Hochschule für Technik und Wirtschaft Berlin
Fachbereich Wirtschaftswissenschaften II – Angewandte Informatik

Diplomarbeit

Entwicklung eines Recommender-Systems auf Basis von Indoor-Tracking-Daten zur Generierung personalisierter Informationssysteme

Benedikt Hebeisen
Mai 2009

Erstprüfer: Prof. Dr. Jürgen Sieck
Zweitprüfer: Prof. Dr. Albrecht Fortenbacher

Ich möchte den Personen danken, die mich beim Erstellen dieser Arbeit unterstützt haben und mir mit Rat und Ideen zur Seite standen.

Ganz besonders danke ich Prof. Dr. Jürgen Sieck für die hilfreiche und sehr engagierte Betreuung dieser Arbeit und die Diskussionen und vielfältigen Denkanstöße. Ebenso danke ich Prof. Dr. Albrecht Fortenbacher, der die Arbeit als Zweitkorrektor betreut hat.

Mein Dank gilt den Menschen, die stets ein offenes Ohr für Diskussionen über diese Arbeit hatten und mich mit Ihren Ideen und Anmerkungen unterstützt haben, dies gilt insbesondere für die Korrektur-LeserInnen.

Inhaltsverzeichnis

1 Einführung

Ein Besuch in einem Museum stellt die Besucher meist vor das Problem, auf der einen Seite aufgrund zeitlicher Beschränkungen nicht die komplette Ausstellung besuchen zu können, auf der anderen Seite aber auch zu einzelnen Exponaten und Themen tiefergehende Informationen zu wünschen. Diese können die Besucher meist durch einen Ausstellungskatalog erhalten, wobei es normalerweise nur eine Version des Ausstellungskatalogs für alle Besucher gibt. Weiterführende Informationen, die auf die Zielgruppe, die persönlichen Interessen oder Schwerpunkte zugeschnitten sind, müssen selbst zusammengestellt werden. Zeitabhängige und aktuelle Informationen, z. B. zu Veranstaltungen und weiterführenden Wanderausstellungen, bleiben in der Regel unberücksichtigt.

Recommender Systeme können die Besucher dabei unterstützen, den Museumsbesuch zu planen, aber auch im Rückblick auf einen Museumsbesuch auf die Themen einzugehen, die besonderes Interesse geweckt haben. Dazu können nähere Informationen zur Verfügung gestellt werden. Derartige Systeme ermöglichen, die in einem Informationspool vorhandenen Daten nach den Vorzügen und Voraussetzungen des Besuchers zu filtern und automatisiert Informationen zu den Themen und Schwerpunkten anzubieten, die den Interessen entsprechen. Dazu ist es notwendig, ein Profil des Nutzers zu kennen, mit dessen Hilfe sich die Informationen filtern lassen.

Digitale Museumsguides erlauben es Besuchern, neben den herkömmlichen Audio-Informationen auch weiterführende Multimedia-Inhalte zu der besuchten Ausstellung, dem jeweiligen Ausstellungsbereich und einzelnen Exponaten zu bekommen. Dabei ermöglicht die RFID-Technologie den Standort der Besucher automatisch zu erkennen. Über digitale Museumsguides, beispielsweise auf Basis von PDA, können die Position erkannt und die passenden Informationen bereit gestellt werden.

Durch die vorherige Eingabe eines Interessenprofils kann ein digitaler Museumsguide die Besucher direkt auf für sie interessanten Objekte aufmerksam machen und den Weg in bestimmte Abteilungen des Museums lenken. Ein Analyse-System auf dieser

Basis ermöglicht es, den Weg eines Besuchers nachzuvollziehen und zeigt, welche Abteilungen und Segmente besucht und welche Exponate betrachtet wurden. Durch eine Analyse des Ganges durch die Ausstellung werden die Interessen und Präferenzen erkennbar aus denen sich ein Interessen-Profil des Nutzers ableiten lässt. Dieses Interessen-Profil kann als Grundlage für den Einsatz eines Recommender Systems dienen, das dem Besucher nach Verlassen der Ausstellung weitere Informationen zur Verfügung stellt. Bei einem derartigen System ist es unerlässlich, hohe datenschutzrechtlichen Standards einzuhalten und Mechanismen zu implementieren, die den Schutz der persönlichen Daten sicher gewährleisten.

1.1 Zielsetzung

Im Rahmen dieser Arbeit wird ein System entwickelt, das mit Hilfe von Recommender Algorithmen ein persönlich zugeschnittenes Informationsportal zusammenstellt. Grundlage für die Zusammenstellung der Informationen ist das persönliche Profil des Nutzers, das auf Basis der über einen digitalen Museumsguide gewonnenen Bewegungsinformationen auf RFID-Basis automatisiert erstellt wird.

Die vorhandenen Informationen müssen für die automatisierte Empfehlung in einer definierten Form aufbereitet sein. Dies geschieht über ein Profil der Objekte, das über Metadaten die Informationen beschreibt. Zwischen Profil des Nutzers und den Informationsbeständen lassen sich so Verknüpfung herstellen, die über Filterprozesse den vorhandene Informationsbestand eingrenzen.

1.2 Aufbau der Arbeit

Zunächst wird in *Kapitel 2* auf die *Grundlagen* dieser Arbeit eingegangen. Dies beinhaltet die Möglichkeiten zur Lokalisierung von mobilen Geräten auf der technischer Seite und die Modelle, die den technischen Konzepten zugrunde liegen. Als zweiter Teil der Grundlagen werden Recommender System eingeführt und die verschiedenen Verfahren vorgestellt. Daran schließt sich eine Betrachtung von bestehenden Systemen an. Abgeschlossen wird das Kapitel mit der Diskussion von Aspekten des Datenschutzes.

Die *Anforderungen* an das zu entwickelnde System werden in *Kapitel 3* anhand von konkreten Szenarien und Anwendungsfällen analysiert und entwickelt.

Als Vorbereitung für den Systementwurf werden in *Kapitel 4* die *Rahmenbedingungen und Lösungsstrategien* beleuchtet. Dies ist in erster Linie die Beantwortung der Frage, wie mit den Anforderungen, die sich aus der Analyse ergeben haben, umgegangen werden kann. Zum einen sind dies die Umgebung und Vorgaben, denen das System ausgesetzt ist. Zum anderen wird betrachtet, welche Strategien zielführend sind und wie sich die Anforderungen sinnvoll umsetzen lassen. Dies betrifft jedoch in erster Linie die konzeptionelle Seite.

Der Aufbau des System und die konkrete Architektur wird in *Kapitel 5 – Systementwurf* diskutiert. Die konkrete Umsetzung und Realisierung dieser Ergebnisse wird dann in *Kapitel 6 – Implementierung* betrachtet.

2 Grundlagen

Als Basis für die Arbeit werden im folgenden Kapitel zunächst die zugrunde liegenden Techniken diskutiert. Dies umfasst, als Grundlage für positionsabhängige Dienste innerhalb von Gebäuden, zunächst die Möglichkeiten der Positionsbestimmung und die hier in Frage kommenden Technologien. Daran schließt sich die Besprechung von Techniken zur Entwicklung von Recommender Systemen an. Hier werden die verschiedenen Verfahren diskutiert und näher erläutert. An dieser Stelle werden auch existierende Lösungen und die dort verwendeten Techniken betrachtet. Die Arbeit mit personenbezogenen Daten bringt zwingend die Notwendigkeit des Schutzes dieser Daten mit sich. Im abschließenden Abschnitt wird daher auf Aspekte des Datenschutzes eingegangen.

2.1 Indoor Lokalisierung

Grundsätzlich eignen sich zur Lokalisierung verschiedene auf Funk-Technologien basierte Verfahren, die im nachfolgenden Abschnitt verglichen und betrachtet werden. Die Auswahl wurde auf Techniken beschränkt, die sich für die Ortung in geschlossenen Räumen eignen. Dies schließt einerseits die Verwendung von Satelliten-gestützten Verfahren wie GPS[1], andererseits auch die Ortung durch Mobilfunk-gestützte Verfahren (GSM[2], UMTS[3]) aus, da diese keine zuverlässigen Angaben zu Verfügbarkeit und Genauigkeit in Gebäuden zulassen. Die Anforderungen an Lokalisierungsverfahren

1 Global Positioning System
2 Global System for Mobile Communications
3 Universal Mobile Telecommunications System

lassen sich durch folgende Kriterien beschreiben:

- Dauer der Messung
- Genauigkeit/Konsistenz
- Zuverlässigkeit
- Kosten
- Einschränkungen

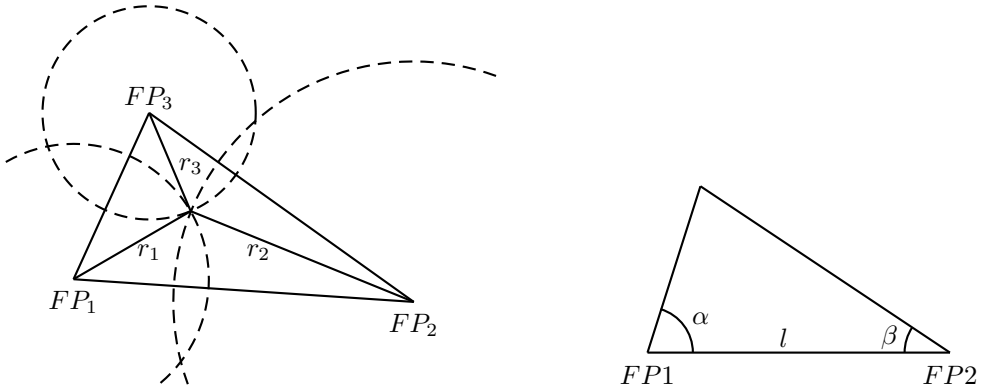
2.1.1 Verfahren zur Positionsbestimmung

Die Positionsbestimmung in Bezug auf bekannte Fixpunkte lässt sich durch die drei Verfahrensklassen *Näherungsverfahren*, *geometrische Verfahren* und *auf Tabellen basierte Verfahren* bestimmen. Die Qualität der Lokalisierung ist dabei von der verwendeten Technik und den damit zur Verfügung stehenden Möglichkeiten abhängig. Die Bestimmung der Position erfolgt entweder über Näherungsverfahren bei Identifizierung des Senders oder durch die Analyse eines oder mehrerer Signale. Diese Verfahren basieren auf der Messung der Feldstärke des Signals (elektromagnetische Lauflänge) oder von geometrischen Eigenschaften.

Näherungsverfahren

Näherungsverfahren wie die Methode *Cell of Origin* (COO) sind die einfachsten Verfahren zur Positionsbestimmung. Voraussetzung ist eine Infrastruktur, die aus einzelnen Sendern mit entsprechenden Funkzellen besteht. Die Bestimmung der Position des Empfängers erfolgt über das Auslesen der Zellen-Identifikation bzw. über die Prüfung, welcher Empfänger sich in Reichweite befindet. Die Genauigkeit ist dabei auf die Bestimmung der Funkzelle beschränkt. Die Position innerhalb der Funkzelle lässt sich nicht genauer feststellen, die absolute Genauigkeit ist also von der Größe der Funkzelle abhängig. Präzisere Positionsbestimmungen erfordern entweder den Empfang mehrerer Sender gleichzeitig oder die Messung des Sendesignals auf die Signal-Laufzeit (*Time of Arrival*) oder Intensität (RSSI[1]).

1 Received Signal Strength Indicator



(a) Trilateration **(b)** Triangulation

Abbildung 2.1: Geometrische Verfahren zur Positionsbestimmung

Geometrische Verfahren

Geometrische Größen und Beziehungen zwischen bekannten Bezugspunkten und dem zu lokalisierenden Punkt erlauben eine genauere Bestimmung der Position als Näherungsverfahren wie COO. Die grundlegenden geometrischen Ansätze sind dabei die *Trilateration* und die *Triangulation*.

Bei der Trilateration wird die Position über die Entfernung zu bekannten Bezugspunkten bestimmt. Um eine Position eines Punktes in einer zweidimensionalen Ebene zu ermitteln, muss die Entfernung des Punktes zu drei Fixpunkten bekannt sein. Zur Bestimmung einer Position in einem Raum sind vier Fixpunkte erforderlich. Über den Schnittpunkt der Kreisgleichungen $r_a^2 = (x - x_a)^2 + (y - y_a)^2$ (wobei $r =$ Entfernung), lässt sich die Position in Bezug auf die Fixpunkte errechnen. Im dreidimensionalen Raum erfolgt die Berechnung über die Kugelgleichungen.

Sind die Winkel zwischen dem zu bestimmenden Punkt und den Fixpunkten bekannt, lässt sich die Position über eine Triangulation bestimmen. Voraussetzung sind hier in der Ebene zwei bekannte Fixpunkte bzw. drei Fixpunkte im Raum und die Position der Fixpunkte zueinander.

Tabellen-basierte Verfahren/Szenenanalyse

Bei Tabellen-basierten Verfahren wird das empfangene Signal mit vorher ausgemessenen Signal-Werten verglichen und die Position mit Hilfe von Vergleichswerten bestimmt. Dazu werden die Signalcharakteristika (üblicherweise die Signalstärke)

einer Umgebung ausgemessen und auf einer *Radio Map* vermerkt. Die Position wird dann über eine Mustererkennung des Signal-Profils erkannt.

Signal-Messungen

Grundlage für geometrische Lokalisierungsverfahren ist die Bestimmung des geometrischen Abstands oder der Richtung zwischen Sender und Empfänger. Beim *Angle of Arrival*-Verfahren (AOA) messen spezielle Antennen den Einfallswinkel der Signale, was die Bestimmung der Position über eine anschließende Triangulation ermöglicht. Die Messung des RSSI erlaubt die Bestimmung des Abstandes zwischen Sender und Empfänger über den Vergleich der Sendeleistung und der empfangenen Signalstärke. Bei der Berechnung des Abstandes muss zusätzlich die Dämpfung des Signals berücksichtigt werden. Diese entsteht durch Hindernisse, die die Ausbreitung elektromagnetischer Wellen beeinflussen, wie z. B. Wände. Die Position lässt sich über eine Trilateration der Abstände ermitteln.

Das Zeit-basierte Verfahren *Time of Arrival* (TOA) bestimmt den Abstand zwischen Sender und Empfänger über die Laufzeit, d. h. die Zeit zwischen dem Senden und Empfangen des Signals. Hier gibt es zwei Möglichkeiten: Beim *Einwegverfahren* müssen die Sender und der Empfänger zeitlich untereinander synchronisiert sein. Die gesendeten Signale enthalten einen Zeitstempel. Aus der Zeitdifferenz zwischen dem Zeitpunkt des Empfangs und dem enthaltenden Zeitstempel lässt sich die Laufzeit ermitteln. Das *Zweiwegverfahren* nutzt die Messung der *Round Trip Time* (RTT) zur relativen Bestimmung des Abstandes. Dies ist die Zeit, die das Signal vom Sender zum Empfänger und zurück benötigt. Das TOA-Verfahren kommt beispielsweise bei GPS zum Einsatz.

Zur Triangulation werden grundsätzlich mindestens zwei, zur Trilateration mindestens drei Messwerte benötigt. Da die Messung elektromagnetischer Signale jedoch immer Wechselwirkungen und Störungen unterworfen ist, verbessert die Einbeziehung von zusätzlichen Messpunkten das Ergebnis und verhindert Fehlmessungen.

2.1.2 WLAN

WLAN[1]-Netze sind zellular ausgebaute Strukturen mit *Access Points*, ähnlich der Struktur von Mobilfunknetzen. Die Größe der Zellen mit dem ausgeleuchteten Bereich ist dabei von der Sendeleistung der Access Points anhängig.

Die zur Zeit üblichen WLAN-Standards IEEE 802.11b/g bzw. 802.11n[2] nutzen das 2,4-2,485 GHz Frequenzband (Mikrowellenbereich). Problematisch ist hier, dass das Signal durch stationäre Objekte (Wände, Körper) und andere Geräte im gleichen Frequenzband stark beeinflusst wird.

Einfache Verfahren zur Positionsbestimmung sind auch hier *Cell of Origin* (COO), wo Genauigkeiten von 25 m erreicht werden (vgl. [DZ02]). Bessere Ergebnisse werden durch die Messung der Signalstärke (RSSI) und Trilateration mehrerer Signale erzielt. Einzelne Ansätze nutzen auch die Messung der Signallaufzeiten (TOA) (z. B. [GH02]), teilweise unter Verwendung angepasster Hardware, da herkömmliche WLAN-Komponenten die Messung der Signallaufzeit nicht direkt unterstützen. Die Messung der Signalstärke ist dagegen beim Empfang von Paketen im Protokoll implementiert, die meisten Ansätze bedienen sich dieser Technik (z. B. *MagicMap* der HU Berlin [UrlMagicMap]). Genauere Ergebnisse lassen sich durch das vorherige Ausmessen der Signalstärke und der Anfertigung einer Karte mit den Signalcharakteristiken (*Radio Map*) erreichen. Durch diese Technik erreicht beispielsweise das *iPOS*-System[3] eine Genauigkeit von 3 m in einer Büroumgebung (vgl. [RMVH06]). Andere Ansätze erreichen durch die Kombination mehrerer Techniken eine genauere Lokalisierung, z. B. durch die Kombination mit einem digitalen Kompass Genauigkeiten von 1,65 m (vgl. [KKH06]).

2.1.3 Bluetooth

Bluetooth ist ein Standard[4] für den Kurzstreckenfunk in dem Frequenzbereich 2,4 GHz. Ziel des Entwicklerkonsortiums war eine kostengünstige und vor allem energiesparende Funktechnik. Die Reichweite ist daher durch die Sendeleistung

1 Wireless Local Area Network
2 Der Standard IEEE 802.11n ist bisher nur als Entwurf verabschiedet, eine endgültige Veröffentlichung ist für November 2009 geplant (vgl. [UrlIEEE]). Auf dem Markt sind jedoch schon zum jetzigen Zeitpunkt Endgeräte verfügbar, die auf dem Entwurf basieren.
3 iPAQ Positioning System, IMST GmbH
4 IEEE 802.15.1

stark beschränkt (Klasse 1: 100 mW/20 dBm, Klasse 2: 2,5 mW/4 dBm, Klasse 3: 1 mW/0 dBm). Übliche Bluetooth-Module gehören der Klasse 3 an, wodurch eine Reichweite von maximal 10 m erreicht wird.

Meist wird Bluetooth nur für eine einfache COO-Ortung genutzt (vgl. [HNS02] und [ABC03]). Die Messung der Signalstärke erlaubt genauere Positionierungen, Versuche (z. B. [WBR05]) erzielen hier eine Genauigkeit von 2 m (bei einer Standardabweichung von 1,2 m), setzen jedoch eine ausgemessene Radio Map voraus.

2.1.4 RFID

Radio Frequency Identification-Systeme (RFID) sind in den Bereich der kontaktlosen ID-Systeme einzuordnen. Sie ermöglichen das drahtlose Auslesen von Informationen die auf einem Datenträger gespeicherten sind über ein magnetisches oder elektromagnetisches Feld (vgl. [Fin06, S. 7f]). Ein RFID-System setzt sich dabei aus einem Lesegerät (RFID-*Reader*) und aktiven oder passiven *Transpondern* (den sogenannten RFID-*Tags*) zusammen.

Zum Auslesen eines Transponders erzeugt der RFID-Reader ein (elektro-) magnetisches Wechselfeld. Dieses wird vom Transponder beeinflusst und vom Reader wieder aufgenommen. Der Transponder sendet kein eigenes Signal. Bei RFID-Transpondern wird zwischen passiven und aktiven Transpondern unterschieden. Passive Transponder kommen ohne eigene Stromversorgung aus und beziehen die benötigte Energie vollständig aus dem Wechselfeld des RFID-Readers. Die vom Chip des Transponders benötigte Energie wird durch das elektromagnetische Feld aufgenommen und kurzzeitig zwischengespeichert. Bei aktiven Transpondern wird zum Auslesen ein wesentlich schwächeres Wechselfeld benötigt, da diese eine eigene Stromversorgung zur Datenverarbeitung auf dem Chip besitzen. Diese wird jedoch nicht zum aktiven Senden eines Signals benutzt, weshalb diese Transponder auch als *semi-passiv* bezeichnet werden (vgl. [Fin06, S. 23f]).

Für RFID-Systeme stehen verschiedene Frequenzbänder zur Verfügung, der Hochfrequenzbereich (13,56 MHz) und der Langwellenbereich (< 135 kHz) sind dabei am weitesten verbreitet. Daneben sind auch Systeme im Mikrowellenbereich (2,45 GHz) und UHF[1]-Bereich (868/91 MHz) verfügbar (vgl. [Fin06, S. 171]). Die Reichweite ist vom Frequenzbereich abhängig, mit passiven Transpondern werden Reichweiten von 3 m, mit aktiven von bis zu 100 m erreicht.

1 Ultra High Frequency

OpenBeacon

Beim System *OpenBeacon* des Herstellers Bitmanufaktur [UrlOpenBeac] bzw. [UrlBit] handelt es sich um ein RFID-System, das mit aktiven Transpondern auf dem 2,4 GHz-Band arbeitet. Die Reichweite beträgt bei einem ungedämpften Signal bis zu 80 m. In dem benutzten Frequenzbereich muss jedoch von starken Dämpfungen und Reflexionen durch Menschen, Wände und andere Objekte im Signalbereich ausgegangen werden (vgl. [FHM08a]).

2.2 Recommender Systeme

Recommender Systeme sind heutzutage in vielen Bereichen anzutreffen, die weiteste Verbreitung finden sie jedoch im Bereich des *E-Commerce*. Recommender Systeme sind Informationssysteme, die aus einer vorhandenen Datenbasis für einen Nutzer individuelle oder gruppenbezogene Empfehlungen gemäß den Präferenzen des Nutzers generieren oder aus einer Menge an möglichen Optionen die für den Nutzer am Interessanten herausfiltern.

Recommender Systeme lassen sich dabei in verschiedene Gruppen klassifizieren, die verschiedene Techniken zur Generierung der Empfehlungen einsetzen. Die Klassifizierung beruht dabei nach [Bur02] auf drei Bestandteilen, anhand denen sich Systeme einordnen lassen:

1. *Datenbasis*, die das System vor Beginn des Recommendation Prozesses besitzt.
2. *Eingabedaten*, die die Nutzer bereitstellen müssen, um eine Empfehlung zu generieren.
3. *Algorithmus*, der die Eingabedaten mit der Datenbasis verknüpft und eine Empfehlung ermittelt.

2.2.1 Regelbasierte Personalisierung

Regelbasierte Systeme lassen sich nicht direkt in den Bereich der Recommender System einordnen, auch wenn sie dem Nutzer eine automatisch angepasste, personalisierte Auswahl von Informationen zur Verfügung stellen. Ihnen liegt ein starres Regelwerk zugrunde (*„wenn – dann“*), das die Inhalte den Vorgaben entsprechend für die Nutzer anpasst. Die zugrunde liegende Technik und Implementierung ist

entsprechend einfach, erfordert jedoch einen hohen Wartungsaufwand. Die Empfehlungen sind sehr generalisiert und lassen sich nicht effizient auf einzelne Nutzer personalisieren, wie dies andere Recommender-Techniken ermöglichen.

2.2.2 Collaborative Filtering

Kollaborative Filtertechniken sind heutzutage die am weitesten verbreitete Grundlage für Recommender Systeme. Die Techniken wurden erstmals 1992 von Goldberg et al. im System *Tapestry* verwendet [GNOT92]. Bei Tapestry handelte es sich um ein bei *Xerox PARC* entwickeltes Mail-System. Das System ermöglicht zusätzlich zu Dokumenten auch Metadaten (Autor, Leser des Dokuments etc.) und Bewertungen der Dokumente zu speichern. Bei der Suche nach Dokumenten wird neben den Metadaten auch die Bewertung der Dokumente in die Ermittlung des Ergebnisses einbezogen.

Bei kollaborativen Systemen handelt es sich um lernende Systeme. Diese nutzen eine vorhandene Datenbasis, um Verhaltensmuster von Nutzereingaben zu erkennen und versuchen, eine Korrelation zwischen eingegebenen und vorhandenen Daten herzustellen (*User-to-User*). Die Empfehlungen werden auf Basis von Bewertungen anderer Nutzer generiert. Dabei wird versucht, das Interessenprofil eines Nutzers auf andere zu übertragen. Die Bewertungen bzw. das Interessenprofil können dabei entweder durch eine explizite Bewertung der Objekte durch die Nutzer erfolgen (z. B. Rating auf einer Skala, nützlich ja/nein) oder durch eine implizite Bewertung anhand des Verhaltens der Nutzer (z. B. welche Objekte wurden betrachtet, wie lange wurden Objekte betrachtet). Die Technik basiert auf dem Grundsatz, dass Informationen bzw. Objekte interessant/relevant für Nutzer sind, wenn diese dies auch für die statistischen Nachbarn waren. Derartige Ähnlichkeiten lassen sich auf einer Ähnlichkeitsmatrix zwischen Nutzern und ihren statistischen Nachbarn oder zwischen Objekten einer Datenbasis abbilden.

	Objekt 1	**Objekt 2**	**Objekt 3**	**Objekt 4**
Nutzer 1	5	5		4
Nutzer 2	4	4	5	3
Nutzer 3			3	2
Nutzer 4	5	2	2	5

Tabelle 2.1: Beispiel-Datenbasis für kollaboratives Filtern

Die Algorithmen für kollaboratives Filtern lassen sich anhand verschiedener Kriterien einteilen. Breese et al. [BHK98] unterscheidet entsprechende Algorithmen in Speicher-basiert und Modell-basiert. Speicher-basierte Algorithmen beziehen die komplette Datenbasis bei der Empfehlung in die Entscheidung mit ein, während Modell-basierte Algorithmen zuvor berechnete Modelle nutzen, um die Komplexität zu verringern. Da rein Speicher-basierte Verfahren bei entsprechender Datenbasis extrem rechenintensiv sind, werden diese in der Praxis normalerweise nicht eingesetzt, es haben sich entweder rein Modell-basierte oder hybride Formen durchgesetzt. Eine andere Art der Einteilung nehmen Schafer et al. in [SFHS07] mit der Einordnung anhand des zugrunde liegenden mathematischen Modells in *probalistische* bzw. *nicht-probalistische* Algorithmen vor.

Nicht-probalistische Algorithmen

Unter nicht-probalistischen Methoden werden Algorithmen verstanden, die nicht auf probalistischen Modellen basieren. Die meisten dieser Algorithmen basieren auf dem Prinzip des *Nearest Neighbour*, entweder auf Basis von Nutzer- oder Objektdaten.

Der *Pearson-Korrelationskoeffizient* gehört zum Bereich des Benutzer-basierten Filterns. Dieser erlaubt die Beschreibung der Ähnlichkeit der Interessen zweier Nutzer, indem ein Koeffizient für die Ähnlichkeit der Bewertungen des Nutzers u und seines Nachbars n berechnet wird. Der Wertebereich des Koeffizienten kann sich dabei zwischen -1 (inverse Korrelation) und 1 (perfekte Korrelation) bewegen; 0 gibt an, dass keine Korrelation besteht.

$$sim(u,n) = \frac{\sum_i (r_{ui} - r_u)(r_{ni} - \overline{r}_n)^2}{\sqrt{\sum_i (r_{ui} - \overline{r}_u)^2} \sqrt{\sum_i (r_{ni} - \overline{r}_n)^2}} \tag{2.1}$$

Der gleiche Algorithmus lässt sich auch zur Bestimmung der Ähnlichkeit zweier Objekte i und j benutzen. Die Ähnlichkeit wird aufgrund der Bewertungen der Objekte durch die Nutzer u errechnet.

$$sim(i,j) = \frac{\sum_u (r_{ui} - \overline{r}_u)(r_{uj} - \overline{r}_u)^2}{\sqrt{\sum_u (r_{ui} - \overline{r}_u)^2} \sqrt{\sum_u (r_{uj} - \overline{r}_u)^2}} \tag{2.2}$$

Zur Einordnung eines Nutzers in Nutzergruppen die automatisch aus den vorhandenen Daten gebildet wurden kann die *Clusteranalyse* genutzt werden. Dazu werden Nutzer in Cluster (Nutzergruppen) eingeteilt. Anfangs besteht jede Gruppe aus genau einem Element (Nutzer), dann werden iterativ jeweils die zwei ähnlichsten Cluster miteinander verschmolzen bis die vorgegebene Clustergröße erreicht ist. Zur Nearest-Neighbour-Bestimmung des Nutzers mit der vorhandenen Datenbasis wird dieser dem Cluster zugeordnet, dem er am ähnlichsten ist. Das Verfahren kann zwar durch Variationen (*k-means-Clusteranalyse, hierarchische Clusteranalyse*) verbessert werden, als Grundproblem bleibt jedoch bestehen, dass zwischen einem Nutzer und den Clustern keine eindeutige Distanzfunktion hergestellt werden kann.

