mashups bereit.
	Mashups müssen als Datenquellen für neue Mashups nutzbar sein.
	Die Mashup-Plattform steuert und kontrolliert den Zugriff auf das Internet in Bezug auf das Datenvolumen, die Anzahl Requests und die erlaubten Datenquellen.
KANN-Ziele	Auf dem Server abgelegte Dokumente können als Datenquellen genutzt werden.
	Besucher können mit Hilfe eines gängigen Webbrowsers auf die Ergebnisse der Mashups zugreifen.
	Ein automatischer Email-Versandmechanismus kann zur Benachrichtigung neuer Mashup-Inhalte eingesetzt werden.
NICHT-Ziele	Ein GUI für das Definieren von Mashups ist aus zeitlichen Gründen nicht zu entwickeln.

	Das Endprodukt dieser Projektarbeit ist keine für den produktiven Einsatz ausgereifte Lösung. Es dient viel mehr als Experimentiersystem, um die technischen Möglichkeiten und Einschränkungen der Mashup-Entwicklung aufzuzeigen.

1.3.1.3 Messbarkeit

Die Mashup-Plattform sollte mindestens ein vorgefertigtes Beispiel-Mashup enthalten, welches die Funktionalität der Plattform demonstriert.
Anhand dieses Beispiel-Mashups kann der Grad der Zielerreichung bewertet werden.

1.3.2 Finanzziele

1.3.2.1 Gesamtziel

Das Endprodukt wird als quelloffene Software unter der GNU General Public License (GPL Version 3) veröffentlicht. Das Projekt ist somit nicht primär gewinnorientiert.

1.3.2.2 Detailziele

Zielart	Ziel
MUSS-Ziele	Entwicklung und Verbreitung des Produktes dürfen keine Lizenzkosten generieren.

1.3.3 Personalziele

1.3.3.1 Gesamtziel

Das Projektteam soll ein vertieftes Verständnis im Bereich Projektmanagement, Software Engineering und Objektorientiertes Programmieren erhalten.
Der erfolgreiche Abschluss soll dem Team Vertrauen in seine Fähigkeiten bringen.

1.3.3.2 Detailziele

Zielart	Ziel
MUSS-Ziele	Jedes Teammitglied hat detaillierte Kenntnisse über den Projektverlauf.
	Jedes Teammitglied ist in der Lage, die an der Schlussprüfung gestellten Fragen fachkundig zu beantworten.
	Alle Projektaufgaben werden termingerecht und in der geforderten Qualität erfüllt.

1.4 Rahmenbedingungen

1.4.1 Technische Restriktionen

- Als Programmiersprache wird hauptsächlich Java eingesetzt.
- Eingesetzte Fremdbibliotheken müssen quelloffen sein.
- Die Mashup-Plattform darf fremde Ressourcen (z.B. externe Datenquellen) nicht übermässig beanspruchen. Dies gilt auch für die Entwicklung und dem Testen der Plattform.

1.4.2 Rechtliche Restriktionen

- Es dürfen nur Datenquellen genutzt werden, welche den maschinellen Zugriff erlauben. Der Datenaustausch sollte über die vorgesehenen Schnittstellen erfolgen, sofern die Fremdquellen entsprechende APIs zur Verfügung stellen.
- Es dürfen keine Datenschutzbestimmungen verletzt werden.
- Eingesetzte Fremdbibliotheken müssen mit der GNU General Public License (GPL Version 3) vereinbar sein.

1.4.3 Personelle Restriktionen

- Das Team muss die Richtlinien gemäss der „Arbeitsmappe für das Modul Projektarbeit“ befolgen.
- Jedes Mitglied ist verpflichtet, sich das notwendige Hintergrundwissen anzueignen, um dieses Projekt erfolgreich durchzuführen.
- Die Arbeitszeiten der Teilnehmer beschränken sich vorwiegend auf Abende und Wochenenden.
- Alle Mitglieder sind zu gleichen Teilen am Projekt beteiligt und müssen mindestens den geforderten Arbeitsaufwand von 210 Stunden (= 7 ECTS-Credits) einhalten.

1.4.4 Technische Ressourcen

Für die Umsetzung des Projekts sind folgende Ressourcen sicherzustellen:

- Einwandfrei funktionierende Entwicklungsplattform mit Anbindung zu einem gemeinsam genutzten GIT-Repository
- Ein Skype-Konto mit entsprechendem Zubehör
- Word-kompatibles Textverarbeitungsprogramm

1.5 Grundlagen

Es sind noch keine Vorarbeiten für dieses Projekt geleistet worden.
Als Grundlagen dienen die aus den vorangegangenen Modulen (Projektmanagement, Software Engineering, Programmieren in JAVA, ...) erworbenen Kenntnisse.
Als wichtigste Informationsquelle in Bezug auf die Web-Programmierung und API-Anbindungen dient das Internet.

1.6 Messbare Lieferobjekte

Titel	Kapitel gemäss Arbeitsmappe
Projektauftrag (vorliegendes Dokument)	AM 2.1
begründeter Lösungsansatz	AM 2.2
Projektstrukturplan	AM 2.3
Phasenplan	AM 2.4
Terminplan	AM 2.5
Systemidee	AM 3.1.1
Systemanwendungsfälle	AM 3.1.2
Domänenmodell	AM 3.1.3
Systemablaufmodelle	AM 3.1.4
Schnittstellenbeschreibung	AM 3.1.5
Softwarearchitektur	AM 3.2.1
Klassenmodelle	AM 3.2.2
Datenmodell	AM 3.2.3
Dynamische Modelle	AM 3.2.4
Testkonzept	AM 3.2.5
Spezifikation der Bedienoberflächen	AM 3.2.6
Lieferobjekte und Installationsanleitung	AM 3.3.1
Coderichtlinien	AM 4.2.1
Quellcode (abgelegt auf USB-Stick)	AM 4.2.2
Ablagestruktur des Quellcodes	AM 4.2.2
Testprotokolle	AM 4.3

1. Statusbericht	AM 5.1
2. Statusbericht	AM 5.1

1.7 Eckwerte

1.7.1 Termine (Meilensteine)

Projektstart / Kick-Off-Meeting	19.08.2011
Initialisierung	
Projektidee fixieren	26.08.2011
Anforderungen und Ziele analysieren	30.08.2011
Kommunikation und Hilfsmittel sicherstellen	02.09.2011
Projektauftrag schreiben	02.09.2011
Vorstudie	
Grobplan erstellen	05.09.2011
Projektstrukturplan erstellen	06.09.2011
Konzept	
Detailziele und Lösungsvarianten festlegen	09.09.2011
System gemäss ausgewählter Lösung entwerfen	16.09.2011
Implementierung	
Mashup-Platfform implementieren	02.12.2011
Test & Einführung	
Testphase abschliessen	15.12.2011
Installations- und Benutzerhandbuch schreiben	19.12.2011
Projektabschluss	
Nachbesprechung des Projektteams durchführen	22.12.2011
Dokumentation ausliefern	23.12.2011
Projekt präsentieren	07.01.2012
Projektende	07.01.2012

1.7.2 Personalplanung / Rollenverteilung

In diesem Projekt sind folgende Rollen vertreten:

- Auftraggeber
- Projektleiter
- Projektmitarbeiter

Jedes Mitglied wird jede Rolle mindestens einmal einnehmen.
Die Rollenverteilung bis zum ersten Status-Meeting sieht wie folgt aus:

Rolle	Name
Auftraggeber	Joël Poyet
Projektleiter	Malte Wintner
Projektmitarbeiter	Alexander Matt
Projektmitarbeiter	Jan Wild

1.8 Risiken

1.8.1 technische Risiken

- Aktualität und Korrektheit der Mashup-Inhalte können nicht garantiert werden, da sie von Fremddatenquellen abhängig sind.

- Eine zu hohe Beanspruchung der Fremddatenquellen könnte zu einer Zugriffs-Sperrung („Blacklisting“) führen.

1.8.2 gesetzliche Risiken

- Die Anzeige von Fremdinhalte könnten Datenschutzbestimmungen verletzen.

- Mashups könnten zu Spam-Zwecken missbraucht werden.

1.8.3 marktbezogene Risiken

- Das allgemeine Interesse an der Mashup-Plattform könnte gering sein, da für die Erstellung der Mashups Fachkenntnisse erforderlich sind.

- Ende Projektauftrag -

2 Begründeter Lösungsansatz

2.1 IST-Situation

Das Team steht vor der Herausforderung, ein System innerhalb von vier Monaten zu entwickeln, das die im Projektauftrag beschriebenen Anforderungen bestmöglich erfüllt.
Als zentraler Faktor für den Erfolg dieses Projektes wird der effiziente Umgang mit der Zeit betrachtet. Jedes Teammitglied ist berufstätig und belegt nebst der Projektarbeit noch weitere Module an der FFHS.
Um den Programmieraufwand zu verkürzen, ist die (Make-or-Buy-)Frage nach dem Einsatz von Fremdkomponenten ein wichtiges Thema geworden.

2.2 SOLL-Situation

Die Mashup-Plattform sollte mit Hilfe von definierten Regeln die gewünschten Daten aus unterschiedlichen Fremddatenquellen herunterladen und sie miteinander verknüpfen. Diese Funktionalität bildet den Kern des Systems und muss mit höchster Priorität implementiert werden.

2.3 IST/SOLL-Analyse

Die IST/SOLL-Analyse hat ergeben, dass der Einsatz von Fremdkomponenten unter Umständen den Programmieraufwand reduzieren könnte. Die in Frage kommenden Fremdkomponenten müssen allerdings die Kernfunktionalität (gemäss SOLL-Situation) unterstützen und dabei keine Rahmenbedingungen (gemäss Projektauftrag) verletzen.

Eine vom Projektteam durchgeführte Internetstudie hat zwei Fremdkomponenten geliefert, welche die geforderten Kriterien erfüllen:

Bezeichnung	Erklärung
Yahoo YQL	Ein von Yahoo zur Verfügung gestellter Webdienst, der den einheitlichen Zugriff auf Fremddatenquellen mit Hilfe einer SQL-ähnlichen Abfragesprache namens „YQL“ erlaubt. (Quelle: http://developer.yahoo.com/yql/)
JOpera	Ein System zur Verknüpfung von Fremddatenquellen. Es besteht aus einer Workflow Engine (Kernel) und aus einer graphischen Benutzeroberfläche. (Quelle: http://www.jopera.org)

Die geringe Anzahl der möglichen Fremdkomponenten bekräftigt das Indiz, dass die Mashup-Entwicklung noch ein junges Forschungsthema ist.

2.4 Lösungsvarianten

Varianten-Nr.	Erklärung
V1	Die zukünftige Mashup-Plattform greift über Yahoo's Webdienst auf das Internet zu. Ein grosser Teil der Mashup-Logik wird in die Abfragesprache „YQL“ verlagert.
V2	Die Workflow Engine von JOpera wird eingesetzt, um Mashups auszuführen.
V3	Es werden keine Fremdkomponenten eingesetzt. Die Mashup-Plattform wird von Grund auf neu entwickelt.

2.5 Bewertungskriterien

Kriterien-Nr.	Gewichtung	Kriterium
B1	10	Konfigurations-/Programmieraufwand
B2	10	Erweiterbarkeit
B3	10	Klarheit und Exaktheit der Mashup-Definierung
B4	7	Unabhängigkeit von Drittentwicklern
B5	6	Sicherheit und Stabilität
B6	5	Dokumentation und Aktivität der Community

B7	5	Lernerfolg für das Team

2.6 Nutzwertanalyse

Die Punktevergabe entspricht der in der Schweiz gebräuchlichen Notenskala (1 sehr schlecht, 6 sehr gut).

	B1	B2	B3	B4	B5	B6	B7	Total (Gewichtung berücksichtigt)
V1	5	5*	5	1	4	5	3	221
V2	3	4	4**	3	5	3	5	201
V3	3	5	5	6	4	1	6	231

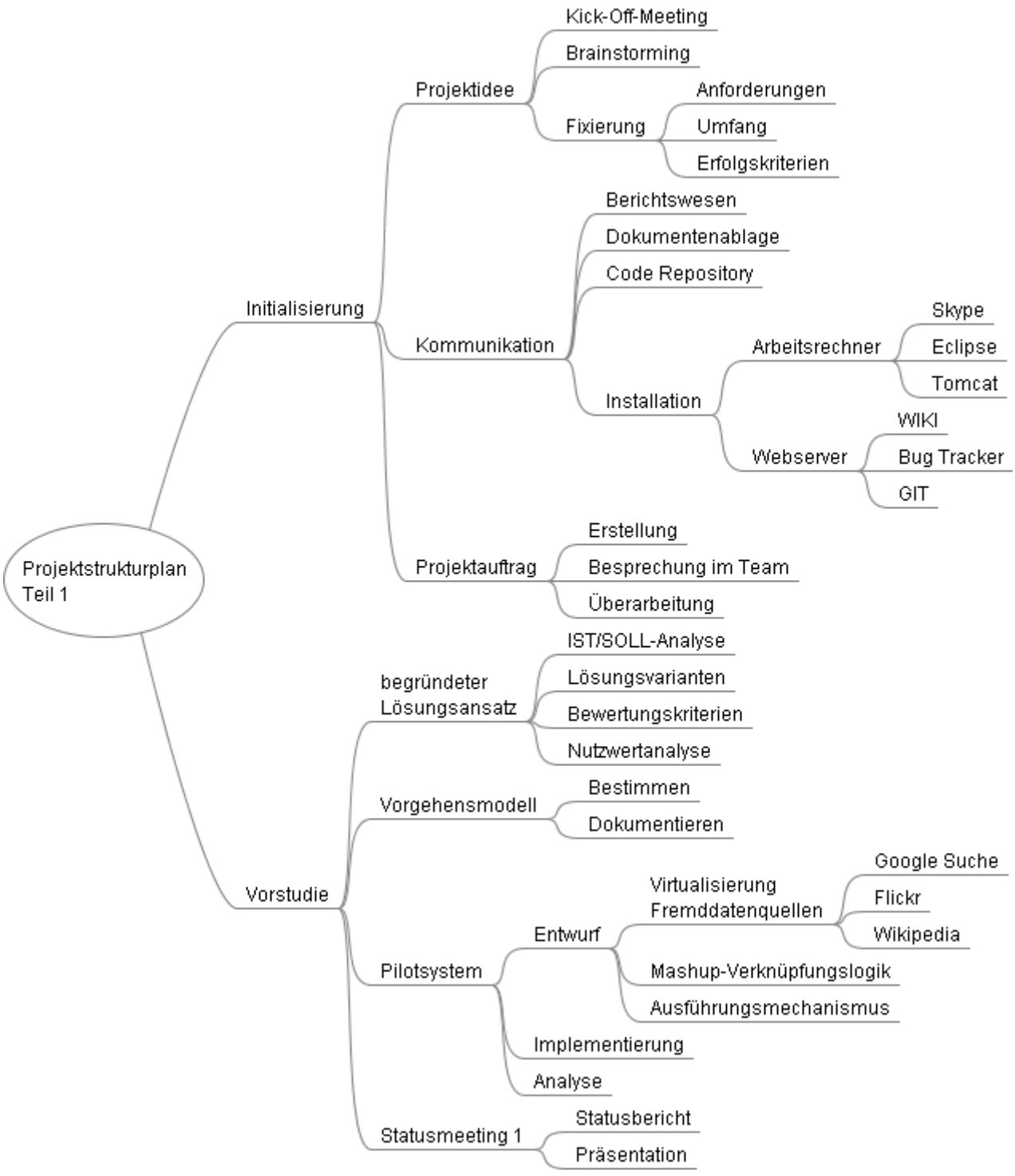
* Yahoo bietet das Konzept der „Open Data Tables“ an (Quelle: http://www.datatables.org/).
** JOpera hat seinen Hauptfokus auf das visuelle Verknüpfen, wodurch die Exaktheit und Flexibilität der Mashup-Definierung etwas verloren geht.

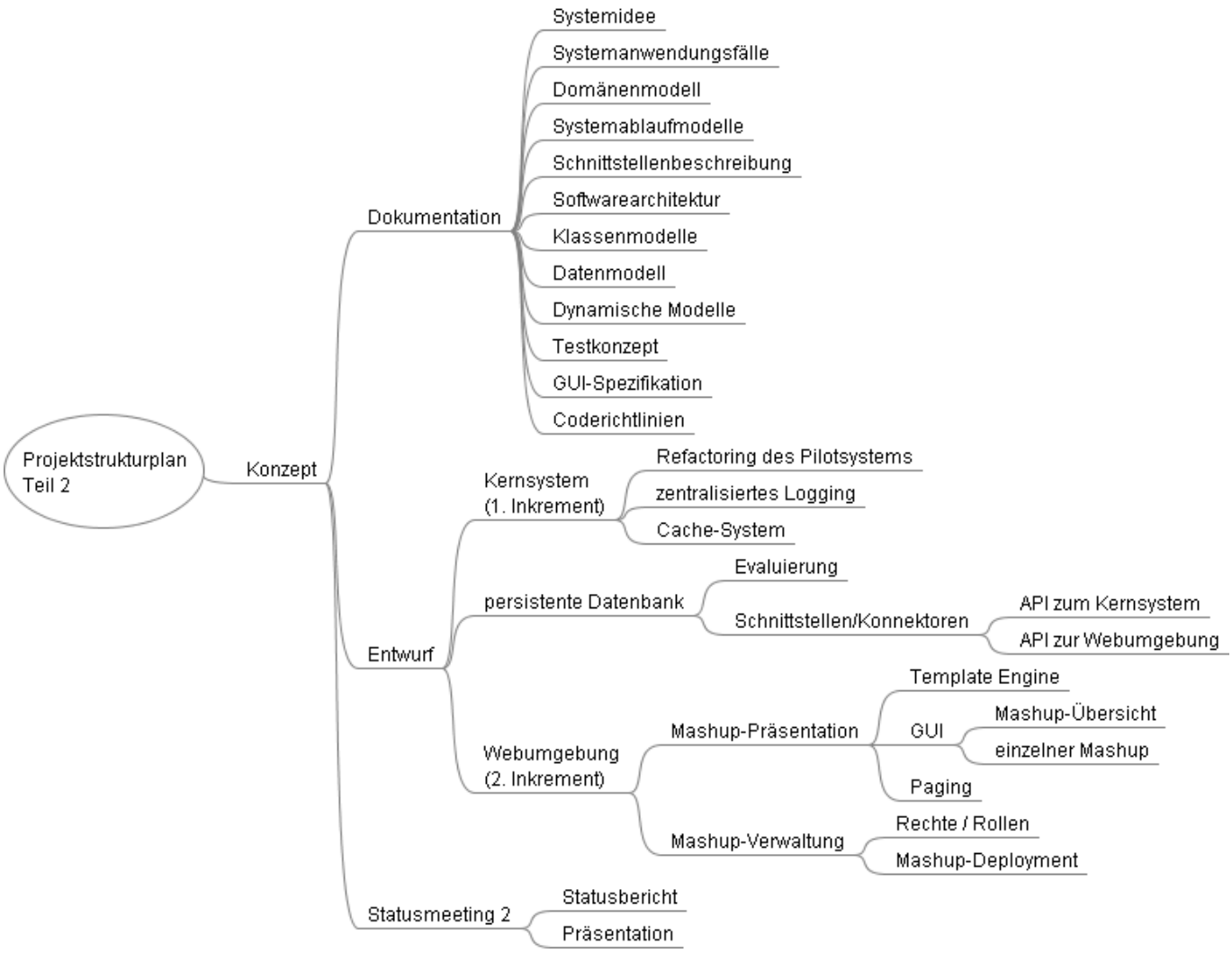
2.7 Ausgewählte Lösung

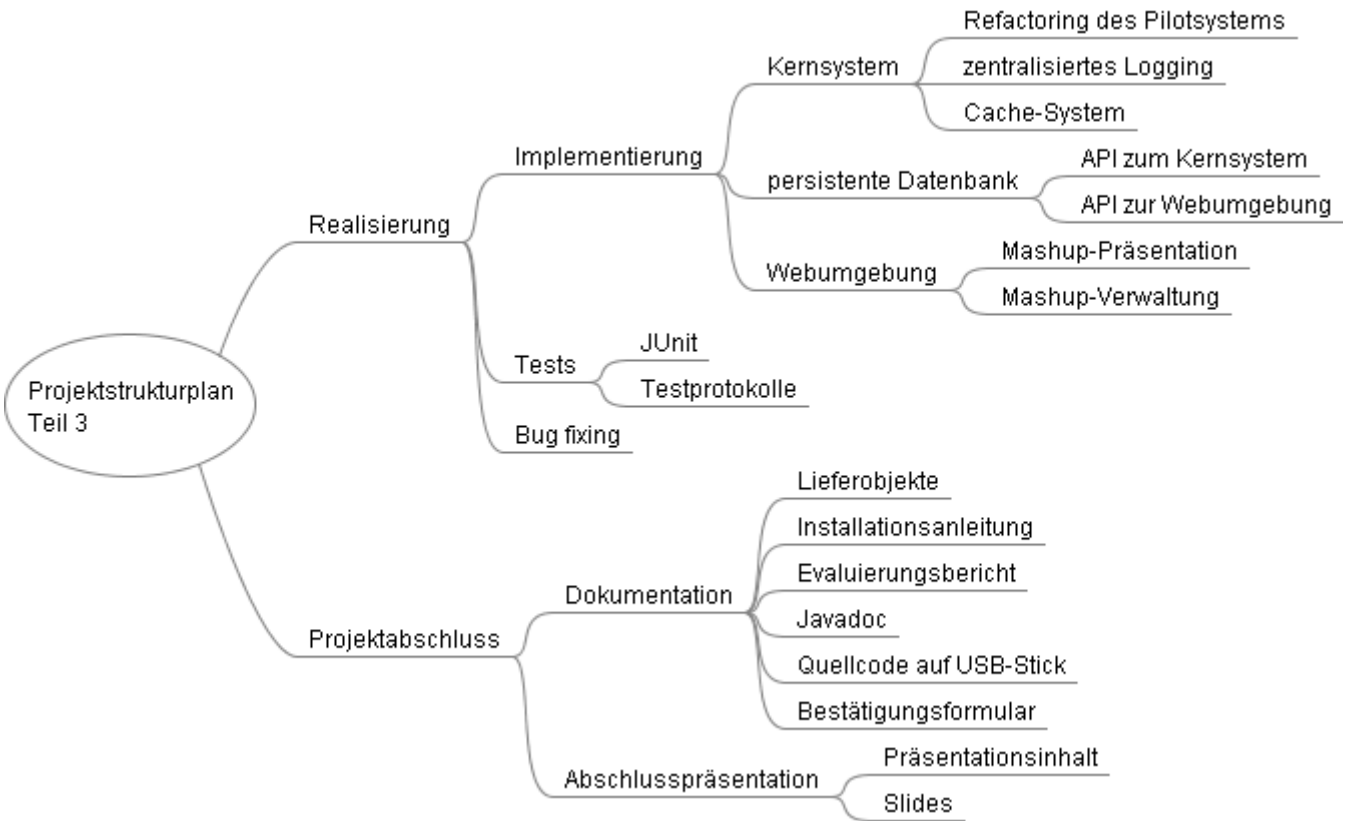
Die Nutzwertanalyse liefert ein interessantes Ergebnis. Die komplette Eigenentwicklung der Mashup-Plattform bringt die meisten Vorteile.
Ein wesentlicher Grund liegt in der kleinen Auswahl an möglichen Fremdkomponenten. Ausserdem hat die Eigenentwicklung den grössten Lerneffekt auf das Team und ermöglicht die grösstmögliche Unabhängigkeit von Dritten.