Andere Ansätze sind *graphenbasierte Algorithmen*, oder *neuronale Netze*.

Probalistische Algorithmen

Über probalistische Algorithmen wird nicht versucht, den Nearest Neighbour bzw. die Ähnlichkeit von Nutzern bzw. Objekten zu bestimmen. Statt dessen wird eine Wahrscheinlichkeit für das Eintreten eines bestimmten Falles berechnet (z. B. wer Objekt a gut bewertet, bewertet mit einer Wahrscheinlichkeit von $E = P(x)$ auch Objekt b gut).

Die meisten dieser Algorithmen berechnen die Wahrscheinlichkeit, dass ein Nutzer u einem Objekt i mit einer Wahrscheinlichkeit von E eine bestimmte Bewertung gibt (nach [SFHS07]):

$$E(r|u,i) = \sum_{r} r \cdot p(r|u,i) \tag{2.3}$$

Ein in diesem Bereich verbreiteter Algorithmus benutzt *Bayessche Netze* zur Abbildung der Wahrscheinlichkeiten. Ein Bayessches Netz besteht aus einem gerichteten zyklischen Graph, der in den Knoten Zufallsvariablen (bedingte Wahrscheinlichkeiten) und in den Kanten die bedingten Abhängigkeiten zwischen den Zufallsvariablen darstellt. Dadurch entsteht eine Wahrscheinlichkeitsverteilung aller Variablen, aus der sich Schlussfolgerungen ableiten lassen.

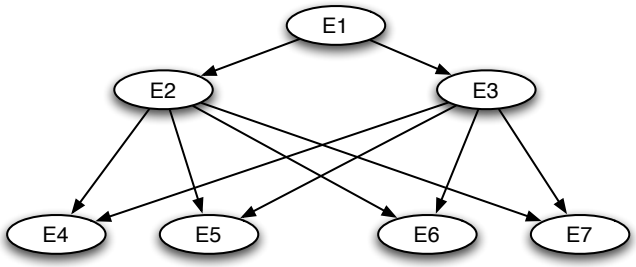


Abbildung 2.2: Bayessches Netz (vereinfacht)

Bayessche Netze verbessern die Objekt-basierten Algorithmen, indem sie die Beziehungen zwischen Objekten nutzen, um Wahrscheinlichkeiten vorherzusagen. Dies gilt jedoch nur für binäre (ja/nein) Bewertungen, für Skalen-basierte Bewertungssysteme konnten bisher nach [SFHS07] noch keine Vorteile gegenüber Nearest Neighbour-Verfahren festgestellt werden.

Probleme

Das Hauptproblem beim kollaborativen Filtern ist das Fehlen einer entsprechenden Datenbasis. Bei neuen Objekten oder Nutzern tritt das sogenannte *Cold-Start*-Problem auf, wenn ein neuer Nutzer oder ein neues Objekt in das System eintritt und noch keine oder zu wenige Bewertungen erhalten hat, bzw. noch kein Nutzerprofil vorliegt. In diesem Fall gibt es keine ausreichende Datenbasis, die für den Filterungsvorgang notwendig ist. Besonders problematisch hierbei ist, dass die meisten Systeme auf Basis von kollaborativen Filtern Objekte ohne Bewertungen nicht berücksichtigen. Ein ähnliches Problem liegt vor, wenn bezogen auf die Zahl der Nutzer (und damit vorliegenden Bewertungen) zu viele Objekte im System vorhanden sind und das Verhältnis zwischen Menge der Nutzer und Zahl Bewertungen gestört ist. Hier kann das *Sparse-Matrix*-Problem entstehen, das besagt, dass es zu wenige Nutzer mit einem gleichen Profil gibt.

2.2.3 Content-based Filtering

Beim inhaltbasierten Filtern beruhen die generierten Empfehlungen auf der Ähnlichkeit von Objekten und Nutzer-Profilen. Die Objekte werden dabei entweder direkt durch ihren Inhalt oder durch ihre Eigenschaften, Beschreibungs-Attribute

und Metadaten beschrieben. Für die Empfehlung wird versucht eine Korrelation zwischen Objekten (*Item-to-Item*) herzustellen. Das Nutzer-Profil wird durch eine Beschreibung von Interessen über Objekt-Attribute erstellt. Dies kann entweder durch explizite Angaben (manuelle Eingabe der Präferenzen) oder implizite Informationen (Verfolgung/Analyse der Tätigkeiten des Nutzers) erfolgen. Für die Empfehlung wird für jedes vorhandene Objekt eine Korrelation zwischen Objekt-Eigenschaften und Nutzer-Präferenzen erstellt, die Nearest Neighbours werden dann dem Nutzer empfohlen.

ID	Titel	Genre	Land	Altersfreigabe
1	Film 1	Komödie	USA	6
2	Film 2	Horror	Frankreich	0
3	Film 3	Action	USA	12
4	Film 4	Komödie	Italien	16

Tabelle 2.2: Beispiel-Datenbasis für inhaltbasiertes Filtern

Die Objekte können entweder strukturiert oder unstrukturiert vorliegen. Bei strukturierten Objekten sind die Beschreibungs-Attribute vorgegeben und klassifiziert. Dies kann beispielsweise ein Buch sein, das durch den Titel, den Autor, die Seitenzahl, Erscheinungsjahr etc. beschrieben wird. Unstrukturierte Objekte besitzen keine manuell generierten Meta-Informationen bzw. Objekt-Eigenschaften, hier wird eine Beschreibung automatisch generiert. Ein Beispiel für die unstrukturierte Beschreibung sind Text-Dokumente, die anhand ihres Inhalts automatisch klassifiziert werden. Der Ursprung dieser Methode liegt im *Information Retrieval*, dabei wird der Inhalt anhand der einzelnen Terme analysiert und anhand der Häufigkeiten des Auftretens der Terme beschrieben. Problematisch bei dieser Art der Beschreibung ist, dass Synonyme, mehrdeutige Terme und grammatikalische Formen des gleichen Wortes nicht berücksichtigt werden. Auch findet der Kontext in dem die Terme auftreten keine Berücksichtigung.

Eine Möglichkeit der Beschreibung von Objekten ist die *Vektorraum-Repräsentation*. Hier werden die Objekte über einen Vektor in einem n-dimensionalen Vektorraum beschrieben, wobei jede Dimension eine Eigenschaft des Objektes darstellt. Der Nearest Neighbour einer Anfrage wird über die Distanz des Anfragevektors zu den Vektoren der beschriebenen Objekte ermittelt.

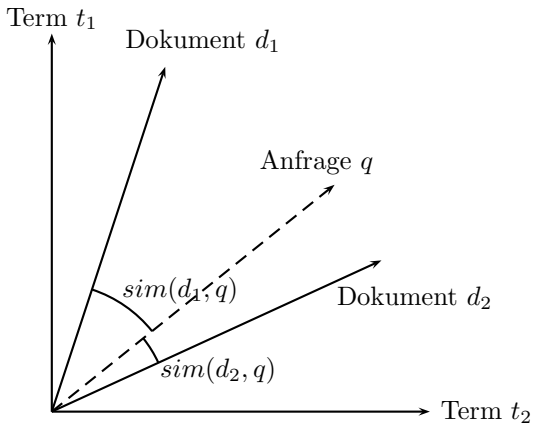


Abbildung 2.3: Vektorraum Repräsentation

Die Vektorraum-Repräsentation eignet sich, um Dokumente anhand ihrer enthaltenen Wörter und der Term-Häufigkeiten zu beschreiben. Je höher die Frequenz des Auftretens eines Terms im Dokument ist, desto höher wird dies gewichtet. Für jeden Term wird ein Wert $w(t, d)$ ermittelt, der diesen für die Relevanz des Terms t_n im Dokument d beschreibt. Dabei steht der Wert in Bezug zu der Gesamthäufigkeit des Terms bei allen vorhandenen N-Dokumenten (nach [PB07]).

$$w(t,d) = \frac{tf_{t,d} \log\left(\frac{N}{df_t}\right)}{\sqrt{\sum_i (tf_{t_i,d})^2 \log\left(\frac{N}{df_{t_i}}\right)^2}} \tag{2.4}$$

Eine anderes Maß für die Gewichtung eines Terms in Dokumenten bezogen auf eine Menge von Dokumenten ist das *tfidf-Maß*[1]. Das Maß setzt sich aus der Term-Häufigkeit und der inversen Dokument-Häufigkeit zusammen.

Die Term-Häufigkeit tf wird aus der Anzahl der Vorkommen n des Terms t_j im Dokument d_j bestimmt und – auf die Länge des Dokuments bezogen – auf das Vorkommen in allen Dokumenten normalisiert. Die Dokumenten-Häufigkeit idf gibt die Relevanz des Terms in allen Dokumenten an. Je geringer das Auftreten des

1 Term Frequency – Inverse Document Frequency

Terms in allen Dokumenten (Anzahl N) ist, desto höher ist dessen Relevanz (vgl. [BS88]).

$$tfidf_{i,j} = tf_{i,j} \cdot idf_i = \frac{n_{i,j}}{\sum_k n_{k,j}} \cdot \frac{\log N}{n_i} \tag{2.5}$$

Problematisch bei diesen Algorithmen ist, dass andere Faktoren nicht abgedeckt werden und nicht in die Bewertung einfließen. Dies sind beispielsweise Position des Terms im Dokument (Gewichtung von Überschriften) oder Termabstand.

Um bei der oben angesprochenen Vektorraum-Repräsentation den Nearest Neighbour zwischen Dokument-Vektoren und Anfrage-Vektor zu finden, eignen sich Distanzfunktionen wie die *Cosine-Similarity-Funktion*. Diese berechnet anhand des Winkels zwischen den Vektoren i, j ein Maß für die Ähnlichkeit.

$$sim(i,j) = \cos(\theta) = \frac{\overrightarrow{i} \cdot \overrightarrow{j}}{\| \overrightarrow{i} \| \| \overrightarrow{j} \|} \tag{2.6}$$

Eine andere Möglichkeit Nutzerprofile mit den vorhandenen Objektdaten zu vergleichen sind *Entscheidungsbäume*. Diese stellen eine einfache Form des maschinellen Lernens dar, bei der die Objekte anhand ihrer Eigenschaften iterativ in Gruppen eingeteilt werden. Dies wird solange wiederholt, bis jede Gruppe nur noch aus einem Objekt besteht. Bei Texten sind die Eigenschaften beispielsweise das Vorkommen eines bestimmten Terms. Mögliche Methoden hierzu sind hier der *ID3*[1]- und der *CART*[2]-Algorithmus [BFAS83]. Beim CART-Algorithmus wird ein Entscheidungsbaum anhand der Entropie der Objekt-Attribute aufgebaut.

2.2.4 Wissensbasierte Recommender Systeme

Wissensbasierte (*Knowledge-based*) Recommender Systeme benutzen funktionales Wissen, um eine Verknüpfung zwischen den Objekten und den Anforderungen der Nutzer herzustellen. Anders als bei inhaltsbasierten Filtern reicht eine einfache Beschreibung der Objekte nicht aus. Es ist funktionales Wissen darüber hinaus

1 Iterative Dichotomiser 3
2 Classification and Regression Trees

notwendig, wie bestimmte Objekte die Anforderungen der Nutzer erfüllen und daraus eine Empfehlung abgeleitet werden kann. Dazu sind Verknüpfungen zwischen den Beschreibungs-Attributen und den Anforderungen notwendig, um diese Zusammenhänge abbilden zu können.

2.2.5 Demographische Recommender Systeme

Um Empfehlungen anhand der demographischen Daten der Nutzer generieren zu können, werden die Nutzer in Gruppen eingeteilt, auf denen die Empfehlung basiert. Grundlage für die Einteilung sind die demographischen Eigenschaften der Nutzer Von den Methoden ist diese Technik dem kollaborativen Filtern sehr ähnlich, nutzt jedoch eine andere Datenbasis. Bei beiden Techniken werden jedoch Korrelationen zwischen Nutzern bzw. Nutzergruppen hergestellt.

2.2.6 Hybride Recommender Systeme

Durch die Kombination verschiedener Recommender Techniken entstehen hybride Recommender Systeme. Dies kann verschiedene Motivationen haben. Zunächst lassen sich die Resultate eines Systems durch die Einbeziehung verschiedener Daten (z. B. Kombination von inhaltsbasierten und kollaborativen Techniken) meist verbessern. Andere Motivationen können das Begegnen von Grundproblemen einzelner Techniken (z. B. Cold-Start-Problem) oder der Steigerung der Performance sein, indem beispielsweise die Datenbasis durch Recommender-Techniken verkleinert bzw. spezialisiert wird.

Die Kombination der Techniken kann auf unterschiedlichen Arten erfolgen, die im nachfolgenden beschrieben werden. Eine Klassifikation der Kombinationsmöglichkeiten wurde durch Burke [Bur02] vorgenommen, an der sich auch die hier verwendeten Bezeichnungen orientieren. Bei einigen Techniken entstehen Redundanzen oder die Techniken schließen sich gegenseitig aus. Tabelle 2.3 gibt einen Überblick über die Kombinationsmöglichkeiten.

Die einfachste Form von hybriden Systemen stellen Verfahren dar, bei der einzelne Recommender Techniken unabhängig voneinander ausgeführt werden, die Resultate jedoch in einem gemeinsamen Ergebnis präsentiert werden. Das Endergebnis wird dabei durch eine fest vorgegebene Gewichtung der Ergebnisse der einzelnen Techniken ermittelt (*Weighted*). Wenn die ermittelten Ergebnisse sich zwar nicht direkt

	Weighted	Mixed	Switching	FC	Cascade	FA	Meta Level
CF/CN	✓*	✓*	✓*	✓*	✓*	✓*	✓
CF/DM	✓*	✓	✓	✓	✓	✓	X
CF/KB	✓*	✓	✓*	X	✓	✓	✓
CN/CF	R	R	R	R	✓	✓	✓*
CN/DM	✓*	✓	✓	✓*	✓	✓	X
CN/KB	✓	✓	✓	—	✓	✓	✓
DM/CF	R	R	R	R	✓	✓	—
DM/CN	R	R	R	R	✓	✓	✓
DM/KB	✓	✓	✓	—	✓	✓	✓
KB/CF	R	R	R	R	✓*	✓*	✓
KB/CN	R	R	R	R	✓	✓	✓
KB/DM	R	R	R	R	✓	✓	—

Tabelle 2.3: Hybride Recommender Systeme, mögliche Kombinationen nach [Bur02] (* = Implementierung existiert, R = redundant, CN = Content-based, CF = Collaborative Filtering, DM = Demographic, KB = Knowledge-based)

miteinander gewichten lassen, also beispielsweise verschiedene Ergebnis-Listen erzeugt wurden, das Endresultat aber dennoch in einer Kombination der Ergebnislisten dargestellt wird, wird von *Mixed*-Verfahren gesprochen.

Beim *Switching* stehen für die Generierung einer Empfehlung verschiedene Recommender Komponenten zur Verfügung. Das System wählt automatisch die Technik aus, die das sinnvollste Ergebnis liefert. Beispielweise kann im ersten Schritt ein einfacher Algorithmus zum Einsatz kommen. Falls das Ergebnis allerdings zu unbestimmt ist (z. B. Nearest Neighbour kann nicht bestimmt werden), wählt das System eine andere Komponente mit anderer Recommender Technik.

Eine andere Möglichkeit ist die Kombination von verschiedenen Recommender Techniken. Dies kann durch die direkte Kombination mehrerer Methoden zu einem gemeinsamen Recommender Algorithmus erfolgen (*Feature Combination*). Bei der *Feature Augmentation* wird das endgültige Resultat in mehreren Schritten generiert. Die Ausgangsdaten für das Resultat werden dabei in einem ersten Schritt über eine Recommender Technik aus der eigentlichen Datenbasis ermittelt und aus diesem Ergebnis die finale Empfehlung generiert. Bei *Cascade*-Verfahren wird das Ergebnis

ähnlich der Feature Augmentation in mehreren Schritten ermittelt. Die einzelnen Schritte sind hierbei jedoch priorisiert und kommen nicht zwangsläufig zum Einsatz, beispielsweise nur bei uneindeutigen Resultaten. Ähnlich der Funktion der Feature Combination sind auch *Meta-level*-Systeme. Diese nutzen für die Eingangsdaten ein Modell, das von einer anderen Recommender Methode erstellt wurde. Der Unterschied ist, dass die eigentliche Recommender Methode nur das erstellte Modell benutzt, nicht die Rohdaten aus der Datenbasis.

2.2.7 Existierende Lösungen

Im Bereich von Recommender Systemen, die auf Lokalisierungsdaten aufsetzen gibt es im Outdoor-Bereich einige Lösungen, die meist Satelliten-gestützte Positionierungsverfahren (GPS) nutzen.

COMPASS *COMPASS*[1] [SPK04] ist ein mobiles Touristen-Informationssystem, das auf Grundlage der aktuellen Position Empfehlungen gemäß den Interessen der Nutzer generiert. Dabei bedient sich das System nicht einem speziellen Lokalisierungsverfahren, sondern ist darauf ausgelegt, verschiedene zur Verfügung stehende Methoden (GSM, UMTS, GPS) zu nutzen. Der Nutzer wählt zunächst die für ihn interessanten Kategorien aus, die ausgewählten und in der Nähe zur Verfügung stehenden *Points of Interest* (POI) werden dann auf einer Karte angezeigt. Über das direkte Auswählen dieser Punkte lassen sich nähere Informationen erfragen bzw. mobile Dienste (z. B. Reservierungen) direkt nutzen. Der Recommendation Engine ermittelt, wie interessant die einzelnen POI für den Nutzer sind. Für die einzelnen Arten der POI (z. B. Restaurants oder Sehenswürdigkeiten) werden dabei verschiedene hybride Recommender Techniken angewandt, die die Ähnlichkeit von Objekten, aber auch das Verhalten des Nutzers in der Vergangenheit berücksichtigen.

CityVoyager Das System *CityVoyager* [TS06] zeichnet die GPS-Positionsdaten mit einem PDA auf und nutzt diese Daten zur Empfehlung von für den Nutzer potentiell interessanten Geschäften in der Umgebung. Die Datenbasis wird aus dem Verlauf der Positionen gewonnen, dazu wird in regelmäßigen Abständen die Position mit einer Genauigkeit von 10 m ermittelt. Darüber wird ermittelt, ob sich der Nutzer in einem Geschäft aufgehalten hat bzw. wie lange die Dauer des Besuches war. Aus diesem

1 Context-Aware Recommendations in the Mobile Tourist Application

Verlauf werden die Interessen des Nutzers abgeschätzt und auf dieser Basis eine Empfehlung generiert. Für die Generierung der Empfehlung werden kollaborative Filtertechniken verwendet, die die Ähnlichkeit von Geschäften mit denen ermitteln, die der Nutzer bereits besucht hat. Als weiterer Faktor fließt der aktuelle Standort und die Entfernung der Geschäfte in die Berechnung ein.

Amazon Das bekannteste Recommender System aus dem E-Commerce Bereich dürfte der Online-Shop von *Amazon* [UrlAmazon] darstellen. In diesem System werden an verschiedenen Stellen Techniken aus dem Recommender Bereich genutzt. Als Ziel formuliert Amazon, für jeden Nutzer einen individuellen Shop anbieten zu wollen. Zum einen können Objekte kommentiert und mit einer 5-skaligen Bewertung versehen werden. Diese Kommentare können nun wiederum von anderen Nutzern als *hilfreich* bzw. *nicht hilfreich* bewertet werden. Zum anderen werden auch direkt andere Objekte angezeigt, nach dem Muster „Nutzer, die Objekt A gekauft haben, haben auch Objekt B gekauft“ bzw. „was kaufen Nutzer, nachdem sie dieses Objekt angesehen haben“. Des weiteren können andere Nutzer Objekte vorschlagen, die zum betrachteten Objekt passen und entsprechend angezeigt werden. Ebenfalls angezeigt werden Objekte, die vom Nutzer schon zuvor angesehen bzw. gekauft wurden. Auch wenn es sich um ein proprietäres System handelt und die eingesetzten Techniken nicht offenliegen wird deutlich, dass hier hauptsächlich Objekt-basierte, kollaborative Methoden eingesetzt werden, um Korrelationen zwischen Objekten zu ermitteln und diese dann entsprechend zu präsentieren. Die Empfehlungen basieren hauptsächlich auf dem Verhalten anderer Nutzer bzw. dem eigenen Verhalten, der Inhalt fließt auf Basis einer semantischen Einordnung (Kategorien, Tags) ein, so dass es sich um ein hybrides System handelt.

Pandora Über das Internet-Radio *Pandora* [UrlPandora] werden anhand der Auswahl eines Musikstückes bzw. Künstlers ähnliche Werke empfohlen. Der Dienst basiert auf dem *Music Genome Project*, auf dessen Basis Musikstücke in eine Taxonomie eingeordnet werden. Dies wird über 400 mögliche Beschreibungsattribute (z. B. Rhythmus, Melodie, Instrumente) realisiert, die eine 5-skalige Bewertung besitzen. Die Anzahl der verwendeten Beschreibungsattribute liegt dabei, je nach Genre, zwischen 150 und 400. Die Einordnung und Beschreibung erfolgt nicht automatisch, sondern komplett manuell. Die Nutzer können ihr Profil durch die explizite Bewertung einzelner Musikstücke (*gefallen/nicht gefallen/keine Meinung*) beeinflussen. Für die Empfehlungen werden inhaltsbasierte Techniken angewandt. Die interne

Beschreibung der Musikstücke erfolgt über die Vektorraum-Repräsentation, ähnliche Stücke werden über einen Distanzvektor gesucht.

2.3 Datenschutz

Das Grundproblem bei Recommender Systemen ist die Notwendigkeit einer Datenbasis mit Informationen über die Präferenzen von Nutzern. Das System muss Wissen über Nutzer erlangen, um Empfehlungen für Nutzer aussprechen zu können. Ob die Daten durch eine explizite Eingabe von Präferenzen oder eine implizite Verfolgung der Nutzeraktionen erfolgt, ist nachrangig. Bei einer expliziten Eingabe stellt der Nutzer wissentlich Informationen zur Verfügung und behält so eine bessere Kontrolle über seine Daten bzw. sein Nutzerprofil. Bei impliziten Nutzerprofilen wird die Datenbasis dagegen ohne Wissen und Kontrolle des Nutzers erstellt.

Es besteht ein Interessenkonflikt zwischen dem Konzept der Datensparsamkeit und Datenvermeidung und den Anforderungen eines Recommender Systems, dessen Qualität neben der technischen und algorithmischen Umsetzung in erster Linie von der Datenbasis, also der Güte und Quantität der bekannten Daten, abhängt. Im Besonderen betrifft das kollaborative Techniken, je größer die Datenbasis hier ist (also je mehr Daten von und über Nutzer vorliegen), desto besser können Empfehlungen generiert werden.

Der Interessenkonflikt besitzt verschiedene Aspekte. Zum einen ist dies die technisch-algorithmische Seite, wie System-Architektur und Konzept des Recommender Systems, zum anderen aber auch nicht-technische Aspekte, die das Vertrauen der Nutzer in das System erhöhen können. Bei der Konzeption von personalisierten Systemen beschreibt Cranor [Cra04] vier Aspekte für den Systementwurf, die Auswirkungen auf die Privatsphäre der Nutzer haben.

1. *Art der Datensammlung*: Die implizite Sammlung ist immer ein stärkerer Eingriff in die Privatsphäre der Nutzer als eine explizite Eingabe.
2. *Dauer der Datenspeicherung*: Bei einer persistenten Speicherung von Daten wird ein Profil des Nutzers dauerhaft im System gespeichert und bei jedem Besuch des Nutzers erweitert. Wesentlich geringere Auswirkungen auf die Privatsphäre besitzen Systeme, die Daten nur für den aktuellen Besuch oder zum Lösen einer Aufgabe verwenden und nicht dauerhaft speichern.

3. *Einbeziehung der Nutzer*: Je stärker die Nutzer aktiv in die Entscheidungsprozesse einbezogen werden, desto transparenter ist das System. Dies kann beispielsweise dadurch erfolgen, dass die Personalisierung die Nutzer selbst aktiviert wird und Möglichkeit besteht, Prozesse zu stoppen.
4. *Eingesetzte Methode*: Kollaborative Techniken basieren auf den Daten von Nutzern, die Qualität hängt sehr stark von der Menge der vorhandenen Daten ab. Inhaltsbasierte Methoden benötigen dagegen nicht zwingend Profile der Nutzer und müssen diese speichern müssen.

2.3.1 Lösungsstrategien

Durch eine entsprechende Architektur des Systems kann die Sicherheit und Vertraulichkeit der Daten im System erhöht werden. Eine Gefahr stellt das Auslesen der Datenbasis durch eine Kompromittierung oder Manipulation der Anwendung dar, was zu einer Offenlegung von Nutzer-Profilen führen kann. Durch eine Verteilung der Datenbasis und Trennung der Daten auf mehrere Systeme, die einzeln kompromittiert werden müssten, lässt sich diese Gefahr zwar verringern. Der Bedrohung muss jedoch in erster Linie mit einer entsprechenden möglichst fehlerfreien Umsetzung und Implementierung mit Sicherheitsmechanismen wie beispielsweise einer Verschlüsselung der Daten begegnet werden.

Werden persönliche Profile für die Empfehlung benötigt ist eine Möglichkeit die Profile pseudonymisiert zu speichern und nutzen, ohne dass sich Rückschlüsse auf einzelne Nutzer ziehen lassen bzw. dies erschwert wird (vgl. hierzu [KS03]).

Auf der konzeptionellen Seite kann eine transparente Gestaltung der Systeme und Offenlegung der vorhandenen Daten Vertrauen bei den Nutzern schaffen, indem sie Kontrolle und Überblick über die über sie gespeicherten Daten erhalten. Dies kann z. B. durch das Client-seitige Sammeln von Daten erfolgen, die erst durch eine explizite Bestätigung in das System übernommen werden. Eine weitere Möglichkeit ist die transparente Darlegung von Entscheidungsprozessen, so dass die Nutzer wissen, auf welcher Grundlage eine Entscheidung getroffen wird, sofern dies technisch möglich ist.

3 Anforderungsanalyse

Ziel ist es, ein System zu entwickeln, dass Besucher nach einem Museumsbesuch bei der Nachbetrachtung unterstützt und bei einem erneuten Gang durch das Museum gezielt auf Aspekte gemäß den Interessen des Besuchers verweist. In der Nachbetrachtung eines Museumsbesuches werden weiterführende Informationen zu einzelnen Themen des Besuches angeboten. Die Informationen sollen dabei auf den vorherigen Besuch abgestimmt sein. Das System soll den Besuchern bei der Auswahl der Informationen assistieren und gezielt die Informationen über die Themen und Bereiche zur Verfügung stellen, an denen der Besucher während seines Ganges durch das Museum Interesse gezeigt hat.

Während dem Besuch im Museum wird der Besucher von dem mobilen, multimedialen Museumsguide (MMG) *EMIKA* unterstützt. Über dieses System werden automatisch Kontext-bezogene Informationen angezeigt. Diese Informationen beziehen sich auf die Abteilung (Segment), in der sich die Person aktuell befindet oder direkt auf einzelne Objekte. Über das Museums-Informationssystem lassen sich nähere Informationen in Form von multimedialen Inhalten (Audio, Video, Multimedia-Objekte) abrufen, die über eine Lokalisierung mittels der RFID-Technologie gezielt auf den Kontext bezogen sind. Sowohl die abgerufenen Informationen, als auch die durchlaufenen Segmente werden von dem Museums-Informationssystem protokolliert. Eine detaillierte Beschreibung des EMIKA-Systems findet sich in Abschnitt 4.1 (S. 33).

Das zu entwickelnde System soll als Informationssystem eine eigene Datenbasis in Form von Texten und multimedialen Inhalten (Bildern, Audio, Video) besitzen, die anhand von beschreibenden Metainformationen inhaltlich eingeordnet werden. Diese Datenbasis wird im folgenden als *Informationspool* bezeichnet. Die Inhalte lassen sich von Benutzern abrufen, bei der Auswahl der Informationen sollen die Besucher gemäß ihren Interessen gezielt vom System unterstützt werden. Da die Datenmenge der Informationsbasis keinen Beschränkungen unterliegen soll, muss berücksichtigt

werden, dass diese durch multimediale Inhalte entsprechend groß werden kann und das System entsprechend skaliert werden muss.

Der Informationspool soll gezielt aufgebaut werden, eine unkontrollierte Übernahme bestehender Artikelsammlungen ist aus konzeptionellen Gründen nicht vorgesehen. Externe Daten können nicht auf die Qualität und die Anforderungen überprüft werden. Sie sind nicht gezielt auf die Zielgruppen zugeschnitten und unterliegen keiner Kontrolle, ob sie den inhaltlichen Anforderungen genügen. Daher wird neben einem System zum Abruf der Informationen auch ein System zur Verwaltung des Informationspools benötigt.

Grundlage für die Unterstützung und automatische Empfehlung von Informationen soll ein Recommender Engine sein, der die protokollierten Museumsbesuche auswertet und zu den Objekten, an denen der Besucher Interesse gezeigt hat, weitergehende Informationen zur Verfügung stellt.

3.1 Szenarien

Aus der Zielsetzung lassen sich konkrete Anwendungsszenarios ableiten. Notwendig ist zum einen der Import von Daten, zum anderen der Abruf, für den verschiedene Szenarios möglich sind.

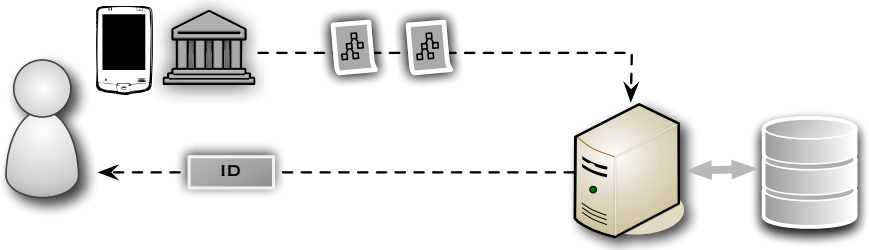


Abbildung 3.1: Anwendungsszenario: Daten-Import

3.1.1 Daten-Import

Voraussetzung für den Abruf von Informationen ist die Übernahme der protokollierten Daten in das System. Dies kann nach Beendigung des Besuches der Ausstellung erfolgen, wenn der MMG an das Museum zurückgegeben wird. Da dieser keine ständige Daten-Verbindung besitzt ist es notwendig eine Verbindung zum System aufzubauen und die Daten in das System zu importieren. Der Nutzer erhält vom System eine Kennung, mit der später die Daten im System identifiziert werden können.

3.1.2 Daten-Abruf

Ein Szenario für den Abruf von vorhandenen Daten aus dem Informationspool ist die Nachbetrachtung des Museumsbesuches über eine Web-Plattform. Eine weitere Möglichkeit soll sein, bei einem erneuten Besuch des Museums gezielt tiefergehende Informationen gemäß den Interessen zu erhalten. Diese können vor dem Besuch auf ein mobiles Gerät geladen werden, um diese während des Besuches angezeigt zu bekommen.

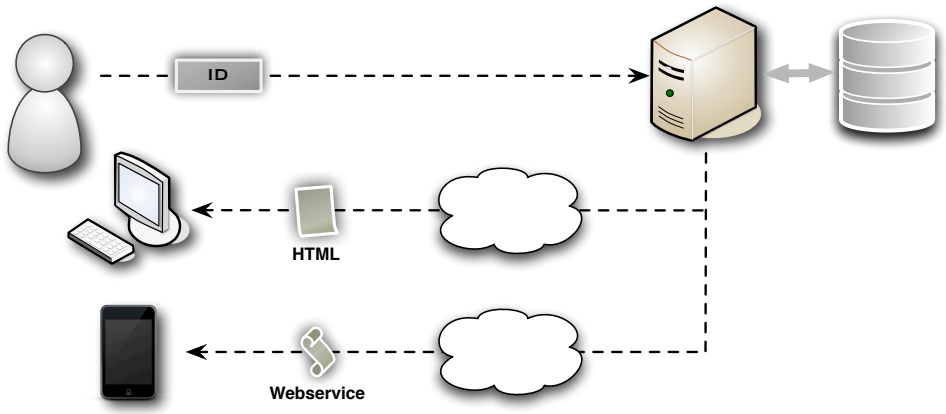


Abbildung 3.2: Anwendungsszenario: Daten-Abruf

3.1.3 Web-Plattform

Eine Web-Plattform bietet die Möglichkeit, auch in einem zeitlichen Abstand des Museumsbesuches Informationen abrufen zu können, ohne dabei örtlich gebunden zu sein. Dies trifft insbesondere auf die touristische Zielgruppe von Museumsbesuchern zu, die sich nur temporär vor Ort aufhalten und während eines Besuches im Museum für sie interessante Themen entdecken und zu diesen in einer Nachbetrachtung weitere Informationen wünschen.

Grundsätzlich wäre die Mitnahme der Informationen auf herkömmlichen Datenträgern (CD-Rom/DVD) denkbar, doch unterliegt dies Beschränkungen und ist daher nicht praktikabel. So soll die Menge der Daten im Informationspool keinen System-bedingten Beschränkungen in Bezug auf Datenmenge und -größe unterliegen. Nur über den direkten Zugriff auf den Informationspool kann eine Aktualität der Daten gewährleistet werden.

Um eine möglichst große Zielgruppe zu erreichen ist es notwendig, den Einstieg für Nutzer auf Seiten der verwendeten Web-Technologien und auf Seiten der Bedienung möglichst niedrig-schwellig und einfach zu halten.

3.1.4 Mobile Anwendung

Um auch während eines Museumsbesuches weiterführende Informationen zu erhalten, lässt sich das Szenario einer mobilen Anwendung entwickeln. Diese Anwendung soll erlauben die Daten auf einem mobilen Endgerät (PDA, Mobiltelefon o.ä.) abrufen zu können und den Nutzer dabei gezielt auf potentiell interessante Themen zu lenken, zu denen weiterführendes Wissen bereitgestellt wird.

Dabei ist denkbar, die Informationen auch direkt zur Planung eines Rundganges zu nutzen, bevor die Ausstellung betreten wird. Hier könnte entweder das Szenario entstehen, dass ein vorheriger Besuch gezielt mit Informationen angereichert und vertieft wird oder zuvor ausgelassene Teile der Ausstellung hervorgehoben werden.

Da die komplette Datenbasis die beschränkten Speicher-Kapazitäten von mobilen Endgeräten übersteigt, ist eine verteilte Anwendung notwendig, um dieses Problem umgehen. Bei der Konzeption muss berücksichtigt werden, dass die Anwendung für den Einsatz in Gebäuden entwickelt wird. Als Technologien für die drahtlose Datenübertragung zwischen Endgerät und Server kommen aufgrund der benötigten Datenrate nur Breitband-Mobilfunk-Technologien (UMTS) und WLAN in Frage, wobei die Verfügbarkeit von UMTS in Gebäuden nicht durchgängig gewährleistet

werden kann. WLAN setzt eine entsprechend aufwändige Infrastruktur im Gebäude voraus. Aus diesen Gründen, kann nicht von einer ständigen Datenverbindung zum Application-Server ausgegangen werden, wodurch nur eine persistente Speicherung der Daten auf dem mobilen Endgerät in Frage kommt.

Unter diesen Voraussetzungen verbleibt nur die vorherige Auswahl und das gezielte Laden von Daten auf das Endgerät vor Beginn des Besuches. Dies bedingt eine verteilte Anwendung. Um der Beschränkung auf eine festgelegte Technologie zu umgehen und zukünftige Entwicklungen zu ermöglichen bieten sich Webservices für den Zugriff auf das System und die Datenbasis an.

3.2 Anwendungsfälle

Aus den Anwendungsszenarios lassen sich zwei Anwendungsfälle ableiten. Dabei soll in der Anforderungsanalyse zwar grundsätzlich auf beide Anwendungsszenarios eingegangen werden, bei der Implementierung wird sich jedoch auf das Szenario einer Web-Anwendung beschränkt. Das System ist jedoch mit geringfügigen Modifikationen erweiterbar und auf eine mobile Anwendung übertragbar. Die dafür notwendigen Voraussetzungen werden bei der Konzeption berücksichtigt.

Der erste Anwendungsfall ist die *Verwaltung der Datenbasis*, welcher die Grundlage für den zweiten Anwendungsfall darstellt. Dieser beinhaltet den *Abruf von Informationen*. Die Anwendungsfälle werden im Folgenden anhand der benötigten Rollen beschrieben.

3.2.1 Verwaltung der Datenbasis

Um überhaupt eine Datenbasis für den Informationspool aufbauen und verwalten zu können ist ein System zur Verwaltung der Inhalte notwendig. Dieses muss alle Operationen beinhalten, die zur Pflege des Informationspools notwendig sind.

Für die Verwaltung der Datenbasis lassen sich zwei Rollen identifizieren. Zum einen ist dies die Rolle des *Administrators*. Dieser besitzt privilegierte Rechte gegenüber anderen Nutzern. Die Rechte beinhalten alle zur Verwaltung der Datenbasis notwendigen Autorisierungen und darüber hinaus die Möglichkeiten der Benutzerverwaltung. Administratoren können andere User anlegen und verwalten und ihnen Rechte zuweisen.

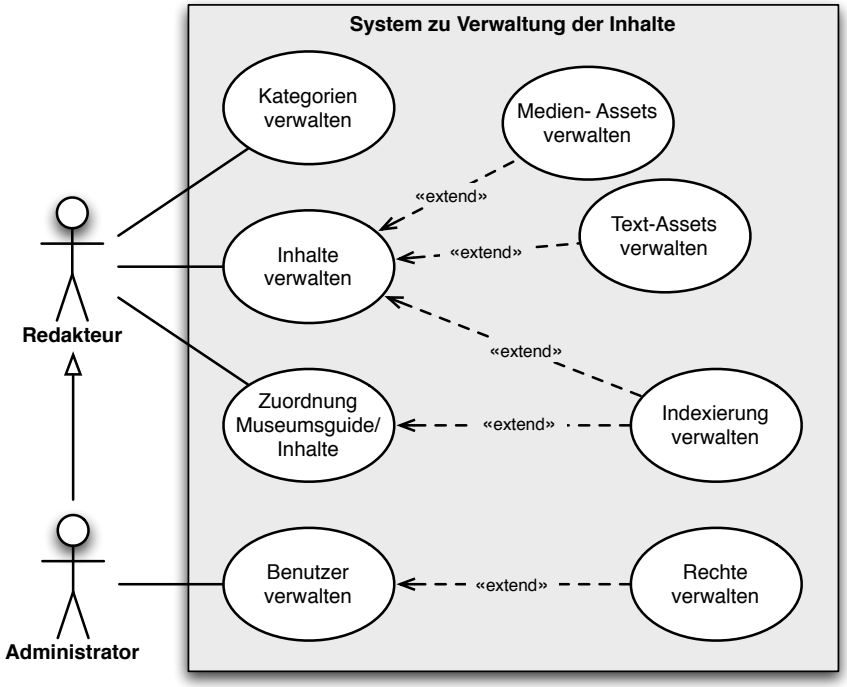


Abbildung 3.3: Anwendungsfall Verwaltung der Datenbasis

Alle anderen Nutzer mit Rechten zur Verwaltung der Datenbasis benötigen die Rolle des *Redakteurs*. Diese Rolle besitzt alle Rechte, die für die Pflege des Datenbestandes notwendig sind. Zur Verwaltung der Datenbasis gehört die Administration von Kategorien, über die Inhalte strukturiert werden. Die Inhaltselemente setzen sich aus einer Sammlung von *Assets* zusammen. Diese Assets werden ebenfalls vom Redakteur verwaltet. Zum einen gibt es Text-Assets für Fließtext-Inhalte. Zum anderen ist es aber auch möglich verschiedene Medien zu integrieren und einem Inhaltselement zuzuweisen. Diese Medien werden über Medien-Assets im System verwaltet, die zunächst keiner Beschränkung auf spezifizierte Medienformen unterliegen. Neben diesen Objektdaten ist die Pflege der Metadaten notwendig. Diese ist Voraussetzung für den Zuordnung von Objekten in der Ausstellung und Inhalten aus der Datenbasis. Dazu ist es notwendig eine Indexierung aufzubauen und zu verwalten, deren Index-Elemente auf die Inhalte zugewiesen werden. Zugleich ist es notwendig, der Struktur des MMG Index-Elemente zuzuordnen.

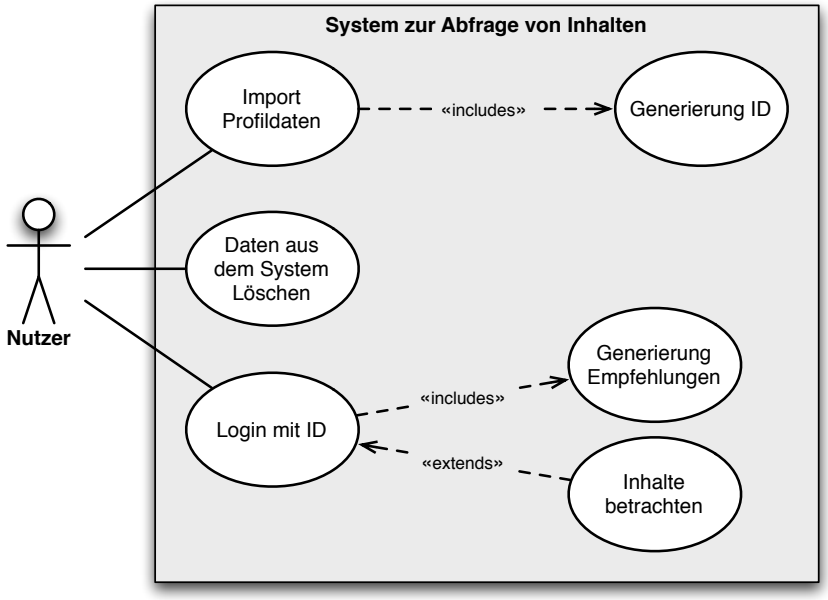


Abbildung 3.4: Anwendungsfall Abruf von Informationen

3.2.2 Abruf von Informationen

Voraussetzung für den Abruf von Informationen sind die Profildaten, die im System vorliegen müssen. Die Profildaten werden während des Museumsbesuchs mit einem mobilen Museums-Informationssystem erzeugt und liegen zunächst auf dem PDA vor.

Der Import der Profildaten in das System sollte automatisch ausgeführt werden, doch kann dies nur durch den MMG realisiert werden, weshalb in diesem System zunächst von einem manuellen Import ausgegangen werden muss. Beim Import der Daten erhält der Nutzer eine vom System generierte ID, die den Zugriff auf das gespeicherte Profil ermöglicht. Der Login in das System und Zugriff auf die Daten erfolgt, ohne dass der Nutzer vorher eine zusätzliche Anmeldungs-Prozedur durchlaufen muss.

Der Zugriff erfolgt dabei weitgehend anonym, weitere Daten über den Nutzer sind für den Login nicht notwendig und werden nicht erhoben. Nach dem Login hat der User die Möglichkeit, Informationen aus der Datenbasis abzurufen. Dabei werden die Informationen gemäß den aus den Profildaten ermittelten Präferenzen aufbereitet

zur Verfügung gestellt. Die Auswahl der Informationen erfolgt dabei implizit über den Recommendation Engine, der die Empfehlungen generiert. Neben Zugriff über die generierten Empfehlungen ist auch ein herkömmlicher Zugriff auf alle Daten des Informationspools möglich. Um die Kontrolle über die Profildaten zu behalten hat der Nutzer auch die Möglichkeit, seine Daten aus dem System komplett zu entfernen.

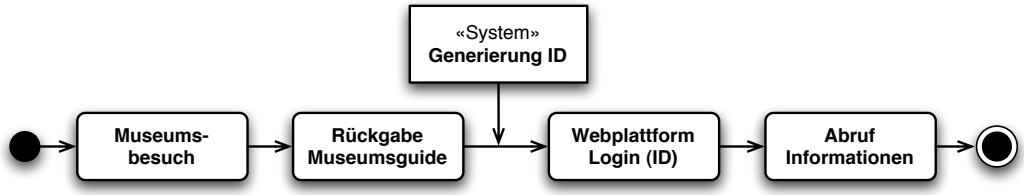


Abbildung 3.5: Abruf von Informationen: Schematischer Ablauf

3.3 Anforderungen

Aus den vorhergehenden Entwürfen der Szenarien und der Anwendungsfälle lassen sich die Anforderungen an das zu entwickelnde System ableiten. Als Grundlage für den Systementwurf muss von einer verteilten Anwendung ausgegangen werden, die eine zentrale Speicherung und Verwaltung der Datenbasis erlaubt. Für den Abruf der Informationen ist neben den für die Szenarien notwendigen Schnittstellen ein Recommender Engine zu entwickeln.

3.3.1 Datenbasis/-haltung

Voraussetzung ist die Nutzung der Protokollierung des Museumsbesuchs, die durch die Nutzung des mobilen, multimedialen Museumsguide des EMIKA-Projektes entsteht. Diese Daten liegen im XML-Format vor, auf die Datenstrukturen wird im Abschnitt 4.1.2 (S. 36) näher eingegangen. Die Datenhaltung muss zum einen die Nutzungsdaten strukturiert vorhalten, zum anderen auch den Informationsbestand verwalten können. Für die Nutzung der auf den mobilen Endgeräten protokollierten Profildaten ist eine Import-Schnittelle notwendig, die die Daten auch ggf. in die erforderliche Struktur transformiert.

Zur Verwaltung des Informationsbestandes wird ein Verwaltungs-System benötigt, mit dem sich verschiedene multimediale Inhalte verwalten lassen und in eine Kategorisierung einpflegen lassen.

3.3.2 Recommender Engine

Durch die Analyse des Protokolls eines Museumsbesuches sollen die für den Besuchenden interessanten Themen herausgefiltert werden und daraus auf potentiell gesuchte Themen geschlossen werden. Dazu ist es notwendig, die im Museum bzw. in den einzelnen Abteilungen verbrachte Zeit und die betrachteten Objekte zu analysieren. Ergebnis der Analyse sollen einzelne Themen sein, zu denen dann aus dem Informationsbestand weitere Objekte ausgewählt und als Empfehlung dem Besucher angeboten werden.

Es muss insbesondere darauf geachtet werden, dass der Datenschutz der Benutzer gewährleistet wird und die Vorgänge transparent für die Benutzer ablaufen, so dass Vertrauen in das System entsteht. Danebn ist wichtig, dass die Vertraulichkeit und Integrität der persönlichen Daten der Nutzer gewährleistet wird.

4 Rahmenbedingungen und Lösungsstrategien

In den folgenden Abschnitten wird auf zweierlei Aspekte eingegangen. Zum einen auf die Rahmenbedingungen, die sich aus der Anforderungsanalyse ergeben. Diese umfassen hauptsächlich eine genauere Betrachtung des multimedialen Museumsguide, der Grundlage für die Analyse der Museumsbesuche ist. Hier wird auf die Funktionsweise, den Aufbau und die verwendeten Datenstrukturen eingegangen, da diese die Grundlage für das zu entwickelnde System bilden. Zum anderen werden Lösungsstrategien diskutiert. Diese umfassen zunächst keinen konkreten Systementwurf, sondern die verschiedenen Möglichkeiten der Umsetzung. Sowohl von technischer Seite, also in Bezug auf Frameworks und Technologien, als auch von konzeptioneller Seite. Es wird besprochen, welche Arten von Recommender Engines sich aus den Rahmenbedingungen als sinnvoll erweisen und wie sich diese integrieren lassen.

4.1 Mobiler, multimedialer Museumsguide

Im Rahmen des Kooperationsprojektes *EMIKA* zwischen dem Jüdischen Museum Berlin und der Forschungsgruppe INKA[1] der HTW Berlin wurde im Zeitraum März 2007 bis September 2008 ein mobiler, multimedialer Museumsguide (MMG) entwickelt. Dieser MMG soll die Besucher des Museums während ihres Besuches der Ausstellung begleiten und mit zusätzlichen, vertiefenden Informationen zu einzelnen Objekten der Dauerausstellung versorgen. Dabei ist das Konzept stark auf das Jüdische Museum Berlin und seine Besonderheiten zugeschnitten. Zum einen gehört dazu die Dauerausstellung. Das Museum gilt als historische Museums, besitzt als solches jedoch nur über eine eingeschränkte Sammlung von historischen Exponaten, was dem erst vergleichsweise kurzen Bestehen des Museums geschuldet ist.[2] Zum anderen

1 Forschungsgruppe Informations- und Kommunikationsanwendungen
2 Das Jüdische Museum Berlin wurde im Jahr 2001 eröffnet.

besitzt das Museum einen hohen Anspruch und eine sehr flexible Präsentationspraxis (vgl. [FHM08b]).

Für die Inhalte des MMG wurden Formate und Konzepte gewählt, die die Benutzer beim Besuch inhaltlich unterstützten und sich zugleich für den Einsatz auf mobilen Endgeräten eignen. Konkret sind dies multimediale Inhalte in Form von Audio-Stücken, Videos und interaktiven Medientypen. Unter interaktiven Medientypen werden verschiedene Formate verstanden, die es den Benutzern erlauben, in die Präsentation einzugreifen und diese zu steuern. Dies erfolgt beispielsweise in Form von interaktiven Büchern, die der Benutzer durchblättern kann, Diashows, Hot-Spot-Bildern etc. Das System wird durch ein Navigationssystem für das Gebäude und positionsabhängige Funktionen ergänzt. Diese ermöglichen es, den Benutzern automatisiert kontextabhängige Inhalte anbieten zu können.

4.1.1 Technische Umsetzung

Auf technischer Seite wurden die Medieninhalte des MMG mittels Verwendung der Adobe-Flash-Technologie umgesetzt. Die Medientypen sind dabei generische Komponenten, die auf Basis von ActionScript 2 realisiert wurden (vgl. [FHM08b]). Die Medieninhalte setzen dabei auf eine Laufzeitumgebung auf, die mit C# umgesetzt wurde. Dies setzt Windows Mobile als Betriebssystem voraus. Als Referenzgerät wählten die Entwickler einen PDA *Hewlett-Packard iPAQ hx2790*.

Als Grundlage für das Navigationssystem und die kontextsensitiven Funktionen ist ein Positionierungssystem notwendig. Der MMG nutzt hier das auf RFID basierende System *OpenBeacon* (vgl. 2.1.4/S. 10). In dieser Lösung kommen aktive

(a) Menü **(b)** Kapitelauswahl **(c)** Multimedia-Inhalt

Abbildung 4.1: Screenshots MMG. Quelle: Forschungsprojekt EMIKA, [FHM08b]

RFID-Transponder zum Einsatz. Die PDA sind mit einem externen RFID-Lesegerät ausgestattet.

Aufgrund der räumlichen Situation des Jüdischen Museums Berlin waren bei der Platzierung der Transponder einige Besonderheiten zu beachten. Die Dauerausstellung des Museums gliedert sich in 14 Abteilungen, die als Segmente bezeichnet werden. Diese verteilen sich auf zwei Obergeschosse und werden im Normalfall von den Besuchern sequenziell durchlaufen. Diese Aufteilung spiegelt sich auch in der Verteilung der Transponder wieder. Jedes Segment ist mit einem oder mehreren Transpondern ausgestattet. Die Transponder senden Signale in maximal 4 verschiedenen Signalstärken aus und besitzen eine eindeutige ID. Darüber hinaus sind einzelne Exponate mit eigenen Transpondern ausgestattet, die in einer geringeren Signalstärke senden. Zur Positionsbestimmung wurde ein eigener Algorithmus entwickelt. Die Position wird nach der Anzahl der empfangenen Signalpakete und ihrer Signalstärke ermittelt (vgl. [FHM08a]). Dabei können über eine Zuordnungstabelle das jeweils aktuelle Segment oder das am nächsten entfernte Objekt erkannt werden. Wird ein Positionswechsel vom Lesegerät erkannt, löst dies ein Event an der C#-Komponente aus.

Auf dem MMG wird die aktuelle Position in Form einer Karte dargestellt, über die einzelne Objekte und multimediale Inhalte ausgewählt werden können. Bei einem erkannten Positionswechsel wird in der Flash-Anwendung der Ausschnitt der Karte angepasst bzw. Inhalte zu einem Objekt angeboten.

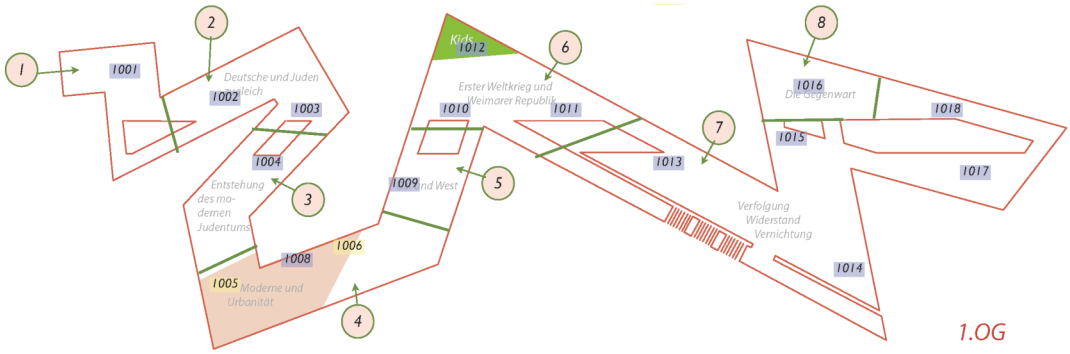


Abbildung 4.2: Verteilung der RFID-Transponder im Jüdischen Museum Berlin, 1. OG. Quelle: [FHM08b]

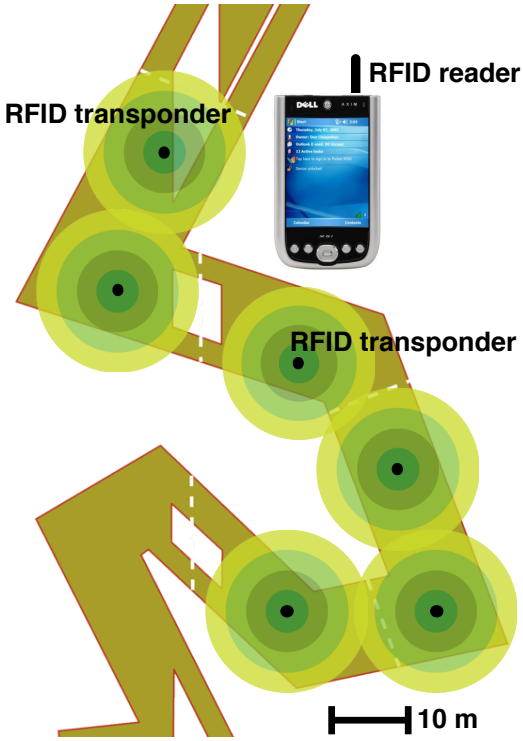


Abbildung 4.3: RFID Lokalisierungsszenario im Jüdischen Museum Berlin. Quelle: [FHM08a]

4.1.2 Datenstrukturen

Im folgenden werden die XML-Datenstrukturen des MMG beschrieben. Dabei wird hauptsächlich auf die für ein Recommender System notwendigen Datenstrukturen eingegangen. Strukturen, die nur für die Darstellung und Steuerung des MMG notwendig sind, werden weitgehend übergangen, da sie in diesem Kontext irrelevant sind.

Inhalte

Die Daten des MMG liegen in einer vorgegebenen Ordnerstruktur im XML-Format vor. Der Ordner `/data/proto_data/XMLData` enthält dabei die Datei `segment.xml`, diese beschreibt die Basisstruktur. Die Datenstruktur orientiert sich dabei an den räumlichen Gegebenheiten und dem Aufbau des Jüdischen Museums Berlin. Die

Dauerausstellung mit ihren 14 Segmenten, die inhaltliche Abteilungen darstellen, erstreckt sich über über zwei Obergeschosse.

Analog zur räumlichen Aufteilung gibt es in der Basisstruktur der Daten das Element `<map>`, das jeweils ein Stockwerk darstellt, mit den Unterelementen `<segment>`. Diese besitzen verschiedene Formatierungs-Attribute, wobei das Attribut `id`, das das entsprechende Segment kennzeichnet. In einem Segment können mehrere Module enthalten sein, die durch das Element `<module>` mit dem Attribut `id` und anderen Attributen zur Darstellung und Formatierung beschrieben werden.