- Ende Begründeter Lösungsansatz -

3 Projektstrukturplan




Projektstrukturplan
Teil 2
Konzept
Dokumentation
Systemidee
Systemanwendungsfälle
Domänenmodell
Systemablaufmodelle
Schnittstellenbeschreibung
Softwarearchitektur
Klassenmodelle
Datenmodell
Dynamische Modelle
Testkonzept
GUI-Spezifikation
Coderichtlinien
Entwurf
Kernsystem
(1. Inkrement)
Refactoring des Pilotsystems
zentralisiertes Logging
Cache-System
persistente Datenbank
Evaluierung
Schnittstellen/Konnektoren
API zum Kernsystem
API zur Webumgebung
Webumgebung
(2. Inkrement)
Mashup-Präsentation
Template Engine
GUI
Mashup-Übersicht
einzelner Mashup
Paging
Mashup-Verwaltung
Rechte / Rollen
Mashup-Deployment
Statusmeeting 2
Statusbericht
Präsentation



- Ende Projektstrukturplan -

4 Phasenplan

4.1 Zeichenerklärung

Ax.x: Arbeitspaket **M**x.x: Meilenstein

4.2 Initialisierung

Nummer	Bezeichnung	Beschreibung
Projektidee		
A1.1	Kick-Off-Meeting	Projektthema wählen
A1.2	Brainstorming	Projektthema konkretisieren
A1.3	Fixierung	Anforderungen, Umfang und Erfolgskriterien grob skizzieren
M1.1	Mindmap	Übersicht über alle vereinbarten Entscheidungen
Kommunikation		
A1.4	Berichtswesen	Berichtswesen etablieren (Wer berichtet wem? In welcher Form?)
A1.5	Dokumentenablage	Software evaluieren
A1.6	Code Repository	Software evaluieren
A1.7	Arbeitsrechner	Entwicklungsumgebung einrichten
A1.8	Webserver	Software (WIKI, Code Repository und Bug Tracker) aufsetzen
M1.2	Arbeitsumgebung	lauffähige Entwicklungsumgebung und Kommunikations-Plattform
Projektmanagement		
A.1.9	Projektauftrag	
A.1.10	Projektstrukturplan	
A.1.11	Phasenplan	
A.1.12	Terminplan	
M1.3	Dokumente	fertig erstellter Projektauftrag

4.3 Vorstudie

Nummer	Bezeichnung	Beschreibung
Dokumentation		
A2.1	begründeter Lösungsansatz	- IST/SOLL-Analyse durchführen - Lösungsvarianten und Bewertungskriterien definieren - Nutzwertanalyse erstellen
A2.2	Vorgehensmodell	Vorgehensmodell bestimmen und dokumentieren
M2.1	Dokumente	schriftlich erfasster Lösungsansatz und Vorgehensmodell.
Entwurf des Pilotsystems		
A2.3	Virtualisierung der Fremddatenquellen	Einheitliche Schnittstellen und Wrapper-Klassen für die Fremddatenquellen (Google-Suche, Flickr und Wikipedia) definieren
A2.4	Mashup-Verknüpfungslogik	Verknüpfungslogik der virtualisierten Fremddatenquellen skizzieren

A2.5	Ausführungsmechanismus	Initialisierung, Ausführung und Speicherung der Ausgabe spezifizieren
M2.2	Klassendiagramme	schriftlich erfasste Klassendiagramme
Implementierung des Pilotsystems		
A2.6	Virtualisierung der Fremddatenquellen	gemäss A2.3
A2.7	Mashup-Verknüpfungslogik	gemäss A2.4
A2.8	Ausführungsmechanismus	gemäss A2.5
M2.3	Pilotsystem	lauffähige Software
Analyse des Pilotsystems		
A2.9	Analyse	- Machbarkeit und Erweiterbarkeit analysieren - Risiken identifizieren
M2.4	Analyse-Ergebnisse	schriftlich erfasste Ergebnisse und Schlussfolgerungen
1. Statusmeeting		
A2.10	Statusbericht	Statusbericht schreiben
A2.11	Präsentation	Präsentation vorbereiten
M2.5	Statusmeeting	durchgeführtes Statusmeeting

4.4 Konzept

Nummer	Bezeichnung	Beschreibung
Dokumentation		
A3.1	Systemidee	
A3.2	Systemanwendungsfälle	
A3.3	Domänenmodell	
A3.4	Systemablaufmodelle	
A3.5	Schnittstellenbeschreibung	
A3.6	Softwarearchitektur	
A3.7	Klassenmodelle	
A3.8	Datenmodell	
A3.9	Dynamische Modelle	
A3.10	Testkonzept	
A3.11	GUI-Spezifikationen	
M3.1	Dokumente	fertig erstellte Dokumente
Entwurf des Kernsystems (1. Inkrement)		
A3.12	Refactoring des Pilotsystems	Codestruktur des Pilotsystems (gemäss M2.4) verbessern

A3.13	zentralisiertes Logging	vollständiges und einheitliches Protokollieren aller wichtigen Prozessschritte des Kernsystems definieren
A3.14	Cache-System	Cache innerhalb der virtualisierten Fremddatenquellen entwerfen
M3.2	Klassendiagramme	fertig erstellte Klassendiagramme
Entwurf der lokalen Datenbank		
A3.15	Evaluierung	- Lösungsvarianten bestimmen - Bewertungskritierien definieren - Nutzwertanalyse durchführen
A3.16	Schnittstellen/Konnektoren	- API zum Kernsystem definieren - API zur Webumgebung definieren
M3.3	Evaluierungsbericht und Schnittstellenbeschreibung	fertig erstellte Dokumente
Entwurf der Webumgebung (2. Inkrement)		
A3.17	Mashup-Präsentation	- Template-Engine definieren - GUI skizzieren
A3.18	Mashup-Verwaltung	- Rechte/Rollen definieren - Mashup-Deployment spezifizieren
M3.4	Klassendiagramme und Mockup-Skizzen	fertig erstellte Klassendiagramme und Mockup-Skizzen
2. Statusmeeting		
A3.19	Statusbericht	Statusbericht schreiben
A3.20	Präsentation	Präsentation vorbereiten
M3.5	Statusmeeting	durchgeführtes Statusmeeting

4.5 Realisierung

Nummer	Bezeichnung	Beschreibung
Implementierung des Kernsystems		
A4.1	Refactoring des Pilotsystems	gemäss A3.12
A4.2	zentralisiertes Logging	gemäss A3.13
A4.3	Cache-System	gemäss A3.14
M4.1	Kernsystem	lauffähiges Kernsystem
Implementierung der lokalen Datenbank		
A4.4	Schnittstellen/Konnektoren	gemäss A3.16
M4.2	lokale Datenbank	lauffähige Datenbank mit definierter API
Implementierung der Webumgebung		
A4.5	Mashup-Präsentation	gemäss A3.17
A4.6	Mashup-Verwaltung	gemäss A3.18

M4.3	Webumgebung	lauffähige Webumgebung
Tests		
A4.7	JUnit	
A4.8	Testprotokolle	
M4.4	Mashup-Plattform	fehlerfreie Mashup-Plattform (Endprodukt)

4.6 Projektabschluss

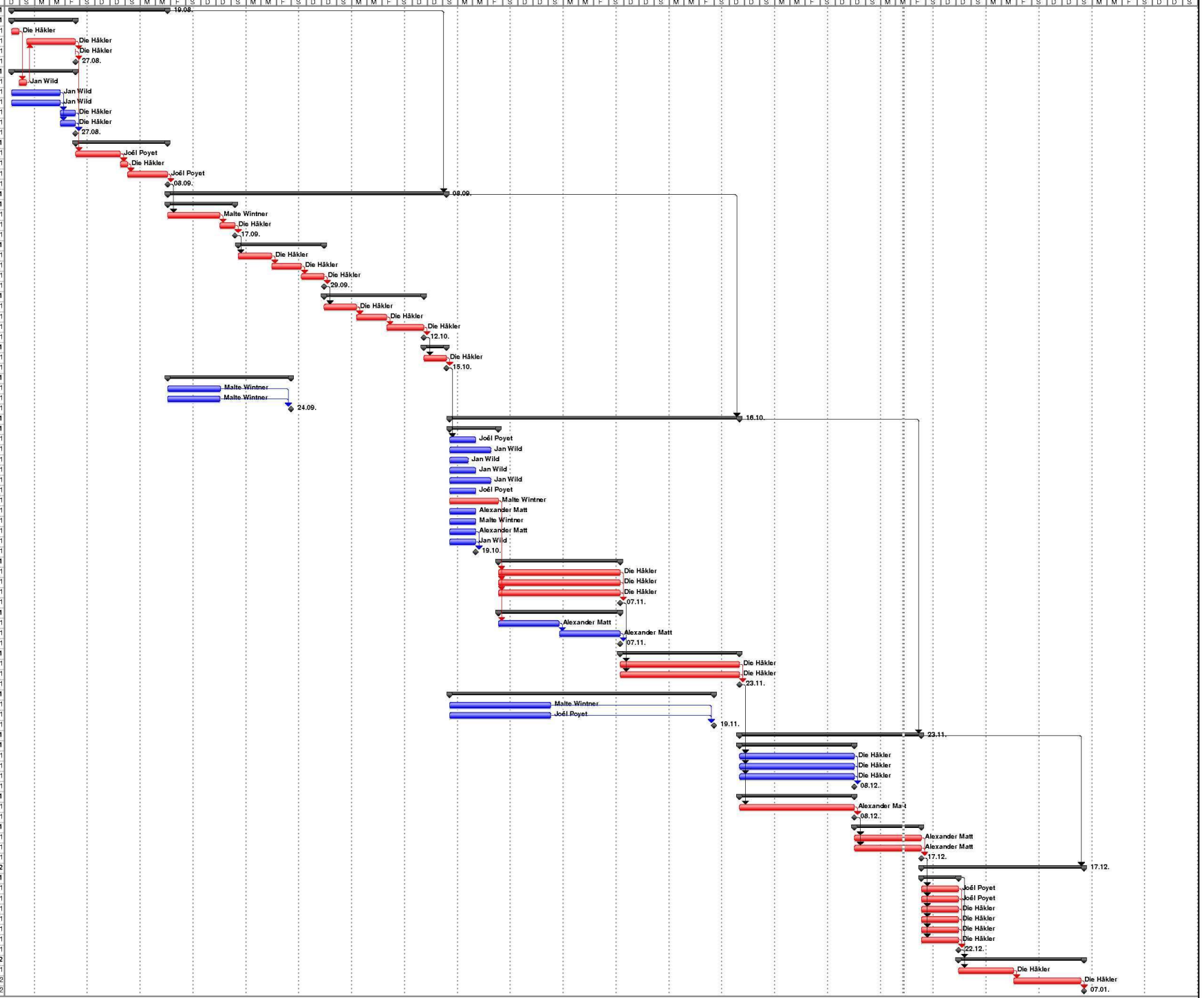
Nummer	Bezeichnung	Beschreibung
Dokumentation		
A5.1	Lieferobjekte	
A5.2	Installationsanleitung	
A5.3	Evaluierungsbericht	
A5.4	Javadoc	
A5.5	Quellcode auf USB-Stick	
A5.6	Bestätigungsformular	
M5.1	Dokumentenmappe	Dokumentenmappe per Post verschickt
Abschlusspräsentation		
A5.7	Präsentationsinhalt	Präsentation vorbereiten
A5.8	Slides	Slides erstellen
M5.2	Abschlusspräsentation	durchgeführte Abschlusspräsentation

- Ende Phasenplan -

5 Terminplan

Siehe Diagramm auf der nächsten Seite.

Nr.	Vorgangsname	Dauer	Anfang	Ende
1	**Initialisierung**	**19 Tage**	**Fr 19.08.11**	**Do 08.09.11**
2	**Projektidee**	**8,1 Tage**	**Fr 19.08.11**	**Sa 27.08.11**
3	Kick-Off-Meeting	1 Tag	Fr 19.08.11	Fr 19.08.11
4	Brainstorming	6 Tage	So 21.08.11	Sa 27.08.11
5	Fixierung	0,1 Tage	Sa 27.08.11	Sa 27.08.11
6	Mindmap	0 Tage	Sa 27.08.11	Sa 27.08.11
7	**Kommunikation**	**8 Tage**	**Fr 19.08.11**	**Sa 27.08.11**
8	Berichtswesen	1 Tag	Sa 20.08.11	Sa 20.08.11
9	Dokumentenablage	6 Tage	Fr 19.08.11	Do 25.08.11
10	Code Repository	6 Tage	Fr 19.08.11	Do 25.08.11
11	Arbeitsrechner	2 Tage	Do 25.08.11	Sa 27.08.11
12	Webserver	2 Tage	Do 25.08.11	Sa 27.08.11
13	Arbeitsumgebung	0 Tage	Sa 27.08.11	Sa 27.08.11
14	**Projektauftrag**	**10,9 Tage**	**Sa 27.08.11**	**Do 08.09.11**
15	Erstellung	5 Tage	Sa 27.08.11	Fr 02.09.11
16	Besprechung	0,9 Tage	Fr 02.09.11	Sa 03.09.11
17	Überarbeitung	5 Tage	Sa 03.09.11	Do 08.09.11
18	Projektauftrag	0 Tage	Do 08.09.11	Do 08.09.11
19	**Vorstudie**	**33 Tage**	**Do 08.09.11**	**Sa 15.10.11**
20	**Dokumentation**	**8 Tage**	**Do 08.09.11**	**Sa 17.09.11**
21	begründeter Lösungsansatz	6 Tage	Do 08.09.11	Do 15.09.11
22	Vorgehensmodell	2 Tage	Do 15.09.11	Sa 17.09.11
23	Dokumente	0 Tage	Sa 17.09.11	Sa 17.09.11
24	**Entwurf des Pilotsystems**	**10 Tage**	**So 18.09.11**	**Do 29.09.11**
25	Virtualisierung der Fremddatenquellen	4 Tage	So 18.09.11	Do 22.09.11
26	Mashup-Verknüpfungslogik	3 Tage	Do 22.09.11	Mo 26.09.11
27	Ausführungsmechanismus	3 Tage	Mo 26.09.11	Do 29.09.11
28	Klassendiagramme	0 Tage	Do 29.09.11	Do 29.09.11
29	**Implementierung des Pilotsystems**	**12 Tage**	**Do 29.09.11**	**Mi 12.10.11**
30	Virtualisierung der Fremddatenquellen	4 Tage	Do 29.09.11	Mo 03.10.11
31	Mashup-Verknüpfungslogik	4 Tage	Mo 03.10.11	Fr 07.10.11
32	Ausführungsmechanismus	4 Tage	Fr 07.10.11	Mi 12.10.11
33	Pilotsystem	0 Tage	Mi 12.10.11	Mi 12.10.11
34	**Analyse des Pilotsystems**	**3 Tage**	**Mi 12.10.11**	**Sa 15.10.11**
35	Analyse	3 Tage	Mi 12.10.11	Sa 15.10.11
36	Analyse-Ergebnisse	0 Tage	Sa 15.10.11	Sa 15.10.11
37	**1. Statusmeeting**	**14,25 Tage**	**Do 08.09.11**	**Sa 24.09.11**
38	Statusbericht	6,25 Tage	Do 08.09.11	Do 15.09.11
39	Präsentation	6 Tage	Do 08.09.11	Do 15.09.11
40	Statusmeeting	0 Tage	Sa 24.09.11	Sa 24.09.11
41	**Konzept**	**34 Tage**	**So 16.10.11**	**Mi 23.11.11**
42	**Dokumentation**	**6 Tage**	**So 16.10.11**	**Sa 22.10.11**
43	Systemidee	3 Tage	So 16.10.11	Mi 19.10.11
44	Systemanwendungsfälle	5 Tage	So 16.10.11	Fr 21.10.11
45	Domänenmodell	2 Tage	So 16.10.11	Di 18.10.11
46	Systemablaufmodelle	3 Tage	So 16.10.11	Mi 19.10.11
47	Schnittstellenbeschreibung	5 Tage	So 16.10.11	Fr 21.10.11
48	Softwarearchitektur	3 Tage	So 16.10.11	Mi 19.10.11
49	Klassenmodelle	6 Tage	So 16.10.11	Sa 22.10.11
50	Datenmodell	3 Tage	So 16.10.11	Mi 19.10.11
51	Dynamische Modelle	3 Tage	So 16.10.11	Mi 19.10.11
52	Testkonzept	3 Tage	So 16.10.11	Mi 19.10.11
53	GUI-Spezifikationen	3 Tage	So 16.10.11	Mi 19.10.11
54	Dokumente	0 Tage	Mi 19.10.11	Mi 19.10.11
55	**Entwurf des Kernsystems (1. Inkrement)**	**14 Tage**	**Sa 22.10.11**	**Mo 07.11.11**
56	Refactoring des Pilotsystems	14 Tage	Sa 22.10.11	Mo 07.11.11
57	zentralisiertes Logging	14 Tage	Sa 22.10.11	Mo 07.11.11
58	Cache-System	14 Tage	Sa 22.10.11	Mo 07.11.11
59	Klassendiagramme	0 Tage	Mo 07.11.11	Mo 07.11.11
60	**Entwurf der persistenten Datenbank**	**14 Tage**	**Sa 22.10.11**	**Mo 07.11.11**
61	Evaluierung	7 Tage	Sa 22.10.11	So 30.10.11
62	Schnittstellen/Konnektoren	7 Tage	So 30.10.11	Mo 07.11.11
63	Evaluierungsbericht und Schnittstellenbeschreibung	0 Tage	Mo 07.11.11	Mo 07.11.11
64	**Entwurf der Webumgebung (2. Inkrement)**	**14 Tage**	**Mo 07.11.11**	**Mi 23.11.11**
65	Mashup-Präsentation	14 Tage	Mo 07.11.11	Mi 23.11.11
66	Mashup-Verwaltung	14 Tage	Mo 07.11.11	Mi 23.11.11
67	Klassendiagramme und Mockup-Skizzen	0 Tage	Mi 23.11.11	Mi 23.11.11
68	**2. Statusmeeting**	**31,25 Tage**	**So 16.10.11**	**Sa 19.11.11**
69	Statusbericht	12 Tage	So 16.10.11	Sa 29.10.11
70	Präsentation	12 Tage	So 16.10.11	Sa 29.10.11
71	Statusmeeting	0 Tage	Sa 19.11.11	Sa 19.11.11
72	**Realisierung**	**22 Tage**	**Mi 23.11.11**	**Sa 17.12.11**
73	**Implementierung des Kernsystems**	**14 Tage**	**Mi 23.11.11**	**Do 08.12.11**
74	Refactoring des Pilotsystems	14 Tage	Mi 23.11.11	Do 08.12.11
75	zentralisiertes Logging	14 Tage	Mi 23.11.11	Do 08.12.11
76	Cache-System	14 Tage	Mi 23.11.11	Do 08.12.11
77	Kernsystem	0 Tage	Do 08.12.11	Do 08.12.11
78	**Implementierung der persistenten Datenbank**	**14 Tage**	**Mi 23.11.11**	**Do 08.12.11**
79	Schnittstellen/Konnektoren	14 Tage	Mi 23.11.11	Do 08.12.11
80	persistente Datenbank	0 Tage	Do 08.12.11	Do 08.12.11
81	**Tests**	**8 Tage**	**Do 08.12.11**	**Sa 17.12.11**
82	JUnit	8 Tage	Do 08.12.11	Sa 17.12.11
83	Testprotokolle	8 Tage	Do 08.12.11	Sa 17.12.11
84	Mashup-Plattform	0 Tage	Sa 17.12.11	Sa 17.12.11
85	**Projektabschluss**	**19 Tage**	**Sa 17.12.11**	**Sa 07.01.12**
86	**Dokumentation**	**4 Tage**	**Sa 17.12.11**	**Do 22.12.11**
87	Lieferobjekte	4 Tage	Sa 17.12.11	Do 22.12.11
88	Installationsanleitung	4 Tage	Sa 17.12.11	Do 22.12.11
89	Evaluierungsbericht	4 Tage	Sa 17.12.11	Do 22.12.11
90	Javadoc	4 Tage	Sa 17.12.11	Do 22.12.11
91	Quellcode auf USB-Stick	4 Tage	Sa 17.12.11	Do 22.12.11
92	Bestätigungsformular	4 Tage	Sa 17.12.11	Do 22.12.11
93	Dokumentenmappe	0 Tage	Do 22.12.11	Do 22.12.11
94	**Abschlusspräsentation**	**15 Tage**	**Do 22.12.11**	**Sa 07.01.12**
95	Präsentationsinhalt	7 Tage	Do 22.12.11	Do 29.12.11
96	Slides	8 Tage	Do 29.12.11	Sa 07.01.12
97	Abschlusspräsentation	0 Tage	Sa 07.01.12	Sa 07.01.12