Listing 4.1: Auszug Datenstruktur Segmente/Module (`segment.xml`)

```
<?xml version="1.0" encoding="utf-8" ?>
<segmentData coformat="flash8">
  [...]
  <map id="2" src="flash/maps/pe_og2.swf">
    <segment id="1" posx="6" posy="4" rotation="9,7" scale="85,1"/>
    <segment id="2" posx="2" posy="4" rotation="9,7" scale="78,4"/>
      <module id="32" posx="15" posy="93" width="15" height="81"
         rotation="135" scale="12" src="flash/maps/P032.swf"/>
      <module id="32" posx="17" posy="95" width="15" height="81"
         rotation="135" scale="12" src="flash/maps/P032.swf"/>
    </segment>
    <segment id="3" posx="5" posy="5" rotation="6,8" scale="100"/>
    [...]
  </map>
</segmentData>
```

Die einzelnen Module werden in separaten XML-Dateien näher beschrieben. Die Zuordnung von Segmenten zu den jeweiligen Dateien erfolgt über die Datei `ModuleCollection.xml`. Dort werden die einzelnen Module über das Attribut `name` betitelt, diese können mehrere Kapitel (`<chapter>`) enthalten, die ebenfalls über die Attribute `id`, `name` und die Sprache (`lang`) beschrieben werden. Darüber hinaus werden die Inhalte der einzelnen Kapitel spezifiziert.

Listing 4.2: Auszug Datenstruktur Modul (Beispiel `P111.xml`)

```
<?xml version="1.0" encoding="UTF-8"?>
<content>
  <module id="111" name="Dybbuk" lang="de">
    <chapter id="0" name="&#8222;Der Dybbuk&#8220;" type="object">
      [...]
    </chapter>
    <chapter id="2" name="Zeitzeugnis" type="today">
      [...]
```

```
    </chapter>
    <menu audio="media/P111/audio/P111_m.mp3"/>
  </module>
</content>
```

Log-Daten

Der Verlauf des Besuches wird vom MMG durch eine eigene Logging-Komponente protokolliert. Dazu werden auf der einen Seite die automatisch ausgelösten Events (Positionswechsel, Übergang in ein neues Segment bzw. Annäherung an ein mit RFID-Transponder versehenes Objekt) aufgezeichnet, auf der anderen Seite aber auch Nutzeraktionen (Start eines Multimedia-Inhaltes etc.).

Das Protokoll wird dabei in XML-strukturierter Form aufgezeichnet und orientiert sich dabei an der Datenstruktur der Inhalte des MMG. Im Protokoll werden der Start und das Ende der Benutzung des MMG protokolliert, sowie die durchlaufenen Segmente, betrachteten Module und Kapitel. Die Segmente, Module und Kapitel werden dabei anhand ihrer ID identifiziert und die jeweilige Startzeit aufgezeichnet, für die einzelnen Module zusätzlich auch noch der Zeitpunkt des Endes, sowie bei Kapiteln die Information, ob sie vollständig bis zum Ende betrachtet oder vorzeitig abgebrochen wurden.

Listing 4.3: Datenstruktur Besuchs-Protokoll (Beispiel)

```
<?xml version="1.0" encoding="us-ascii"?>
<logging>
  <visit device="iPaq 7" start="05.08.2008 11:23:21" end="
     05.08.2008 12:17:50">
    <segment id="6" start="05.08.2008 11:23:51">
      <modul nr="61" start="05.08.2008 11:26:51" end="05.08.2008 11
         :34:54">
        <chapter nr="0" type="object" start="05.08.2008 11:26:51"
           aborted="false"/>
        <chapter nr="1" type="archive" start="05.08.2008 11:29:22"
           aborted="false"/>
        <chapter nr="0" type="object" start="05.08.2008 11:33:17"
           aborted="true"/>
      </modul>
    </segment>
    <segment id="6" start="05.08.2008 11:51:34"/>
  </visit>
</logging>
```

Bei der Analyse des Protokolls müssen einige Besonderheiten beachtet werden. Zum einen können alle Elemente mehrfach durchlaufen werden und sich doppeln. Es kann ebenso vorkommen, dass Module auch außerhalb des Segments betrachtet werden, dem sie eigentlich zugeordnet sind. Diese lassen sich jedoch über die ID identifizieren. Die ID der Module setzt sich zusammen aus der ID des Segments, an die als laufender Zähler eine ID für das Modul angehängt wird. Kapitel lassen sich nur innerhalb eines Moduls betrachten, aus diesem Grund reicht hier eine einfache ID aus. Daneben wird noch der Typ des Kapitels (*Object, Story, Quotation, Today, Archive, Exhibition*) als Attribut aufgezeichnet.

4.2 Technologien

Aus der Anforderungsanalyse ergeben sich einige Vorgaben, die bei der Wahl der zu verwendenden Technologien Berücksichtigung finden müssen. Dort wird eine Anwendung für Benutzer gefordert, mit der auf die Datenbasis zugegriffen und Inhalte abgerufen werden können (*Frontend*). Der zweite Teil ist eine Verwaltungs-Anwendung zur Administration des Systems und der Inhalte.

Als erste Vorgabe ist zu beachten, dass das Frontend als Webanwendung konzipiert sein soll, für das Backend werden keine konkreten Vorgaben gemacht. Bei der Auswahl muss der Aspekt der mobilen Anwendung Berücksichtigung finden. Auch wenn im Prototyp der mobile Abruf von Informationen nicht umgesetzt wird, muss dieser Aspekt in die Auswahl der Technologien und die Konzeption einbezogen werden.

Da zumindest ein Teil der Applikation als Webanwendung umzusetzen ist, ist es sinnvoll, die komplette Applikation derartig zu konzipieren, d. h. auch die Verwaltung-Applikation. Die Umsetzung über eine andere GUI (z. B. über die Java Bibliotheken *Swing* oder das *Standard Widget Toolkit – SWT*) wäre durch die Architektur der Applikation prinzipiell jedoch problemlos möglich.

4.2.1 Web-Frameworks

Die Verwendung eines Frameworks für Web-Applikationen ist heutzutage bei Konzeption und Entwurf komplexer Web-Applikationen als Prämisse anzusehen. Ihnen liegt eine Trennung der verschiedenen Aspekte einer Anwendung in Form des MVC[1]-

1 Model View Controller

Patterns in direkter oder angepasster Form zugrunde. Einzelne Frameworks sind dabei auf bestimmte Verfahren des Softwareentwicklungsprozesses (z. B. *Agile Development*) ausgelegt und unterstützen diesen Prozess. Darüber hinaus umfassen Web-Frameworks meist Funktionalitäten für einen vereinfachten Datenbankzugriff und ermöglichen die Kommunikation über offene Schnittstellen. Ebenfalls integrieren sie meist ein Templating-System zur Generierung des Benutzerinterfaces.

An dieser Stelle werden verschiedene Web-Frameworks diskutiert und verglichen, um eine geeignete Auswahl zu finden. Dabei wird sich auf Frameworks beschränkt, denen die Programmiersprache Java zu Grunde liegt, oder die JVM[1] nutzen. Die Auswahl der zu vergleichenden Frameworks muss aufgrund der Vielzahl der Verfügbaren weiter eingeschränkt werden, in erster Linie orientiert sich die Auswahl am Vergleich unterschiedlicher Ansätze und Technologien.

Web-Frameworks nutzen meist das MVC-Entwurfspattern in der klassischen oder angepassten Form. Mit Hilfe dieses Entwurfsmusters werden die Komponenten in das *Model*, den *View* und den *Controller* aufteilt. Das Model wird dabei von den Daten gebildet, die den anderen Komponenten zur Verfügung gestellt werden und dient dem Austausch von Daten zwischen Anwendungs- und Präsentationsschicht. Dabei besitzt das Model keine Kenntnis über die Komponenten des Views oder Controllers. Über die Komponenten des Views findet die Interaktion mit den Benutzern statt. Dies ist zum einen die Darstellung des Models, sowie die Bereitstellung von GUI-Komponenten zur Auslösung von Aktionen, wie der Änderung des Models. Die Aktionen werden von den Komponenten des Views dabei nur ausgelöst und vom

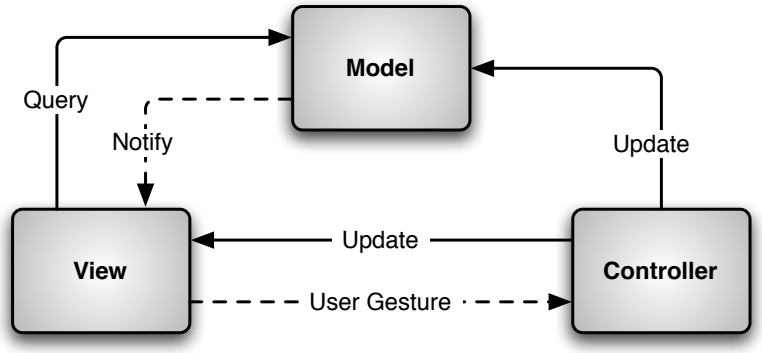


Abbildung 4.4: Klassisches MVC-Pattern

1 Java Virtual Machine

Controller ausgeführt. Dieser stellt die Schnittstelle zwischen Model und View dar und löst die Aktionen auf der Anwendungslogikschicht aus, sowie stellt dem View die Daten des Model zur Verfügung.

Apache Struts

Das in Java implementierte Web-Framework *Struts* [UrlStruts], das von der *Apache Software Foundation* entwickelt wird, ist weit verbreitet. Es folgt dem MVC-Pattern und nutzt dafür bereits vorhandene Java-Techniken, wie *Java Server Pages* (JSP), *Java Beans* und *Servlets*. Struts bietet eine Architektur, die sich einfach ergänzen lässt. Java Beans dienen dem Zugriff auf das Model, das innerhalb der Anwendungslogik gespeichert wird. Die View-Komponente wird über JSP realisiert, dazu verfügt Struts über eigene Tag-Libraries. Gesteuert wird der View vom Controller, der über Java Servlets implementiert wird. Diese verarbeiten die Anfragen und generieren die Antworten in Form von JSP.

Verschiedene andere Frameworks setzen auf Struts auf bzw. ergänzen die Funktionalitäten verschiedener Aspekte des Web-Frameworks. Dies sind beispielsweise Ergänzungen durch Tag-Libraries oder die Integration eines Objektrelationalen Mappings (ORM) für den Datenbankzugriff, das nicht Teil des Funktionsumfanges von Struts ist. Nachteilig an Struts ist die teilweise komplexe Konfiguration der Komponenten, insbesondere bei umfangreichen Applikationen.

Grails

Groovy [UrlGroovy] ist eine an Java angelegte objektorientierte Scriptsprache, die bei der Ausführung in Java-Bytecode übersetzt wird und damit in der JVM ausgeführt werden kann. Dies ermöglicht es, Java Code und Java Bibliotheken in Groovy zu nutzen bzw. umgekehrt kompilierte Groovy-Klassen in Java.

Das Framework *Grails* versucht den Entwicklungsprozess zu verschlanken und zu vereinfachen. Ziel ist es, das Paradigma *Convention over Configuration* mit Groovy zu verbinden (vgl. [UrlGrails]). Dazu werden Techniken und Methoden aus dem Bereich der agilen Softwareentwicklung verwendet. Grails benutzt das MVC-Pattern um die verschiedenen Schichten zu trennen. Über *Scaffolding* wird ermöglicht, nach der Erstellung eines Models die grundlegenden *CRUD*[1]-Datenbankoperationen ohne

1 „Create, Read, Update, Delete“

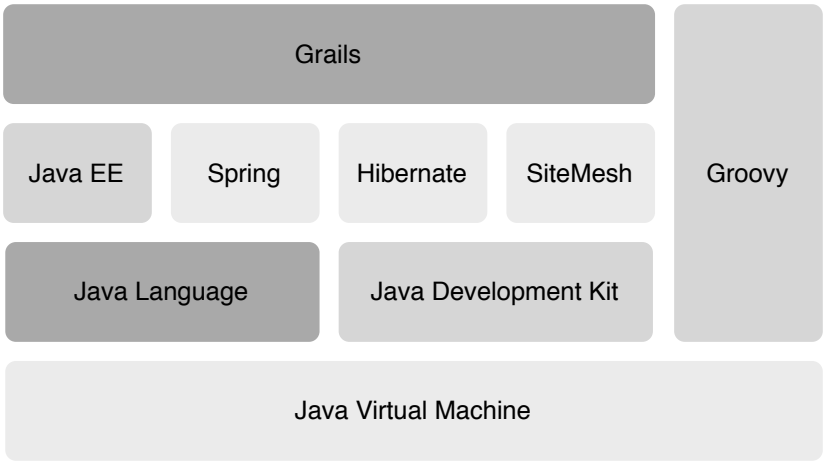


Abbildung 4.5: Grails Stack, nach [BR09, S. 3]

weitergehende Implementierung eines Views und Controllers im Prototyping-Prozess direkt ausführen zu können. Grails setzt auf bestehende und verbreitete Frameworks auf. So nutzt es standardmäßig *Hibernate* [UrlHiber] als Persistenz-Framework und Schnittstelle zur Datenbank. Über den Ansatz *Convention over Configuration* wird versucht, die Konfiguration des Frameworks auf ein Minimum zu reduzieren. Die Nutzung von Hibernate benötigt in Grails keinerlei eigene Konfigurationen, sondern besitzt dafür mit GORM[1] einen eigenen Mapper. Darüber hinaus nutzt es das Spring-Framework [UrlSpring]. Grails integriert dadurch verschiedene Konzepte, wie eine Internationalisierung (*i18n*). Der Grails-eigene Template-Engine *GSP* setzt auf JSP auf.

Apache Cocoon

Das von der *Apache Software Foundation* herausgegebene Web Framework *Cocoon* ([UrlCocoon]) nutzt das Konzept *Separation of Concerns* und der Komponenten-basierten Entwicklung. Die geschieht bei Cocoon, indem die verschiedenen Komponenten klar voneinander getrennt sind. Die Komponenten sind über sogenannten *Pipelines* verbunden, wobei eine Anfrage die verschiedenen Komponenten durchläuft. Eine Besonderheit von Cocoon ist, dass es stark auf XML basiert. Seit der letzten

1 Grails' object relational mapping

Version 2.2[1] ist es möglich, das Spring-Framework zu integrieren, was die Verwendung von externen Standard-Komponenten (z. B. ORM) vereinfacht.

Entscheidung

Für eine prototypische Umsetzung bieten Frameworks, die sich an einem verschlankten Software-Entwicklungsprozess orientieren, Vorteile, da sie Konzepte für eine schnelle Umsetzung mitbringen und den Konfigurationsaufwand gering halten. Diese Vorteile zeigen sich im Vergleich von herkömmlichen Frameworks wie Struts und agilen Frameworks wie Grails besonders stark. Durch den Ansatz der vorgegebenen Konventionen wird der herkömmliche Konfigurationsaufwand um ein Vielfaches reduziert. Die vorkonfigurierte Integration von externen Frameworks wie etwa zur Umsetzung einer Persistenzschicht oder Generierung von Layout-Templates ermöglichen eine Konzentration auf die zentralen Elemente der Anwendungslogik. Die vorhandenen Konventionen erfüllen die Standard-Funktionalitäten, lassen sich aber trotzdem je nach Bedarf anpassen. Mit Mechanismen wie *Scaffolding* und *CRUD* sind schon nach der Erstellung eines Models prototypische Aktionen ausführbar. Aus diesem Grund ist die Verwendung des Grails-Frameworks hier praktikabel und anderen Techniken vorzuziehen.

4.2.2 Frontend-Technologien

In der Anforderungsanalyse wird zum einen eine Web-Applikation als Benutzer-Schnittstelle, zum anderen die Bereitstellung der Daten in Form von Web Services gefordert. Hier kommen jeweils unterschiedliche Technologien in Betracht, die im folgenden diskutiert werden.

Adobe Flash

Interaktive Benutzerschnittstellen auf Basis Adobe Flash benötigen eine eigene Laufzeitumgebung, den sogenannten Flash-Player. Dieser wird normalerweise über ein entsprechendes Plugin im Webbrowser bereitgestellt. Es handelt sich dabei um eine proprietäre Technik, deren Spezifikationen lange Zeit nicht offen gelegt

1 Die Version 2.2 wurde im März 2008 veröffentlicht.

waren.[1] Im Flash-Player werden die zuvor kompilierten Dateien abgespielt, die im SWF[2]-Format vorliegen. Das erstmalige Laden eines Flash-Films erfolgt mittels eines normalen HTTP-Requests, bei dem die HTML-Seite mit eingebetteter SWF-Datei geladen wird. Die weitere Kommunikation des Flash-Films findet dann zwischen Flash-Player und Server statt. Dafür stehen entweder das von Adobe spezifizierte Protokoll RTMP[3] zur Übertragung von Audio, Video oder Objekten oder Web Services (REST, SOAP, siehe Abschnitt 4.2.3/S. 46) zur Verfügung.

Die Entwicklung der Anwendung erfolgt entweder herkömmlich über die Flash-eigene Programmiersprache *ActionScript* oder über das *Flex*-Framework. Dieses stellt neben ActionScript mit der Auszeichnungssprache *MXML* noch eine weitere Möglichkeit für die Entwicklung der Client-seitigen Darstellungsschicht zur Verfügung. Server-seitig erfolgt die Kommunikation über Serverkomponenten, die die Anbindung an weitere Applikationen zulassen. Dafür gibt es verschiedene Frameworks, die Schnittstellen für die gängigen Programmiersprachen (Java, PHP etc.) bereitstellen. Dies sind z. B. *Adobe LiveCycle Data Services ES* [UrlAdobe] oder *Red5* [UrlRed5], die beide Unterstützung für Java bieten.

HTML/CSS mit JavaScript/Ajax

Die Auszeichnungssprache HTML[4] beschreibt strukturiert Dokumente zur Darstellung in einem Webbrowser. Dabei wird sowohl der Inhalt der Dokumente beschrieben, als auch das Dokument selbst über Metadaten. Grafiken und andere Medien lassen sich über Verweise referenzieren. Die Darstellung von HTML[5]-Dokumenten im Webbrowser lässt sich über CSS[6] steuern, indem Elementen des Dokuments Stil-Definitionen zur Formatierung zugewiesen werden (vgl. [Hen07]).

Mit der Scriptsprache *JavaScript* wird es ermöglicht, HTML-Dokumente zu dynamisieren. Das bedeutet, dass auf Eingaben von Nutzern und auf Events reagiert werden kann und Aktionen ausgelöst werden, z. B. die dynamische Änderung der Struktur eines HTML-Dokumentes durch Manipulation des DOM[7]. Dabei wird JavaScript in-

1 Die Spezifikationen werden von Adobe seit Anfang 2009 im Rahmen des *Open Screen Projects* veröffentlicht, siehe [UrlOpenScre].
2 Shockwave Flash
3 Real Time Messaging Protocol
4 Hypertext Markup Language
5 Hypertext Markup Language
6 Cascading Style Sheets
7 Document Object Model

nerhalb einer Sandbox im Webbrowser ausgeführt, womit nur Zugriffe auf Elemente des Webbrowsers erlaubt werden, tiefergehende Zugriffe, beispielsweise auf Objekte des Betriebssystems oder das Dateisystem, sind nicht möglich (vgl. [UrlSelfJava]). Der Kern von JavaScript ist von der Normierungsorganisation *ECMA International* als Standard ECMA-262 genormt (vgl. [Hen07]).

Neben der herkömmlichen Kommunikation mit Webservern stellen moderne Webbrowser eine Schnittstelle über ein *XMLHttpRequest*-Objekt zur Verfügung. Dies ermöglicht Daten vom Webserver über die herkömmlichen HTTP-Request/-Response Methoden anzurufen, diese aber zu verarbeiten, ohne das dargestellte HTML-Dokument neu laden zu müssen. Die Generierung der Anfragen und Verarbeitung der Antworten erfolgt dabei über JavaScript und das Ergebnis wird erst nach der Verarbeitung der Antwort durch die Manipulation des DOM o.ä. dargestellt. Die Technik wird daher als Ajax[1] bezeichnet. Der HTTP-Response besteht nicht wie bei einem normalen HTTP-Request über einen Webbrowser aus einem HTML-Dokument, sondern aus einer nicht vorgegebenen Datenstruktur. Meist werden die Anfragen aber über Web Services realisiert. Die übertragenen Datenmengen sind dadurch wesentlich geringer als bei der Übertragung eines kompletten HTML-Dokuments, was die Geschwindigkeit der Verarbeitung von Nutzeraktionen erhöht. Dies erhöht die Usability der Benutzerschnittstelle und bei Benutzern entsteht der Eindruck, eine herkömmliche Desktop-Anwendung zu bedienen.

Bewertung

Prinzipiell eignen sich beide Techniken zur Umsetzung einer Benutzerschnittstelle, sowohl Adobe Flash als auch die herkömmlichen Techniken. Die Vorteile einer Flash-Anwendung liegen in einer einheitlichen Darstellung der GUI-Komponenten, unabhängig des zugrunde liegenden Webbrowsers (und seinen Einstellungen) und Rahmenbedingungen des Systems (System-Einstellungen, installierte Schriften etc.). Jedoch entstehen auch einige Probleme. So sind zum einen zusätzliche Komponenten notwendig (Flash-Plugin, Server-Framework), zum anderen werden keine offenen Standards benutzt. Zwar liegen mittlerweile die meisten Spezifikationen offen, doch werden diese weiterhin vom Hersteller Adobe entwickelt. Im Gegensatz dazu sind HTML und CSS offene Standards des W3C[2], die in einem transparenten Prozess weiterentwickelt werden und frei von Patenten veröffentlicht werden (vgl. [UrlW3C]).

1 Asynchronous JavaScript and XML
2 World Wide Web Consortium

Die Nutzung ist mit jedem Webbrowser der aktuellen Generation möglich. Die Darstellung im HTML-Format erlaubt eine einfache Anpassung der Inhalte an spezielle Rahmenbedingungen, wie beispielsweise für Webbrowser auf mobilen Geräten, die nur über eine geringere Bildschirmauflösung verfügen.

4.2.3 Web Services

Für die Nutzung des Systems über mobile Endgeräte eignen sich Web Services. Da dieser Bereich zwar in der Systemarchitektur berücksichtigt, jedoch in der prototypischen Umsetzung nicht implementiert wird, wird das grundsätzliche Prinzip und die einzelnen Möglichkeiten von Web Services andiskutiert, jedoch nicht abschließend bewertet.

Die W3C-Arbeitsgruppe zum Thema Web Services beschreibt diese als Applikationen, die durch eine URI angesprochen werden, deren Schnittstellen als XML-Artefakte beschrieben werden können und die die Interaktion mit anderen Software-Systemen über Internet-Protokolle auf Basis von XML-basierten Nachrichten ermöglichen (nach [UrlWebservi]).

SOAP

SOAP (*Simple Object Access Protocol*) ist ein Standard des W3C für Web Services, um strukturierte Informationen in verteilten Systemen auszutauschen (vgl. Spezifikation [UrlSoap]). Das Protokoll ist dabei nicht festgelegt, üblicherweise wird zwar HTTP benutzt, es sind aber auch andere Protokolle, wie SMTP und POP3 möglich. Die Beschreibung der Nachrichten erfolgt im XML-Format, eine Nachricht besteht aus dem *Envelope*, der *Header* (optional) und *Body* enthält. Der Header enthält dabei Metainformationen, wie etwa Kontrollnachrichten, der Body die eigentlichen Informationen.

Die Schnittstellen können über die XML-basierte Beschreibungssprache *Web Services Description Language* (WSDL) beschrieben werden. Die Beschreibung enthält sowohl die angebotenen Dienste und Methoden als auch die technischen Spezifikationen. Konkret werden die Eingabe-und Ausgabedaten, Nachrichten und Datentypen definiert und die Schnittstellen mit Protokollen und Ports beschrieben.

XML-RPC

Ein alternativer Ansatz für Web Services ist das der XML-RPC (*XML-Remote Procedure Call*). Mit Hilfe von XML-RPC wird das Konzept der *Remote Procedure Calls* (RPC) auf Web Services adaptiert. Das Konzept der RPC wurde für verteilte Systeme entwickelt und besagt das Aufrufen von Methoden und Prozeduren auf einem entfernten System. Dazu wird von einem Client eine Nachricht an einen Server gesendet, die die aufzurufende Prozedur und die benötigten Parameter enthält. Die Prozedur wird auf dem Server ausgeführt und das Resultat als Antwort an den Client gesendet.

Bei XML-RPC wird für die Beschreibung der Nachrichten XML, für die Datenübertragung das HTTP-Protokoll genutzt. Eine Anfrage erfolgt dabei über ein HTTP-`POST` Request, das Vokabular zur Beschreibung ist dabei vordefiniert. Die Spezifikation [UrlXmlRpc] beschreibt eine Liste von Standard-Datentypen und erlaubt die Verwendung von Structs und Arrays für strukturierte Daten. Binäre und andere Datenformate können *Base64* kodiert werden.

REST

Anders als die zuvor diskutierten Techniken SOAP und XML-RPC beschreibt REST (*Representational State Transfer*) keinen Standard oder konkrete Spezifikation, sondern einen Architekturstil für Web Services.[1] Dabei basiert REST auf den Prinzipien des Internets, die Anfragen und Antworten werden technisch mit den Methoden des HTTP (`GET`, `POST`, `PUT`, `DELETE` etc.) realisiert. Da der Server dabei zustandslos (*stateless*) ist, müssen die Anfragen der Clients alle benötigten Parameter enthalten. Jede Ressource auf dem Server besitzt dabei eine eindeutige URI. Das Format der Antwort auf die Anfrage ist nicht weiter spezifiziert. Diese erfolgt in Form eines beliebigen Dokuments (z. B. XML, Medientypen), das wiederum Verweise auf andere Ressourcen enthalten kann.

1 Das Konzept von REST wurde im Jahr 2000 in der Dissertation *Architectural Styles and the Design of Network-based Software Architectures* von R.T. Fielding veröffentlicht, vgl. [UrlRest].

4.3 Recommender System

Im folgenden Abschnitt werden die zur Verfügung stehenden Daten betrachtet und die Möglichkeiten zur Verwendung als Datenbasis und Eingabedaten für das Recommender System diskutiert. Anhand des Anwendungsfalls werden die in Frage kommenden Recommender Techniken betrachtet und eine konzeptionelle Entscheidung getroffen, die auch den geforderten Datenschutz-Aspekten gerecht wird.

4.3.1 Datenbasis und Eingabedaten

Wie unter 2.2 (S. 10) eingeführt sind Datenbasis und Eingabedaten integrale Bestandteile eines Recommender Systems.

Als Eingabedaten stehen zunächst die durch den MMG protokollierten Log-Daten (vgl. Abschnitt 4.1/S. 33) zur Verfügung. Diese beinhalten Daten zu angesehenen Bereichen und Objekten und den damit verbundenen Informationen (Dauer, Reihenfolge etc.). Die Protokolle enthalten eine eindeutige ID zu allen Objekten, jedoch keine weitere Bezeichnung oder inhaltliche Einordnung. Eine nähere Bezeichnung ist jedoch in den Datenstrukturen der Inhalte zu finden. Diese lassen sich über eindeutige IDs auf die protokollierten Daten zuordnen.

Die XML-Dateien mit den Datenstrukturen stellen zunächst die Grundlage für den MMG dar und enthalten dessen Aufbau und bei Modulen und Kapiteln Ansätze einer Beschreibung in Form eines Namens. Segmente besitzen keinen Namen, hier ist eine Identifikation über die ID möglich. Um die protokollierten Log-Daten einzuordnen ist daher eine Verknüpfung mit den inhaltlichen Strukturen des MMG notwendig. Dies wird über die benutzten eindeutigen IDs ermöglicht.

Die Datenbasis wird durch die Informationen dargestellt, die nicht im mobilen Museumsguide enthalten sind und den Besuch inhaltlich ergänzen sollen (*Informationspool*). Eine Beziehung zwischen Museumsguide und diesen Daten besteht zunächst nicht. Als Grundlage für den Einsatz eines Recommendation Engines ist es notwendig, eine Beziehung zwischen den Eingabedaten und der Datenbasis – dem durch den MMG ermittelten Besuchsprofil und dem Informationspool – herzustellen. Diese Beziehung darf dabei nicht statisch sein. Eine direkte, feste Zuordnung von Inhalten des Informationspools zu Objekten des MMG würde bei einer späteren Ergänzung oder Veränderung der Struktur oder Inhalte des MMG eine entsprechende Änderung bei den Inhalten des Informationspools nach sich ziehen. Daher wird eine

Art der Zuordnung benötigt, die robust und unabhängig gegenüber Änderungen der Struktur der Eingabedaten und der Datenbasis ist.

Indexierung

Für die Zuordnung ist eine inhaltliche Erschließung der Objekte und Inhalte notwendig. Eine inhaltliche Erschließung kann grundsätzlich entweder bei Texten über den Volltext oder eine strukturierte Form erfolgen. Bei Nicht-Text-Objekten, wie im vorliegenden Fall, bleibt nur die Erschließung über eine Beschreibung der Objekte. Diese Beschreibung muss dabei sowohl für die Datenbasis als auch die Eingabedaten einheitlich und geeignet sein.

Um Objekte und Dokumente inhaltlich zu beschreiben werden *Dokumentationssprachen* genutzt. Diese lassen sich in *natürlich-sprachlich basierte* und *künstlich (nicht natürlich-sprachlich) basierte* Dokumentationssprachen unterscheiden.

Zu den natürlich-sprachlich basierten Beschreibungsmethoden zählen *Thesauri* und *Schlagwortsysteme* (vergleiche hierzu [UrlLuckh]). Hier werden die Objekte durch den Inhalt beschreibende Deskriptoren erschlossen. Bei Schlag- oder Stichwortsystemen[1] erfolgt dies durch eine Zuordnung von Beschreibungsworten aus einer vorgegebenen Menge (*kontrolliertes Vokabular*) oder freien Menge (*freies Vokabular*). Ein Thesaurus besteht aus einem kontrollierten Vokabular, wobei die einzelnen Attribute durch Relationen in Beziehung zueinander stehen. Damit werden Synonyme, Abkürzungen etc. in Beziehung zueinander gesetzt (*Äquivalenzrelation*) und Attribute hierarchisch eingeordnet (*Hierarchierelation*).

Klassifikationen sind in den Bereich der künstlich basierten Dokumentationssprachen einzuordnen. Sie bestehen aus einem festen Vokabular und ordnen die Objekte anhand ihrer Eigenschaften in Klassen ein. Die Klassen können zueinander in hierarchischen Beziehungen stehen. Anders als beim Thesaurus muss die Bezeichnung der Klasse nicht mit der inhaltlichen Einordnung verbunden sein, die Klassen spiegeln lediglich eine zuvor festgelegte Art der Systematik wieder. Bei der Bildung von Hierarchien sind verschiedene Strategien grundsätzlich möglich. Zum einen *monohierarchische Strukturen*, die eine Baumstruktur abbilden. Dies bedeutet, dass ein Element immer nur ein übergeordnetes Element besitzt, also eindeutig zugeordnet ist. Zum anderen

1 Der Unterschied zwischen Schlag- und Stichwörtern liegt in ihrem Vorkommen im Inhalt; Schlagwörter kommen nicht darin vor, Stichwörter hingegen schon (vgl. [UrlLuckh]).

sind *flache Strukturen* möglich, bei denen die Elemente nicht in eine Hierarchie eingeordnet sind, d. h. keine Oberklasse und/oder Unterklassen besitzen.

Tagging erweitert die Möglichkeiten der Indexierung von Objekten, die im Zuge des sogenannten *Web 2.0* entwickelt wurde und in diesem Umfeld eine starke Bedeutung besitzt. Unter dem Begriff Tagging wird das gemeinschaftliche Indexieren von Objekten verstanden. Die Methode fällt in den Bereich der natürlich-sprachlich basierten Beschreibungsmethoden, wobei das Vokabular nicht vorgegeben ist, sondern von den Nutzern frei gewählt wird. In Abgrenzung zur herkömmlichen Verschlagwortung wird Tagging auch als *Folksonomy* bezeichnet (vgl. [UrlMathes]). Die Besonderheit der Folksonomy liegt im kollaborativen Erstellen der Indizes. Das Hinzufügen von Tags ist unkontrolliert, also nicht wie bisher einer redaktionellen Instanz vorbehalten, sondern kann durch die Nutzer erfolgen. Des Weiteren ist spezifisch, dass keine hierarchischen Strukturen oder Relationen zwischen Tags existieren.

Inhaltliche Erschließung der Datenbasis und Eingabedaten

Alle zuvor diskutierten Methoden eignen sich grundsätzlich dazu, Inhalte und Objekte inhaltlich zu erschließen, doch stoßen diese auch an Grenzen und eignen sich abhängig von Randbedingungen besser oder weniger gut. Klassifikationen, Thesauri und alle anderen Methoden mit einem kontrollierten Vokabular erfordern eine hohe Vorleistung vor der eigentlichen Erschließung für das Erstellen des Indexes. Das Anlegen einer vorgefertigten Klassifikation oder eines neuen Thesaurus ist sehr aufwändig und nicht praktikabel, sofern dieser nicht schon vorliegt. Ähnliches gilt für das Anlegen eines neuen kontrollierten Vokabulars. Hier müssen bei der Erstellung alle möglichen Sachgebiete berücksichtigt werden. Die Vorteile liegen jedoch in einer präzisen, eindeutigen Beschreibung der Objekte bzw. Inhalte.

Automatische Erschließungsverfahren aus dem Bereich des Information Retrieval, wie beispielsweise das *tfidf-Maß* (vgl. 2.2.3/S. 16) kommen grundsätzlich zur automatischen Beschreibung von Texten in Betracht. Da im hier vorliegenden Anwendungsfall jedoch auch hauptsächlich Objekte ohne textuelle Beschreibung Verwendung finden, scheiden diese Verfahren aus. Aus diesem Grund ist einzig eine manuelle Einordnung praktikabel. Automatische Verfahren der Textanalyse wären ergänzend als Auswahlhilfen bei der manuellen Einordnung denkbar. Diese werden hier aber nicht weiter berücksichtigt, da sie auf die Mehrzahl der Objekte nicht anwendbar sind.

Aufgrund der Einschränkungen – das Fehlen einer vorhandenen Grundlage (Thesaurus, kontrolliertes Vokabular etc.) zur Erschließung der Eingabedaten und Da-

tenbasis – ergibt sich die inhaltliche Erschließung mittels Schlag- und Stichworte. Diese werden im folgenden auch als *Tags* bezeichnet, auch wenn die kollaborativen Tagging-Methoden nicht benötigt werden. Die Indexierung der Inhalte und Objekte soll nicht von den normalen Nutzer ausgeführt werden, sondern von den Redakteuren/Administratoren des Systems. Die Redakteure können eigene Inhalte am Präzisesten einordnen. Da eine hierarchische Struktur eines Vokabulars nicht bereits existiert ist, soll eine flache Struktur benutzt werden, bei der die Attribute nicht in Beziehung zueinander stehen. Durch die Erschließung ergibt sich für jedes Objekt eine Sammlung von Tags.

Die Objekte sind dabei entweder Inhalte des Informationspools oder Strukturen des MMG (Sektoren, Module, Kapitel), die durch einen Wort-Vektor von Tags beschrieben werden.

Die Gewichtung der Tags eines Objektes ist bei herkömmlichen Verfahren zur Verschlagwortung gleich, von Text Retrieval-Verfahren ist jedoch eine Gewichtung der beschreibenden Attribute bekannt. Dies soll auch hier genutzt werden, um die Objekte des MMG genauer zu beschreiben, je nachdem ob sie Interesse bei dem Nutzer gefunden habe.

4.3.2 Recommender Techniken

Prinzipiell kommen für die Umsetzung dieser Applikation verschiedene Recommender Ansätze durch die gegebenen Eingabedaten und Datenbasis in Betracht.

Regelbasierte Systeme *Regelbasierte Systeme* benötigen eine starre Zuordnung der möglichen Eingabedaten zu Daten aus Datenbasis. Aufgrund der komplexen Struktur der Eingabedaten und der unbegrenzten Größe der Datenbasis sind derartige Ansätze hier ungeeignet.

Collaborative Filtering Systeme auf Basis von *Collaborative Filtering* verfolgen das Prinzip, Empfehlungen aus dem Profil anderer Nutzer abzuleiten. Dies bedeutet sie basieren auf der Annahme, dass die Informationen relevant sind, die Nutzer mit einem ähnlichen Profil der Eingabedaten angesehen haben. Meist geschieht dies in Verbindung mit einer zusätzlichen Bewertung der Informationen (explizit oder implizit). Hier bestehen zwei Grundprobleme. Zum einen wird eine bestehende

Datenbasis benötigt, die Qualität der Empfehlung hängt auch von der Größe Datenbasis ab. Zum anderen werden neue Informationen der Datenbasis zunächst nicht berücksichtigt, da diese noch nicht bewertet wurden.

Wissensbasierte Systeme *Wissensbasierte Recommender Techniken* erfordern eine aufwändigere Verknüpfung zwischen Beschreibungs-Attributen und Anforderungen, was den Aufwand der Zuordnung extrem erhöht.

Inhaltsbasierte Systeme Einen anderen Ansatz stellt das *inhaltsbasierte Filtern* dar, bei dem Korrelationen zwischen den Eingabedaten und der Datenbasis gesucht werden. Voraussetzung dazu ist die Eingabedaten und die Datenbasis in einer einheitlichen Form zu beschreiben und inhaltlich einzuordnen. Dabei kommen verschiedene Arten der Beschreibung in Betracht.

Die zuvor eingeführte Erschließung der Eingabedaten und der Datenbasis mittels von Schlagwörtern ermöglicht die Beschreibung eines Objektes mittels eines Vektors in einem Vektorraum über alle möglichen Beschreibungs-Attribute. Dies lehnt sich an automatisierte Information Retrieval-Verfahren an, wobei die Wahl der Schlagworte manuell erfolgt, die Art der Beschreibung jedoch gleich ist. Die Gewichtung der Werte im Vektor kann dabei die Gewichtung der Schlagworte im Objekt ausdrücken. Dies trifft hier insbesondere auf die Eingabedaten zu, wobei die Gewichtung aus verschiedenen Faktoren ermittelt wird (Anzahl der betrachteten Kapitel und Module eines Segments, Länge des Besuchs eines Segments in Bezug auf die Gesamtbesuchsdauer etc.). Bei einer einfachen Verschlagwortung von Objekten ohne Gewichtung der Schlagworte wird von einem Normalwert 1 ausgegangen. Es entsteht so eine Vektorraum-Repräsentation der, mit den Vektoren $\overrightarrow{d_i}$ für jedes Element des Informationspools (*Dokument-/Objektvektoren*) und dem Vektor $\overrightarrow{q}$ zur Beschreibung der Eingabedaten (*Anfragevektor*). Die Richtung des Vektors gibt dabei die thematische Einordnung an, diese wird durch die Gewichtung der Attribute bestimmt.

	$\mathbf{d_1}$	$\mathbf{d_2}$	$\mathbf{d_3}$	$\mathbf{d_4}$	**q**
Tag 1	1	1	1	0	0.91
Tag 2	0	1	0	0	0.00
Tag 3	0	0	0	1	0.38
Tag 4	1	0	1	0	0.69
Tag 5	0	1	0	1	0.05

Tabelle 4.1: Objektvektoren d_i und Anfragevektor q

Die Ähnlichkeit zweier Vektoren in einem Vektorraum lässt sich mathematisch über ein Ähnlichkeitsmaß beschreiben, bei denen eine numerischer Wert für die Ähnlichkeit ermittelt wird. Dieser Wert ein probalistischer Wert. Zur Ermittlung der Ähnlichkeit gibt es verschiedene Verfahren. Eine gängige und stark verbreitete Methode ist die Berechnung des *Cosinus-Koeffizienten* (auch: *Cosine-Similarity*).

$$sim(d,q) = \cos(\theta) = \frac{\langle d,q \rangle}{\|d\| \ \|q\|} \tag{4.1}$$

Der Ähnlichkeitswert $sim(d,q)$ entspricht dem Schnittwinkel θ der beiden Vektoren. Der Zähler setzt sich aus dem Skalarprodukt der beiden Vektoren zusammen, der Nenner aus dem Betrag des Vektors. Es gilt:

$$\langle d,q \rangle = d * q \tag{4.2}$$

$$\|d\| = \sqrt{d * d} \tag{4.3}$$

Aufgelöst bedeutet dies:

$$blasim(d,q) = \cos(\theta) = \frac{\sum_{i=1}^{n} d_i \ q_i}{\sqrt{\sum_{i=1}^{n} d_i^2} \ \sqrt{\sum_{i=1}^{n} q_i^2}} \tag{4.4}$$

Die Vektoren bewegen sich im positiven Wertebereich, daher ergeben sich Ähnlichkeitswerte zwischen 0 und 1 als Maximum. Je kleiner der Winkel zwischen Anfrage- und Objektvektor ist, desto größer ist der Ähnlichkeitswert.

Bewertung und konzeptionelle Entscheidung

Prinzipiell kommen drei verschiedenen Systemtypen in Betracht. Dies sind kollaborative, inhaltsbasierte und hybride Recommender Systeme. Die Verwendung eines inhaltsbasierten Systems hat mehrere Vorteile. Das System kann sofort eingesetzt werden, ohne dass zunächst Informationen über das Verhalten von anderen Nutzern bekannt sein müssen. Lediglich das Profil des Nutzers wird benötigt. Die Empfehlung ist nicht vom Verhalten der Nutzer im System abhängig (z. B. welche Informationen von Nutzern abgerufen wurden). Eine explizite Bewertung von Objekten ist nicht notwendig, was die Bedienung vereinfacht. Die Qualität des Empfehlungsprozesses ist also von der Qualität der vergebenen Objekt-Attribute abhängig und der Qualität der Eingabedaten (dem Nutzerprofil).

Aspekte des Datenschutzes

Aus Sicht des Datenschutzes sind mehrere Aspekte zu berücksichtigen. Auf die Grundproblematiken wurde bereits in Abschnitt 2.3 (S. 22) eingegangen. Wichtig ist zunächst, dass die Nutzer Kontrolle über die Daten haben, die über sie gespeichert werden. Bei der Nutzung des MMG werden die Aktivitäten der Nutzer protokolliert. Diese Protokollierung erfolgt Client-seitig auf dem Gerät und ist zunächst nicht weiter zugänglich. Erst durch den Import in das System werden diese Daten weiterverarbeitet und die Daten dem System zugänglich gemacht. Hier ist zunächst das explizite Zutun der Nutzer notwendig, ansonsten wird das Protokoll nicht weiterverarbeitet. Beim Import liegen die Daten zwar persistent gespeichert im System vor, doch erfolgt zunächst keine weitere Auswertung. Der Nutzer erhält beim Import lediglich eine ID, mit der eine spätere Identifikation des gespeicherten Protokolls ermöglicht wird. Eine Rückschluss von den Protokoll-Daten auf die Nutzer ist nicht möglich. Der Nutzer erhält weiterhin die Kontrolle über die Daten und kann diese rückstandslos aus dem System entfernen, indem das Protokoll gelöscht wird. Die weitere Analyse und Verarbeitung der Daten wird erst beim Login und der Generierung von Empfehlungen ausgeführt.

5 Systementwurf

Im folgenden Kapitel werden die Anforderungen, die sich aus der Analyse ergeben haben und die entworfenen Lösungsstrategien in ein technisches Konzept überführt. Dies umfasst zunächst die Festlegung der Architektur. Als Grundlage für das System wird ein Datenmodell entworfen und die Struktur der Anwendungslogik festgelegt. Im letzten Teil wird die Benutzeroberfläche konzipiert.

5.1 Architektur

Bei der Architektur des Systementwurfs wird als Grundkonzept eine *Drei-Schichten-Architektur* (*Three-Tier-Architecture*) benutzt. Dieses Architektur-Muster trennt die verschiedenen Aspekte eines Systems logisch voneinander. Die *Datenhaltungsschicht* enthält die persistente Speicherung der Daten und ist für den Zugriff auf die Daten zuständig. Die Geschäftslogik mit den Verarbeitungskomponenten wird in der *Anwendungslogikschicht* gekapselt, während die *Präsentationsschicht* die Benutzerschnittstelle mit der Repräsentation der Daten darstellt.

Abbildung 5.1: Drei-Schichten-Architektur

Eine mehrschichtige Architektur ist Voraussetzung für eine verteilte Anwendung. Die Trennung der verschiedenen logischen Aspekte ermöglicht erst eine physische

Trennung auf verschiedene Systeme, was beispielsweise durch eine Client-Server-Anwendung umgesetzt werden kann. Daneben erlauben derartige Architekturen eine gute Skalierbarkeit, da die einzelnen Schichten unabhängig voneinander auf die Umgebung angepasst werden können, indem sie ihren benötigten Ressourcen entsprechend auf unterschiedliche Systeme verteilt werden.

Die Kapselung von zusammengehörigen Funktionalitäten innerhalb der einzelnen Schichten ermöglicht eine eigenständige Wiederverwendbarkeit der Funktionalitäten auf der einen Seite. Auf der anderen Seite erlaubt diese Aufteilung eine gute Wartbarkeit der einzelnen Aspekte und eine einfache Erweiterbarkeit, ohne in andere Komponenten des Systems eingreifen zu müssen.

5.2 Datenhaltungsschicht

Die Ebene der Datenhaltungsschicht steuert den Zugriff der anderen Komponenten auf die Daten und die persistente Vorhaltung der Daten. Die persistente Vorhaltung bedeutet in diesem Zusammenhang eine Speicherung der Daten unabhängig von der Laufzeit des Systems. Die Daten müssen unabhängig von den anderen Komponenten in strukturierter Form vorliegen. Dies bedeutet, dass diese nicht an die anderen Schichten des Systems gebunden und von diesen abhängig sind.

Diese Anforderungen werden durch den Einsatz von *relationalen Datenbanksystemen* erfüllt. Datenbanksysteme legen die Daten in einer *Datenbank* ab und ermöglichen den Zugriff und die Verwaltung über ein *Datenbankmanagementsystem* (DBMS). Relationale Datenbanksysteme nutzen das Konzept von Entitäten und Relationen und der Ausführung von Operationen auf den Relationen (*Relationale Algebra*). Eine Entität wird dabei durch eine Tabelle repräsentiert, die die Datensätze (*Tupel*) beinhaltet. Ein Datensatz besteht dabei aus verschiedenen *Attributen* und muss über Schlüsselattribute eindeutig adressierbar sein. Beziehungen zwischen Entitäten werden über die Relationen ermöglicht.

5.2.1 Datenmodell

Das zentrale Datenmodell bildet die Datenkomponenten des kompletten Systems ab. Die zentrale Element des Datenmodells wird durch die Entität *Tag* dargestellt. Diese bildet die Verbindung zwischen Datenmodell der Verwaltungsanwendung und dem mobilen multimedialen Museumsguide (MMG). Dieser wird durch seine Struktur (siehe 4.1/S. 33) im Datenmodell repräsentiert. Die Struktur setzt sich zusammen aus *Segmenten*, *Modulen* und *Kapiteln*. Die einzelnen Elemente des MMG werden durch Tags inhaltlich eingeordnet. Das gleiche gilt für *Inhaltselemente*, die ebenfalls durch Tags beschrieben werden. Die Inhaltselemente stellen Container dar, die mehrere

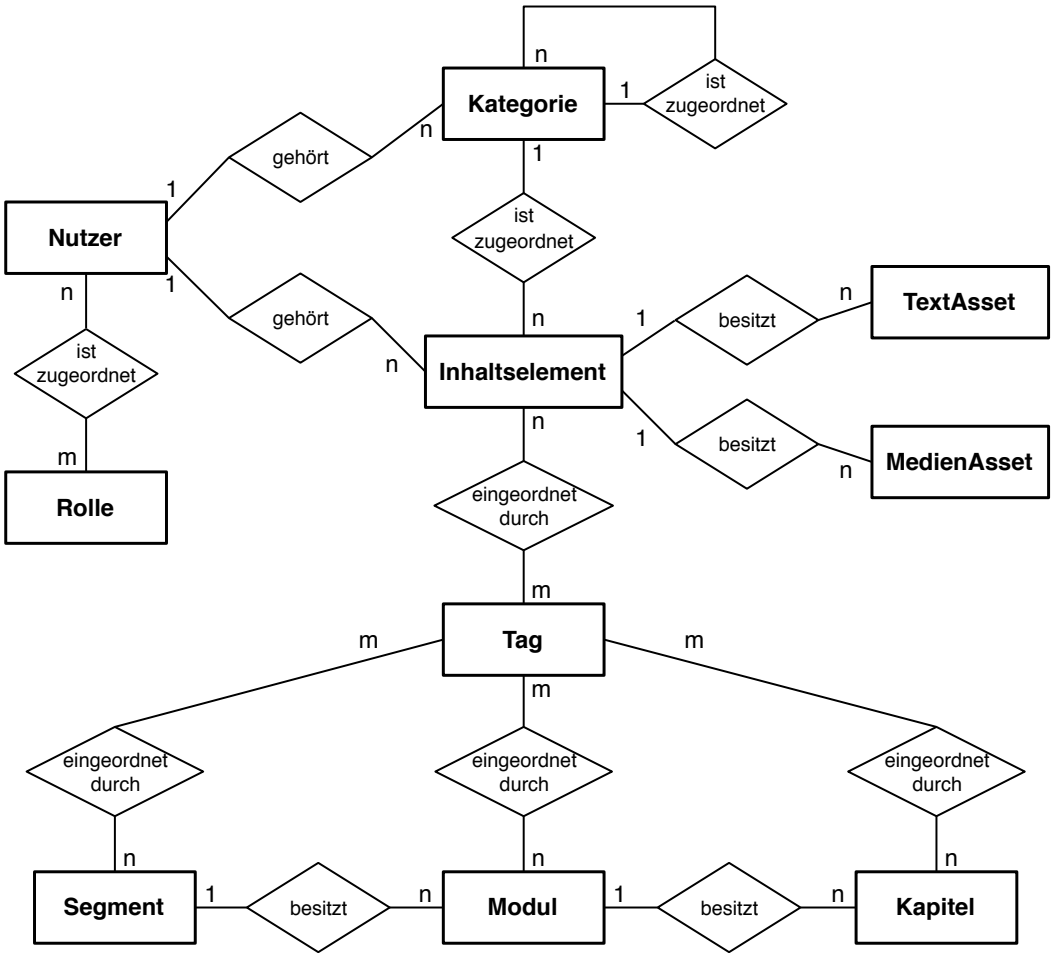


Abbildung 5.2: Zentrales Datenmodell in Chen-Notation (ohne Attribute)

TextAssets und *MedienAssets* enthalten können, was durch entsprechende Entitäten im Datenmodell umgesetzt wird. Die Inhaltselemente werden durch *Kategorien* strukturiert. Die Kategorien stellen dabei eine monohierarchische Struktur dar, in der jede Kategorie einer anderen Kategorie zugeordnet werden kann („Eltern-Kategorie“). Kategorien und Inhaltselemente gehören jeweils einem *Nutzer*. Den Nutzern werden eine oder mehrere *Rollen* zugeordnet.

Das Datenmodell ist in zwei Teile aufgeteilt, in den Teil des Informationspools und den der Struktur des MMG und dem damit verbundenen Besuchsprotokoll. Diese logische Trennung spiegelt sich auch auf der technischen Ebene wieder. Während die Daten der Verwaltungsanwendung in einer relationalen Datenbank persistent abgelegt werden, liegen die Besuchsprotokolle und die Struktur und Beschreibung des MMG bereits beim Import in XML-strukturierter Form vor und werden auch so belassen.

Datenmodell des Informationspools

Das Datenmodell des Informationspools stellt einen wesentlichen Teil der Datenstrukturen des Systems dar. Es wird an dieser Stelle gesondert besprochen, da dieses Modell in der Datenbank persistent abgelegt wird. Durch die Verwendung eines ORM liegen die Entitäten in gleicher Form auch als Domain-Klassen vor.

Zentrales Element im System sind die Inhaltselemente. Diese stellen ein Dokument mit Inhalten dar. Die Inhalte werden durch Assets dargestellt, wobei es verschiedene Typen von Assets gibt. Zentrales Element ist daher die Entität `ContentElement`. Diese repräsentiert ein Inhaltselement, das als Eigenschaften das Attribut `title` besitzt. Die Assets werden über die Entitäten `TextAsset` und `MediaAsset` repräsentiert. Ein `ContentElement` kann dabei n-`TextAsset` bzw. n-`MediaAsset` besitzen. Ein `TextAsset` stellt einen einfachen Texte dar, der durch einen Titel (Attribut `title`) und den Inhalt (Attribut `content`) beschrieben wird. Medien werden über die Entität `MediaAsset` dargestellt. Diese Entität enthält eine Title (Attribut `title`) und eine Referenz auf die Mediendatei (Attribut `file`). Die Mediendatei als solches wird nicht in der Datenbank gespeichert, sondern separat im Dateisystem abgelegt. Eine Besonderheit stellt das Attribut `sort` dar, das in beiden Asset-Entitäten vorhanden ist. Dieses ist für die Sortierung der Assets eines Inhaltselements notwendig. Die Sortierung erfolgt dabei über einfache `int`-Werte, wobei jeder Wert nur einmal pro Inhaltselement auftreten darf.

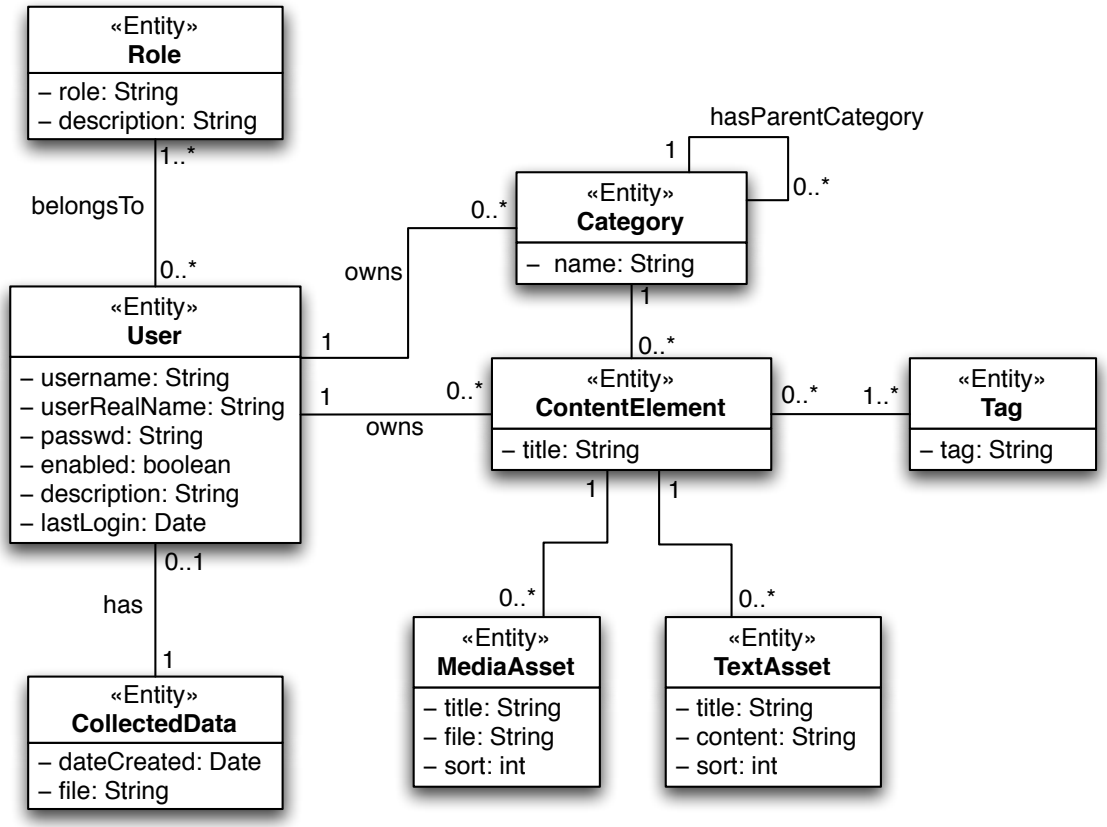


Abbildung 5.3: Datenmodell Inhaltsverwaltung in UML-Notation

Inhaltselemente werden über Tags inhaltlich eingeordnet. Im Datenmodell wird dies über die Entität `Tags` abgebildet, von der eine n:m-Relation zur Entität `ContentElement` besteht.

Die Inhaltselemente werden über eine monohierarchische Struktur eingeordnet. Diese Struktur wird durch Kategorien gebildet, die durch die Entität `Category` abgebildet werden. Die Kategorien können einer übergeordneten Kategorie zugeordnet sein, wobei jede Kategorie beliebig viele Unterkategorien besitzen darf. Umgesetzt wird diese Struktur durch eine Referenz auf die gleiche Entität. Besitzt eine Kategorie keine Referenz auf eine andere Ober-Kategorie, so gehört diese der obersten Hierarchie-Ebene an. Jeder Kategorie können beliebig viele Inhaltselemente zugeordnet werden.