Projekt: Mashup Plattform
Datum: Do 15.12.11

Vorgang | In Arbeit | Sammelvorgang | Rollup: Kritischer Vorgang | Rollup: In Arbeit | Externe Vorgänge | Gruppenkopf
Kritischer Vorgang | Meilenstein | Rollup: Vorgang | Rollup: Meilenstein | Unterbrechung | Projektsammelvorgang | Stichtag

6 Systemidee

Die Mashup-Plattform ist ein System, das dem Benutzer das Erstellen, Ausführen und Betrachten von Mashups ermöglicht.
Benutzer mit Grundkenntnissen in der Objektorientierten Programmierung können auf elegante Weise Mashups definieren, indem sie vorgefertigte Bausteine (sogenannte „virtualisierte Datenquellen“) miteinander verknüpfen.
Die daraus entstehenden Mashups bilden wiederum Bausteine, die für die Erstellung neuer Mashups genutzt werden können. Somit ist es möglich, eine ganze Bibliothek aus Bauelementen zu entwickeln.

Das System besitzt ein konfigurierbares Cache-System. Mit seiner Hilfe können die Zugriffe auf die Fremddatenquellen auf ein Minimum begrenzt werden. Darüber hinaus gewährleistet der Cache die Verfügbarkeit der Daten.

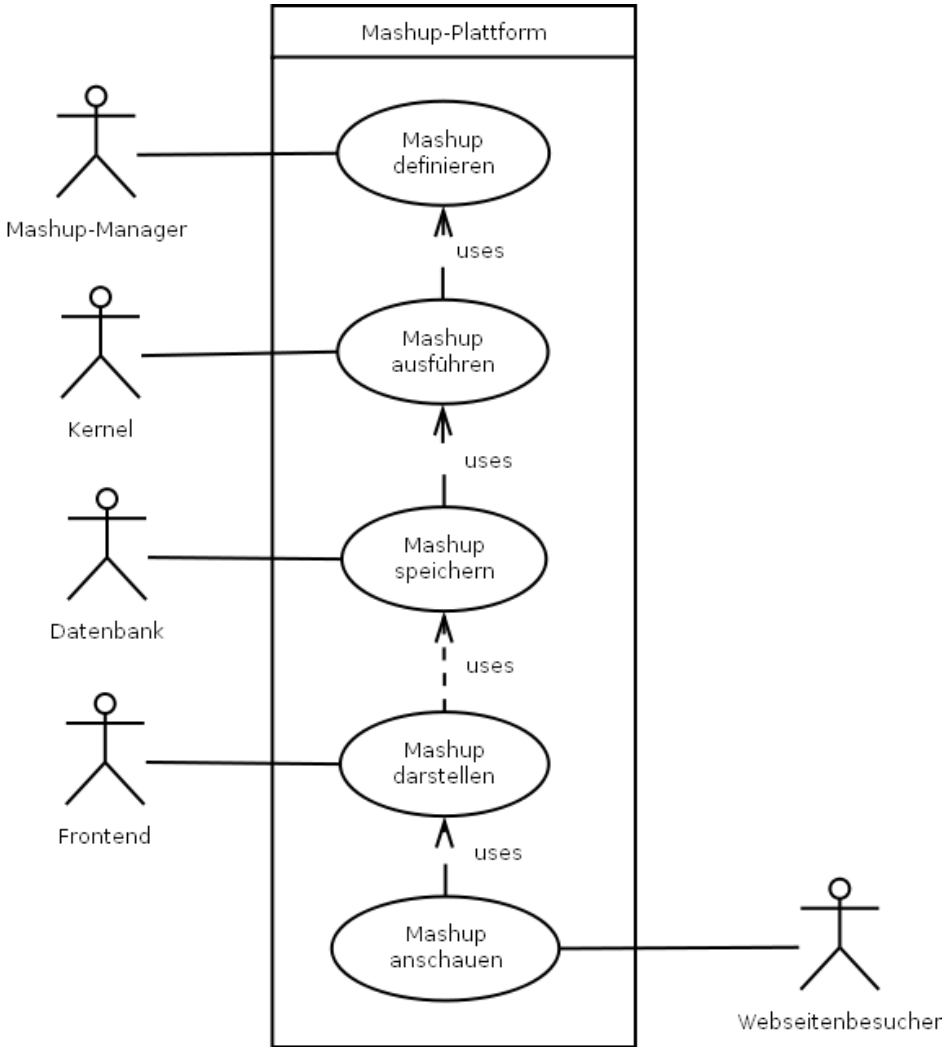
Die Mashup-Plattform hat ein Frontend, das die dynamisch generierten Mashup-Daten in naher Echtzeit im Web anzeigt. Internetbesucher können in einem gängigen Webbrowser die Mashups betrachten, ohne zusätzliche Software installieren zu müssen.

Dank einer klar definierten Architektur sind die einzelnen Systemkomponenten entkoppelt und arbeiten unabhängig voneinander. Der Kernel läuft als Hintergrundprozess und generiert in gleichmässigen Zeitintervallen die einzelnen Mashup-Datensätze. Diese Datensätze werden in eine performante und verteilbare Datenbank abgespeichert.
Bei ausgeschalteter Datenbank weicht der Kernel automatisch auf einen Zwischenspeicher aus, der erst geleert wird, sobald die Datenbank wieder läuft.

Die Mashup-Plattform ist quelloffen und steht unter der GNU General Public License (GPL Version 3). Sie ist überwiegend in Java programmiert und somit plattformunabhängig.

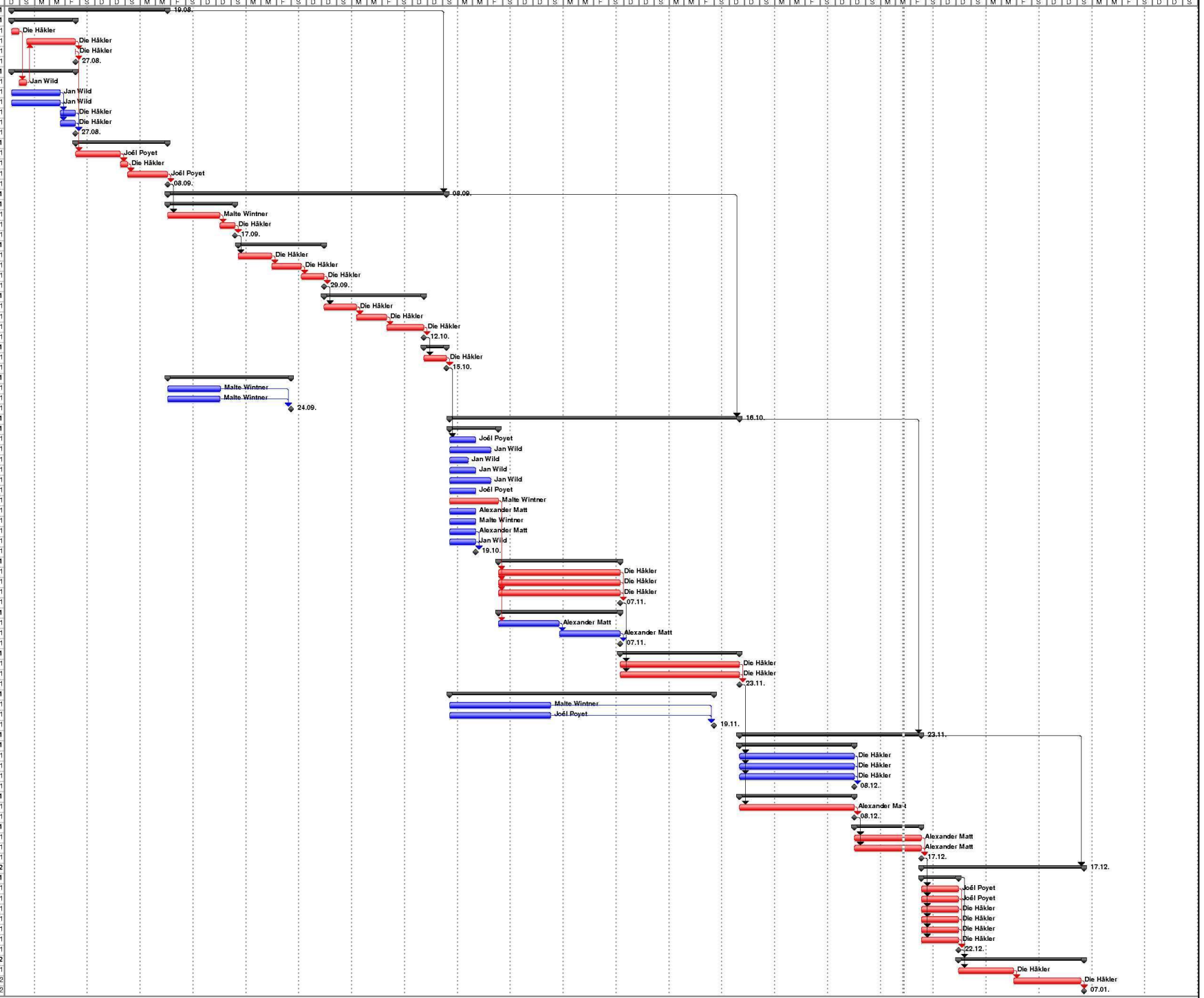
- Ende Systemidee -

7 Systemanwendungsfälle



Name	**Mashup-Manager (MM) definiert Mashup**
Kurzbeschreib.	Der MM definiert einen neuen Mashup.
Akteure	MM
Auslöser	Der MM möchte einen neuen Mashup definieren
Vorbedingungen	installierte Java-Umgebung, Klassenbibliothek der Mashup-Plattform
Eingehende Informationen	-
Ergebnisse	„virtualisierte“ Datenquelle (= Java Klasse), Konfigurationsparameter
Nachbedingung	Der Mashup sollte konfigurierbar und wiederverwendbar sein
Ablauf	**1. Mashup-Ordner erstellen** Der MM erstellt einen neuen Unterordner im var-Verzeichnis. Der Name des Ordners entspricht dem Mashup-Identifikator. **2. Klassengerüst erstellen**

Nr.	Vorgangsname	Dauer	Anfang	Ende
1	**Initialisierung**	**19 Tage**	**Fr 19.08.11**	**Do 08.09.11**
2	**Projektidee**	**8,1 Tage**	**Fr 19.08.11**	**Sa 27.08.11**
3	Kick-Off-Meeting	1 Tag	Fr 19.08.11	Fr 19.08.11
4	Brainstorming	6 Tage	So 21.08.11	Sa 27.08.11
5	Fixierung	0,1 Tage	Sa 27.08.11	Sa 27.08.11
6	Mindmap	0 Tage	Sa 27.08.11	Sa 27.08.11
7	**Kommunikation**	**8 Tage**	**Fr 19.08.11**	**Sa 27.08.11**
8	Berichtswesen	1 Tag	Sa 20.08.11	Sa 20.08.11
9	Dokumentenablage	6 Tage	Fr 19.08.11	Do 25.08.11
10	Code Repository	6 Tage	Fr 19.08.11	Do 25.08.11
11	Arbeitsrechner	2 Tage	Do 25.08.11	Sa 27.08.11
12	Webserver	2 Tage	Do 25.08.11	Sa 27.08.11
13	Arbeitsumgebung	0 Tage	Sa 27.08.11	Sa 27.08.11
14	**Projektauftrag**	**10,9 Tage**	**Sa 27.08.11**	**Do 08.09.11**
15	Erstellung	5 Tage	Sa 27.08.11	Fr 02.09.11
16	Besprechung	0,9 Tage	Fr 02.09.11	Sa 03.09.11
17	Überarbeitung	5 Tage	Sa 03.09.11	Do 08.09.11
18	Projektauftrag	0 Tage	Do 08.09.11	Do 08.09.11
19	**Vorstudie**	**33 Tage**	**Do 08.09.11**	**Sa 15.10.11**
20	**Dokumentation**	**8 Tage**	**Do 08.09.11**	**Sa 17.09.11**
21	begründeter Lösungsansatz	6 Tage	Do 08.09.11	Do 15.09.11
22	Vorgehensmodell	2 Tage	Do 15.09.11	Sa 17.09.11
23	Dokumente	0 Tage	Sa 17.09.11	Sa 17.09.11
24	**Entwurf des Pilotsystems**	**10 Tage**	**So 18.09.11**	**Do 29.09.11**
25	Virtualisierung der Fremddatenquellen	4 Tage	So 18.09.11	Do 22.09.11
26	Mashup-Verknüpfungslogik	3 Tage	Do 22.09.11	Mo 26.09.11
27	Ausführungsmechanismus	3 Tage	Mo 26.09.11	Do 29.09.11
28	Klassendiagramme	0 Tage	Do 29.09.11	Do 29.09.11
29	**Implementierung des Pilotsystems**	**12 Tage**	**Do 29.09.11**	**Mi 12.10.11**
30	Virtualisierung der Fremddatenquellen	4 Tage	Do 29.09.11	Mo 03.10.11
31	Mashup-Verknüpfungslogik	4 Tage	Mo 03.10.11	Fr 07.10.11
32	Ausführungsmechanismus	4 Tage	Fr 07.10.11	Mi 12.10.11
33	Pilotsystem	0 Tage	Mi 12.10.11	Mi 12.10.11
34	**Analyse des Pilotsystems**	**3 Tage**	**Mi 12.10.11**	**Sa 15.10.11**
35	Analyse	3 Tage	Mi 12.10.11	Sa 15.10.11
36	Analyse-Ergebnisse	0 Tage	Sa 15.10.11	Sa 15.10.11
37	**1. Statusmeeting**	**14,25 Tage**	**Do 08.09.11**	**Sa 24.09.11**
38	Statusbericht	6,25 Tage	Do 08.09.11	Do 15.09.11
39	Präsentation	6 Tage	Do 08.09.11	Do 15.09.11
40	Statusmeeting	0 Tage	Sa 24.09.11	Sa 24.09.11
41	**Konzept**	**34 Tage**	**So 16.10.11**	**Mi 23.11.11**
42	**Dokumentation**	**6 Tage**	**So 16.10.11**	**Sa 22.10.11**
43	Systemidee	3 Tage	So 16.10.11	Mi 19.10.11
44	Systemanwendungsfälle	5 Tage	So 16.10.11	Fr 21.10.11
45	Domänenmodell	2 Tage	So 16.10.11	Di 18.10.11
46	Systemablaufmodelle	3 Tage	So 16.10.11	Mi 19.10.11
47	Schnittstellenbeschreibung	5 Tage	So 16.10.11	Fr 21.10.11
48	Softwarearchitektur	3 Tage	So 16.10.11	Mi 19.10.11
49	Klassenmodelle	6 Tage	So 16.10.11	Sa 22.10.11
50	Datenmodell	3 Tage	So 16.10.11	Mi 19.10.11
51	Dynamische Modelle	3 Tage	So 16.10.11	Mi 19.10.11
52	Testkonzept	3 Tage	So 16.10.11	Mi 19.10.11
53	GUI-Spezifikationen	3 Tage	So 16.10.11	Mi 19.10.11
54	Dokumente	0 Tage	Mi 19.10.11	Mi 19.10.11
55	**Entwurf des Kernsystems (1. Inkrement)**	**14 Tage**	**Sa 22.10.11**	**Mo 07.11.11**
56	Refactoring des Pilotsystems	14 Tage	Sa 22.10.11	Mo 07.11.11
57	zentralisiertes Logging	14 Tage	Sa 22.10.11	Mo 07.11.11
58	Cache-System	14 Tage	Sa 22.10.11	Mo 07.11.11
59	Klassendiagramme	0 Tage	Mo 07.11.11	Mo 07.11.11
60	**Entwurf der persistenten Datenbank**	**14 Tage**	**Sa 22.10.11**	**Mo 07.11.11**
61	Evaluierung	7 Tage	Sa 22.10.11	So 30.10.11
62	Schnittstellen/Konnektoren	7 Tage	So 30.10.11	Mo 07.11.11
63	Evaluierungsbericht und Schnittstellenbeschreibung	0 Tage	Mo 07.11.11	Mo 07.11.11
64	**Entwurf der Webumgebung (2. Inkrement)**	**14 Tage**	**Mo 07.11.11**	**Mi 23.11.11**
65	Mashup-Präsentation	14 Tage	Mo 07.11.11	Mi 23.11.11
66	Mashup-Verwaltung	14 Tage	Mo 07.11.11	Mi 23.11.11
67	Klassendiagramme und Mockup-Skizzen	0 Tage	Mi 23.11.11	Mi 23.11.11
68	**2. Statusmeeting**	**31,25 Tage**	**So 16.10.11**	**Sa 19.11.11**
69	Statusbericht	12 Tage	So 16.10.11	Sa 29.10.11
70	Präsentation	12 Tage	So 16.10.11	Sa 29.10.11
71	Statusmeeting	0 Tage	Sa 19.11.11	Sa 19.11.11
72	**Realisierung**	**22 Tage**	**Mi 23.11.11**	**Sa 17.12.11**
73	**Implementierung des Kernsystems**	**14 Tage**	**Mi 23.11.11**	**Do 08.12.11**
74	Refactoring des Pilotsystems	14 Tage	Mi 23.11.11	Do 08.12.11
75	zentralisiertes Logging	14 Tage	Mi 23.11.11	Do 08.12.11
76	Cache-System	14 Tage	Mi 23.11.11	Do 08.12.11
77	Kernsystem	0 Tage	Do 08.12.11	Do 08.12.11
78	**Implementierung der persistenten Datenbank**	**14 Tage**	**Mi 23.11.11**	**Do 08.12.11**
79	Schnittstellen/Konnektoren	14 Tage	Mi 23.11.11	Do 08.12.11
80	persistente Datenbank	0 Tage	Do 08.12.11	Do 08.12.11
81	**Tests**	**8 Tage**	**Do 08.12.11**	**Sa 17.12.11**
82	JUnit	8 Tage	Do 08.12.11	Sa 17.12.11
83	Testprotokolle	8 Tage	Do 08.12.11	Sa 17.12.11
84	Mashup-Plattform	0 Tage	Sa 17.12.11	Sa 17.12.11
85	**Projektabschluss**	**19 Tage**	**Sa 17.12.11**	**Sa 07.01.12**
86	**Dokumentation**	**4 Tage**	**Sa 17.12.11**	**Do 22.12.11**
87	Lieferobjekte	4 Tage	Sa 17.12.11	Do 22.12.11
88	Installationsanleitung	4 Tage	Sa 17.12.11	Do 22.12.11
89	Evaluierungsbericht	4 Tage	Sa 17.12.11	Do 22.12.11
90	Javadoc	4 Tage	Sa 17.12.11	Do 22.12.11
91	Quellcode auf USB-Stick	4 Tage	Sa 17.12.11	Do 22.12.11
92	Bestätigungsformular	4 Tage	Sa 17.12.11	Do 22.12.11
93	Dokumentenmappe	0 Tage	Do 22.12.11	Do 22.12.11
94	**Abschlusspräsentation**	**15 Tage**	**Do 22.12.11**	**Sa 07.01.12**
95	Präsentationsinhalt	7 Tage	Do 22.12.11	Do 29.12.11
96	Slides	8 Tage	Do 29.12.11	Sa 07.01.12
97	Abschlusspräsentation	0 Tage	Sa 07.01.12	Sa 07.01.12