Der Bereich der Benutzer und Rollen wird über die Entitäten `User` und `Role` im

Datenmodell abgebildet. Für die Anmeldung der Benutzer an das System gibt es das eindeutige Attribut `username` mit einem Attribut für das Passwort (`passwd`), von dem aus Sicherheitsgründen nur ein Hash-Wert abgelegt wird, der mit dem Hash-Algorithmus MD5[1] erzeugt wird. Zusätzlich gibt es noch einige Beschreibungsattribute (Realname, Beschreibung, Zugang aktiviert, letzter Login). Ein Benutzer kann mehrere Rollen besitzen, dies wird durch die Beziehung zwischen `User` und `Role` abgebildet. Durch die Anforderungsanalyse sind drei Rollen vorgegeben:

`ROLE_EDITOR`	Redakteur, Verwaltung der Inhalte
`ROLE_ADMIN`	Administrator, erweiterte Privilegien zum Redakteur: Zuordnung von Tags zu Datenstrukturen des MMG, Benutzer-Verwaltung
`ROLE_USER`	Normaler Benutzer

Das Rollenmodell stellt im jetzigen Zustand ein Basiskonzept dar und kann durch die Einführung neuer Rollen einfach verfeinert werden, ohne in die Datenstrukturen eingreifen zu müssen. Siehe dazu auch Abschnitt 6.4.1 (S. 74).

Benutzer in der Rolle `ROLE_USER` werden über die Entität `CollectedData` erweitert. Diese Entität enthält eine Referenz auf eine Datei und ein Datum der Erstellung des Datensatzes. Die referenzierte Datei liegt im Dateisystem und enthält das während eines Museumsbesuches vom MMG aufgezeichnete Besuchsprotokoll.

Datenmodell des Besuchsprotokolls und MMG

Auf die Struktur der vom MMG geschriebenen Log-Dateien (Besuchsprotokoll) und der Konfigurationsdateien wurde bereits in Abschnitt 4.1.2 (S. 36) eingegangen, daher sollen an dieser Stelle nur die für das System relevanten Datenstrukturen betrachtet werden.

Die Daten des Besuchsprotokolls werden in einer eigenen XML-Datei persistent gespeichert (Teil A in Abbildung 5.4), ebenso wie die Struktur des MMG (Teil B in Abbildung 5.4).

Aus der in `Collected Data` referenzierten XML-Datei kann eine Datenstruktur extrahiert werden, die den Verlauf eines Besuches widerspiegelt. Ein Besuchsprotokoll wird durch die Entität `Visit` repräsentiert. Diese enthält Attribute zum Verlauf des

1 Message-Digest Algorithm 5

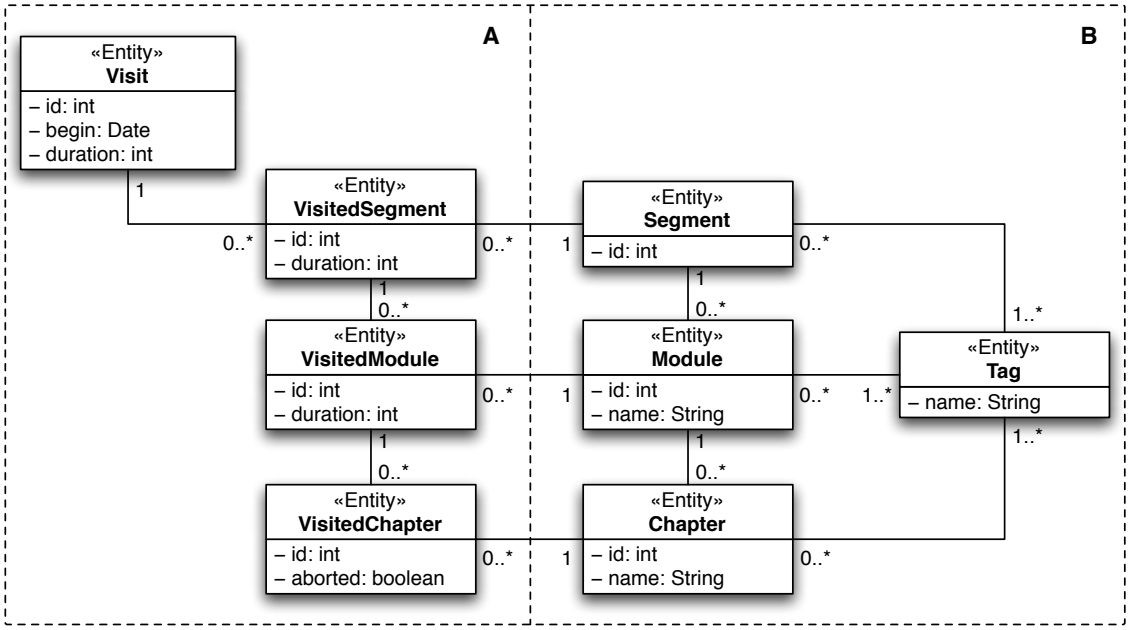


Abbildung 5.4: Aus den XML-Vorgaben extrahiertes Datenmodell in UML-Notation

Besuches (Beginn, Dauer in Minuten) und Referenzen zu den besuchten Segmenten (Entität `VisitedSegment`), die wiederum auf die besuchten Module (`VisitedModule`) und Kapitel (`VisitedChapter`) referenzieren.

Die Struktur des MMG wird beim Import in die Verwaltungsanwendung analysiert und nur die für das System relevanten Informationen gespeichert. Dafür wird eine Datenstruktur benötigt, die sich aus den Segmenten, Modulen und Kapiteln zusammensetzt. Diesen Elementen werden die Tags zur inhaltlichen Einordnung zugeordnet.

Die Objekte des Besuchsprotokolls besitzen über die ID eine Referenz auf die ihnen entsprechenden Objekte des MMG, wodurch sich die Tags auf die Objekte das Besuchsprotokoll zuordnen lassen,

Alle Entitäten werden durch entsprechende Domain-Klassen mit der gleichen Struktur (Attributen, Relationen) repräsentiert.

5.2.2 Datenzugriff

Wie bei der Diskussion des Datenmodells deutlich wurde, sind verschiedene Datenstrukturen zu berücksichtigen. Zum einen sind dies die Daten für die Verwaltungs-Applikation, deren Daten in einer relationalen Datenbank persistent abgelegt werden. Zum anderen sind dies die Daten für die Recommendation. Für diesen Bereich des Systems werden die Daten als Eingabedaten in einer XML-Struktur importiert und als Datei im Dateisystem abgelegt.

Die Implementierung des Systems erfolgt in einer objektorientierten Sprache, da das Datenmodell jedoch in einer relationalen Datenbank abgelegt wird, erfolgt der Zugriff über ein objektrelationales Mapping (ORM). Dieses ermöglicht die Abbildung einer objektorientierten Struktur auf eine relationale Datenbank. Für die Anwendung ist dabei der Zugriff transparent und scheinbar objektorientiert, da das ORM als Zwischenschicht zwischen persistenten Model-Klassen und der Datenbank existiert. An dieser Stelle übersetzt es die Speicher- und Ladeoperation in den Dialekt der Datenbank (*SQL*). Der Vorteil bei der Verwendung eines ORM liegt in dem einheitlichen Zugriff. Es ist keine weitere Sprache für den Datenzugriff im Quellcode notwendig, dies erleichtert die Pflegbarkeit. Eine weiterer Vorteil liegt in der Unabhängigkeit der Anwendung vom verwendeten DBMS, da dieses nicht direkt aus der Anwendung, sondern nur über die Zwischenschicht angesprochen wird.

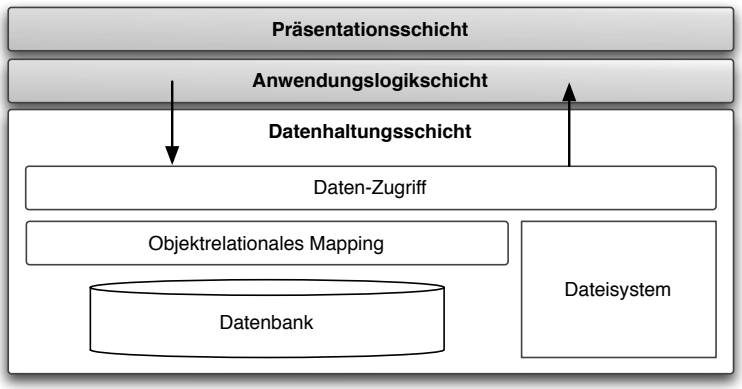


Abbildung 5.5: Datenhaltungsschicht

5.3 Anwendungslogikschicht

Das System umfasst zwei Hauptfunktionalitäten: Die Administration der Daten und den Abruf der Daten, wobei mit dem Abruf auch die Generierung von Empfehlungen verbunden ist. Diese beiden Hauptfunktionalitäten sind getrennt voneinander zu betrachten. Die Administration umfasst die Benutzerverwaltung, Verwaltung der Inhalte und Konfiguration der Tag-Zuordnungen als Vorbereitung der Datenbasis für eine Empfehlung.

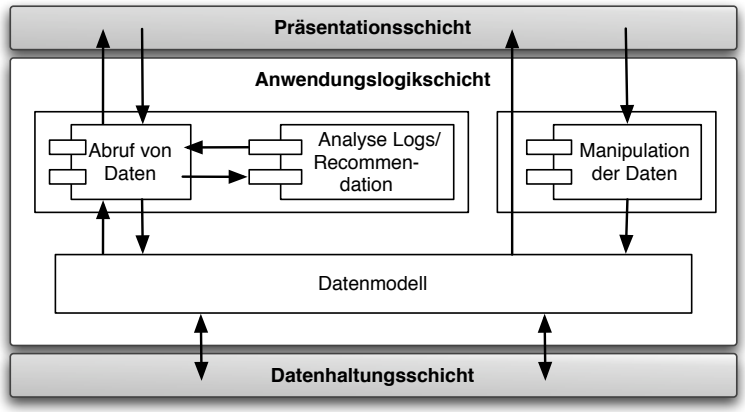


Abbildung 5.6: Anwendungslogikschicht

Diese logische Trennung der Funktionalitäten spiegelt sich auch weitgehend in der Aufteilung der Komponenten des Systems wieder. Dabei ist zu beachten, dass es ein zentrales Datenmodell gibt, das Basis für die verschiedenen Komponenten ist. Das Datenmodell stellt die Verbindung zwischen Anwendungslogik- und Datenhaltungsschicht dar.

5.3.1 Anwendungslogik Verwaltungs-Applikation

Die Anwendungslogik der Verwaltungs-Applikation unterteilt sich in mehrere Controller, die die Verwaltung der verschiedenen Aspekte der Datenhaltung steuern. Die Controller nehmen die Anfragen (HTTP-Requests) entgegen die von der Benutzeroberfläche ausgelöst wurden und kontrollieren die Verarbeitung dieser.

Die Anmeldung an das System wird vom `LoginController` verarbeitet. Dieser steuert das Erzeugen eine User-Session, beim Abmelden übernimmt der `LogoutController`

das Entfernen dieser Session. Zur Verarbeitung von Kategorien, Inhaltselementen, Text- und MedienAssets, sowie den Tags existieren eigene Controller, die jedoch alle nach dem gleichen Muster arbeiten. Sie übernehmen die Anfrage auf Anzeige Erstellung, oder Manipulation von Elementen und reichen diese an die Datenbasis weiter. Die Trennung in separate Controller erlaubt das unabhängige Steuern einzelner Elemente. Dazu werden die Controller aufgeteilt in sie Klassen:

- `CategoryController`
- `ContentElementController`
- `TextAssetController`
- `MediaAssetController`
- `TagController`

5.3.2 Anwendungslogik Frontend-Applikation

Daten-Import Der Import der Protokoll-Daten in das System wird von einem eigenen Controller gesteuert. Hierfür ist die Controller-Klasse `ImportDataController` zuständig. Sie übernimmt die Daten und überprüft diese. Dazu wird die empfangene Zeichenkette in eine Datenstruktur überführt. Die Datenstruktur wird als Model von Instanzen der Klasse `Visit`, mit den Unterklassen `VisitedSegment`, `VisitedModule` und `VisitedChapter` dargestellt. Die Logik für die Überführung der Zeichenkette in eine Datenstruktur wird in einer eigenen Klasse `Xml` gekapselt. Wenn die vom Controller empfangenen Daten validiert werden konnten, werden diese im Dateisystem abgelegt und der Controller veranlasst die Generierung einer ID, was über eine Instanz der Hilfsklasse `MemorableID` geschieht.

Daten-Abruf Die Anwendungslogik des Daten-Abrufs wird durch die Controller-Klasse `QrypsController` gesteuert. Die einzelnen Komponenten liegen dabei als Module vor. Dazu notwendige Komponenten sind ein Modul zur Analyse der Profildaten und ein Modul zur Generierung einer Empfehlung

Das Modul zur Analyse der Profildaten wird durch die Klasse `Evaluation` repräsentiert. Es benötigt die Daten eines Besuchs (`Visit`-Objekt) zur Generierung eines Anfragevektors. Der Anfragevektor wird als Objekt der Model-Klasse `TagList` erzeugt.

Mit dem erzeugten Objekt des Anfragevektors `TagList` kann der Controller dann eine Instanzen der Klasse `CosineSim` erzeugen, über die die Vergleiche zwischen Inhaltselementen mit ihrem Beschreibungsvektor und dem Anfragevektor durchgeführt werden.

Daneben ermöglicht es der `QrypsController` einzelne Inhaltselemente direkt aus dem Datenhaltung zu extrahieren und darzustellen und eine Übersicht dieser zu erzeugen.

5.4 Präsentationsschicht

Die Präsentationsschicht stellt die Benutzerschnittstelle für die Interaktion mit den Anwendern bereit und ist dabei von der Anwendungslogikschicht entkoppelt. Diese Trennung ermöglicht eine unabhängige Erweiter- und Wartbarkeit der verschiedenen Schichten, da Änderungen in der Präsentation keine Auswirkungen auf die Anwendungslogik haben.

5.4.1 Benutzeroberfläche

Bei der Diskussion des Bedienkonzeptes müssen zwei unterschiedliche Oberflächen betrachtet werden. Zunächst wird die Benutzerschnittstelle der Verwaltungs-Applikation, dann die der Frontend-Applikation diskutiert. Design-Ziel der Benutzeroberfläche ist, allen Benutzern ein möglichst intuitives Bedienkonzept zu bieten, das möglichst geringe Vorkenntnisse und Schulungen voraussetzt. Dies betrifft insbesondere die Frontend-Anwendung. Darüber hinaus wird Wert darauf gelegt, die Navigationsstrukturen möglichst offen und nicht verschachtelt zu halten, so dass ein direkter Zugriff auf die wichtigsten Bedienelemente direkt von allen Unterseiten aus möglich ist.

Benutzeroberfläche Verwaltungs-Applikation

Das Layout-Konzept der Verwaltungs-Applikation kann zunächst in zwei Teile unterteilt werden. Im Kopfbereich sind die grundlegenden Navigationselemente für Funktionen wie Hilfe, Logout etc. zu finden. Dort ist ebenfalls die erste Navigationsebene untergebracht, die mit Hilfe von Tabs realisiert ist. Dies bietet einen einfachen Einstieg in die drei Hauptfunktionalitäten der Anwendung: Verwaltung

der Inhalte, Benutzerverwaltung und Verwaltung der Tag-Zuordnung. Für Benutzer mit eingeschränkten Rechten können Tabs ausgeblendet werden. Der Haupt-Bereich bleibt von seiner Aufteilung her bei allen Navigationspunkten gleich. Hier wurde ein 3-spaltiges Layout gewählt, um alle wichtigen Funktionen stets auf einer Bildschirmseite zu halten. Die linke Spalte dient als erweiterte Navigation, mit der das zu bearbeitende Objekt ausgewählt wird. Die Darstellung dieser Objektauswahl ist jedoch bewusst flexibel gehalten, um den Anforderungen der jeweiligen Funktionalität gerecht zu werden. Im Bereich der Inhaltsverwaltung werden in der Objektauswahl die Kategorien angezeigt. Dies wird über eine Baumstruktur umgesetzt, um auch verschachtelte Kategoriestrukturen darstellen zu können. In den anderen Bereichen (Benutzerverwaltung, Tag-Mapping) wird in der Objektauswahl eine einfache Liste dargestellt. Um alle wichtigen Elemente gleichzeitig auf dem Bildschirm darstellen zu können werden sämtliche Dialoge über Popup-Fenster bzw. Overlays realisiert. Dazu gehören auch Eingabe- und Bearbeitungsmasken.

Mach Auswahl im Kategoriebaum wird ein Inhaltselement mit allen zugehörigen Assets dargestellt. Die Assets können sortiert und bearbeitet werden, dem Inhaltselement können Tags zugeordnet werden. Dies geschieht in der rechten Spalte, wo auch eine Liste der Tags dargestellt wird.

Bei der Benutzerverwaltung werden in der rechten Spalte (Objektauswahl) die Rollen, im Mittelteil eine Liste der Benutzer angezeigt. Die Benutzer können über eine Detailansicht direkt angewählt werden, die Details werden in der rechten Spalte angezeigt. Bei Frontend-Benutzern wird dort auch das aus einem Besuch ermittelte Profil angezeigt (s.u.).

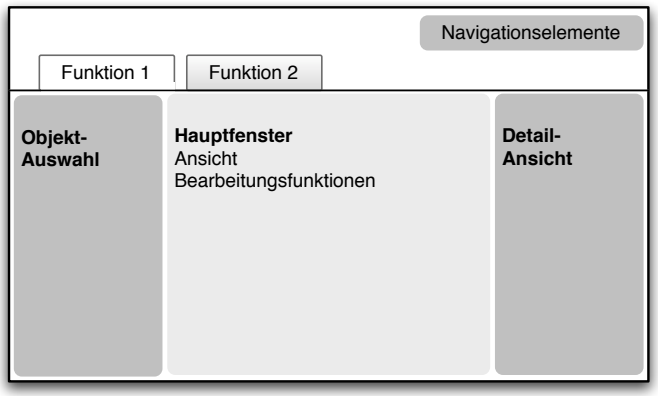


Abbildung 5.7: Layout-Konzept Verwaltungs-Applikation

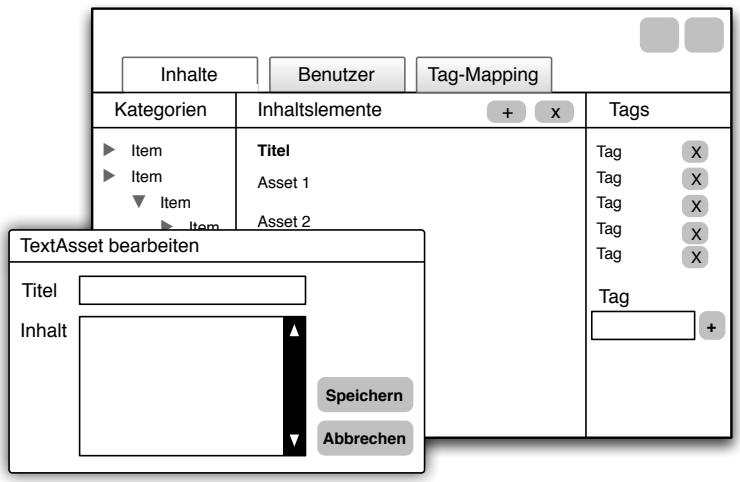


Abbildung 5.8: Skizze Verwaltungs-Applikation

Bei der dritten Funktionalität, der Zuweisung von Tags zur Inhaltsstruktur des MMG wird über die Objektauswahl die zu bearbeitende Version ausgewählt bzw. eine der vorhandenen Versionen als Produktiv-Version (Version, die für die Empfehlung im Frontend benutzt wird) gesetzt. Die Produktiv-Version kann dabei nicht direkt bearbeitet werden, um Inkonsistenzen einer Version vorzubeugen, die gerade bearbeitet wird. Die zu bearbeitende Version wird im Hauptfenster gemäß ihrer Struktur und den Strukturelementen zugeordneten Tags dargestellt. Nach Auswahl eines Strukturelements findet die Bearbeitung der Tags in der rechten Detailansicht statt (analog zur Bearbeitung von Tags bei Inhalts-Elementen).

Benutzeroberfläche Frontend-Applikation

Der Aufbau der Frontend-Applikation muss gesondert betrachtet werden, da er sich von dem der Verwaltungs-Applikation konzeptionell unterscheidet. Zwar gelten auch hier die gleichen Ziele von Bedienkonzept und Design, doch ist die Flexibilität bei Gestaltung beim Bedienkonzept größer. Das Layout-Konzept sieht einen großen Navigationsbereich vor, um auch eine komplexe Seitenstruktur darstellen zu können. Dabei kann im Navigationsbereich entweder eine Sitemap mit allen Inhalten angezeigt werden, was dem Kategoriebaum mit den Inhaltselementen entspricht. Im Normalfall werden jedoch die generierten Empfehlungen für Inhaltselemente dargestellt. Die Empfehlungen sind dabei nach Segmenten aufgeteilt, wobei die Segmente chronologisch anhand ihrer ID sortiert werden. Die Gewichtung der Interessen an den den einzelnen Segmenten wird durch die Darstellung der einzelnen Segmente visualisiert. Hierzu wird die Breite der Segmente an die Gewichtung der Segmente

angepasst, ebenso die Anzahl der angezeigten Inhaltselemente pro Segment. Um die Chronologie deutlich machen zu können, sind die Segmente horizontal nebeneinander angeordnet, um diese horizontale Navigation nicht zu durchbrechen wird dies auch bei der Darstellung der Sitemap beibehalten. Unterhalb der Navigationsebene wird das Inhaltselement mit den Assets angezeigt, in der rechten Spalte sind weiterführende Informationen verfügbar. Zum einen sind dies dem angezeigten Inhaltselement ähnliche Inhalte, zum anderen eine Visualisierung des Interessen-Profils.

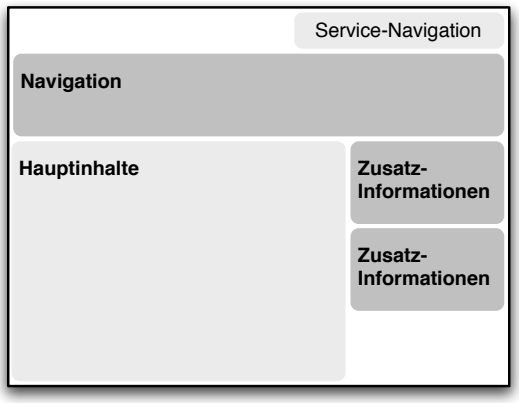


Abbildung 5.9: Layout-Konzept Frontend-Applikation

Tag-Cloud

Wie unter Abschnitt 4.3.1 (S. 50) beschrieben, wird das Interessen-Profil als Schlagwort-Vektor, der aus der Menge aller Tags und ihrer Gewichtung besteht, beschrieben. Dieser wird über eine *Tag-Cloud* (dt. *„Schlagwortwolke“*) visualisiert. Dazu wird die Gewichtung der einzelnen Tags veranschaulicht, indem jedes Tag mit einer der Gewichtung entsprechend angepassten Schriftgröße dargestellt wird. Die Sortierung der Tags erfolgt alphabetisch. Auf diese Weise sind Interessen-Schwerpunkte sehr schnell zu erfassen.

Abbildung 5.10: Beispiel Tag-Cloud

6 Implementierung

Das in der Anforderungsanalyse geforderte System wurde im Systementwurf entwickelt und konkretisiert. Im folgenden Kapitel wird die prototypische Umsetzung und Implementierung beschrieben. Dabei wird zunächst auf die technischen Rahmenbedingungen und die Laufzeitumgebung eingegangen und dann relevante Aspekte der Implementierung beleuchtet. Diese setzen sich zusammen aus den wichtigsten Aspekten der Anwendungslogik, der Datenhaltung und den verwendeten Technologien bei der grafischen Benutzeroberfläche bzw. deren Integration in die Applikation.

6.1 Laufzeitumgebung

Die Implementierung ist mit dem Web-Framework *Grails* in der Programmiersprache *Groovy* realisiert (vgl. 4.2.1/S. 41). Dabei wird auf Seiten des Web-Frameworks die Version Grails 1.1, auf Seiten von *Groovy* die Version 1.6.0 benutzt.[1] Der Groovy-Code wird bei der Kompilierung in Java Bytecode übersetzt, daher wird eine Java-Laufzeitumgebung benötigt. Hier kommt *Java SE* Version 5 zum Einsatz.[2] Als Laufzeitumgebung der Applikation wird außerdem ein Java-Servlet-Container benötigt, hier wurde der Webserver und Servlet-Container *Jetty* in der Version 6.4 [UrlJetty] benutzt, da dieser standardmäßig in Grails integriert ist. Die Verwendung anderer Servlet-Container ist jedoch problemlos möglich.

1 Die verwendeten Versionen waren zum Zeitpunkt der Implementierung (März 2009) die jeweils aktuellsten Versionen.

2 Zum Zeitpunkt der Implementierung war schon seit geraumer Zeit die aktuelle Java-Version 6 verfügbar. Die Implementierung erfolgte jedoch mit der vorherigen Version, da Apple für ältere Hardware (32-Bit-Prozessoren) die aktuellen Java-Versionen nicht bereitstellt. Nachteile entstanden in Folge dessen jedoch nicht.

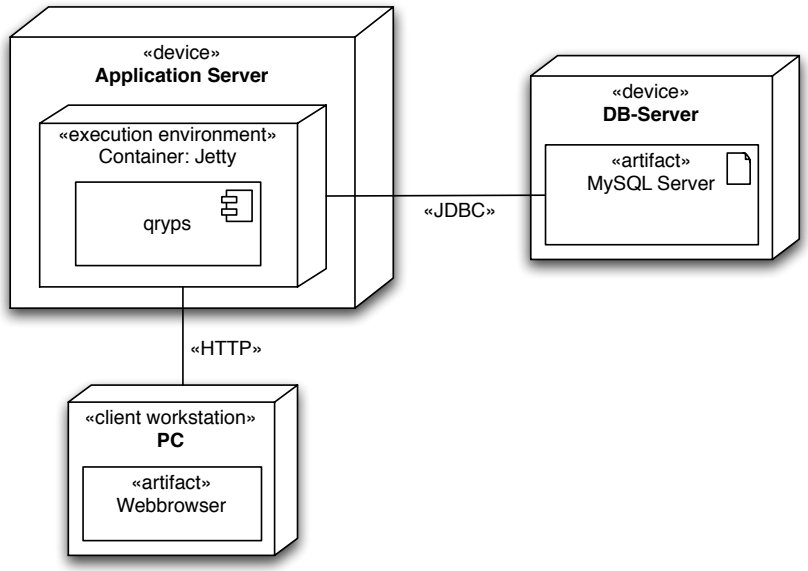


Abbildung 6.1: Deployment-Diagramm des Systems

Die persistente Speicherung des Datenmodells erfolgt teils im Dateisystem, teils in einer relationalen Datenbank. Hier kommen verschiedene Datenbanksysteme in Frage, die von Grails nativ unterstützt werden (z. B. MySQL, Oracle, Microsoft SQL Server). In der prototypischen Umsetzung wurde das Datenbanksystem *MySQL Community Server* Version 5.1 [UrlMysql] verwendet.

6.2 Struktur der Applikation

Die Struktur des Projektes wird in erster Linie vom verwendeten Web-Framework *Grails* vorgegeben. Projekte in Grails unterliegen festen Konventionen, was die Verzeichnisstruktur anbelangt. Die eigentliche Anwendung liegt dabei im Ordner `grails-app`, daneben gibt es Ordner für externe Bibliotheken, die eingebunden werden (`lib`) und Quellcode, der im Scope der Anwendung liegt (`services`) oder als externe Komponente eingebunden wird (`src/java` bzw. `src/groovy`). Der Ordner `web-app` enthält statische Dateien, wie beispielsweise Bilder, JavaScript- und CSS-Dateien und für das *Deployment* notwendige Konfigurationsdateien (z. B. *Application-Context*). Er wird daher in das beim Deployment erstellte *Web Application Archive* (WAR-Datei) integriert.

```
+ grails-app/
    + conf/
    + controllers/
    + domain/
    + i18n/
    + services/
    + taglib/
    + util/
    + views/
+ lib/
+ src/
    + groovy/
    + java/
+ web-app/
    + META-INF/
    + WEB-INF/
    + css/
    + files/
    + images/
    + js/
```

In der Verzeichnisstruktur der eigentlichen Anwendung spiegelt sich das in Grails verwendete MVC-Pattern wieder. So enthält der Ordner `grails-app` entsprechende Unterordner für Model (`domain`), View und Controller. Im Ordner `conf` finden sich die Konfigurationsdateien des Projektes, wie die Einstellungen für Datenquellen, Zugriffsschutz und allgemeine Projekteinstellungen.

Paketstruktur

Die Klassen der Applikation sind gemäß ihrer Funktionalität und Zugehörigkeit in Pakete strukturiert. Zum einen ist dies die Frontend- und Verwaltungsapplikation, zum anderen die Module. Die einzelnen Module spiegeln sich in den Paketen wieder. Die Pakete sind im einzelnen:

`org.qryps`	Basis-Paket
`org.qryps.app`	Klassen der Frontend- und Verwaltungs-Applikation
`org.qryps.emika`	Datenstrukturen EMIKA-MMG und Module zum Zugriff
`org.qryps.engine`	Klassen der Recommender-Engine-Komponente
`org.qryps.visit`	Datenstrukturen Besuchs-Protokoll und Module zum Zugriff
`org.qryps.util`	Verschiedene, allgemeine Hilfsklassen

6.3 Datenhaltung

Die persistente Datenhaltung besteht aus zwei Teilen. Zum einen aus dem Bereich der Verwaltungsanwendung, deren Daten in einer relationalen Datenbank gespeichert werden, zum anderen aus dem Bereich der Benutzerdaten, die in XML-Dateien im Dateisystem vorgehalten werden.

Die Domain-Klassen der Verwaltungs-Anwendung werden über ein ORM[1] auf die Datenbank abgebildet. Realisiert wird dies mit dem Grails-eigenen ORM *GORM*, das auf Hibernate aufsetzt und die entsprechende Funktionalität im Grails-Framework zur Verfügung stellt. Die Anbindung von GORM an das relationale Datenbanksystem wird über die Datenbankschnittstelle *JDBC* (*Java Database Connectivity*) unter Verwendung des JDBC/MySQL-Treibers *MySQL Connector/J 5.1* [UrlConnectJ] umgesetzt.

Listing 6.1: GORM Datenbank-Konfiguration (`DataSource.groovy`)

```
environments {
  development {
    dataSource {
      pooled = true
      driverClassName = "com.mysql.jdbc.Driver"
      username = "root"
      password = ""
      dbCreate = "update"
```

1 Object-Relational Mapping

```
9         url = "jdbc:mysql://localhost:3306/qryps"
10      }
11    }
12    [...]
13  }
```

Im Gegensatz zur üblichen Verwendung von Hibernate ist GORM weitgehend konfigurationsfrei, hier wird die Umsetzung des Ansatzes *Convention over Configuration* von Grails deutlich. Neben der Konfiguration der Datenbankschnittstelle sind für die Standard-Einstellung keine weiteren Schritte zwingend notwendig. Anders als bei Hibernate wird kein manuelles Mapping zwischen den Klassen und Datenbanktabellen benötigt, dieses wird direkt aus den Domain-Klassen abgeleitet. Dazu ist es erforderlich die Datentypen statisch zu typisieren, anders als normalerweise Groovy-Klassen der Fall ist. Zusätzliche Mapping-Informationen wie beispielsweise standardmäßige Sortierungen für das Auslesen der Domain-Objekte werden direkt in der Klasse festgelegt.

Listing 6.2: Domain-Klasse `Tag.groovy`

```
1  package org.qryps.app
2
3  class Tag {
4    String name
5
6    static belongsTo = ContentElement
7    static hasMany = [contentElements: ContentElement]
8
9    static constraints = {
10     name(blank: false, unique: true)
11   }
12
13   static mapping = {
14     sort 'name'
15   }
16 }
```

Sogenannte *Constraints* in den Domain-Klassen definieren Validierungs-Regeln. Objekte können nur in die Datenbank geschrieben werden, wenn die Regeln erfüllt sind. Dazu implementiert GORM eine `validate`-Methode, mit der die Validierung überprüft wird.

Das Lesen von XML-Dateien über die Implementierung des SAX[1]-Parsers in Groovy *XMLSlurper*. Dieser erlaubt das sequentielle Parsen von XML-Strukturen.

Listing 6.3: Beispiel XMLSlurper

```
def results = new XmlSlurper().parse(file);

results.segment.each { seg ->
  Segment segment = new Segment(
    id: seg.@id.toInteger(),
    tags: new ArrayList<String>()
  );

  seg.tag.each { tag ->
    segment.tags.add(tag);
  }
}
```

6.4 Anwendungslogik

Die wichtigsten Bereiche der Anwendungslogik umfassen die Benutzerverwaltung, sowie den Import von Profil-Daten in das System, sowie die Verarbeitung dieser Daten. Unter der Verarbeitung wird die Vorbereitung der Daten für das Recommender System durch Analyse der Daten und die Generierung der eigentlichen Empfehlung zusammengefasst.

6.4.1 Benutzerverwaltung

Die Umsetzung des Zugriffsschutzes auf Teile der Applikation erfolgt über eine rollenbasierte Benutzerverwaltung. Wie im Systementwurf beschrieben, existieren dabei die drei Benutzerrollen *Redakteur* (`ROLE_EDITOR`), *Administrator* (`ROLE_ADMIN`) und *normaler Nutzer* (`ROLE_USER`). Die Rollen und die Benutzer sind persistent in der Datenbank abgelegt (s. Abschnitt 5.2.1/S. 57). In der Implementierung werden die Benutzer durch die Domain-Klasse `User`, die Benutzerrolle durch die Domain-Klasse `ROLE` umgesetzt.

1 Simple API for XML

Die Realisierung erfolgt über das *Spring Security Framework*, das als Plugin für das Grails-Framework verfügbar ist (*Spring Security Plugin*, [UrlGraSec]). Dieses stellt Security Funktionen über Servlet-Filter bereit. Dies umfasst die Authentifizierung, also Bestätigung der Identität und Überprüfung auf Korrektheit, und Autorisation über Zuweisung von entsprechenden Rechten in Form einer Rolle, sowie den Zugriffsschutz. Mit dem Zugriffsschutz wird sichergestellt, dass nur nur Nutzer mit den entsprechenden Rechten auf Ressourcen des Systems zugreifen dürfen. Neben der lokalen Benutzerverwaltung ermöglicht das Spring Security Framework auch die Anbindung an Verzeichnisdienste über LDAP[1] oder die Nutzung *Single-Sign-on*-Dienste, wie OpenID [UrlOpenID].

Die Implementierung des Zugriffsschutzes kann auf verschiedenen Ebenen erfolgen. Prinzipiell ist es möglich diesen auf Ebene der Views innerhalb der Templates zu realisieren, innerhalb der Services und Controller (über *Annotations* und *Access Control Lists*) und über sogenannte *Requestmaps* auf Basis der URL.

In der vorliegenden Applikation werden Requestmaps zum Schutz von Ressourcen vor unberechtigtem Zugriff verwendet. Diese können entweder in der Datenbank oder über eine Konfigurationsdatei definiert werden, was hier verwendet wird.

Listing 6.4: Requestmap in `SecurityConfig.groovy` (Auszug)

```
1  requestMapString = """
2    CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
3    PATTERN_TYPE_APACHE_ANT
4
5    /error*=IS_AUTHENTICATED_ANONYMOUSLY
6    /login*=IS_AUTHENTICATED_ANONYMOUSLY
7    /logout/**=IS_AUTHENTICATED_REMEMBERED
8    /admin/**=ROLE_EDITOR,ROLE_ADMIN
9    /admin/user/**=ROLE_ADMIN
10   /importdata/**=IS_AUTHENTICATED_ANONYMOUSLY
11   /qryps/**=ROLE_USER
12   /css/**=IS_AUTHENTICATED_ANONYMOUSLY
13 """
```

Jede Anfrage wird auf ihre Zugriffsberechtigung hin überprüft, bevor die Anfrage vom Controller verarbeitet wird. Dazu wird der Pfad der angefragten Ressource extrahiert und versucht, diesen in der Requestmap zu identifizieren und dann die dort zugeordnete Rolle mit der Rolle in der Session vergleichen. Die Rolle des

1 Lightweight Directory Access Protocol

angemeldeten Nutzers wird bei der erfolgreichen Authentifizierung in der Session abgelegt. Bei nicht angemeldeten Nutzern ist keine keine Session vorhanden, diese werden über die Rolle `IS_AUTHENTICATED_ANONYMOUSLY` identifiziert.

Kann die benötigte Rolle nicht in der Session identifiziert werden, so wird die Anfrage auf eine Fehler-Seite weitergeleitet. Der Vorteil der Verwendung von Requestmaps ist, dass alle Anfragen vor der eigentlichen Verarbeitung, also vor der Weiterleitung der Anfrage an den Controller, auf ihre Rechte hin überprüft werden. Dies ermöglicht es den Zugriffschutz außerhalb der Controller-Klassen zu implementieren. Alle Einstellungen diesbezüglich erfolgen an einer zentralen Stelle in der Datei `SecurityConfig.groovy` durch Anpassung des Attributs `requestMapString`. Dies hat zur Folge, dass die Wartbarkeit erheblich vereinfacht wird, da Änderungen des Zugriffsschutzes umgesetzt werden können, ohne tiefergehend in den Code eingreifen zu müssen. Auch kann der Zugriffsschutz für die einzelnen Ressourcen (komplette Controller, einzelne Controller-Actions oder auch Verzeichnisse und Dateien) spezifisch gesetzt werden.

Teilweise ist es erforderlich, einzelne Templates aufgrund der Rolle des Nutzers anzupassen. Dies wird über die Abfrage der Rolle direkt im Template realisiert. Das *Spring Security Plugin* stellt hierfür entsprechende Tags in einer Tag-Library zur Verfügung. Dies wird beispielsweise bei der Generierung der Navigation der Verwaltungs-Anwendung benötigt, diese soll bestimmte Funktionen nur für die Rolle `ROLE_ADMIN` anzeigen. Der Zugriffsschutz der entsprechenden Controller ist jedoch in der Requestmap realisiert, hier wird nur die View-Komponente entsprechend angepasst

Listing 6.5: Abfrage der Rolle im GSP-View (Auszug `_header.gsp`)

```
1  <li id="tab2li">
2    <a href="#" id="tab2"><em><g:message code="backend.contentadmin"
       /></em></a>
3  </li>
4  <g:ifAllGranted role="ROLE_ADMIN">
5    <li id="tab3li">
6      <a href="#" id="tab3"><em><g:message code="backend.useradmin"
         /></em></a>
7    </li>
8    <li id="tab4li">
9      <a href="#" id="tab4"><em><g:message code="backend.mapping"
         /></em></a>
10   </li>
11 </g:ifAllGranted>
```

6.4.2 Datenimport

Der Datenimport der Protokolldaten, die in Form einer XML-Datei vorliegen, wird über ein HTML-Formular ausgeführt. Die Verarbeitung des ausgelösten `POST`-Requests erfolgt dann über den `ImportDataController`. Dort wird der MIME[1]-Type (erlaubte MIME-Types sind `application/xml`, `text/xml`) und die Größe der übertragenen Daten überprüft und die XML-Struktur des DOM-Objektes validiert. Dann wird das DOM-Objekt im Dateisystem gespeichert.

Listing 6.6: Speicherung des DOM-Objektes (Auszug `XmlConf.groovy`)

```
File file = new File (filePath + fileName);

file.withWriter("UTF-8") { w ->
  w << "<?xml version=\"1.0\" encoding=\"UTF-8\" ?>\n"
  w << "<!-- " << comment << " -->\n"
  w << content
}
```

Die Speicherung der Datei erfolgt unter einem zuvor generierten Namen, der zugleich dem Nutzer als ID zum Login dient. Die ID wird dabei als zufällige Zeichenfolge erzeugt, die jedoch mittels eines eigenen Algorithmus nach einem bestimmten Muster erzeugt wird.

Listing 6.7: Generierung der ID

```
Random rand = new Random();
String id = new String();

static String [] consonants = { "B", "D", "F", "G", "H", "K",
  "L", "M", "N", "P", "R", "S", "T", "V", "W", "X"};
static String [] vowels = { "A", "E", "I", "O", "U" };

for (int $i=0; $i<length; $i++) {
  if ($i % 2 == 0) {
    id = id + consonants[rand.nextInt(consonants.length)];
  }
  else {
    id = id + vowels[rand.nextInt(vowels.length)];
  }
}
```

1 Multipurpose Internet Mail Extensions

Die Generierung der ID ist als Methode der Hilfsklasse `MemorableID` implementiert. Die ID setzt sich aus einer Aneinanderreihung von Konsonanten und Vokalen zusammen. Dabei beginnt die ID mit einem Konsonanten und einer daran anschließenden abwechselnden Reihenfolge von Vokalen und Konsonanten. Die Menge der Buchstaben ist dabei eingeschränkt, Buchstaben deren Laute ähnlich klingen oder die nicht eindeutig „aussprechbar“ sind, werden nicht berücksichtigt (z. B. *C*, *Z*). Dadurch entstehen Zeichenketten die „aussprechbar“ und dadurch relativ gut merkbar sind. Beispiele für derart erzeugte IDs sind *VAHEMABI* oder *KATEBINE*.

Beim Datenimport wird ebenfalls ein Benutzer mit der zugeordneten Rolle `ROLE_USER` angelegt. Der Benutzername entspricht dabei der ID, das Passwort wird nicht gesetzt, so dass ein Login alleine mit der ID ermöglicht wird. Die Profildaten liegen in der Datei `<ID>.xml`, wodurch diese eindeutig zugeordnet werden kann.

6.4.3 Datenverarbeitung

Die in das System importierten Daten müssen zunächst aufbereitet und analysiert werden, bevor sie für den Recommendation Prozess genutzt werden können.

Datenbasis und Eingabedaten

Um die Datenbasis und die Eingabedaten im Recommender Engine verwenden zu können ist eine gleichartige Struktur beider Daten notwendig. Wie unter 4.3.1 (S. 50) beschrieben erfolgt die inhaltliche Einordnung der Daten-Objekte mittels eine Tag-Vektors. Die Schlagworte besitzen dabei jeweils einen Wert, der sich aus der Gewichtung ergibt und die Richtung des Vektors beeinflusst. Die Werte sind dabei normalisiert und bewegen sich im Wertebereich 0–1.

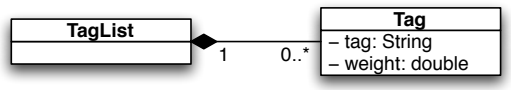


Abbildung 6.2: Klassen `TagList` und `Tag`

Implementiert ist die Beschreibung der Inhaltsobjekte und des Anfrageprofils über die Klasse `TagList`, die das öffentliche Attribut `tags` besitzt. Dieses repräsentiert einen Tag-Vektor und wurde daher über das *Collection*-Interface umgesetzt. Dazu wird die Implementierung `ArrayList` genutzt, da diese einen wahlfreien Zugriff und

eine Sortierung ermöglicht. Die `ArrayList`-Collection nimmt Objekte des Typs `Tag` auf.

Algorithmus zur Daten-Analyse

Der Beschreibungs-Vektor der Eingabedaten wird aus dem Besuchs-Protokoll gewonnen, das in XML-strukturierter Form vorliegt. Während die Datenbasis mit der Beschreibung der Objektdaten bereits vorliegt, muss der Beschreibungs-Vektor der Eingabedaten zunächst ermittelt werden. Die Analyse der Eingabedaten erfolgt beim Login in das System.

Grundsätzlich können jedem Objekt des MMG Tags zugeordnet werden, also Segmenten, Modulen und Kapiteln. Wie in Abschnitt 4.1.2 (S. 38) beschrieben werden zu den einzelnen Objekten unterschiedliche Informationen aufgezeichnet. Diese Informationen werden zur Ermittlung des Profils und der dort enthaltenen Gewichtung der einzelnen Objekte, respektive der einzelnen Tags, genutzt. Implementiert ist dies in der Klasse `Evaluation`, die mit den Profildaten eines Protokolls instantiiert wird.

Aufbereitung der Daten Zunächst ist es notwendig die Daten des Profils aufzubereiten, da die verschiedenen Elemente im Besuchsprotokoll ungeordnet vorliegen. Dies ist Folge davon, dass die Elemente des MMG mehrmals durchlaufen (Segmente) bzw. betrachtet (Kapitel, Module) werden können. Dazu werden zur Vereinfachung die Elemente gleichen Typs (also Segmente bzw. Module mit der gleichen ID) zusammengefasst. Um das mehrfache Auftreten in der Wertung zu berücksichtigen, werden die einzelnen Zeitspannen für gleiche Segmente/Module addiert. Doppelte Kapitel bleiben erhalten. Implementiert ist dies in der Methode `revise`.

Analyse-Algorithmus Die Analyse der bereinigten Protokoll-Daten wird in einem zweiten Schritt über die Methode `analyze` durchgeführt. Dazu werden die verschiedenen Elemente des Besuchs-Protokolls nach einem entwickelten Algorithmus gewichtet. Dieser Algorithmus verwendet verschiedene Schwellwerte, die an zentraler Stelle gewartet werden.

- *Segmente* werden nur einbezogen, wenn die Dauer des Besuchs des Segments den Schwellwert `segmentThreshold` überschreitet. Dies ist notwendig, um Segmente, die beispielsweise nur durchschritten und nicht näher beachtet werden, aus der Bewertung auszuschließen.

Die Gewichtung der den Segmenten zugeordneten Tags wird ermittelt durch den Anteil der Dauer des Besuchs des Segments an der Dauer des kompletten Besuchs. Dies wird dem festgelegten Faktor `segmentFactor` multipliziert.

- Für *Module* gilt ähnliches wie für Segmente. Hier gibt es ebenfalls eine Schwellwert für die Dauer der Betrachtung des Moduls (`moduleThreshold`). Wenn dieser unterschritten wird, fällt das Modul aus der Wertung heraus.

  Für die Gewichtung gibt es den Faktor `moduleFactor`.

- Bei *Kapiteln* gibt es eine zusätzliche Information im Protokoll, ob das jeweilige Kapitel komplett angeschaut oder vorher abgebrochen wurde. Im Falle eines vorzeitigen Abbruchs werden die Tags des Kapitels nicht in die Bewertung einbezogen, ansonsten geschieht das mit dem Faktor `chapterFactor`.

  Gleiche Kapitel können im Protokoll mehrfach auftreten und fließen entsprechend mehrfach bewertet in die Gesamt-Einordnung ein. Dies wird bewusst nicht bereinigt, um das mehrfache Betrachten eines Kapitels zu berücksichtigen.

- Als Ergebnis ergibt sich ein Tag-Vektor (mehrfach auftretende Tags werden aufaddiert), der das Profil eines Besuchs inhaltlich einordnet.

Listing 6.8: Profil-Analyse (vereinfacht; Auszug `Evaluation.groovy`)

```
%\begin{lstlisting}[caption=Profil-Analyse (vereinfacht; Auszug)]
visit.segments.each{ currentSegment ->
  // only if segment has modules or is above threshold
  if (currentSegment.duration > segmentThreshold || currentSegment.
     modules.size() != 0 ) {

    // process modules in visit segment
    currentSegment.modules.each { currentModule ->
      // only if module has chapter or is above threshold
      if (currentModule.duration > moduleThreshold || currentModule
         .chapter.size() != 0) {

        // process chapter in modules
        currentModule.chapter.each { currentChapter ->
          // but only if aborted is false
          if (!currentChapter.aborted) {

            // add tags with weight to profileVector
            profileVector.addTags(chapterFactor, confChapter.tags);
          }
        }

        // add tags with weight to profileVector
```

```
22          double moduleWeight = currentModule.chapter.size() /
                confModule.chapter.size() * moduleFactor;
23          profileVector.addTags(moduleWeight, confModule.tags);
24        }
25      }
26
27      // add tags with weight to profileVector
28      double segmentWeight = currentSegment.duration / totalDuration
            * segmentFactor;
29      profileVector.addTags(segmentWeight, confSegment.tags);
30    }
31  }
```

Die verwendeten Schwellwerte wurden für die Analyse des Protokolls wurden durch eine Untersuchung von Protokoll-Testreihen ermittelt, die vom EMIKA-Projekt zur Verfügung gestellt wurden. Hierbei wurden folgende Werte ermittelt (Werte in Sekunden):

- `segmentThreshold = 30`
- `moduleThreshold = 10`

Die Faktoren für die Gewichtung der Objekte wurden als sinnvolle Werte in Testreihen ermittelt:

- `segmentFactor = 4.0`
- `moduleFactor = 2.0`
- `chapterFactor = 1.0`

Recommendation Engine

Die eigentliche Generierung der Empfehlung ist als ein gekapseltes Modul realisiert. Dies hat den Vorteil, dass der Recommendation Engine einfach auf andere Anwendungsfälle übertragen werden kann, ohne dass an dieser Stelle weitere Änderungen nötig werden. Darüber hinaus wird so ermöglicht, den Recommendation Engine zu erweitern oder anzupassen, ohne in Teile des restlichen Systems eingreifen zu müssen.

Implementiert ist der Recommender Engine in der Klasse `CosineSim`. Diese wird mit einem Anfragevektor des Typs `TagList` instantiiert. Über die Methode `cosineSimilarity` kann dann ein Ähnlichkeitswert des übergebenen Tag-Vektors

mit dem Anfragevektor ermittelt werden. Die Funktionsweise des verwendete Algorithmus (*Cosinus-Koeffizient*) wird in Abschnitt 4.3.2 (S. 51) genauer beschrieben.

6.5 GUI

Das grafische Benutzerinterface setzt sich aus verschiedenen Komponenten zusammen, die kombiniert bzw. in das Basis-Layout eingebettet werden. Die Aufteilung in Komponenten ermöglicht die Wiederverwendung sowohl in der Verwaltungs- als auch in der Frontend-Applikation. Konzeptionell unterscheiden sich beide Teile aber im Benutzerinterface.

Die Aufteilung in verschiedene View-Komponenten wird durch das in Grails integrierte Layout-Framework *SiteMesh* [UrlSiteMesh] unterstützt. SiteMesh ermöglicht die Trennung der Inhalts-Komponenten von der Gestaltung und Platzierung im Layout. Dies bedeutet, dass einzelne Komponenten, wie z. B. Formulare oder Listen unabhängig von ihrer Gestaltung und Einbettung in das Layout gestaltet und als einfache HTML-Elemente generiert werden.

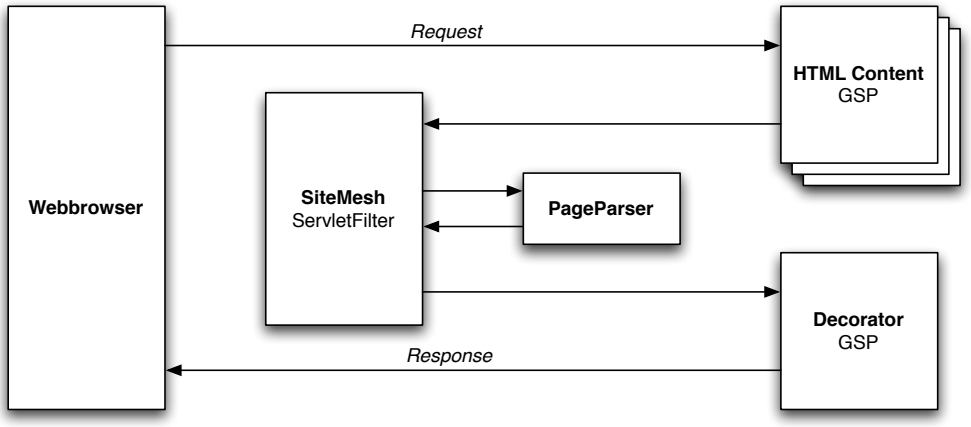


Abbildung 6.3: Funktionsweise von SiteMesh, nach [UrlWalnes, S. 19]

SiteMesh arbeitet als *Servlet-Filter*. Requests werden zunächst normal verarbeitet und das Ergebnis vom Template-System als normales HTML-Dokument mit einfachem Markup ohne weitergehende Layout-Informationen erzeugt. In einem zweiten Schritt wird der Response vom Servlet-Filter weiterverarbeitet und der *Body*-Teil des erzeugten Responses extrahiert. Dieser wird als Inhaltselement an den *Decorator* übergeben, der als Mapper die Inhalte in das Layout einbettet.

Listing 6.9: Sitemesh Layout-Template zur Einbindung von HTML-Code

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "
    http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

  <head>[...]</head>

  <body>
    <div class="page">

      <g:layoutBody />
      <g:render template="/qryps/footer" />

    </div>
  </body>
</html>
```

Listing 6.10: Festlegung des Layout-Templates im HTML-Header

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "
    http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

  <head>
    <meta name="layout" content="frontendSingle"></meta>
    <title><g:message code="frontend.title.import" /></title>
  </head>

  <body>
    <h1><g:message code="importData.title" /></h1>
    <p><g:message code="importData.textUpload" /></p>
    [...]
  </body>
</html>
```

Als Platzhalter für die Einbindung des Codes dient im Layout-Template das Tag `<g:layoutBody />`. Dieses wird bei der Generierung des Ausgabedokuments durch den eingebundenen Code ersetzt. Das vom Decorator zu verwendende Layout ist in Grails per Konvention festgelegt, kann alternativ aber auch über ein Meta-Tag im HTML-Header bestimmt werden (`<meta name='layout' content='frontendSingle'></meta>`). Für den Request und die aufrufende URL arbeitet SiteMesh transparent, die URL ändert sich nicht durch die Verwendung.

Bei Grails liegen die Layouts standardmäßig im Ordner `grails-app/views/layouts`. Insgesamt werden in der Applikation drei verschiedene Layout-Templates verwendet

und zwar für ein Einzelfenster (`Single.gsp`), die Frontend- (`frontendMain.gsp`) und die Verwaltungs-Applikation (`adminMain.gsp`).

(a) Login-Fenster **(b)** Daten-Import

Abbildung 6.4: Layout Single-Screen

6.5.1 Frontend-Applikation

Die Frontend-Applikation besteht aus einem Grundlayout mit der grundlegenden Struktur, in das verschiedene Komponenten geladen werden. Die Hauptnavigation ist zugleich das Ergebnis der generierten Empfehlung. In einer horizontalen Anordnung werden die Segmente des Museumsbesuches dargestellt, die in der Profil-Analyse als die Interessantesten ermittelt wurden. Dabei sind die Segmente anhand ihrer Abfolge im Museum geordnet, um den Besuch visuell darzustellen. Die Gewichtung der dargestellten Segmente untereinander wird über die Breite ihrer Darstellung illustriert. Dazu wird die Gewichtung in Relation zu den anderen Segmenten gesetzt, allerdings besteht eine Mindest-Breite um auch schwach gewichtete Elemente anzeigen zu können. Die Anzahl der aufrufbaren Inhalte innerhalb der dargestellten Elemente ist ebenfalls von der Gewichtung abhängig.

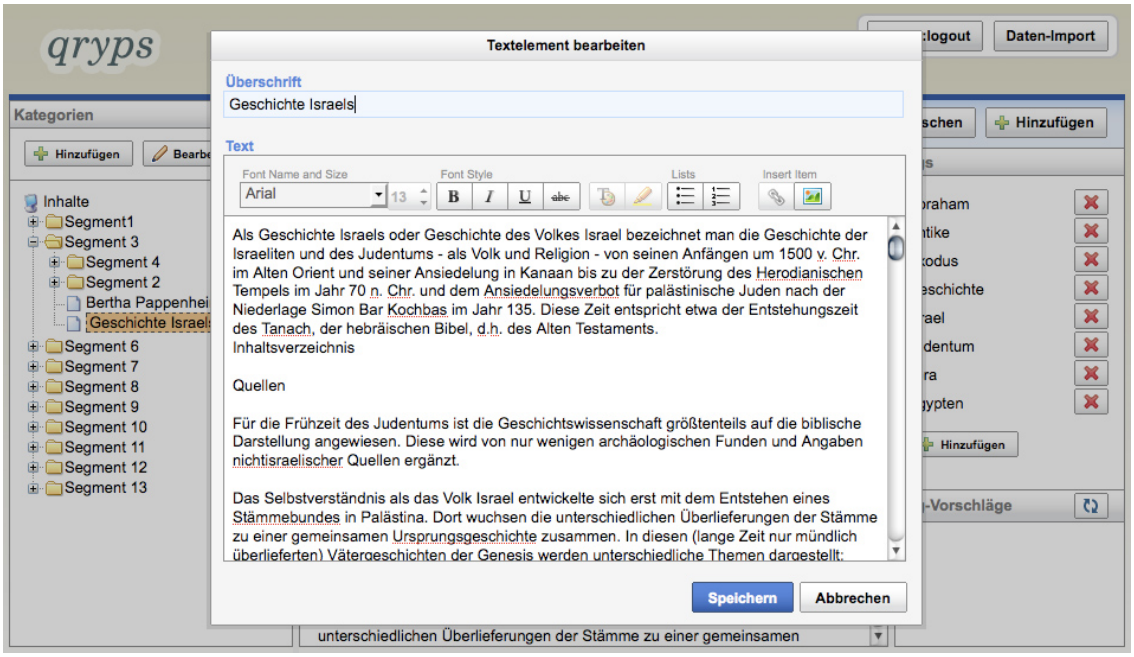
In der rechten Spalte neben den Inhaltselementen werden Zusatzinformationen in Form von Komponenten angezeigt. Zum einen ist dies die Visualisierung des Profils, das in der Auswertung des Museumsbesuches entstanden ist, in Form einer Tag-Cloud (vgl. Abschnitt 5.4.1/ S. 68).