Projekt: Mashup Plattform
Datum: Do 15.12.11

Vorgang | In Arbeit | Sammelvorgang | Rollup: Kritischer Vorgang | Rollup: In Arbeit | Externe Vorgänge | Gruppenkopf
Kritischer Vorgang | Meilenstein | Rollup: Vorgang | Rollup: Meilenstein | Unterbrechung | Projektsammelvorgang | Stichtag

	Der MM erstellt das Klassengerüst im Mashup-Ordner. Als Vorlage kann das vorgefertigte Beispiel-Mashup dienen. **3. Verknüpfungslogik definieren** Der MM definiert die Verknüpfungslogik innerhalb einer erweiterten Systemklasse.		
Ansprechpartner	Jan		
Risiko	Strukturänderungen der Fremddatenquellen		
Verbindlichkeit	unverzichtbar		
Aufwand	30 h		
Stabilität	abhängig von den Programmierfähigkeiten des MM		
Zeitpunkt	Release 0.1		
Änderungen	Mitarb.	Status	Kommentar
13.10.2011	Jan	Entwurf	.
28.10.2011	Jan	1. Über-arbeitung	Verschiedene Dinge angepasst, hauptsächlich den Ablauf.
03.11.2011	Malte	Review	kleinere Anpassungen

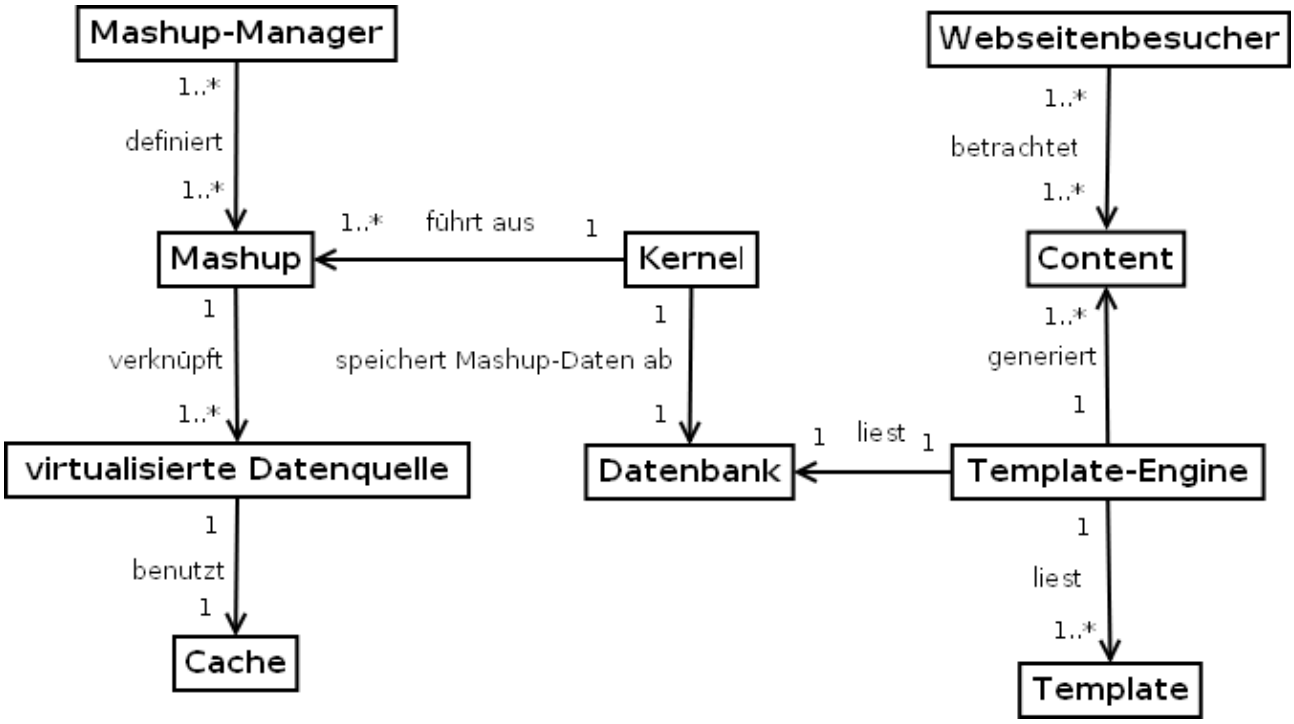
Name	**Kernel führt Mashup aus**
Kurzbeschreib.	Der Kernel führt einen Mashup aus
Akteure	Kernel (= Komponente der Mashup-Plattform)
Auslöser	Der MM möchte Mashup-Daten generieren
Vorbedingungen	installierter Kernel, definierter Mashup, laufende Datenbank (optional)
Eingehende Informationen	Mashup-Name, max. Anzahl Datensätze, Ausführungsgeschwindigkeit des Kernels, Lebenszeit des Caches
Ergebnisse	generierte Mashup-Daten
Nachbedingung	optimierte Konfigurationsparameter für ressourcenschonenden Zugriff auf Fremddatenquelle
Ablauf	**1. Mashup-Klassen instanziieren** Der Kernel instanziiert die Mashup-Klassen. **2. Output speichern** Der Kernel legt den Output in einen Zwischenspeicher ab. **3. Datenbank starten (optional)** Der Kernel transferiert die Daten aus dem Zwischenspeicher in die Datenbank.
Ansprechpartner	Jan
Risiko	Überlastung des Systems/Fremddatenquelle bei schlechter Konfiguration
Verbindlichkeit	unverzichtbar
Aufwand	30 h

Stabilität	stabil		
Zeitpunkt	Release 0.1		
Änderungen	Mitarb.	Status	Kommentar
13.10.2011	Jan	Entwurf	
03.11.2011	Malte	Review	kleinere Anpassungen

Name	**Webseitenbesucher betrachtet Mashup**		
Kurzbeschreib.	Der Besucher wählt auf der Website einen Mashup aus.		
Akteure	Webseitenbesucher		
Auslöser	Der Webseitenbesucher möchte einen Mashup anschauen		
Vorbedingungen	installiertes Frontend (Mashup-Plattform)		
Eingehende Informationen	Mashup-Identifikator		
Ergebnisse	Ausgabe der ersten Mashup-Seite.		
Nachbedingung	Die Daten des Mashups wurden angezeigt. Der Webseitenbesucher hat die Möglichkeit weitere Seiten auszuwählen oder auf die Übersichtsseite zurückzukehren.		
Ablauf	**1. Mashup-Identifikator entgegennehmen** Der Webseitenbesucher wählt einen Mashup aus einer Liste aus. **2. Mashup-Seite an den Browser senden** Die erste Mashup-Seite wird im Webbrowser angezeigt.		
Ansprechpartner	Jan		
Risiko	Internetprobleme		
Verbindlichkeit, Priorität	hoch		
Aufwand	20 h		
Stabilität	stabil		
Zeitpunkt, Dringlichkeit	Release 0.1		
Änderungen	Mitarb.	Status	Kommentar
13.10.2011	Jan	Entwurf	
28.10.2011	Jan	1. Über-arbeitung	„Mashup Benutzer“ in „Webseitenbesucher“ umbenannt. Weitere, kleinere Änderungen.
03.11.2011	Malte	Review	kleinere Anpassungen

- Ende Systemanwendungsfälle -

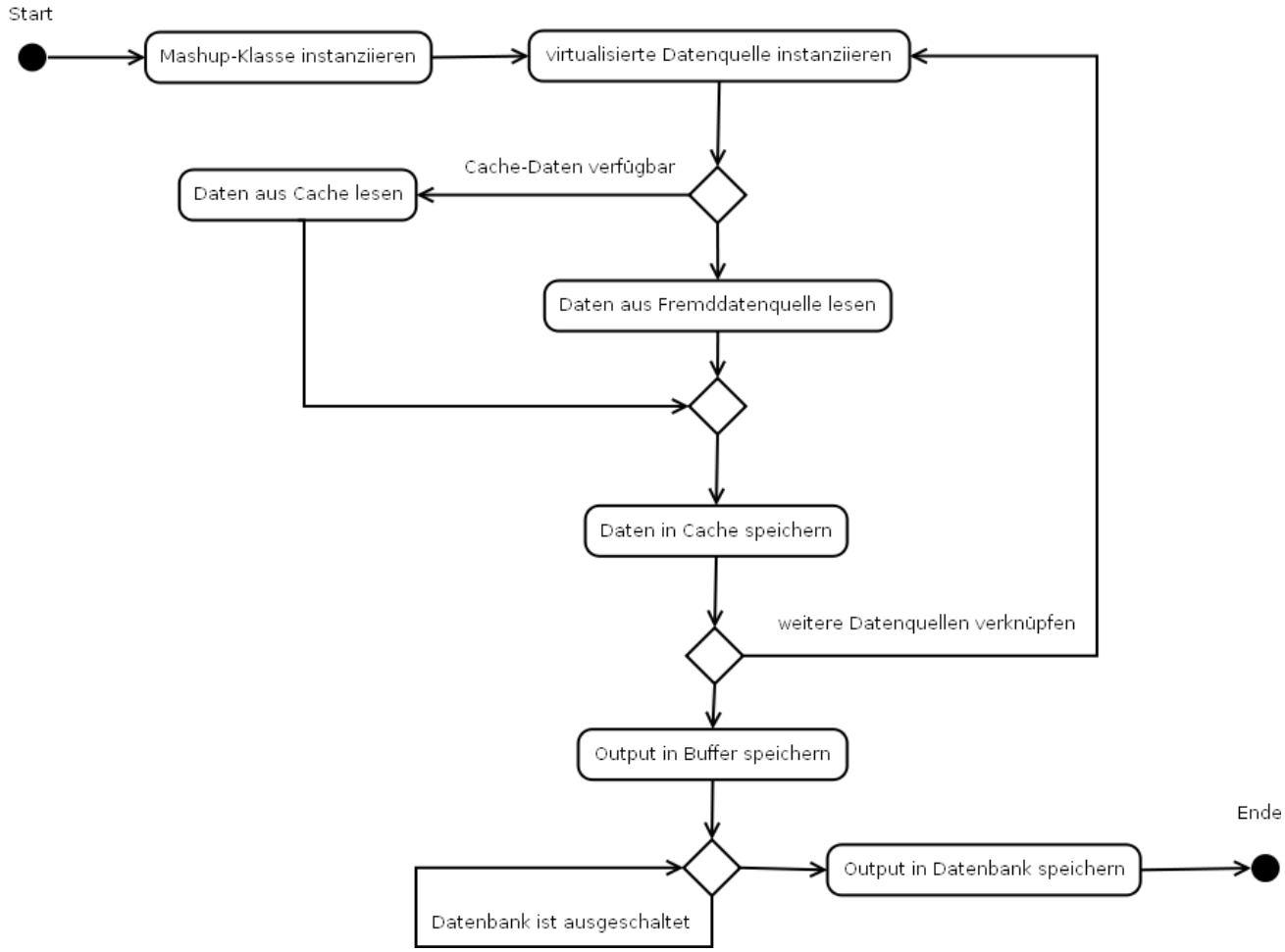
8 Domänenmodell



- Ende Domänenmodell -

9 Systemablaufmodelle

Mashup ausführen



- Ende Systemablaufmodelle -

10 Schnittstellenbeschreibung

Anwendungsfallschritt	involvierte Schnittstellenelemente
Mashup definieren	API der Klassenbibliothek
Mashup ausführen	Konfigurationsdateien
Mashup betrachten	Website-Navigation

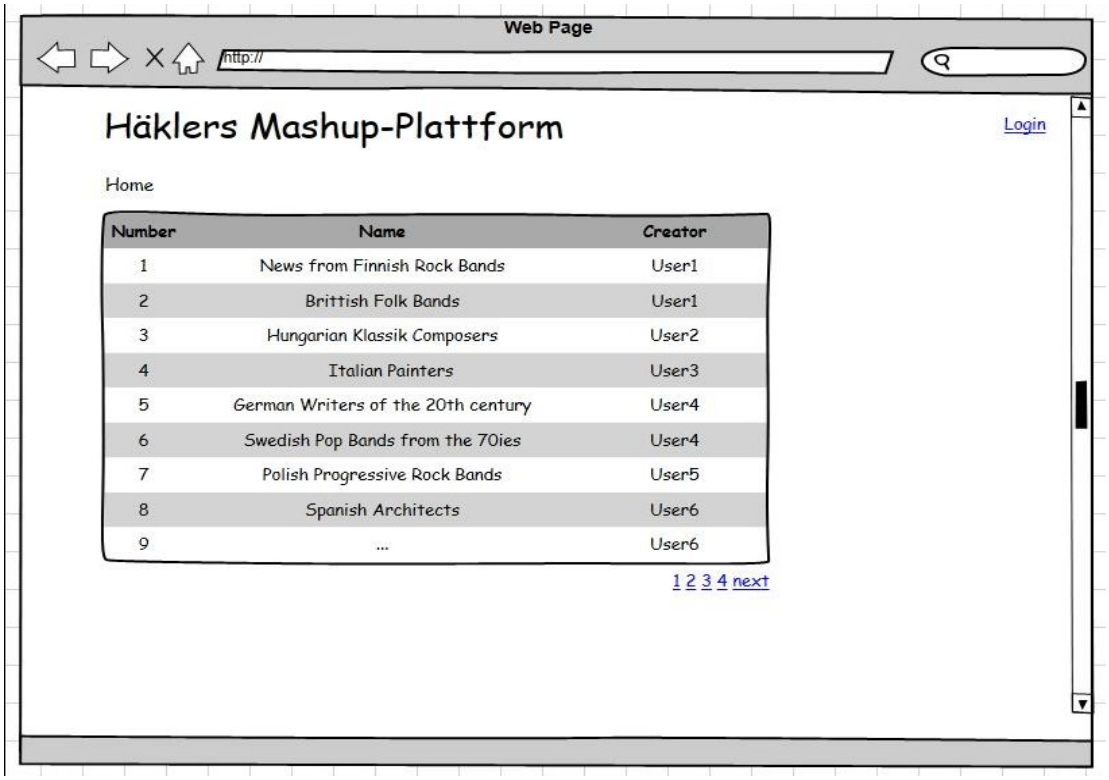
Dialogbeschreibung	
Name	**Mashup definieren**
Kurzbeschreibung	Durch Vererbung/Nutzung von Systemklassen können Mashups definiert werden
Verwendung	Jeder Mashup stellt eine virtualisierte Datenquelle dar und wird durch eine Java-Klasse repräsentiert.
Komplexität	hoch
Eingabefelder	Um eine virtualisierte Datenquelle zu definieren, muss eine (abstrakte) Hauptklasse erweitert werden. Die Hauptklasse schreibt die zu implementierenden Schnittstellen vor.
Anzeigefelder	-
Verzweigungs-möglichkeiten	-
Aktionen	Verknüpfungslogik programmieren, Konfigurationsparameter definieren

Dialogbeschreibung	
Name	**Mashup ausführen**
Kurzbeschreibung	Zum Ausführen des Mashups muss der Kernel (= Hintergrundprozess) gestartet werden.
Verwendung	Der Kernel instanziiert den Mashup in regelmässigen Abständen und speichert den Output des Mashups ab. Bei jedem Durchgang wird genau **eine** Mashup-Seite generiert. D.h. der Kernel erhöht die Seitennummer bei jeder Instanziierung.
Komplexität	hoch
Eingabefelder	Seitennummer
Anzeigefelder	Der Output des Mashups kann in der Datenbank betrachtet werden
Verzweigungs-möglichkeiten	-
Aktionen	Kernel starten, Datenbank starten (optional)

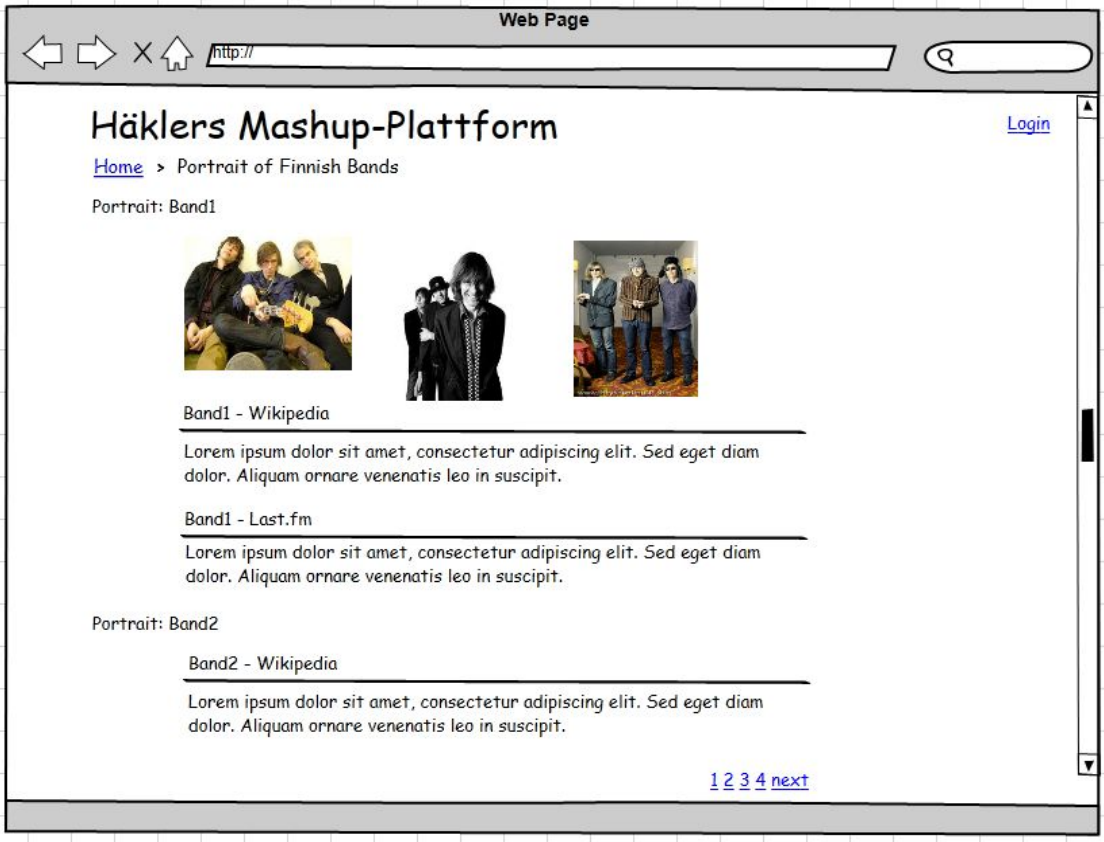
Dialogbeschreibung	
Name	**Mashup betrachten**
Kurzbeschreibung	Webseitenbesucher können den generierten Output des Mashups im Webbrowser betrachten.
Verwendung	Die Template-Engine generiert (dynamische) HTML-Seiten aus dem Output des Mashups. Diese HTML-Seiten beinhalten eine Navigation, um zwischen den einzelnen Mashup-Seiten zu blättern.
Komplexität	normal
Eingabefelder	Mashup-Identifikator, Seitennummer
Anzeigefelder	Mashup-Gesamtübersicht bzw. einzelne Mashup-Seite
Verzweigungs-möglichkeiten	Mashup auswählen (Gesamtübersicht), Seite blättern (Mashup-Seite)
Aktionen	Website aufrufen, Mashup aus der Gesamtübersicht wählen, Mashup-Seite betrachten, Seite blättern

10.1 Skizzen

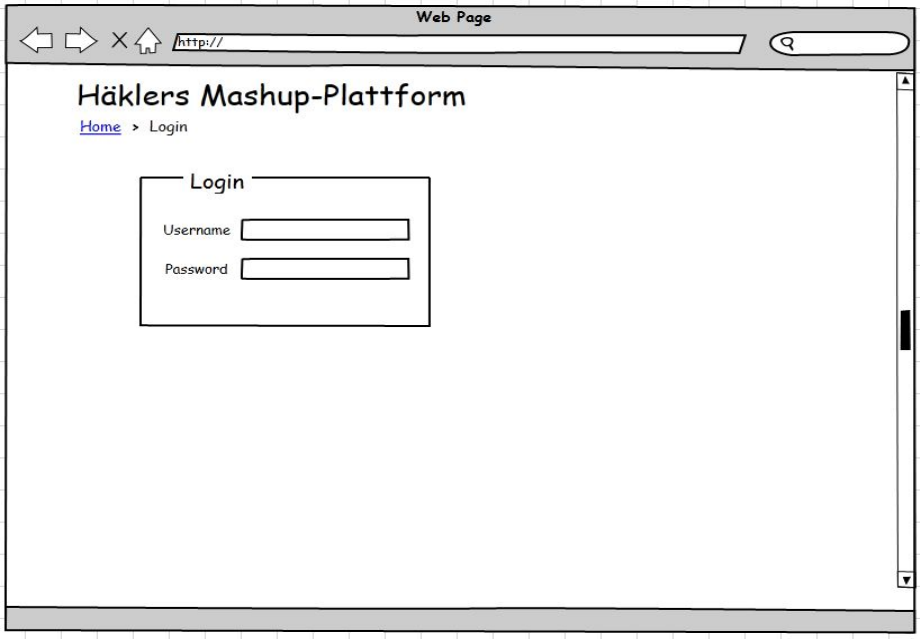
10.1.1 Mashup-Gesamtübersicht



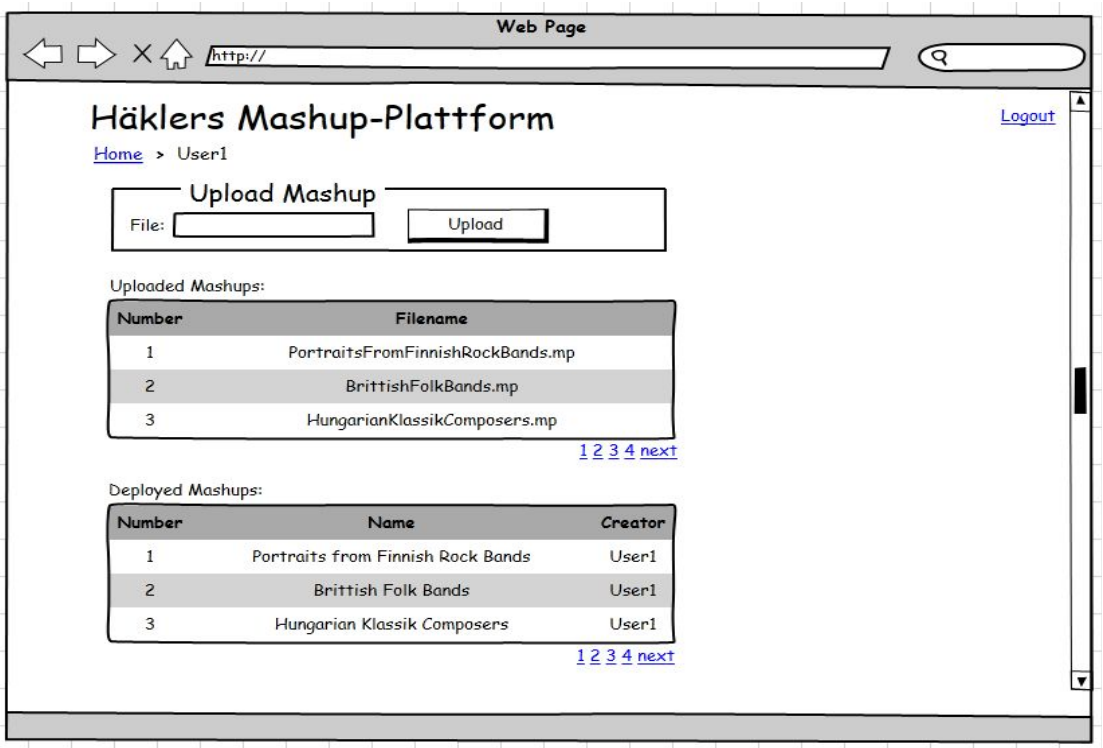
10.1.2 einzelne Mashup-Seite



10.1.3 Login („Kann“-Ziel)



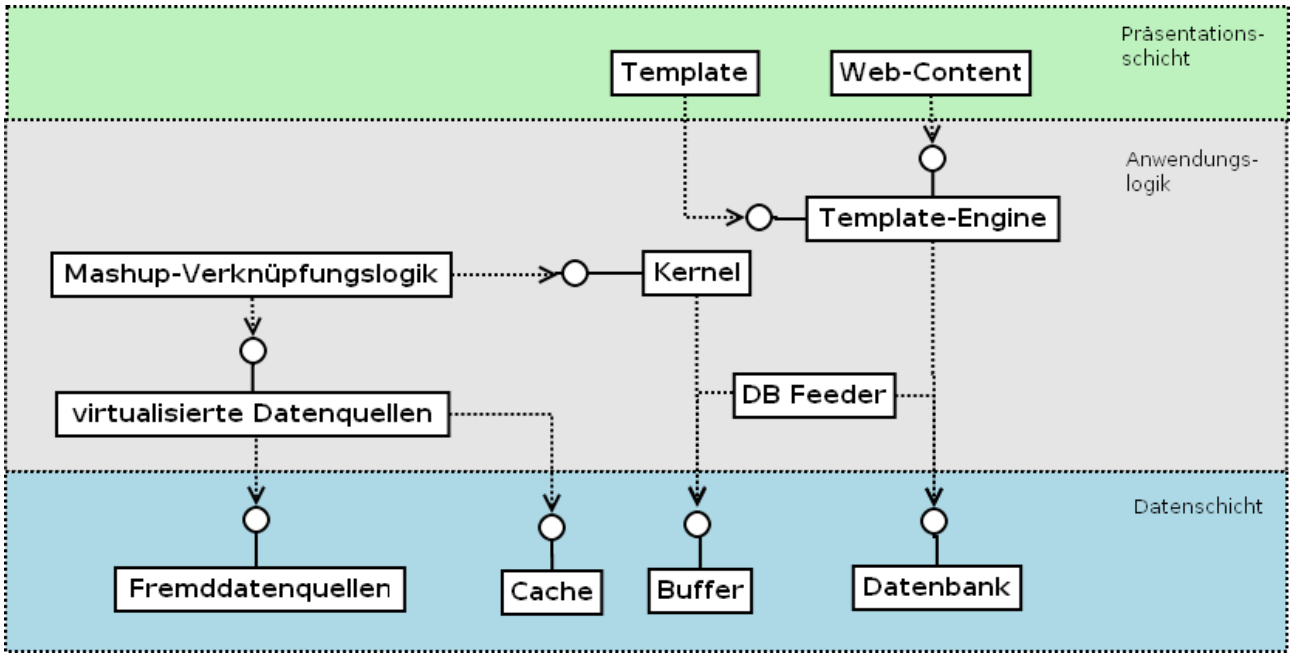
10.1.4 webbasiertes Mashup-Deployment („Kann-Ziel“)



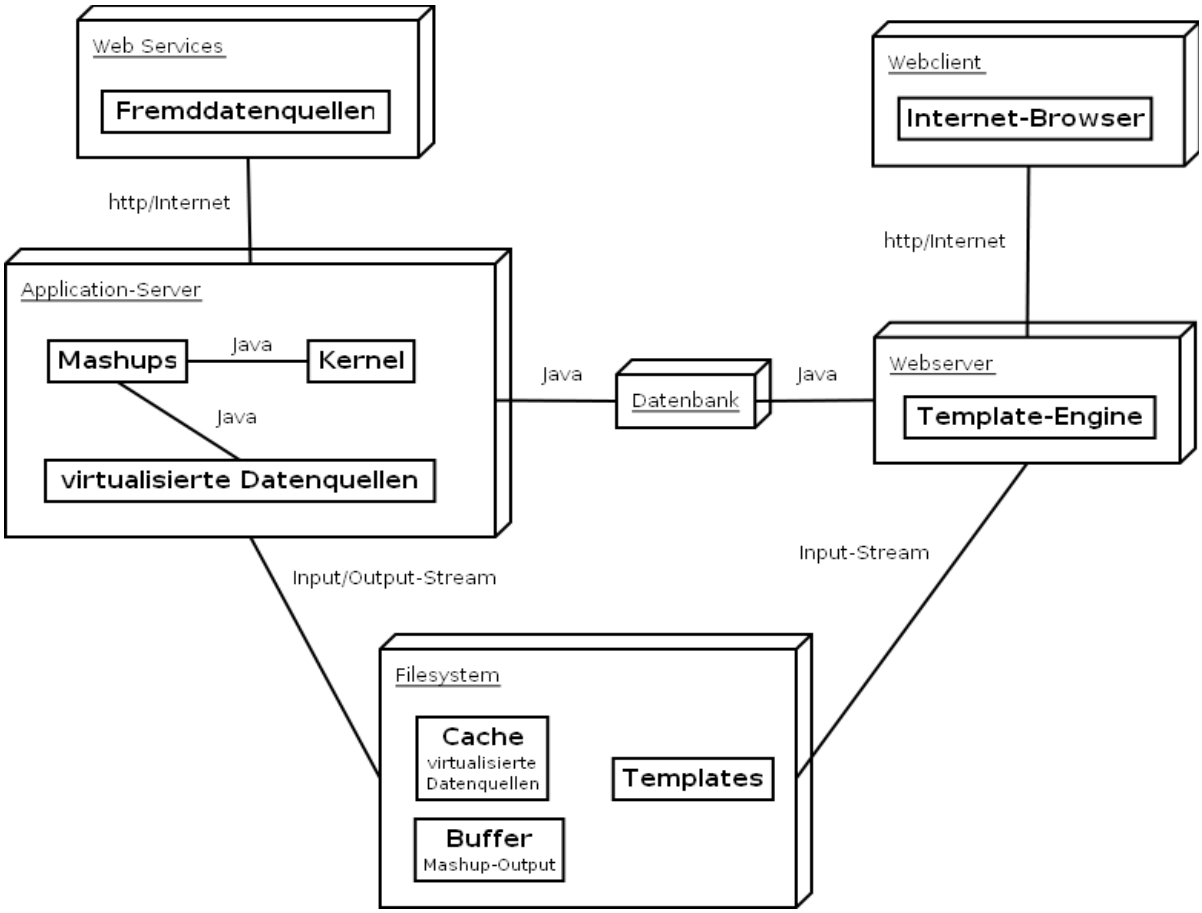
- Ende Schnittstellenbeschreibung -

11 Softwarearchitektur

11.1 Schichtenmodell

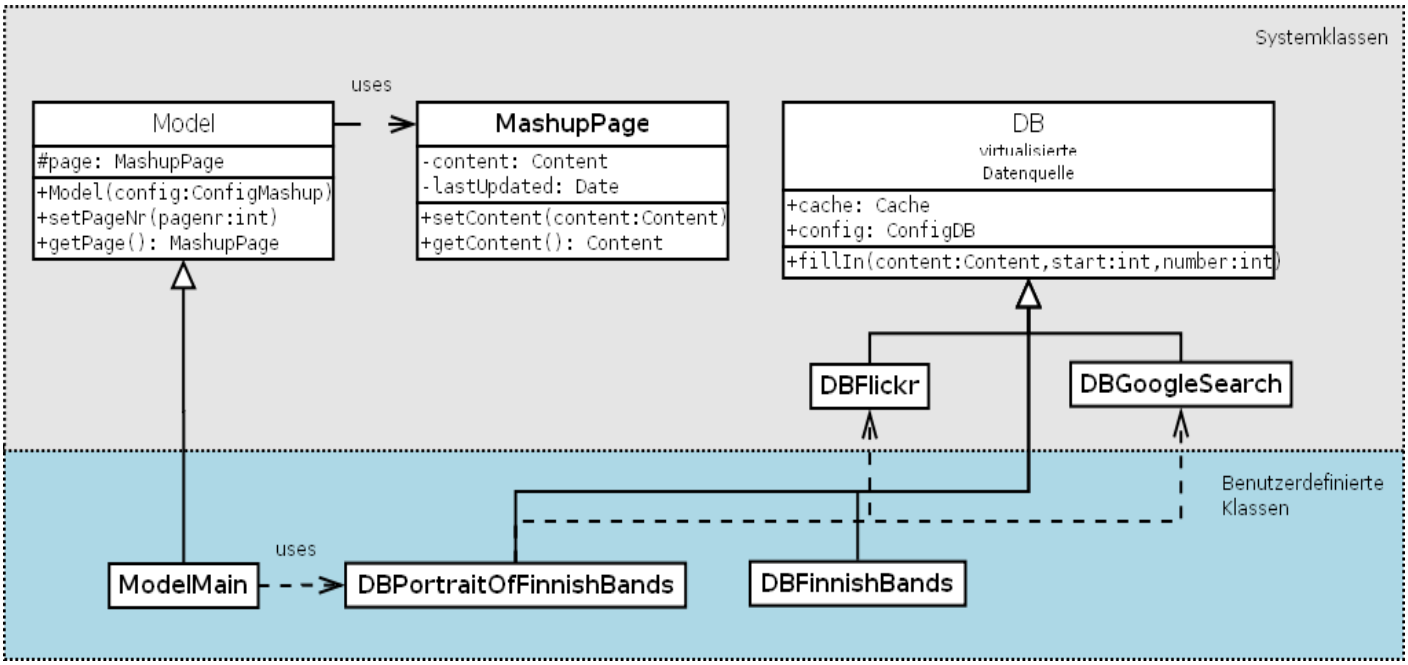


11.2 Verteilungsmodell

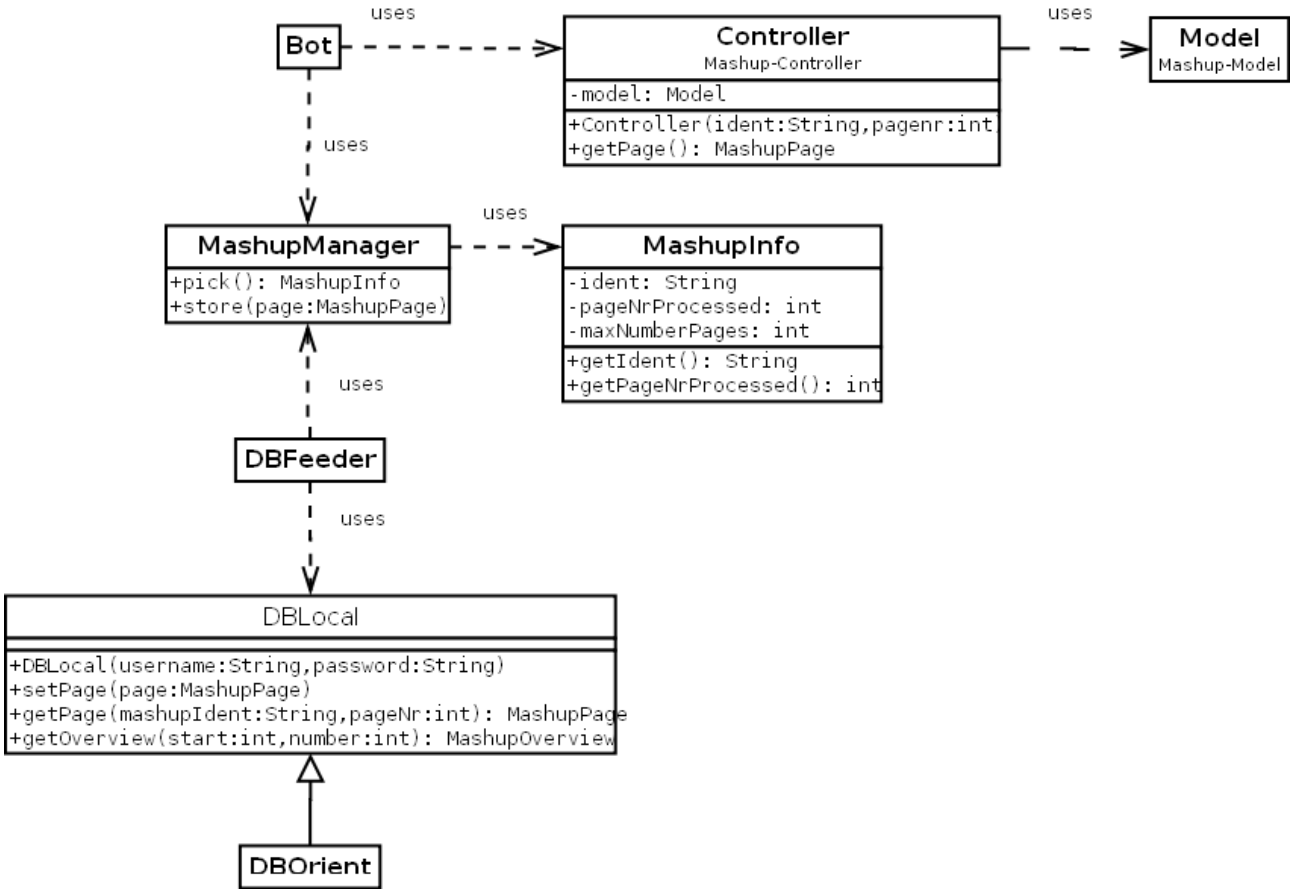


12 Klassenmodelle

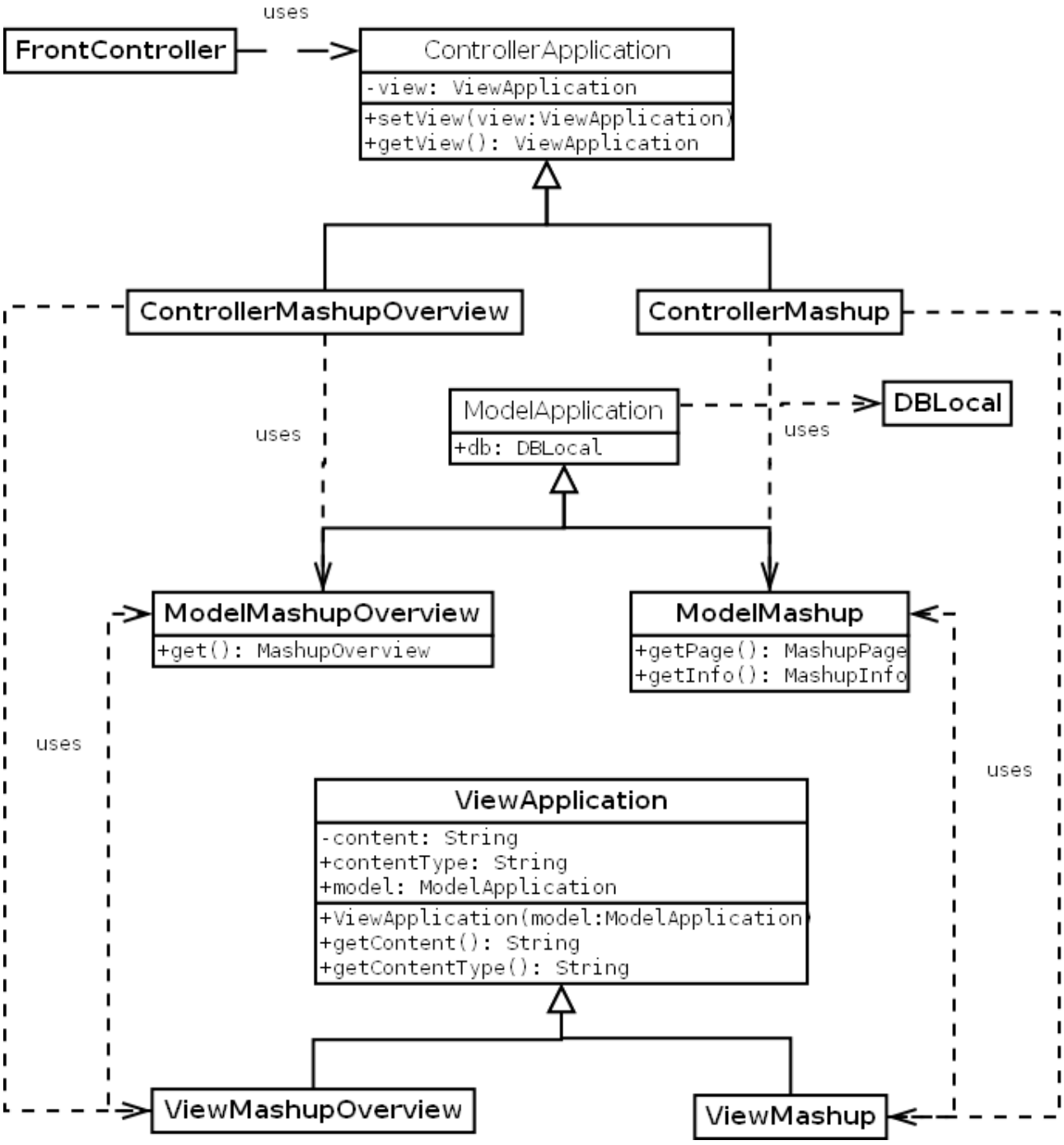
12.1 Mashup definieren



12.2 Mashup ausführen



12.3 Mashup anzeigen

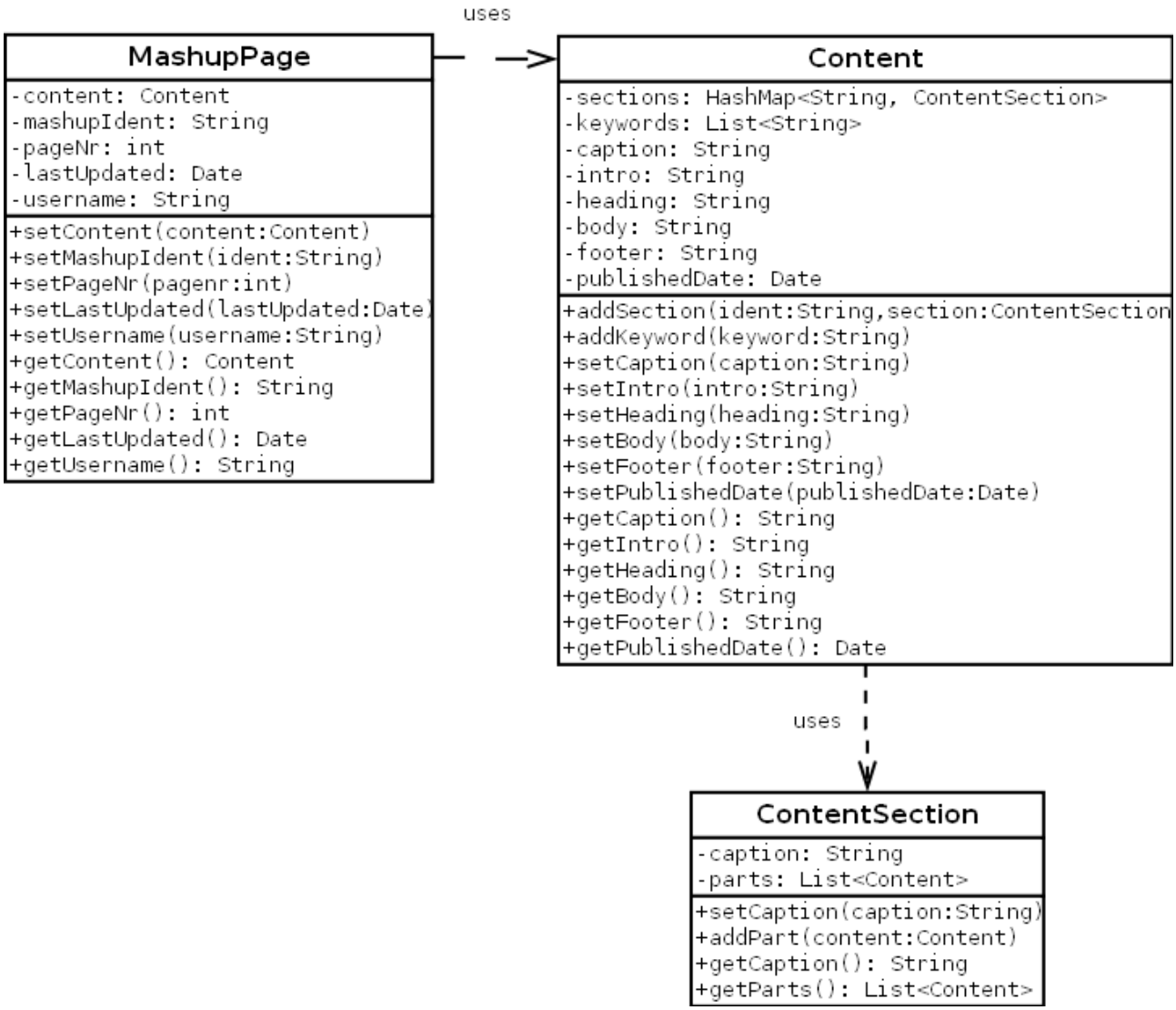


- Ende Klassenmodelle -

13 Datenmodell

13.1 Persistenzklassen (objektbasierte Datenhaltung)

Bei diesem Projekt wurde eine objektbasierte Datenhaltung gewählt, da die durch die Mashups erzeugten Content-Objekte dynamische Baumstrukturen aufweisen, welche sich nur sehr schwer auf Relationen abbilden lassen.



13.2 Einschub: Wahl der Datenbank

13.2.1 IST-Situation

Die Mashup-Generierung aus mindestens zwei Datenquellen wurde bereits erfolgreich implementiert, ist aber (vor allem bei mehreren Quellen) ein sehr zeitaufwändiger Vorgang. Dem Webseitenbesucher könnte bei zu langer Wartezeit das Interesse an der Mashup-Plattform vergehen.

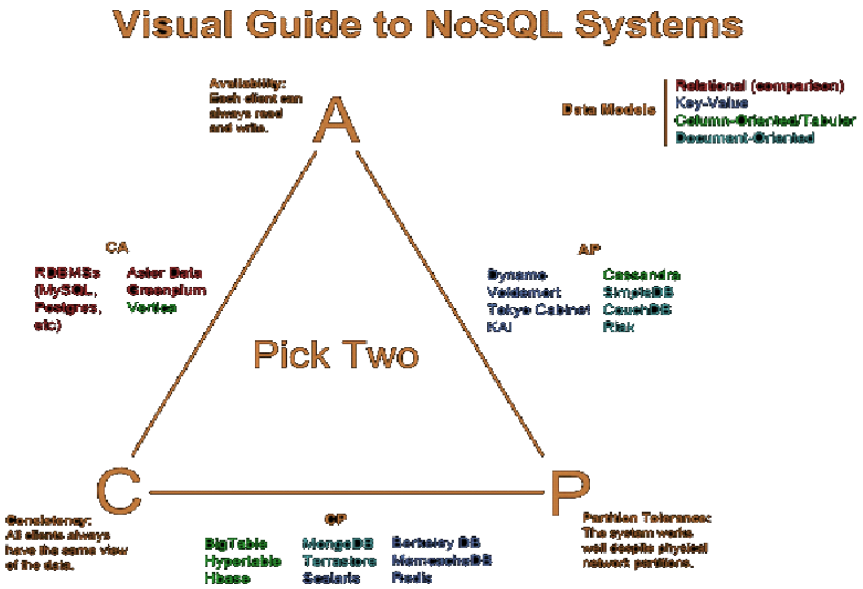
13.2.2 SOLL-Situation

Um die Performance bei der Betrachtung von Mashups zu optimieren, sollen diese in einer Datenbank abgelegt werden. Die Mashup-Inhalte müssen so nicht bei jedem Aufruf aufs Neue generiert werden, sondern können direkt aus der Datenbank gelesen werden, was dem Webseitenbesucher letztendlich eine kürzere Wartezeit ermöglicht und das System somit entlastet.

13.2.3 IST/SOLL-Analyse

Im Rahmen der IST/SOLL-Analyse wurde das Projektteam zum ersten Mal auf die sogenannten NoSQL(**N**ot **O**nly **SQL**)-Systeme aufmerksam, die eine interessante Alternative zu den häufiger verwendeten relationalen Datenbanksystemen darstellen. RDBMS sind stark strukturiert, aufwändig zu entwerfen und nachträglich nur schwer zu ändern, während NoSQL-Systeme sehr flexibel und gut einsetzbar sind für unstrukturierte Daten, bei denen man sich nicht um Relationen zu kümmern braucht. Aus diesen Gründen waren sie ideal für die Persistierung unserer Mashups geeignet.

Da wir allerdings noch keine Erfahrung mit derartigen Systemen hatten und es im Internet schwierig war, sich in den Unmengen von Informationen über die verschiedensten Arten von NoSQL-Systemen zurechtzufinden, orientierten wir uns zu Beginn an folgender Übersicht:



Quelle: http://blog.nahurst.com/visual-guide-to-nosql-systems

Die Grafik stellt im sogenannten CAP-Theorem die drei für Datenbankmanagementsysteme wichtigsten Eigenschaften dar:

Konsistenz (C): Alle sehen zur selben Zeit dieselben Daten.

Verfügbarkeit (A): Alle Anfragen an das System werden stets beantwortet.

Partitionstoleranz (P): Das System arbeitet trotz willkürlicher Verluste von Nachrichten weiter.

Das CAP-Theorem besagt nun, dass ein verteiltes System lediglich zwei dieser drei Eigenschaften erfüllen kann, jedoch niemals alle drei. Die Systeme die jeweils zwei der Eigenschaften erfüllen sind in der Grafik zwischen selbigen ersichtlich.

Als nächster Schritt für uns galt es also zu entscheiden, welche der drei Eigenschaften in unserem System vernachlässigt werden könnte. Im Hinblick auf unsere Hauptdaten (die Inhalte der einzelnen

Mashups) einigten wir uns an dieser Stelle auf die Konsistenz, da es den meisten Webseitenbesuchern relativ egal sein dürfte, ob ein anderer Benutzer zum selben Zeitpunkt genau dieselben Daten auf dem Bildschirm hat oder nicht.

Unsere zu wählende Lösung lag demnach zwischen der Verfügbarkeit und der Partitionstoleranz. Da unsere Mashups letztendlich nichts anderes waren als Dokumente in unterschiedlichen Formaten wie JSON oder HTML, schränkten wir unsere Wahl weiter ein auf NoSql-Systeme vom Typ der dokumentenorientierten Datenbanken. Demzufolge mussten wir uns nur noch zwischen drei Systemen entscheiden, was wir am besten mithilfe einer Nutzwertanalyse durchführen konnten.

13.2.4 Lösungsvarianten

Varianten-Nr.	Erklärung
V1	SimpleDB
V2	CouchDB
V3	Riak

13.2.5 Bewertungskriterien

Kriterien-Nr.	Gewichtung	Kriterium
B1	10	Kosten
B2	9	Skalierbarkeit
B3	8	Partitionstoleranz
B4	8	Verfügbarkeit
B5	7	Performance
B6	7	Einfache Programmierschnittstelle
B7	5	Konsistenz

13.2.6 Nutzwertanalyse

Die Punktevergabe entsprach der in der Schweiz gebräuchlichen Notenskala (1 sehr schlecht, 6 sehr gut).

	B1	B2	B3	B4	B5	B6	B7	Total (Gewichtung berücksichtigt)
V1	1	4	3	3	6	5	5	196
V2	6	6	5	6	3	5	6	**281**
V3	3	4	6	6	6	4	5	257

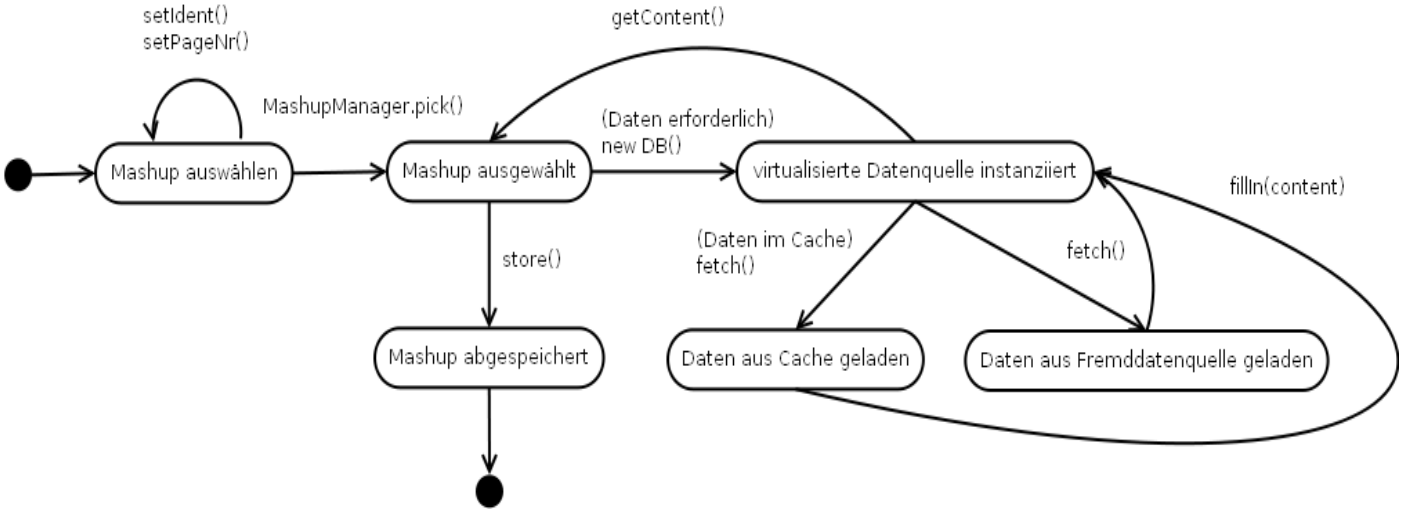
13.2.7 Ausgewählte Lösung

Die Nutzwertanalyse brachte CouchDB als eindeutigen Sieger hervor. Vor allem die Tatsache, dass dieses Produkt komplett Open Source ist, aber auch die sehr gute Skalierbarkeit waren im Vergleich zu den anderen Systemen entscheidend.

- Ende Datenmodell -

14 Dynamische Modelle

14.1 Mashup ausführen



- Ende Dynamische Modelle -

15 Testkonzept

15.1 Allgemein

Die Softwaretests gliedern sich in zwei Bereiche.

1. Zum Testen der Klassen und deren Methoden werden Testmethoden auf der Basis von JUnit entwickelt.

2. Für die allgemeine Prüfung der Software werden Testfälle erstellt, welche die Benutzerinteraktionen simulieren.

15.2 Junit

Um die Qualität der Software zu gewährleisten, werden automatisierte JUnit-Tests implementiert.
Die Testklassen werden sich im Unterordner „test“ (siehe Kapitel „Ablagestruktur des Quellcodes“) befinden. Jede wichtige Java-Klasse erhält eine entsprechende Testklasse, welche die vorhandenen Methoden auf Korrektheit überprüft.

Da innerhalb des Teams keine Erfahrung im Bereich des Test Driven Developments vorliegt, werden sogenannte Regressions-Tests durchgeführt. Die Komponenten werden erst nach der Programmierung getestet und (falls notwendig) überarbeitet.

Zur Überprüfung der Code-Abdeckung wird das Programm „eclEmma“ (http://www.eclemma.org/) eingesetzt.
Eine Code-Abdeckung von mindestens 75% ist für dieses Projekt geplant.

Die JUnit-Tests dienen der Funktionskontrolle des implementierten Codes. Bei jeder zukünftigen Codeänderung müssen die Tests erneut durchlaufen werden, um die korrekte Funktionsweise der Software sicherzustellen.

15.3 zentralisiertes Logging

Ein sehr wichtiges Hilfsmittel zum Testen des Systems ist der zentralisierte Logger.
Jeder wichtige Prozessschritt wird protokolliert und in einer Log-Datei abgespeichert. Zur Laufzeit aufgetretene Fehler (Exceptions) werden dabei in einer separaten Error-Log-Datei ausgelagert.
Mit Hilfe dieser Log-Dateien können Programm-/Logikfehler sehr schnell aufgedeckt werden.

- Ende Testkonzept -

16 Spezifikation der Bedienoberfläche

Die Skizzen für die Bedienoberfläche befinden sich im Kapitel „Schnittstellenbeschreibung“.

- Ende Spezifikation der Bedienoberfläche -

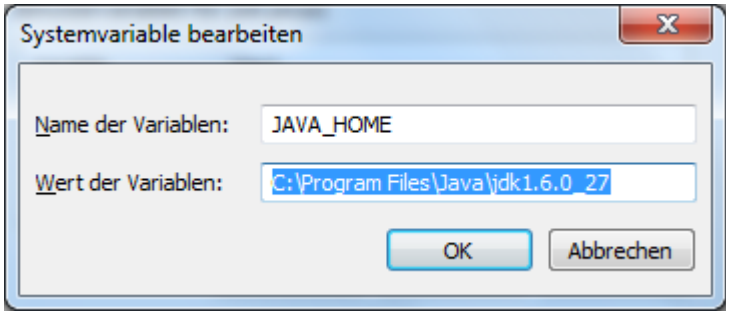
17 Inbetriebnahme

17.1 Installation (Ausführen von „MP_setup.exe“)

17.1.1 Voraussetzungen

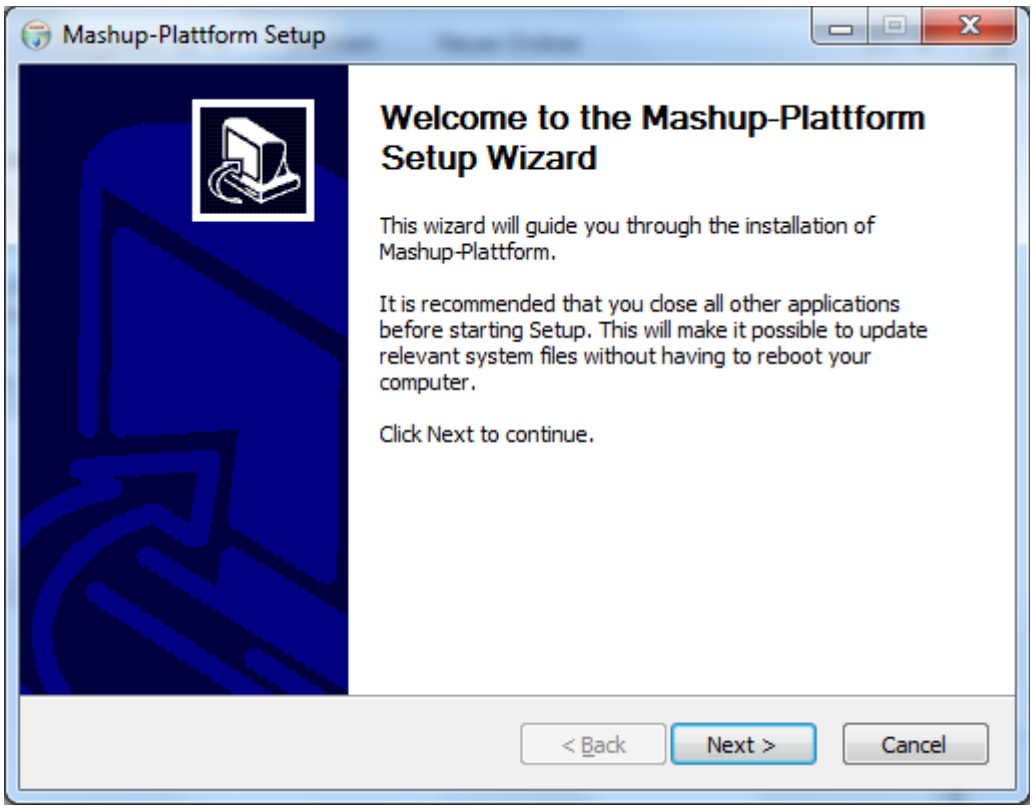
- Windows XP oder höher (getestet mit „Windows 7 64bit & Windows XP 32bit“)
- **JDK** mit JRE **Version 6** (getestet mit „jdk1.6.0_27“ & „jdk1.6.0_30“)

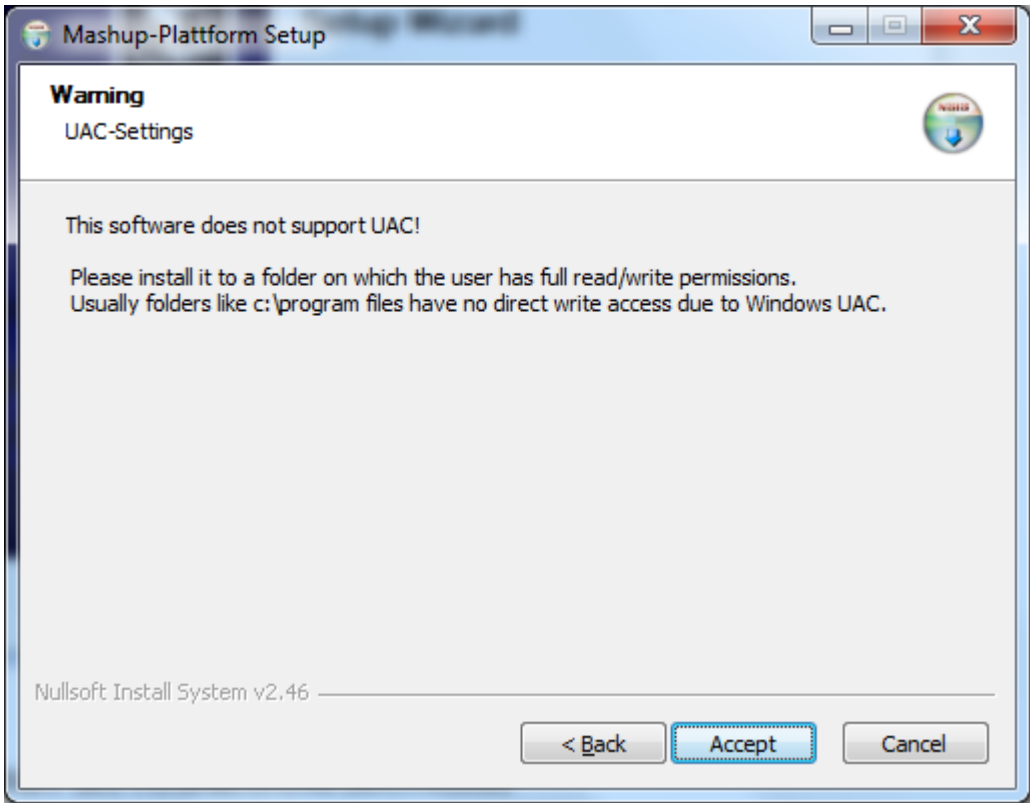
Vor der Installation sollte kontrolliert werden, ob die Umgebungsvariable „JAVA_HOME“ auf ein gültiges Java-Verzeichnis (JDK Version 6) zeigt.



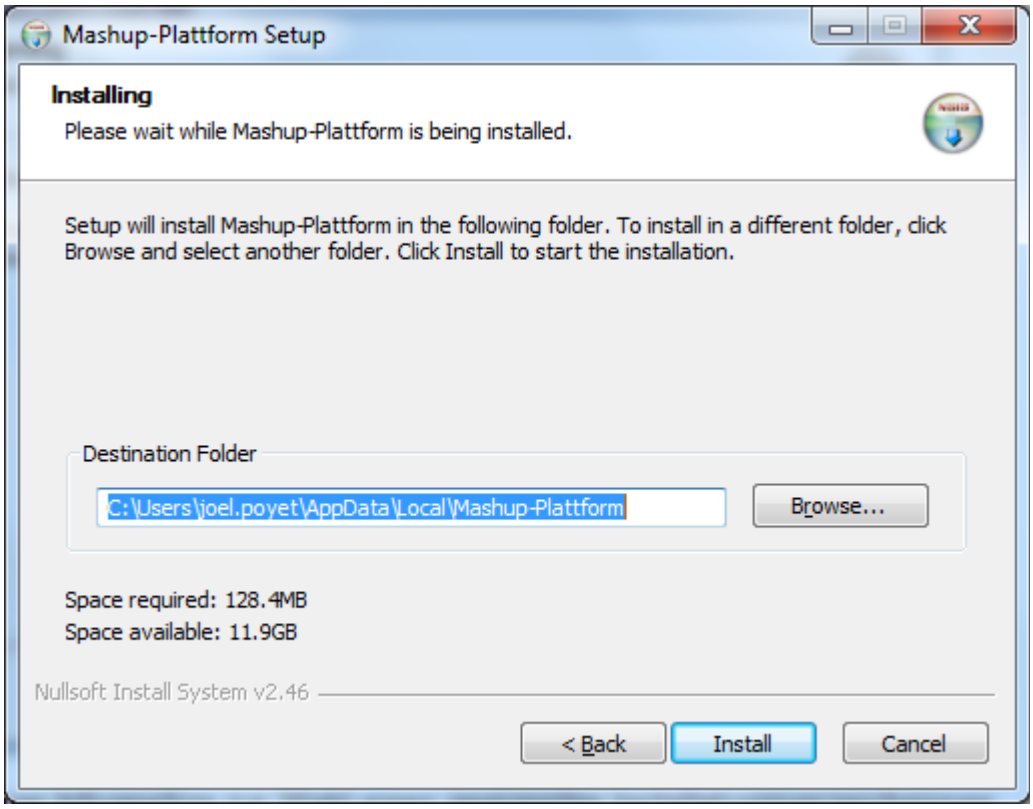
17.1.2 Ausführen der Installationsdatei

Die Installation kann in ein beliebiges Verzeichnis erfolgen. Für die Webumgebung der Mashup-Plattform wird Apache Tomcat in der Version 7 mitgeliefert.

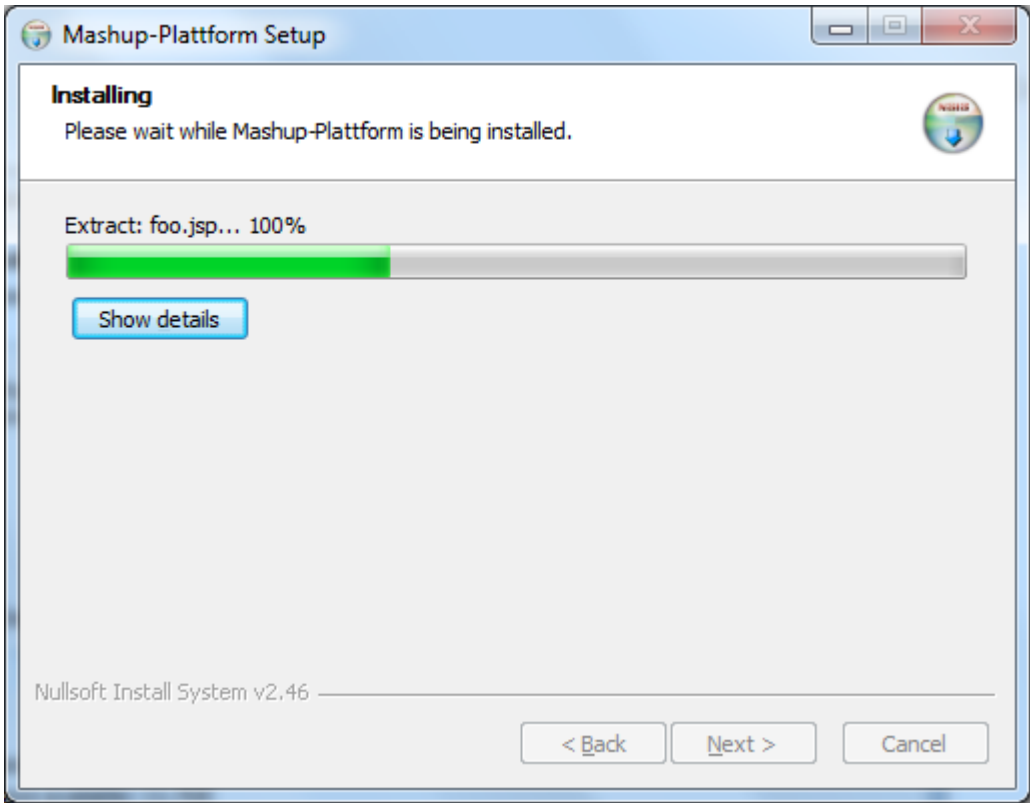




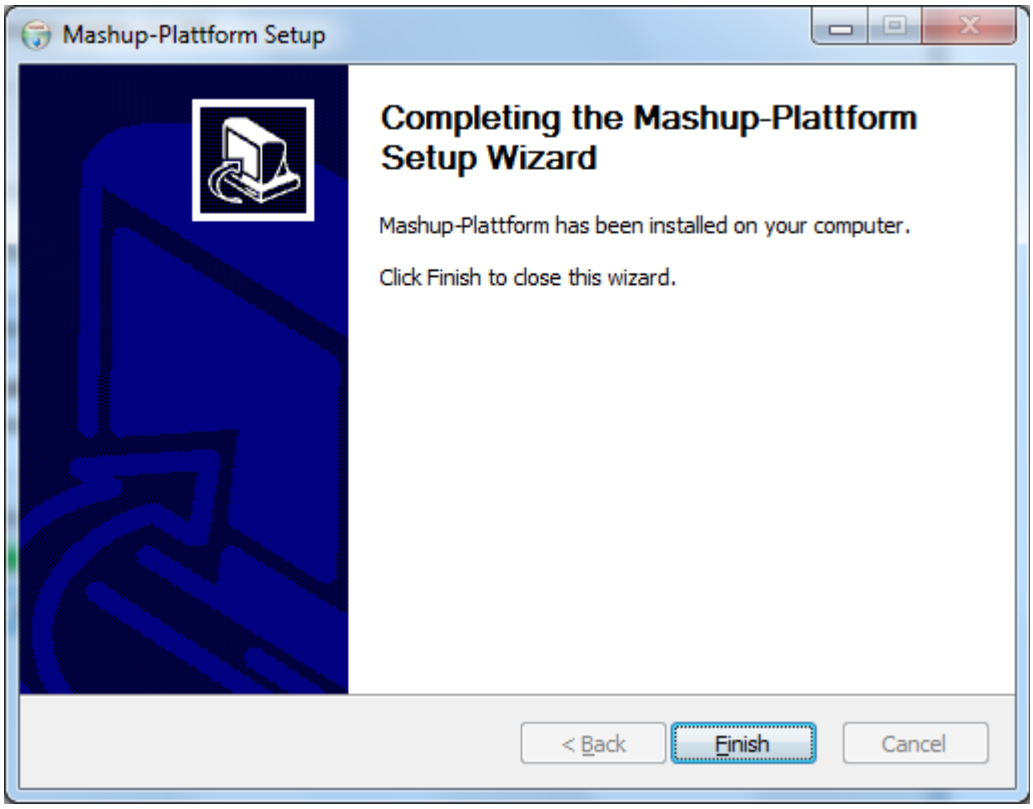
Wichtige Information zur Wahl eines geeigneten Installationsverzeichnisses



Freie Wahl des Installationsverzeichnisses



Fortschrittsanzeige des Entpackens (Binaries, Source Code, Templates, Tomcat)



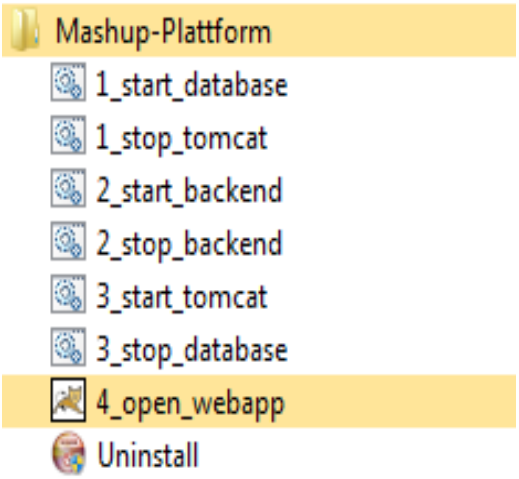
Die Mashup-Plattform wurde erfolgreich installiert.

17.2 Bedienungsanleitung

Die Bedienungsanleitung beschreibt alle für den Betrieb notwendigen Schritte.

17.2.1 Starten der Applikation

Für den Start des Systems stehen entsprechende Verknüpfungen im Startmenü zur Verfügung.



Um die Mashup-Plattform zu starten, öffnen Sie bitte die Verknüpfungen in dieser Reihenfolge:

1. „1_start_database“
2. „2_start_kernel“
3. „3_start_tomcat“

17.2.2 Stoppen der Applikation

Um die Mashup-Plattform zu stoppen, öffnen Sie bitte die Verknüpfungen in dieser Reihenfolge:

1. „1_stop_tomcat“
2. „2_stop_kernel“
3. „3_stop_database“

17.2.3 Öffnen der Website

Um zur Website zu gelangen, öffnen Sie bitte die folgende URL in ihrem Browser:
http://localhost:8180/MP_final/

Alternativ können Sie auch die Verknüpfung „4_Open in Browser (WebApp)“ im Startmenü auswählen.

„MP_final“ ist der Name der Apache Tomcat-Instanz (WebApp).

Bemerkung: Auf der Website sollten Sie eine Tabelle mit den verfügbaren Mashups sehen. Bitte haben sie ein wenig Geduld (~ 1-2 min) bis die ersten Mashup-Seiten erscheinen. Der Kernel generiert (per Default) alle 20 Sekunden eine neue Seite.

17.2.4 Fehleranalyse

Der Kernel ist verantwortlich für das Herunterladen und Zusammenführen der Mashup-Daten. Sämtliche zur Laufzeit entstandenen Fehler werden in den Logdateien („log/error_<date>.txt“) protokolliert.

Zur allgemeinen Fehlerreduktion beachten Sie bitte folgende Punkte:

- Der Rechner, auf dem die Mashup-Plattform installiert ist, sollte über eine direkte Internetverbindung verfügen (kein Proxy).
- Die Datenbank sollte gestartet sein.

- Die Applikation muss zur Laufzeit Schreibrechte auf folgende Verzeichnisse haben:
    1. cache
    2. log
    3. output
    4. TOMCAT/logs
    5. TOMCAT/temp

17.3 Ausführen der Mashup-Plattform aus Eclipse heraus

Falls Sie die Mashup-Plattform als Java-Projekt in Eclipse importiert haben, dann können Sie das Programm auch aus der Eclipse-Umgebung heraus starten.
Die Ausführung von „MP_setup.exe“ ist in diesem Fall nicht mehr notwendig.

Schritte:

1. Die Konfigurationsdateien „/MP_final/config/config.properties“ und „/MP_final/WebContent/config/config.properties“ öffnen und die Parameter gemäss der lokalen Umgebung anpassen.

2. PlattformResetter.java (als „Java-Applikation“) ausführen.
   Hinweis: Dieses Programm setzt die Mashup-Plattform zurück. Der Cache und die Einträge der lokalen Datenbank werden dabei gelöscht.

3. Die „OrientDB“ („/MP_final/db/orientdb_x.x/bin/server.bat“) von einem DOS-Fenster aus starten

4. BotStarter.java (als „Java-Applikation“) starten

5. FrontController.java (als „Server-Applikation“) starten.
   Hinweis: Im Webbrowser sollten Sie jetzt die Mashup-Übersichtsseite sehen. Bitte warten Sie ein paar Minuten, bis die ersten Mashup-Seiten generiert worden sind. Durch Aktualisieren des Webbrowsers können Sie jeweils den aktuellen Stand der generierten Mashup-Seiten sehen.

Bitte kontrollieren Sie gelegentlich auch die Logdateien im Ordner „/MP_final/log/“, um sicherzustellen, dass die Mashup-Plattform ordnungsgemäss läuft.

17.4 Erstellen der Installationsdatei

Die Installationsdatei „MP_setup.exe“ ermöglicht die Installation einer betriebsbereiten Mashup-Plattform auf Windows-Rechnern.

17.4.1 Voraussetzungen

- Die Mashup-Plattform als Java-Projekt in Eclipse
- Windows XP oder höher
- Software „NSIS“ (http://nsis.sourceforge.net)

17.4.2 Anleitung

In diesem Kapitel werden die notwendigen Schritte zur Erstellung der Installationsdatei beschrieben. Gewisse Grundlagen werden dabei vorausgesetzt.

Relevante Ordner/Datei-Struktur für die Paketierung	
/	ROOT
/package.nsi (NSIS Compilation)	Source für die Kompilierung von MP_setup.exe

MP_setup.exe	generierte Installer-Datei
/installer_files	Ordner/Dateien für die Installation
/installer_files/TOMCAT	Tomcat v7
/installer_files/TOMCAT/webapps	Webumgebung
/installer_files/	Start&Stop Batch-Dateien

17.4.2.1 Export des Java-Projekts aus Eclipse

Export Format	Runnable Jar
Launch configuration	BotStarter
Library handling	Package required libraries
Dateiname	kernel.jar

17.4.2.2 Herunterladen & Extrahieren von Apache Tomcat v7

Für den Betrieb der Webumgebung wird Tomcat (ohne Installation) integriert.

Download	http://tomcat.apache.org/download-70.cgi (Windows 32-bit als ZIP-Download)
Pfad	Tomcat v7 Ordner (siehe oben)

17.4.2.3 Export des Tomcat Projekts aus Eclipse

Export Format	WAR
Optimize for	Apache Tomcat v7
Dateiname	MP_final.war
Pfad	*Webumgebung

*Die WAR-Datei muss als Ordner (MP_final) in der Webumgebung extrahiert werden.

17.4.2.4 Integration der Start/Stop-Skripte

Alle Batch-Skripte müssen auf oberster Ebene im installierten Verzeichnis liegen (/installer_files/*.*).
Folgende Befehle werden in den Batch-Skripten verwendet:

Start Kernel:
```
java -classpath "kernel.jar;lib\*"
ch.ffhs.inf09.pa.mashup_platform.core.BotStarter
```

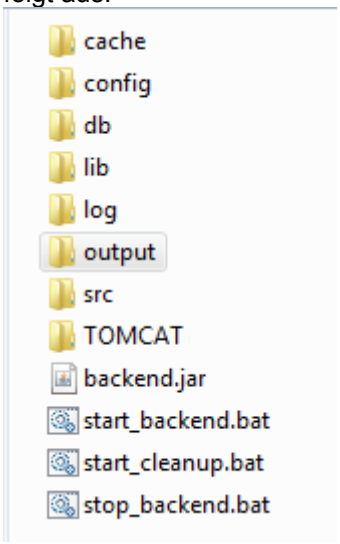
Stop Kernel:
```
java -classpath "kernel.jar;lib\*"
ch.ffhs.inf09.pa.mashup_platform.core.BotStopper
```

Cleanup Kernel:
```
java -classpath "kernel.jar;lib\*"
ch.ffhs.inf09.pa.mashup_platform.core.PlatformResetter
```

17.4.3 Ordner-/Dateistruktur

Für den einwandfreien Betrieb der Plattform müssen gewisse Ordner/Dateien aus der Entwicklungsumgebung übernommen werden.

Nach Übernahme der Ordner/Dateien aus der Entwicklungsumgebung sieht die Ordnerstruktur wie folgt aus:



cache	Speicherung von temporären Mashup-Daten
config	globale Konfiguration der Plattform
db	lokale Datenbank („OrientDB“)
lib	extern benötigte Libraries
log	Logdateien
src\ch\ffhs\inf09\pa\mashup_platform\var	Standardordner für neue Mashups
TOMCAT	Webumgebung (siehe oben)

17.4.4 Kompilierung

Für die Kompilierung einer ausführbaren Installationsdatei muss eine NSI-Datei mit Kompilierungsanweisungen erstellt werden.

Mit Hilfe dieser NSI-Datei lässt sich eine ausführbare Installationsdatei mit folgenden Eigenschaften generieren:

- Standarddialog zur Installation der Software in ein beliebiges Verzeichnis
- wichtige Mitteilungen an die Benutzer
- Überprüfen der installierten Java-Version (Version 6 ist Voraussetzung)
- Überprüfen der Environment-Variable „JAVA_HOME“
- Automatische Anpassung der Java Properties Dateien während der Installation
- Erweiterung des Startmenüs für die einfache Handhabung
- Uninstaller wird bereitgestellt

- Ende Inbetriebnahme -

18 Coderichtlinien

Zur Formatierung des Codes wurde der in Eclipse eingebaute Formatierer („Eclipse [built-in]“) verwendet.

18.1 Begründung

Der Formatierer bietet unter anderem folgende Vorteile:

1. genaue Darstellung der logischen Struktur des Codes

2. Konsequente Darstellung der logischen Struktur

3. Verbesserung der Lesbarkeit des Codes

- Ende Coderichtlinien -

19 Ablagestruktur des Quellcodes

Der Quellcode befindet sich als exportiertes Eclipse-Projekt (Projektname: „MP_final“) auf dem mitgelieferten USB-Stick.
Folgende Software-Komponenten werden benötigt, um das Projekt erfolgreich in Eclipse zu importieren:

- JDK 1.6
- Eclipse JEE
- Tomcat 7 (Anbindung an Eclipse)

Pfad (relativ zum Hauptprojektordner)	Beschreibung
cache	Cache der virtualisierten Datenquellen
config	Ordner mit Konfigurationsdateien (Properties)
config/config.properties	Konfigurationsdatei mit lokalen Einstellungen
db	Binaries der lokalen Datenbank
design	(HTML-)Templates
doc	Javadoc
lib	Fremdbibliotheken (jar-Files)
log	Ordner für die Logdateien
output	Buffer der Mashup-Outputs
src	Quellcode-Ordner
src/ch/ffhs/inf09/pa/mashup_platform/core/	Quellcode des Kernels. Mit Hilfe der ausführbaren Java-Klassen „BotStarter“ und „BotStopper“ kann der Kernel gestartet/gestoppt werden.
src/ch/ffhs/inf09/pa/mashup_platform/var/	Hauptordner der erstellten Mashups
src/ch/ffhs/inf09/pa/mashup_platform/web/	Quellcode der Webumgebung
src/ch/ffhs/inf09/pa/mashup_platform/web/controller/FrontController.java	Servlet-Klasse, welche **alle** HTTP-Requests entgegennimmt. Mit „Run as => Run on server“ kann die Klasse in Tomcat gestartet werden. Als Startseite erscheint eine Übersicht über die vorhandenen Mashups.
test	JUnit-Tests
WebContent	Hauptordner der Webumgebung
WebContent/config	Ordner mit Konfigurationsdateien (Properties)
WebContent/css	Ordner mit CSS-Dateien
WebContent/images	Ordner mit Bildern
WebContent/WEB-INF/lib/	Fremdbibliotheken

20 Testprotokolle

20.1 Testfälle

Nr.	Testfall	Erwartungen
1	Datenbank	
1.1	Start Datenbank: • Sind die lokalen Netzwerk Ports geöffnet? • Dos-Fenster gestartet?	1. Port 127.0.0.1:2424 & 127.0.0.1:2480 wird abgehört (netstat -an) 2. DOS-Fenster bleibt erhalten und die letzte Zeile lautet: `Listening for distributed nodes on IP multicast /235.1.1.1:2424`
1.2	Stop Datenbank: • Sind die lokalen Netzwerk Port geschlossen? • DOS-Fenster	1. Port 2424 und 2480 nicht mehr geöffnet (keinen Eintrag mit netstat -an) 2. DOS-Fenster bleibt erhalten und die letzte Zeile lautet: `PAUSE`
2	Backend	
2.1	Vorbedingungen: – Datenbank gestartet Start Backend: 1. Dos-Fenster gestartet? 2. Logdatei 3. Initialstart: Wurden die Datenbanken angelegt?	1. DOS-Fenster bleibt erhalten mit der Meldung: `Mashup Backend is running...` 2. Loggdatei wurde im Format log_<heutiges Datum>.txt im Ordner log angelegt → bei Fehler wird Error-Logdatei im Format error_<heutiges Datum>.txt angelegt 3. Im Verzeichnis db/orientdb-<version>/databases müssten die Ordner „mashups“ und „users“ angelegt worden sein
2.2	Stop Backend: 1. Dos-Fenster	1. DOS-Fenster bleibt erhalten und die letzte Zeile lautet: `Mashup Backend is stopped!`
2.3	Ablage: 1. Wurden die Cache Dateien angelegt? 2. Wurde der Mashup-Inhalt in die Datenbank übertragen?	1. Jeder Mashup generiert nach dem Start des Backend Cache Dateien im Verzeichnis cache/ mit dem Mashup-Namen. 2. Über das OrientDB Studio http://localhost:2480 lässt sich der Mashup-Inhalt überprüfen
2.4	Mashup-Properties: 1. Änderung des Mashup-Namens 2. Änderung der primären Quelle 3. Änderung der Limits	1. Der neue Mashup-Name darf die Stabilität/Funktion des Plattform nicht beeinträchtigen → Neuer Name muss auf der Weboberfläche ersichtlich sein 2. Die Inhalte werden durch die neue Quelle ersetzt 3. Stabilität/Funktion darf nicht beeinträchtigt werden. Weboberfläche wird entsprechend den neu gesetzten Limits aktualisiert
3	Webumgebung	
3.1	Vorbedingungen: – Datenbank gestartet – Backend gestartet Start Tomcat: 1. Dos-Fenster gestartet? 2. Wurden die Netzwerk-Ports geöffnet? 3. Apache Tomcat Defaultseite erreichbar? 4. WebApp „MP_final“ erreichbar?	1. DOS-Fenster bleibt erhalten mit der Meldung: `INFO: Server startup in <time> ms` 2. TCP Port 0.0.0.0:8180 & 0.0.0.0:8109 müssen auf Verbindungen warten (netstat -an) 3. Beim Öffnen der URL „http://localhost:8180“ sollte die Standardseite von Apache Tomcat angezeigt werden 4. Beim Öffnen der URL „http://localhost:8180/MP_final/“ sollte eine Übersichtseite mit allen bestehenden Mashups angezeigt werden.

3.2	Mashup-Overview: 1. Anzeige fehlerfrei? 2. Filter-Funktionen aktiv? 3. Suchfunktion aktiv? 4. Link „Show“ funktionstüchtig?	1. Design und Layout fehlerfrei mit IE & Firefox 2. Alle Filter-Funktionen reagieren auf Input (ohne Nachladen der Seite) 3. Suchfunktion muss auf Input reagieren (ohne Nachladen der Seite). Alle Felder werden in die Suche miteinbezogen 4. Der Link muss bei allen aufgelisteten Mashups erscheinen und auf den korrekten Mashup verweisen
3.3	Mashup-Page/Template: 1. Prüfen des Inhalts 2. Prüfen der Links 3. Prüfen des Pagination	1. Stimmt der Inhalt mit dem Inhalt in der DB überein 2. Keine Dead-Links, Links stimmt mit Bild/Text überein 3. Es kann durch den Mashup geblättert werden. Die Anzahl Seiten sind in den Properties limitiert. Die Links müssen auf auf Dead-Links geprüft werden.
3.4	Menü: 1. Herunterklappen/Hochklappen 2. Login 3. Logout	1. Menü muss auch beim Wechsel auf andere Seiten die Darstellung beibehalten 2. Nach Login erweitert sich das Menü um den Eintrag „Account“. 3. Nach Logout soll man wieder aus die Mashup-Overview Seite gelangen. Das Menü ändert sich in den Ursprung.

20.2 Testprotokolle

Testfallnr.	Programm Release	Datum	Tester	Resultat
1.1	0.1	9.12.2011	Jan	OK
1.2	0.1	9.12.201	Jan	OK
2.1	0.1	10.12.2011	Joël	OK
2.2	0.1	10.12.2011	Joël	OK
2.3	0.1	10.12.2011	Joël	OK
2.4	0.1	10.12.2011	Joël	OK
3.1	0.1	15.12.2011	Jan	OK
3.2	0.1	15.12.2011	Jan	OK
3.3	0.1	15.12.2011	Jan	OK
3.4	0.1	15.12.2011	Jan	OK

20.3 JUnit-Tests

Die JUnit-Tests sind erfolgreich durchgeführt und decken 85% des Codes ab.

- Ende Testprotokolle -

21 Statusbericht I

21.1 Gesamtstatus

Das Team konkretisierte in der ersten Woche nach dem Kick-Off-Meeting vom 19.08.2011 die Projektidee mit Hilfe von Brainstorming und Online-Diskussionen.

Bei einem persönlichen Treffen wurden aus den eingegangenen Vorschlägen ein gemeinsames Ziel abgeleitet.

Gesamtziel

Das Endprodukt dieser Projektarbeit ist eine Software, welche dem User das Erstellen und Ausführen eigener Mashups* ermöglicht.
Jeder Mashup bildet eine Datenquelle, die genutzt werden kann, um daraus neue Mashups zu kreieren.
Auf diese Weise können User kollaborativ Daten aus dem Internet filtern und strukturieren.

Nach der Fixierung der Projektidee wurden die Rollen für den weiteren Projektverlauf verteilt.

Rollenverteilung

Alexander Matt (Projektmitarbeiter)
Joël Poyet (Auftraggeber)
Jan Wild (Projektmitarbeiter)
Malte Wintner (Projektleiter)

Der Fokus in der zweiten Woche lag bei der Einführung eines Informations- und Berichtswesens. Da die Zusammenarbeit zum grössten Teil nur über das Internet erfolgen konnte, wurden geeignete Web-Programme (Skype, WIKI, Bug Tracker, ...) auf dem lokalen Rechner, beziehungsweise auf einem gehosteten Webserver installiert.

Nach Verabschiedung des Projektauftrages definierte das Team die Teilziele und Anwendungsfälle.

Als Teil der Machbarkeitsstudie wurde der Beschluss gefasst, einen Prototypen zu entwickeln.

In den darauf folgenden Wochen wurde der Prototyp entworfen und einzelne Basis-Klassen implementiert.

21.1.1 Bewertung

Das Projekt verlief bis an hin planmässig. Besonders in den wöchentlich durchgeführten Online-Treffen wurde ersichtlich, dass das Team begeistert bei der Projektarbeit war und viele Ideen einbrachte.

Beim Entwurf des Prototypen wurde das Konzept der „Virtualisierung“ verfolgt, um die Heterogenität der Web-Datenquellen hinter einer Abstraktionsschicht zu verbergen.

Dieses Konzept legte die Basis für die weitere Entwicklung.

* Unter Mashup wird bei dieser Projektarbeit das Verknüpfen von Web-Inhalten unterschiedlicher Datenquellen verstanden.

21.2 Termine

Bezeichnung	Status	Deadline
Projektidee fixieren	erledigt	26.08.11
Kommunikation etablieren	erledigt	02.09.11
Anwendungsfälle aufzeichnen	erledigt	05.09.11
Anforderungen und Teilziele festlegen	erledigt	09.09.11
Prototyp entwerfen	erledigt	16.09.11
Prototyp implementieren	offen	30.09.11
Prototyp analysieren	offen	07.10.11
Kernsystem (1. Inkrement) entwerfen	offen	21.10.11
Kernsystem implementieren	offen	28.10.11
Kernsystem testen	offen	04.11.11
Gesamtsystem (2. Inkrement) entwerfen	offen	18.11.11
Gesamtsystem implementieren	offen	02.12.11
Gesamtsystem testen und Bugs beheben	offen	16.12.11

21.2.1 Bewertung

Status: planmässig
Besonderen Wert wurde auf einen klaren und vollständigen Entwurf des Prototypen gelegt, um die Implementierungszeit zu verkürzen und die Qualität zu sichern.

21.3 Lieferobjekte

Bezeichnung	Status	Deadline
Projektauftrag	erledigt	02.09.11
grober Projektstrukturplan (wird mit der Zeit detaillierter)	(erledigt)	02.09.11
Systemidee	erledigt	02.09.11
Systemanwendungsfälle	erledigt	05.09.11
Lösungsansatz	erledigt	09.09.11
Softwarearchitektur (Prototyp)	erledigt	16.09.11
Klassenmodelle (Prototyp)	erledigt	16.09.11
Coderichtlinien	erledigt	16.09.11
1. Statusbericht (vorliegendes Dokument)	erledigt	20.09.11
Softwarearchitektur (1. Inkrement)	offen	14.10.11
Klassenmodelle (1. Inkrement)	offen	21.10.11
2. Statusbericht	offen	15.11.11
Softwarearchitektur (2. Inkrement)	offen	15.11.11
Klassenmodelle (2. Inkrement)	offen	18.11.11
Domänenmodell	offen	01.12.11
Systemablaufmodelle	offen	01.12.11
Schnittstellenbeschreibung	offen	01.12.11
Datenmodell	offen	01.12.11
Dynamische Modelle	offen	01.12.11
Testkonzept	offen	01.12.11
GUI-Spezifikationen	offen	01.12.11

21.3.1 Bewertung

Status: planmässig
Für die Software-Entwürfe wurden UML-Klassendiagramme als bevorzugtes Mittel eingesetzt.

21.4 Qualität

Status: planmässig
Das Know-how des Teams wurde gebündelt, um Design-Fehler frühzeitig zu erkennen.

Durch strenge Richtlinien beim Entwurf des Klassendesigns konnte die Qualität stetig verbessert werden.

21.5 andere Probleme

Das Synchronisieren und Committen des Quellcodes in das Git-Repository bereitete anfangs Probleme. Aber diese konnten mit etwas Übung behoben werden.

21.6 Nächste Schritte

Der nächste Schritt ist die Fertigstellung des Prototypen.
Dieser Meilenstein ist sehr wichtig, da er nicht nur Aufschluss über die technische Machbarkeit, sondern auch die Stärken und Schwächen bei der Teamarbeit aufzeigen wird.
Allfällige Schwachpunkte können dann rechtzeitig korrigiert werden, um das Endprodukt erfolgreich zu implementieren.

21.7 aktualisierte Rollenorganisation

Alexander Matt (Projektleiter)
Joël Poyet (Projektmitarbeiter)
Jan Wild (Projektmitarbeiter)
Malte Wintner (Auftraggeber)

21.8 Vorgehensmodell

Das Team entschied sich für eine inkrementelle Entwicklung, um die Komplexität zu reduzieren. Geplant wurden zwei Inkremente, wobei für das Kernsystem (1. Inkrement) zuerst ein Prototyp entworfen wurde.

Die Qualität des Prototypen wird voraussichtlich schon sehr nahe an das endgültige Kernsystem heran kommen.

Die anstehende Programmierung wird von gegenseitigem Code Review begleitet, um allfällige Fehler frühzeitig zu erkennen und eine einheitliche Codierung zu erreichen.

- Ende Statusbericht I -

22 Statusbericht II

22.1 Gesamtstatus

Das Team implementierte den Prototypen gemäss den Vorgaben. Die aus der Analyse des Prototypen gewonnenen Erkenntnisse flossen in das Kernsystem ein.
Zu den wichtigsten Erweiterungen des Kernsystems gehörten:

Bezeichung	Details
Cache-System	Durch das Zwischenspeichern der Daten konnten die Zugriffe auf die Fremddatenquellen reduziert und gesteuert werden. Auf diese Weise wurde auch die CPU des Rechners entlastet. Der Cache berücksichtigte ausserdem die Beschaffenheit der Daten. Im Gegensatz zu aktuellen Nachrichten, müssen beispielsweise historische Fakten nur sehr selten erneuert werden.
Logging-System	Um die Qualität bei ansteigender Komplexität des Systems zu sichern, wurde ein zentralisiertes Protokollieren aller wichtigen Prozessschritte eingeführt.

Nach der Fertigstellung des Kernsystems wurde die bisherige Zusammenarbeit im Team ausgewertet. Es zeigte sich, dass die einzelnen Mitglieder sich besonders für bestimmte Bereiche der Mashup-Plattform interessierten. Unter Berücksichtigung dieser Tatsache konnten die Verantwortlichkeiten für die zukünftigen Komponenten des Systems den einzelnen Mitgliedern zugewiesen werden:

Komponente	verantwortlich	Details
Datenbank-System	Alexander Matt	- Evaluierung und Auswahl der Datenbanktechnologie - Administration der Datenbank - Bereitstellung der Datenbank-Anbindungen - Implementierung des Caches
Front-End	Joël Poyet	- Administration des Webservers - Implementierung der Template-Engine - GUI-Programmierung (Anzeige und Navigation der Mashups)
Benutzer-verwaltung	Jan Wild	- (webbasierte) Authentifizierung - Rechte- und Rollenverwaltung - Deployment von Mashups
Kernel	Malte Wintner	- Systemklassen zum Definieren von Mashups - Virtualisierung der Fremddatenquellen - Mashup-Ausführungsmechanismus

22.2 Bewertung

Status: **planmässig**
Mit der Implementierung des Kernsystems wurde ein wichtiger Meilenstein erreicht.
Die erfolgreichen Tests bestätigten die im Vorfeld aufgestellten Behauptungen, dass der objektorientierte Ansatz (im Gegensatz zu einer Abfragesprache wie SQL oder YQL) Vorteile beim Definieren von Mashups im Bezug auf Übersichtlichkeit und Erweiterbarkeit hatte.

22.3 Termine

Bezeichnung	Status	Deadline
Projektidee fixieren	erledigt	26.08.11

Kommunikation etablieren	erledigt	02.09.11
Anwendungsfälle aufzeichnen	erledigt	05.09.11
Anforderungen und Teilziele festlegen	erledigt	09.09.11
Prototyp entwerfen	erledigt	16.09.11
Prototyp implementieren	erledigt	30.09.11
Prototyp analysieren	erledigt	07.10.11
Kernsystem (1. Inkrement) entwerfen	erledigt	21.10.11
Kernsystem implementieren	erledigt	28.10.11
Kernsystem testen	erledigt	04.11.11
Webanwendung (2. Inkrement) entwerfen	erledigt	18.11.11
Webanwendung implementieren	offen	02.12.11
Webanwendung testen und Bugs beheben	offen	16.12.11

22.4 Bewertung

Status: **planmässig**

22.5 Lieferobjekte

Bezeichnung	Status	Deadline
Projektauftrag	erledigt	02.09.11
Projektstrukturplan	erledigt	02.09.11
Phasenplan	erledigt	02.09.11
Terminplan	erledigt	02.09.11
Systemidee	erledigt	02.09.11
Systemanwendungsfälle	erledigt	05.09.11
begründeter Lösungsansatz	erledigt	09.09.11
Softwarearchitektur (Prototyp)	erledigt	16.09.11
Klassenmodelle (Prototyp)	erledigt	16.09.11
Coderichtlinien	erledigt	16.09.11
1. Statusbericht (vorliegendes Dokument)	erledigt	20.09.11
Softwarearchitektur (1. Inkrement)	erledigt	14.10.11
Klassenmodelle (1. Inkrement)	erledigt	21.10.11
2. Statusbericht	erledigt	15.11.11
Softwarearchitektur (2. Inkrement)	erledigt	15.11.11
Klassenmodelle (2. Inkrement)	erledigt	18.11.11
Domänenmodell	offen	01.12.11

Systemablaufmodelle	offen	01.12.11
Schnittstellenbeschreibung	offen	01.12.11
Datenmodell	offen	01.12.11
Dynamische Modelle	offen	01.12.11
Testkonzept	offen	01.12.11
GUI-Spezifikationen	offen	01.12.11
Lieferobjekte und Installationsanleitung	offen	15.12.11
Quellcode (abgelegt auf USB-Stick)	offen	15.12.11
Testprotokolle	offen	15.12.11

22.6 Bewertung

Status: **planmässig**
Das Verfassen der Dokumente beanspruchte mehr Zeit als anfangs erwartet.

22.7 Qualität

Status: **planmässig**
Die Qualität wurde durch das gegenseitige Überprüfen der Dokumente und des Programmcodes gesichert. Ausserdem konnten Fehler durch das Einführen eines zentralisierten Logging-Systems viel früher erkannt und behoben werden.

22.8 andere Probleme

keine

22.9 nächste Schritte

Der nächste Schritt ist die Implementierung des Gesamtsystems (2. Inkrement). Jedes Teammitglied konzentriert sich dabei auf die ihm zugewiesene Komponente.

22.10 aktualisierte Rollenorganisation

Alexander Matt (Auftraggeber)
Joël Poyet (Projektmitarbeiter)
Jan Wild (Projektleiter)
Malte Wintner (Projektmitarbeiter)

- Ende Statusbericht 2 -

23 Evaluierungsbericht

Das Projekt ist aus Sicht des Teams sehr gut verlaufen. Alle Mitglieder waren engagiert und unterstützten sich gegenseitig.
Dank der wöchentlichen Meetings konnten Unklarheiten und Probleme schnell behoben werden.
Ein entscheidender Meilenstein in diesem Projekt war die Fertigstellung des ersten Prototypen. Seine erfolgreichen Resultate gaben dem Team Vertrauen.
Das Projektergebnis ist eine lauffähige Software, die mit Hilfe des Beispiel-Mashups „Portrait Of Finnish Bands“ übersichtliche und informative Mashup-Webseiten generieren kann.
Die Planungsziele sind erreicht. Um die Leistungsfähigkeit der Mashup-Plattform zu demonstrieren, hätte das Team gerne noch weitere Beispiel-Mashups definiert, was aus zeitlichen Gründen nicht mehr möglich war.

Das Projektresultat ist für das Team eine Bestätigung, dass auch komplexere Systeme in kurzer Zeit realisierbar sind. Der Erfolg begründet sich auf den konsequenten Einsatz von Projektmanagement, Software-Engineering und Objektorientierter Programmierung.