Abbildung 6.5: Screen Frontend-Applikation

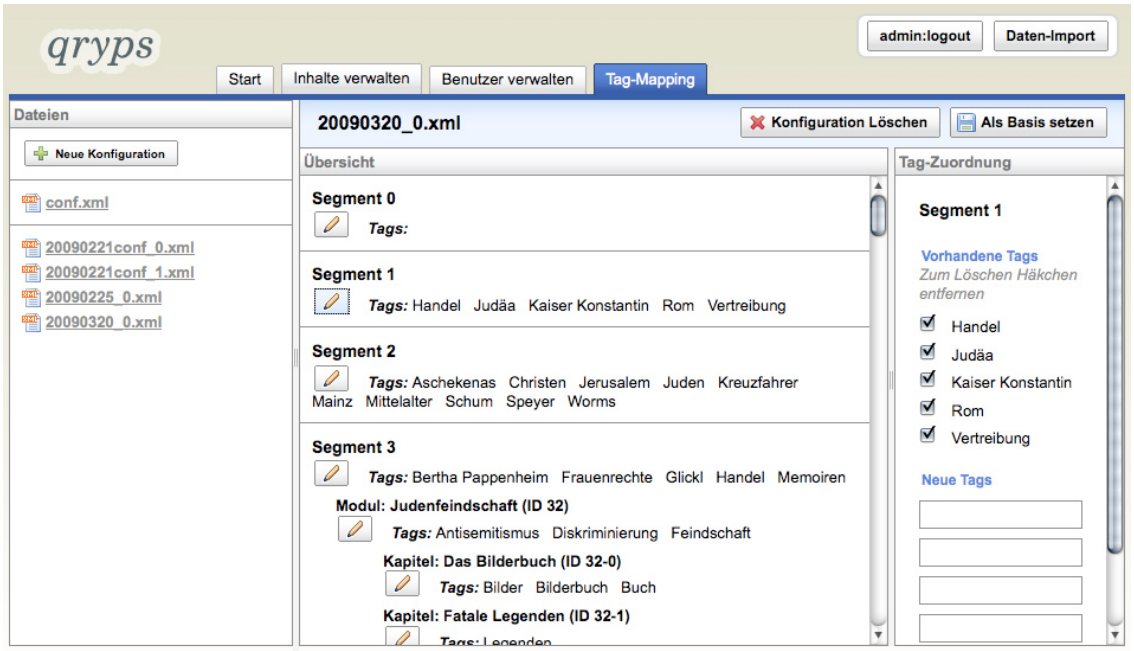
6.5.2 Verwaltungs-Applikation

Die Erstellung von dynamischen Web-Applikationen lässt sich über JavaScript-Frameworks vereinfachen. Diese stellen Funktionen bereit, um statische HTML-Dokumente durch den Einsatz von Ajax und DOM-Scripting interaktiv zu gestalten und dynamisch zu ändern. Dies umfasst die Änderung bzw. Anpassung der Darstellung durch Zuweisung von Formatierungs-Attributen über CSS, die Bereitstellung von zusätzlichen interaktiven Elementen (z. B. Popup-Dialoge innerhalb des Browserfensters, dynamische Navigation) und Änderung der Seiteninhalte, beispielsweise durch die Verwendung von Ajax und Manipulation des DOM-Baumes.

Die Benutzeroberfläche der Verwaltungs-Applikation ist grundlegend anders aufgebaut als die Frontend-Applikation. Im Gegensatz zu dieser ist die Verwaltungs-Applikation weitgehend dynamisch aufgebaut, alle Komponenten werden über Ajax



(a) Dialog



(b) Aufteilung

Abbildung 6.6: Screens Verwaltungs-Applikation

nachgeladen und durch Änderung des DOM-Baumes dargestellt. Realisiert wird dies mit dem JavaScript-Framework *Yahoo! UI Library Version 2.6.1* (*YUI*) [UrlYUI].

Der Systemumfang des Frameworks YUI umfasst verschiedene Bereiche. Der Core enthält die Basis Funktionalitäten, was die Verarbeitung von Events und die DOM-Funktionen (Manipulation des DOM-Baumes und Gestaltung über CSS) umfasst. Diese werden durch zahlreiche Bibliotheken erweitert, deren Komponenten sich in die drei Bereiche *Controls/Widgets* (GUI-, Dialogelemente), *CSS-Tools* (Layout-/Gestaltungsfunktionen) und *Utilities* (allgemeine Hilfs- und Ergänzungsfunktionen) unterteilen lassen.

Die Oberfläche der Verwaltungs-Applikation ist mit Hilfe verschiedener Bibliotheken des YUI-Frameworks realisiert. So werden für die Umsetzung des Layouts mit Aufteilung in drei Spalten und mehrere Tabs (vgl. Abbildung 6.6b) die Komponenten *Grids CSS* und *TabView* verwendet. Die Dialoge sind über Popups innerhalb des Webbrowser-Fensters realisiert (vgl. Abbildung 6.6a), wofür die Bibliothek *Container* benutzt wird. Der verwendete WYSIWYG[1]-Editor gehört als Komponenten (*Rich Text Editor*) ebenfalls zum YUI-Framework.

Tag-Library `YuiButtonTagLib`

Da über ein Plugin zwar einzelne Komponenten des YUI-Frameworks als Tag-Library zur Verfügung gestellt werden, diese aber für die Umsetzung nicht ausreichen, wurden eine eigene Tag-Library entwickelt. Diese umfasst die YUI-Komponente `YAHOO.widget.Button` und stellt verschiedene Funktionen für diese Komponente zur Verfügung. Dies sind die neuen Tags `yuiLinkButton` für die Integration eines Buttons als normalen Link, `yuiRemoteLinkButton` für die Auslösung eines Ajax-Remote-Requests und `yuiSubmitButton` zum Absenden eines Formulars.

6.5.3 Medienelemente

Neben normalen Textinhalten werden von der Applikation auch Medienelemente unterstützt. Dabei erfolgt die Verwaltung unabhängig vom Format, um hier die Möglichkeit einer einfachen Erweiterbarkeit zu haben. Die Unterstützung verschiedener Medienformate ist lediglich in der Darstellung des Mediums im View eingeschränkt.

1 What You See Is What You Get

Es wurden exemplarisch mehrere Bild, Audio- und Videoformate ausgewählt und entsprechende Schnittstellen zur Darstellung über eine Tag-Library implementiert.

Tag-Library `QrypsTagLib`

Die Schnittstellen zur Darstellung werden über die entsprechenden Templates realisiert. Dort wird der Code der darzustellenden Medienelemente über eigene Tags in der Tag-Library `QrypsTagsLib.groovy` generiert. Diese stellt Tags für die verschiedenen Medien zur Verfügung.

Für Bildformate, die von Webbrowsern unterstützt werden (GIF, JPEG, PNG) erfolgt dies durch das einfache Verknüpfen über das entsprechende `img`-HTML-Tag (Tag `mediaImage`). Dazu wurde das Tag `mediaImage` entwickelt.

Für die Einbindung von Audio- und Videoelementen werden zusätzliche Plugins im Webbrowser benötigt. Um dies möglichst einfach zu halten, werden auf eine Flash-basierte Abspielprogramme benutzt. Das Flash-Plugin eignet sich, da es sehr weit verbreitet ist[1] und nur ein einziges Webbrowser-Plugin benötigt wird.

Für das Abspielen der Audio-Elemente im MP3-Format wird der *Flash MP3 Player* [UrlFlashMP3] benutzt (Tag `mediaAudio`), für Videos der *FLV-Scrubber 3.0* [UrlFlvScrub], der mit den von Flash unterstützten Video-Codecs[2] benutzt werden kann (Tag `mediaVideo`).

Listing 6.11: Tag-Library `QrypsTagLib.groovy` mit dem Tag `mediaVideo`

```
1 def mediaVideo = { attrs ->
2   def writer = out
3   writer << '<object id="FLVScrubber" width="450" height="253"
       classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" odebase="
       http://download.macromedia.com/pub/shockwave/cabs/flash/
       swflash.cab#version=9,0,0,0">\n'
4   writer << '  <param name="movie" value="'
5   writer << resource(dir:'extras/FLVScrubber3', file:'FLVScrubber.
       swf') << '"/>\n'
6   writer << '  <param name="bgcolor" value="#000000"/>\n'
7   writer << '  <param name="allowScriptAccess" value="sameDomain
       "/>\n'
8   writer << '  <param name="allowFullScreen" value="true"/>\n'
```

1 Laut einer Studie des Herstellers Adobe werden 99% der Internet-PCs erreicht, mit Apple Quicktime beispielsweise nur 62% (Dezember 2008, vgl. [UrlAdobeCen]).

2 Sorensen, H.264, VP6

```
9   writer << '  <param name="flashVars" value="file='
10  writer << resource(dir:attrs.path, file:attrs.file) << '&
        secondsToHide=2"/>\n'
11  writer << '  <embed src="'
12  writer << resource(dir:'extras/FLVScrubber3', file:'FLVScrubber.
        swf') << '" bgcolor="#000000" width="450" height="253" name="
        FLVScrubber" allowScriptAccess="sameDomain" allowFullScreen="
        true" flashVars="file='
13  writer << resource(dir:attrs.path, file:attrs.file) << '&&
        secondsToHide=2" type="application/x-shockwave-flash"
        pluginspage="http://www.adobe.com/go/getflashplayer">\n'
14  writer << '</object>\n'
15 }
```

6.5.4 Internationalisierung

Das Projekt ist für eine Internationalisierung vorbereitet. Dies bedeutet, dass sämtliche Meldungen (*Messages*) nicht hardgecoded in die Templates oder den Code integriert sind, sondern in einer separaten *Message-Properties*-Datei abgelegt werden. Bei Anpassung der Applikation an andere Sprachen ist es so nur noch nötig, eine Übersetzung der Messages in eine lokalisierte Datei abzulegen. Auch Änderungen einzelner Messages müssen nicht im Code oder den Templates erfolgen, sondern können an zentraler Stelle ausgeführt werden.

Die Dateien befinden sich bei Grails per Vorgabe im Ordner `grails-app/i18n`. Neben der standardmäßigen Default-Version der Property-Datei `messages.properties` liegen dort auch die lokalisierten Versionen. Bei den einzelnen Sprach-Versionen wird die Sprache in den Dateinamen integriert (z. B. für deutsche Spracheinstellungen `messages_de.properties`).

Bei einer HTTP-Anfrage wird die im Webbrowser eingestellte Sprach-Präferenz im Request-Header mitgesendet. Aufgrund der im Request ausgelesenen Sprache erfolgt die Auswahl der Sprache bei der Antwort. Die automatische Auswahl kann manuell angepasst werden, beim Fehlen der mitgesendeten Sprach-Version werden die Default-Werte übertragen..

Listing 6.12: Auszug `messages.properties`

```
backend.mapping.newtags=Neue Tags
backend.mapping.setbaseconf=Als Basis setzen
backend.mapping.delconf=Konfiguration entfernen
backend.mapping.delete=Soll die Konfiguration {0} wirklich entfernt
    werden?
```

Der Zugriff auf die Elemente über die GSP-Templates des Views erfolgt über ein `message`-Tag: `<g:message code="backend.mapping.newtags"/>`.

7 Fazit und Ausblick

Im Rahmen dieser Arbeit wurde ein System entwickelt, dass es ermöglicht Dokumente und Medieninhalte in einen Informationspool einzustellen und diese Daten dort zu verwalten. Diese Daten können dann über eine Web-Plattform von Nutzern abgerufen werden, die zuvor einen multimedialen mobilen Museumsguide benutzt haben und dort ein Profil angelegt haben. Beim Abruf werden die Nutzer unterstützt, indem sie die Dokumente entsprechend ihres Interessenprofils angepasst angeboten bekommen. Dies ermöglicht ihnen einen schnellen Einstieg und Informationen über die Themengebiete, an denen sie zuvor beim Museumsbesuch Interesse gezeigt haben.

Das gestellte Ziel konnte damit erreicht und prototypisch umgesetzt werden. Dazu wurden zunächst die Grundlagen, auf denen ein derartiges System basiert, betrachtet. Bei einer Analyse der in diesem Umfeld bestehenden Systeme wurde festgestellt, dass es die einzelnen Komponenten alleine betrachtet zwar bereits in verschiedenen Formen gibt. Dies betrifft Lokalisierungs- und Recommender Systeme, doch eine Zusammenführung in einem musealen Kontext auf einem mobilen Gerät noch nicht existiert.

Die Anforderungsanalyse war Basis für den späteren Systementwurf, in dem das technische Umfeld entworfen und die Implementierung konzeptioniert wurde. Das Konzept unterlag dabei stets den Rahmenbedingungen des bereits existierenden digitalen Museumsguides, der die Eingabedaten für das System liefert. Die Eingabedaten mussten zunächst aufbereitet werden. Dazu wurden Lösungsstrategien entwickelt, wie die Daten mit dem Informationspool am Besten verknüpft werden können und sich so als Grundlage für eine ein Recommender System eignen. Hier zeigte sich, dass für diesen Anwendungsfall inhaltsbasierte Filter eine sinnvolle Strategie sein können.

Diese Erkenntnisse konnten im Rahmen der Implementierung praktisch umgesetzt werden. Hier hat das entworfene Konzept als praxistauglich erwiesen hat, was in dem entwickelten Prototyp demonstriert wird.

Von vornherein war geplant, dass System neben der Web-Anwendung auch für für eine mobiles Szenario zu konzeptionieren, doch die Umsetzung dieses Szenarios nicht im Rahmen dieser Arbeit durchzuführen. Dieser Teil bietet somit Potential für eine Weiterentwicklung.

Daneben sind Ergänzungen des Informationspools denkbar. Dieser beinhaltet im Moment textuelle und multimediale Inhalte, eine Integration weiterer Datenquellen und Dienste, wie z. B. Veranstaltungskalender oder externe Informationsdienste würden das Informationsangebot aber erweitern.

Abkürzungsverzeichnis

Ajax	Asynchronous JavaScript and XML
AOA	Angle of Arrival
COO	Cell of Origin
CSS	Cascading Style Sheets
DBMS	Datenbankmanagementsystem
DOM	Document Object Model
GIF	Graphics Interchange Format
GPS	Global Positioning System
GORM	Grails' object relational mapping
GUI	Graphical User Interface
GSM	Global System for Mobile Communications
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
JVM	Java Virtual Machine
JPEG	Joint Photographic Experts Group
JSP	Java Server Pages
LDAP	Lightweight Directory Access Protocol
MD5	Message-Digest Algorithm 5
MIME	Multipurpose Internet Mail Extensions
MMG	Mobiler multimedialer Museumsguide
MP3	MPEG-1 Audio Layer 3

MVC	Model View Controller
ORM	Object-Relational Mapping
PNG	Portable Network Graphics
PDA	Personal Digital Assistant
POI	Point of Interest
POP3	Post Office Protocol 3
REST	Representational State Transfer
RFID	Radio Frequency Identification
RPC	Remote Procedure Call
RSSI	Received Signal Strength Indicator
RTMP	Real Time Messaging Protocol
RTT	Round Trip Time
SMTP	Simple Mail Transfer Protocol
SAX	Simple API for XML
SOAP	Simple Object Access Protocol
SWF	Shockwave Flash
TOA	Time of Arrival
UHF	Ultra High Frequency
UML	Unified Modeling Language
UMTS	Universal Mobile Telecommunications System
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WAR	Web Application Archive
WLAN	Wireless Local Area Network
WSDL	Web Services Description Language
WYSIWYG	What You See Is What You Get
XML	Extensible Markup Language
XML-RPC	Extensible Markup Language Remote Procedure Call

Literaturverzeichnis

[ABC03] Anastasi, G. ; Bandelloni, R. ; Conti, M. ; Delmastro, F. ; Gregori, E. ; Mainetto, G.: Experimenting an Indoor Bluetooth-Based Positioning Service. In: *Proceedings of the 23rd International Conference on Distributed Computing Systems Workshops*, 2003, S. 480–483

[BFAS83] Breiman, L. ; Friedman, J. H. ; A., Olshen R. ; Stone, C. J.: *CART: Classification and Regression Trees*. Belmont, CA : Wadsworth, 1983

[BHK98] Breese, J.S. ; Heckerman, D. ; Kadie, C.: Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In: *Uncertainty in Artificial Intelligence: Proceedings of the Fourteenth Conference*, 1998, S. 43–52

[BR09] Brown, J. ; Rocher, G.: *The Definitive Guide to Grails*. Second Edition. Berkeley : Apress, 2009

[BS88] Buckley, C. ; Salton, G.: Term Weighting Approaches in Automatic Text Retrieval. In: *Information Processing & Management* 24 (1988), Nr. 5, S. 512–534

[Bur02] Burke, R.: Hybrid Recommender Systems: Survey and Experiments. In: *User Modeling and User-Adapted Interaction* 12 (2002), Nr. 4, S. 331–370

[Cra04] Cranor, L.F.: 'I Didn't Buy it for Myself': Privacy and Ecommerce Personalization. In: Karat, C.-M. (Hrsg.) ; Blom, J.O. (Hrsg.) ; Karat, J. (Hrsg.): *Designing Personalized User Experiences in eCommerce*, Kluwer Academic Publishers, 2004

[DZ02] Dornbusch, P. ; Zündt, M.: Realisierung von Positionsortungen

in WLAN. In: *ITG-Fachtagung Technologie und Anwendungen für die mobile Informationsgesellschaft*, 2002

[FHM08a] Fiedler, A.; Hohendorf, A.; Meriac, M.; Mohnke, J.; Reinhardt, J. ; Starostik, M.: Lokalisierungstechniken für ein mobiles Museumsinformationssystem. (2008)

[FHM08b] Fiedler, A.; Hohendorf, A.; Mohnke, J.; Reinhardt, J. ; Starostik, M.: Projekt EMIKA – Bereich IT. Projektdokumentation. (2008)

[Fin06] Finkenzeller, K.: *RFID-Handbuch.* 4. Auflage. München : Hanser, 2006

[GH02] Günther, A.; Hoene, C.: Measuring round trip times to determine the distance between WLAN nodes. In: *Proceedings of 4th International IFIP-TC6 Networking Conference, Waterloo, Canada* Bd. 3462. Berlin, Heidelberg, 2002, S. 768–779

[GNOT92] Goldberg, D.; Nichols, D.; Oki, B.M. ; Terry, D.: Using Collaborative Filtering to Weave an Information Tapestry. In: *Communications of the ACM* 35 (1992), Nr. 12, S. 61–70

[Hen07] Henning, P.: *Taschenbuch Multimedia.* 4. Auflage. München : Hanser, 2007

[HNS02] Hallberg, J.; Nilsson, M. ; Synnes, K.: Bluetooth Positioning. In: *Proceedings of the Third Annual Symposium on Computer Science and Electrical Engineering* (2002)

[KKH06] King, T.; Kopf, S.; Haenselmann, T.; Lubberger, C. ; Effelsberg, W.: COMPASS: A probabilistic indoor positioning system based on 802.11 and digital compasses. In: *Proceedings of the 1st international workshop on Wireless network testbeds, experimental evaluation & characterization.* New York, 2006, S. 34–40

[KS03] Kobsa, A.; Schreck, J.: Privacy Through Pseudonymity in User-Adaptive Systems. In: *ACM Transactions on Internet Technology* 3 (2003), Nr. 2, S. 149–183

[PB07] Pazzani, M.J.; Billsus, D.: Content-based Recommendation Systems. In: Brusilovsky, P. (Hrsg.); Kobsa, A. (Hrsg.) ; Nejdl, W. (Hrsg.): *The Adaptive Web: Methods and Strategies of Web Personalization.* Berlin, Heidelberg : Springer, 2007

[RMVH06] Retscher, G. ; Moser, E. ; Vredeveld, D. ; Heberling, D.: Performance and Accuracy Test of the WLAN Indoor Positioning System "ipos". In: *Proceedings of the 3rd Workshop on Positioning, Navigation and Communication (WPNC'06), Hannover* Bd. 16, 2006, S. 7–15

[SFHS07] Schafer, J.B. ; Frankowski, D. ; Herlocker, J. ; Sen, S.: Collaborative Filtering Recommender Systems. In: Brusilovsky, P. (Hrsg.) ; Kobsa, A. (Hrsg.) ; Nejdl, W. (Hrsg.): *The Adaptive Web: Methods and Strategies of Web Personalization.* Berlin, Heidelberg : Springer, 2007

[SPK04] Setten, M. van ; Pokraev, S. ; Koolwaaij, J.: Context-Aware Recommendations in the Mobile Tourist Application COMPASS. In: *Adaptive Hypermedia and Adaptive Web-Based Systems*, Springer, 2004, S. 235–244

[TS06] Takeuchi, Y. ; Sugimoto, M.: CityVoyager: An Outdoor Recommendation System Based on User Location History. In: *Lecture Notes in Computer Science* Bd. 4159. Berlin, Heidelberg : Springer, 2006, S. 625–636

[UrlAdobe] Adobe Systems Inc.: *Adobe LiveCycle Data Services ES.* `http://www.adobe.com/de/products/livecycle/dataservices/`. – Online–Ressource, Abruf: 15.04.2009

[UrlAdobeCen] Adobe Systems Inc.: *Player Census – Flash Player Penetration.* `http://www.adobe.com/products/player_census/flashplayer`. – Online–Ressource, Abruf: 15.04.2009

[UrlAmazon] Amazon.com Inc.: *Amazon.com.* `http://www.amazon.com`. – Online–Ressource, Abruf: 15.11.2008

[UrlBit] Bitmanufaktur: *Bitmanufaktur.* `http://www.bitmanufaktur.de`. – Online–Ressource, Abruf: 10.12.2008

[UrlCocoon] Apache Software Foundation: *Apache Cocoon.* `http://cocoon.apache.org`. – Online–Ressource, Abruf: 27.03.2009

[UrlConnectJ] MySQL AB, Sun Microsystems Inc.: *MySQL®Connector/J.* `http://www.mysql.com/products/connector/j`. – Online–Ressource, Abruf: 02.03.2009

[UrlFlashMP3] nesiumdotcom: *Flash MP3 Player.* `http://www.nesium.com/blog/2006/10/05/flash-mp3-player/`. – Online–Ressource, Abruf: 25.02.2009

[UrlFlvScrub] Topfstedt, F.: *FLV-Scrubber*. http://www.topfstedt.de/weblog/?page_id=208. – Online–Ressource, Abruf: 25.02.2009

[UrlGrails] Grails Team: *Grails*. http://www.grails.org. – Online–Ressource, Abruf: 25.03.2009

[UrlGraSec] Yamamoto, T.: *Spring Security Plugin*. http://www.grails.org/plugin/acegi. – Online–Ressource, Abruf: 25.04.2009

[UrlGroovy] Codehaus Foundation: *Groovy*. http://groovy.codehaus.org. – Online–Ressource, Abruf: 25.03.2009

[UrlHiber] Red Hat Middleware, LLC: *Hibernate – Relational Persistence for Java and .NET*. http://www.hibernate.org. – Online–Ressource, Abruf: 25.03.2009

[UrlIEEE] IEEE P802.11, Task Group N: *Status of Project IEEE 802.11n*. http://grouper.ieee.org/groups/802/11/Reports/tgn_update.htm. – Online–Ressource, Abruf: 05.11.2008

[UrlJetty] Mort Bay Consulting: *Jetty WebServer*. http://jetty.mortbay.com. – Online–Ressource, Abruf: 02.03.2009

[UrlLuckh] Luckhardt, H.-D.: *Virtuelles Handbuch Informationswissenschaft: Automatische und intellektuelle Indexierung*. http://is.uni-sb.de/studium/handbuch/exkurs.ind.html. – Online–Ressource, Abruf: 08.01.2009

[UrlMagicMap] HU Berlin, Lehrstuhl Rechnerorganisation und Kommunikation: *MagicMap*. http://www.informatik.hu-berlin.de/rok/MagicMap. – Online–Ressource, Abruf: 18.12.2008

[UrlMathes] Mathes, A.: *Folksonomies – Cooperative Classification and Communication Through Shared Metadata*. http://www.adammathes.com/academic/computer-mediated-communication/folksonomies.html. – Online–Ressource, Abruf: 08.01.2009

[UrlMysql] MySQL AB, Sun Microsystems Inc.: *MySQL Downloads*. http://dev.mysql.com/downloads. – Online–Ressource, Abruf: 02.03.2009

[UrlOpenBeac] OpenBeacon: *OpenBeacon*. http://www.openbeacon.org. – Online–Ressource, Abruf: 10.12.2008

[UrlOpenID] OpenID: *OpenID Foundation*. http://www.openid.net. – Online–Ressource, Abruf: 25.04.2009

[UrlOpenScre] Adobe Systems Inc: *Open Screen Project – Publications*. http://www.openscreenproject.org. – Online–Ressource, Abruf: 20.04.2009

[UrlPandora] Pandora Media Inc.: *Pandora Internet Radio*. http://www.pandora.com. – Online–Ressource, Abruf: 15.11.2008

[UrlRed5] OSFlash: *Red5 : Open Source Flash Server*. http://osflash.org/red5. – Online–Ressource, Abruf: 15.04.2009

[UrlRest] Fielding, R.T.: *Architectural Styles and the Design of Network-based Software Architectures*. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm. – Online–Ressource, Abruf: 15.03.2009

[UrlSelfJava] SELFHTML e.V.: *SELFHTML: Einführung / Web-Technologien / JavaScript/DOM*. http://de.selfhtml.org/intro/technologien/javascript.htm. – Online–Ressource, Abruf: 25.03.2009

[UrlSiteMesh] OpenSymphony: *SiteMesh*. http://opensymphony.com/sitemesh. – Online–Ressource, Abruf: 02.04.2009

[UrlSoap] Gudgin, M. and Hadley, M. and Mendelsohn, N. and Moreau, J.-J. and Nielsen, H.F. and Karmarkar, A. and Lafon, Y.: *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. http://www.w3.org/TR/soap12-part1. – Online–Ressource, Abruf: 15.03.2009

[UrlSpring] SpringSource: *Spring*. http://www.springsource.org. – Online–Ressource, Abruf: 25.03.2009

[UrlStruts] Apache Software Foundation: *Apache Struts*. http://struts.apache.org. – Online–Ressource, Abruf: 25.03.2009

[UrlW3C] W3C: *Overview and Summary of W3C Patent Policy*. http://www.w3.org/2004/02/05-patentsummary.html. – Online–Ressource, Abruf: 17.03.2009

[UrlWalnes] Walnes, J.: *Prasentation: Advanced Layouts with SiteMesh*. http://www.grails-exchange.com/files/JoeWalnes-SiteMesh.pdf. – Online–Ressource, Abruf: 20.04.2009

[UrlWebservi] Austin, D. and Barbir, A. and Ferris, C and Garg, S.: *Web Services Architecture Requirements, W3C Working Draft 11 October*

2002. http://www.w3.org/2004/02/05-patentsummary.html. – Online–Ressource, Abruf: 17.03.2009

[UrlXmlRpc] WINER, D.: *XML-RPC Specification*. http://www.xmlrpc.com/spec. – Online–Ressource, Abruf: 19.03.2009

[UrlYUI] YAHOO! INC.: *The Yahoo! User Interface Library (YUI)*. http://developer.yahoo.com/yui. – Online–Ressource, Abruf: 01.02.2009

[WBR05] WENDLANDT, K.; BERBIG, M. ; ROBERTSON, P.: Indoor localization with probability density functions based on Bluetooth. In: *IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications, 2005* Bd. 3, 2005, S. 2040–2044

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings