Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Anwendungen II

Ruben Schempp

Interaktive Karten als Rich Internet Applications

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Ruben Schempp

Interaktive Karten als Rich Internet Applications

Ausarbeitung eingereicht im Rahmen der Veranstaltung Anwendungen 2
im Wintersemester 2007/2008
im Studiengang Master of Science Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Kai von Luck

Abgegeben am 28. Februar 2008

Inhaltsverzeichnis

1 Einleitung **1**

2 Szenario **3**
2.1 Beschreibung 3
2.2 Vision 5

3 Verwandte Anwendungen und Projekte **6**
3.1 Google Maps 6
3.2 eyeOS & Google Documents 7
3.3 MapWiki 8

4 Technik **11**
4.1 AJAX-Technologie 12
4.2 Client/Server Kommunikationsmodell 13
4.3 Ruby on Rails 14
4.4 Blackboard Architektur & iROS 16

5 Zusammenfassung **18**
5.1 Fazit 18
5.2 Ausblick 18

Literaturverzeichnis **20**

Abbildungsverzeichnis

2.1 Powerwall 4

3.1 Google Maps 6
3.2 eyeOS 7
3.3 Google Documents 8
3.4 MapWiki Übersicht 9
3.5 MapWiki Anwendung 10

4.1 AJAX 12
4.2 Client/Server Trennung 14
4.3 Rails Framework 15
4.4 Blackboard Architektur 16

1 Einleitung

Diese Ausarbeitung beschreibt die Web-Anwendung Maps on Rails und mit ihr verwandte Arbeiten und Projekte. Maps on Rails ist ein Prototyp, der auf interaktiven Karten basiert und als Rich Internet Application implementiert wird. Als Grundlage für die prototypische Erstellung der Anwendung wird das schon in der Ausarbeitung Anwendungen I [Schempp, 2007] vorgestellte Szenario der Einsatzleitstelle dienen.

Für die Umsetzung der Anwendung wird das Ruby on Rails Framework, die AJAX-Technologie und ein Kartendienst verwendet. Ruby on Rails ist noch relativ jung. Und auch die AJAX-Technologie und Kartendienste, nach Art von Yahoo! oder Google Maps ([Yahoo! Inc., 2007] bzw. [Google Inc., 2007b]), kamen erst in den letzten Jahren auf. Aus diesem Grund gibt es noch sehr wenige Web-Anwendungen, die große Ähnlichkeiten zu Maps on Rails aufweisen bzw. eng mit Maps on Rails verwandt sind. Nichtsdestotrotz gibt es einige Mashup-Anwendungen, die demonstrieren, was mit Hilfe der AJAX-Technologie oder mit Kartendiensten mittlerweile machbar ist. Im Folgenden werden ein paar ausgewählte Anwendungen und Projekte vorgestellt, die mit der hier beschriebenen Anwendung thematisch und/oder technisch verwandt sind. Sie verdeutlichen, welche Möglichkeiten und Grenzen die Technologien bieten und in welche Richtung sich die Anwendung Maps on Rails entwickeln soll. Weiter wird – soweit es im Rahmen dieser Ausarbeitung möglich ist – ein Blick hinter die Kulissen der vorgestellten bzw. eingesetzten Techniken geworfen.

Einordnung und Bezug

Diese Ausarbeitung stellt die Fortsetzung der in Anwendungen I [Schempp, 2007] begonnenen Projektarbeit im Rahmen des Masterstudiums der Informatik an der Hochschule für Angewandte Wissenschaften Hamburg dar. Sie behandelt den Bereich des Computer Supported Collaborative Work bzw. des Collaborative Workplace[1] im Rahmen der bestehenden und aktuell in Bearbeitung befindlichen Arbeiten im Ubicomp-Labor der Hochschule für Angewandte Wissenschaften Hamburg. Das Ziel der Arbeit ist, durch technische Unterstützung die Zusammenarbeit zwischen mehreren Personen an einer übergeordneten Aufgabe zu erleichtern bzw. zu verbessern oder gar zu ermöglichen. Diese übergeordnete Aufgabe wird im

[1] Das Konzept des Collaborative Workplace wird in [Köckritz, 2006] beschrieben.

Rahmen des Szenarios in Anwendungen I genauer beschrieben und hier in Kapitel 2 erneut kurz vorgestellt.

In der Veranstaltung Seminar/Ringvorlesung des aktuellen Studiensemesters [Schempp, 2008b] entsteht parallel zu dieser eine weitere Ausarbeitung, die eine stärkere Fokussierung auf die anstehende Masterarbeit im Rahmen des Szenarios hat. Ergänzend zu diesen beiden Ausarbeitungen ist der Projektbericht [Schempp, 2008a] gedacht, der die Umsetzung der Anwendung Maps on Rails vorbereitet und die gewählte Technik auf ihre Eignung hin untersucht. Im Zusammenhang gesehen sollen diese drei Arbeiten den thematischen Rahmen für die Masterarbeit vorgeben und so auf deren Erstellung vorbereiten.

Sofern im Folgenden besondere Berührungspunkte zwischen diesen Arbeiten bestehen, wird darauf an der entsprechenden Stelle hingewiesen. Im folgenden Kapitel wird das Szenario für Maps on Rails beschrieben. In Kapitel 3 werden verwandte Arbeiten vorgestellt und daraufhin in Kapitel 4 einige Techniken vorgestellt. Nachfolgend schließt die Zusammenfassung diese Ausarbeitung ab.

2 Szenario

Diese Ausarbeitung des Leitstand-Szenarios steht in Verbindung zu den vorangegangenen Arbeiten zum Thema „RESCUE: Leitstand für Desaster-Szenarien“. Es besteht eine große Ähnlichkeit zu dem in [Piening, 2006] beschriebenen Szenario und der in [Piening, 2007] beschriebenen Technologien. Jedoch unterscheiden sich die Szenarien in ihrer Ausrichtung und Umsetzung voneinander. Das hier gewählte Szenario wurde bereits in [Schempp, 2007] beschrieben. Es wird an dieser Stelle kurz (aber nicht vollständig) zusammengefasst und ergänzend behandelt.

2.1 Beschreibung

Eine Einsatzleitstelle dient als zentraler Ausgangspunkt für die Leitung, Organisation, Planung und Koordination eines Einsatzes, an dem Einsatzkräfte und -fahrzeuge teilnehmen und evtl. Gefahren zu berücksichtigen sind. In der Leitstelle laufen nach Möglichkeit alle Informationen zusammen und werden ausgewertet um neue Entscheidungen zu fällen. Die leitenden Personen, die diese Entscheidungen treffen, stehen vor der schwierigen Aufgabe, sich innerhalb kürzester Zeit Informationen zu beschaffen und auf deren Grundlage unter Zeitdruck die korrekten Entscheidungen zu treffen. Denkbare konkrete Ausprägungen des Szenarios sind Einsätze von Polizei und/oder Feuerwehr sowie auch das Überwachen bzw. Verfolgen von Geldtransportern oder Demonstrationen.

Im Rahmen dieser Ausarbeitung wird von einer abstrakten Betrachtung des Szenario ausgegangen, bei der es in erster Linie um die Abbildung der Positionen von Personen bzw. Fahrzeugen geht. Durch die Anwendung Maps on Rails sollen diese Positionsangaben zusammen mit anderen Informationen übersichtlich auf einer Land- bzw. Stadtkarte dargestellt werden[1]. So soll die Entscheidungsfindung erleichtert werden, indem die Beschaffung sowie die Aufbereitung der Informationen verbessert wird. Das Fällen von Entscheidungen kann und soll die Anwendung dem Menschen nicht abnehmen.

[1] Die Beschaffung der Information liegt nicht im Fokus dieses Szenarios. Solche Informationen können entweder manuell oder automatisiert erfasst werden. Positionen können beispielsweise mit handelsüblichen GPS-Systemen bestimmt und danach automatisiert weiterverarbeitet werden.

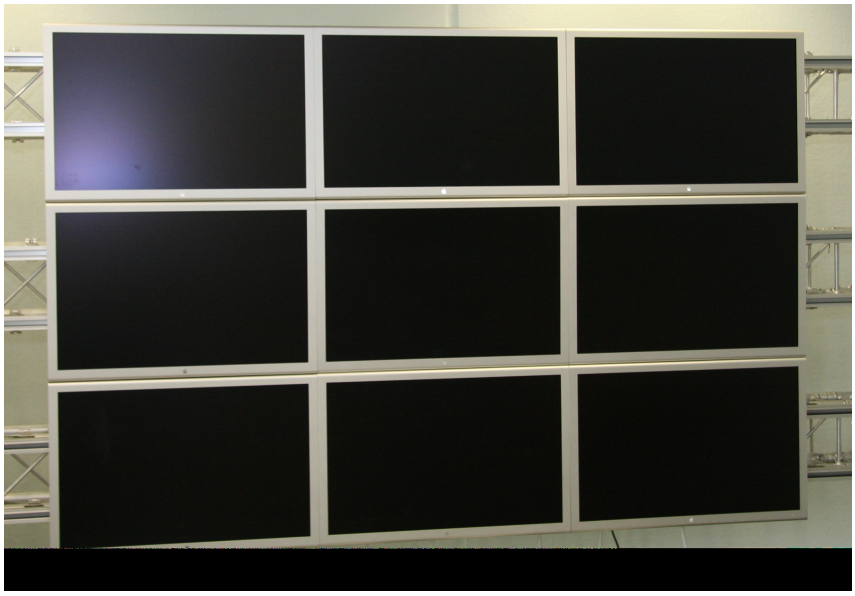

Abbildung 2.1: Neun Monitore als Powerwall an der HAW-Hamburg.

In der Einsatzleitstelle kann die Anwendung beispielsweise auf einer hochauflösenden Bildschrimwand (Powerwall) dargestellt werden. In Abbildung 2.1 ist die Powerwall an der Hochschule für Angewandte Wissenschaften Hamburg abgebildet, die für eine Darstellung in einer Leitzentrale gut geeignet ist. Die auf der Karte visualisierten Informationen des Systems müssen nicht allein den Personen in der Leitstelle vorbehalten sein. Aufgrund der Auslegung der Anwendung Maps on Rails als Rich Internet Application ist es mit einem entsprechenden Internetbrowser möglich die Informationen auch auf mobilen Geräten in angepasster Weise zur Verfügung zu stellen. Weitere Überlegungen und Voraussetzungen hierzu finden sich in [Schempp, 2008b].

Die Darstellung der verfügbaren (Positions-)Informationen findet auf einer Karte statt. Das Kartenmaterial hierzu ist mittlerweile durch Internetdienste wie Google oder Yahoo! Maps frei für jedermann zugänglich. Früher waren solche Informationen nur in Expertensystemen, die für spezielle Aufgaben Verwendung fanden, verfügbar[2]. Die Verfügbarkeit dieser Informationen in Form von Karten, Straßenplänen und Satellitenbildern und der ungehinderte Zugang zu ihnen soll in Maps on Rails genutzt werden.

[2] Diese Expertensysteme werden auch Geoinformationssysteme (GIS) genannt. Sie werden beispielsweise für die Stadtplanung eingesetzt. Google und Yahoo! Maps können (trotz ihrer grundverschiedenen Ausrichtung) auch als GIS angesehen werden.

2.2 Vision

Die Vision basiert auf der Umsetzung des Szenarios Einsatzleitzentrale, die prototypisch geschehen soll. Für den entstehenden Prototypen wird von den konkreten Anforderungen der Einsatzleitzentrale abstrahiert, sodass prinzipiell auch andere Szenarien für den Einsatz der Anwendung in Frage kommen, bei denen es darum geht, einen Überblick über ortsgebundene Informationenen zu bekommen[3]. Maps on Rails soll als Prototyp eines interaktiven kartenbasierten Mehrbenutzer-Informationssystems dienen. Für die Umsetzung sollen die Programmiersprache Ruby, das Ruby on Rails Framework, die AJAX-Technologie und ein Kartendienst verwendet werden.

Bezogen auf die Masterarbeit besteht neben der Integration des Kartenmaterials und der Austauschbarkeit der Endgeräte das eigentliche Ziel darin, die Sprache Ruby und insbesondere das Rails Framework im praktischen Einsatz zu evaluieren.

[3] Eine detailliertere Beschreibung der Anforderungen des Szenarios findet sich in [Schempp, 2007] und [Schempp, 2008b]

3 Verwandte Anwendungen und Projekte

Dieses Kapitel stellt einige Anwendungen des Web2.0 [O'Reilly, 2005] vor, die entweder wegen ihrer Technik oder ihres Themas mit der Anwendung Maps on Rails verwandt sind. Es handelt sich dabei um Rich Internet Applications, die zeigen sollen, was mit Kartendiensten und/oder der AJAX-Technologie inzwischen machbar ist.

3.1 Google Maps

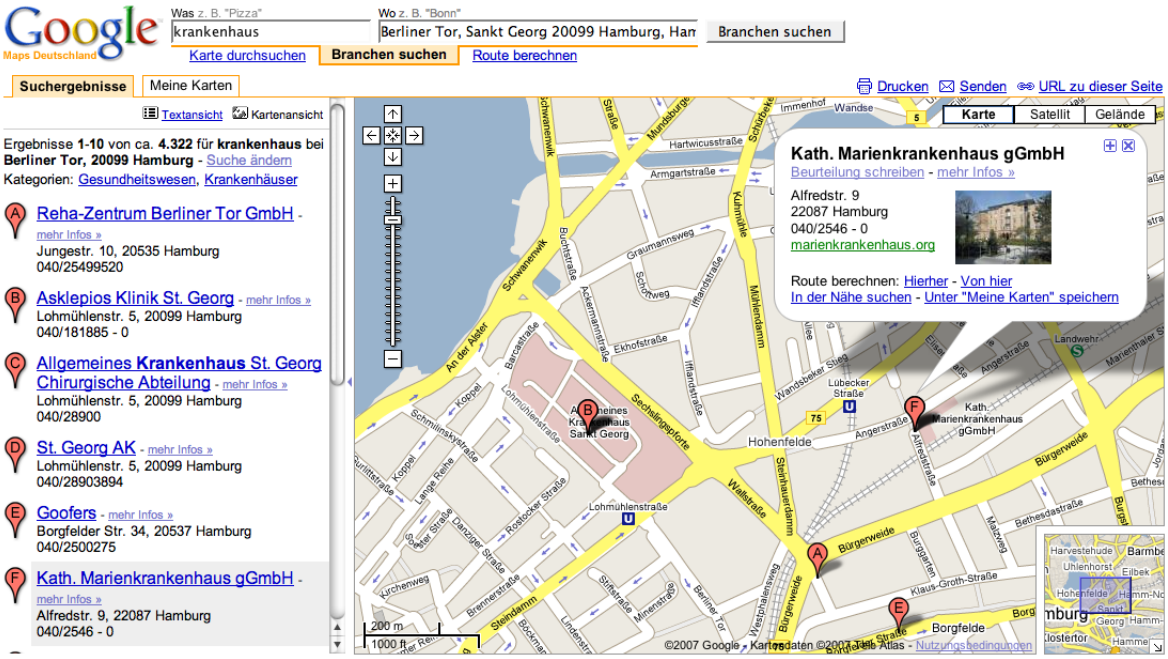

Abbildung 3.1: Die Umkreissuche mit Google Maps im Hamburger Stadtteil St.Georg zeigt eine der Fähigkeiten von Geoinformationssystemen. [Google Inc., 2007b]

Google Maps ist einer der am meisten verbreiteten Kartendienste. Er stellt Karten, Satellitenbilder und das Straßennetz dar. Das Kartenmaterial kann auch über eine entsprechende Programmierbibliothek (API, [Google Inc., 2007c]) abgerufen werden und so in eigene

Anwendungen integriert werden. Man spricht bei der entstehenden Anwendung von einem Mashup.

Die Fähigkeiten eines Geoinformationssystems werden deutlich, wenn man etwa den Routenplaner verwendet oder die Umkreissuche benutzt. Bei der Umkreissuche wird nur – und hier liegt die geographische Besonderheit – in dem aktuell sichtbaren Kartenausschnitt nach relevanten Informationen gesucht, die dann optisch hervorgehoben werden. Suchen kann man etwa nach Supermärkten, Bahnhöfen, Kirchen, Tankstellen oder Krankenhäusern, wie in Abbildung 3.1 dargestellt.

Yahoo! bietet ebenfalls einen Kartendienst[1] an, der sich in seinem Erscheinungsbild und seiner Funktionalität nur gering von Google Maps unterscheidet. Es gibt noch weitere im Internet verfügbare Kartendienste. Meist fehlt diesen eine Bibliothek oder diese verfügt über eine zu geringe Funktionalität. Auf diese Dienste wird wegen der erheblich geringeren Relevanz für die Anwendung nicht näher eingegangen.

3.2 eyeOS & Google Documents

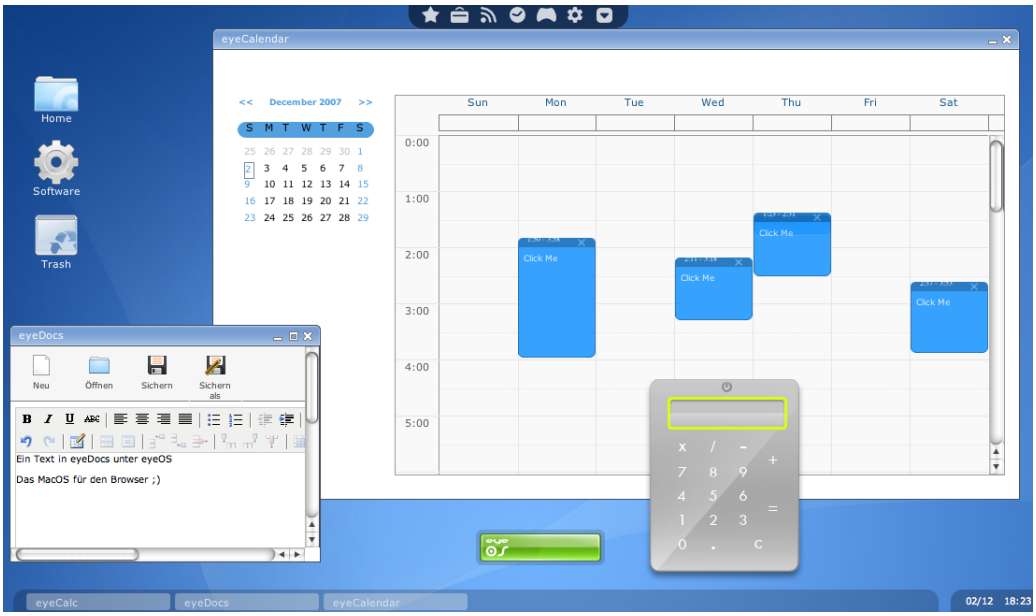

Abbildung 3.2: Die Oberfläche von eyeOS erinnert stark an MacOS. [eyeOS Project, 2007]

In diesem Abschnitt werden zwei Anwendungen gezeigt, die massiv auf die AJAX-Technologie bauen. eyeOS ist nach [eyeOS Project, 2007] ein Betriebsystem, wel-

[1] Für den Kartendienst siehe [Yahoo! Inc., 2007] und die API siehe [Yahoo! Inc., 2008].

ches im Browser läuft. Es zeigt eindrucksvoll, dass es inziwschen mit den Mitteln der AJAX-Technologie möglich ist eine grafisch ansprechende Benutzeroberfläche für Web-Anwendungen zu programmieren. Jenseits von reinen HTML-Seiten sind in eyeOS Anwendungen, wie man sie von seinem Desktopcomputer kennt, integriert (siehe Abbildung 3.2). Diese werden direkt vom Internetserver geladen, aber weiterhin lokal ausgeführt.

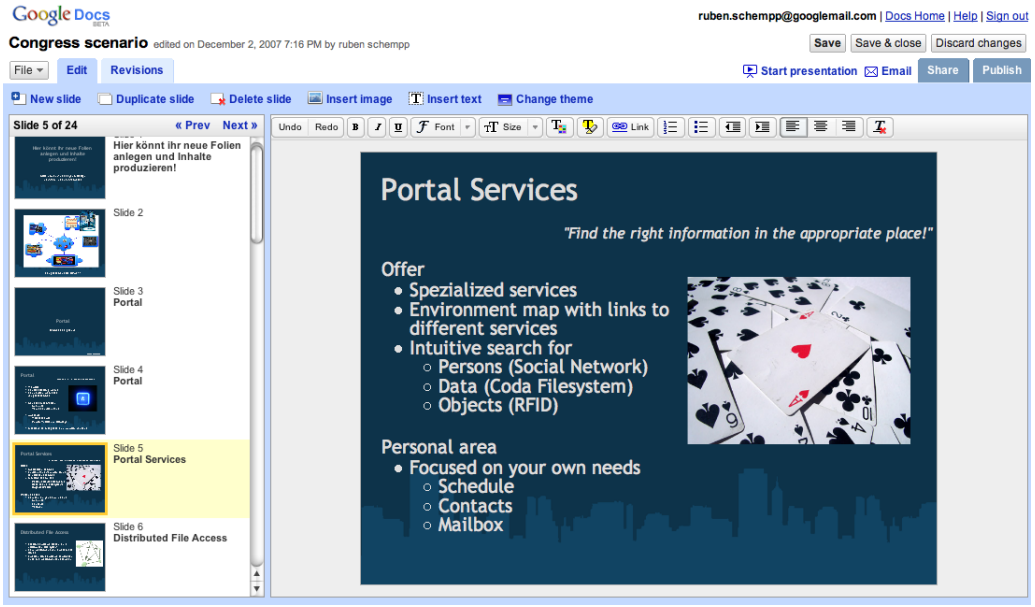

Abbildung 3.3: Google Documents bietet Text-, Tabellen- und Präsentationsbearbeitung. [Google Inc., 2007a]

Google ist mittlerweile mit einer ganzen Produktpalette an Bürosoftware im Internet vertreten. Von der E-Mail-Software bis zum mehrbenutzerfähigen Präsentationsprogramm (siehe Abbildung 3.3) finden sich mehr oder weniger ausgereifte Anwendungen. Je nach Nutzung der Anwendung werden die benötigten Programmteile zur Laufzeit vom Server nachgeladen.

3.3 MapWiki

MapWiki ist eine Mashup-Anwendung, die an zwei japanischen Hochschulen entstanden ist. Sie ermöglicht verschiedenen Benutzern Informationen auf einer Karte miteinander zu teilen und auszutauschen (siehe Abbildung 3.4). Beispielsweise können Benutzer, die zu Hause an ihrem Computer arbeiten, in MapWiki Hinweise über Sehenswürdigkeiten in die Karte eintragen. Der Wanderer, der sein mobiles Gerät mit Internetanbindung dabei hat, kann diese

Abbildung 3.4: Skizze des Konzepts von MapWiki. [Teranishi u. a., 2006]

Informationen abrufen und neue hinzufügen. So entsteht ein Kreislauf von Informationen zwischen der virtuellen und realen Welt.

Abbildung 3.5 zeigt ein Bildschirmfoto der Web-Anwendung. Bei ihr wurde darauf geachtet, dass geographisch nahe liegende Informationen zeitlich bevorzugt vor weit entfernten Informationen aktualisiert werden. Das Konzept des Informationsaustauschs mit der Karte als Medium ist dabei eng verwandt mit Maps on Rails. Der Einsatzzweck, die Informationsarten und deren Beschaffungswege sind jedoch unterschiedlich. MapWiki nutzt die Maps Bibliothek von Google.

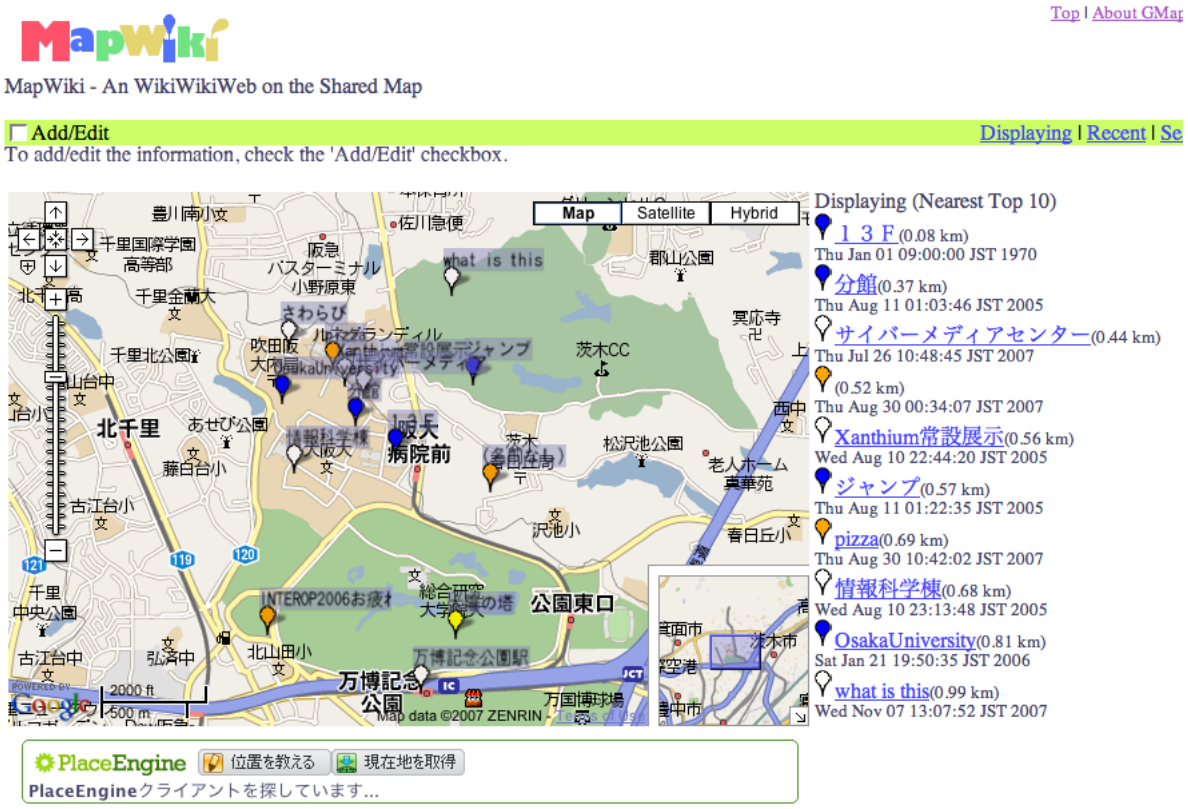

Abbildung 3.5: MapWiki zeigt die im Kartenausschnitt markierten Orte. [Teranishi u. a., 2006]

4 Technik

Dieses Kapitel stellt die Techniken näher vor, die für Maps on Rails eingesetzt werden sollen. In der Ausarbeitung des Seminars [Schempp, 2008b] wird näher auf den Aufbau und das Zusammenspiel der Komponenten eingegangen.

Java für Rich Internet Applications

Neben den im Folgenden genannten Techniken gibt es noch weitere Ansätze zur Konstruktion von Rich Internet Applications. Diese basieren auf der Programmiersprache Java. Eine Übersicht findet sich in [Hartmann, 2008] wieder. Dazu gehören das Google Web Toolkit[1] und das Echo Framework[2]. Sie basieren auf der Idee, dass Java-Programmierer mit einem Framework in der Lage sein sollen eine Web-Anwendung zu erstellen. Das soll ohne größere Umschulungen und in der gewohnten Entwicklungsumgebung möglich sein.

Im Rahmen des Eclipse Projekts existiert die Rich Client Platform[3] als Framework. Es lädt eine Java-Anwendung als Plugin in die Eclipse-Platform und führt diese auf einem Desktopcomputer als eigenständige Applikation aus. Neben der Rich Client Platform existiert auch die sehr ähnliche Rich AJAX Platform[4]. Sie unterscheidet sich von der Rich Client Platform darin, dass die Anwendungen auf einem Server ausgeführt werden und der Client-Zugriff über einen Web-Browser erfolgt.

Im Folgenden sollen Java-basierte Ansätze nicht weiter diskutiert werden, da Ruby on Rails (aus bereits in Anwendungen I erwähnten Gründen) für das Szenario eingesetzt werden soll.

[1] Siehe: http://code.google.com/webtoolkit/ und [Carfagno, 2007].
[2] Siehe: http://echo.nextapp.com/site/echo2
[3] Siehe: http://wiki.eclipse.org/index.php/Rich_Client_Platform
[4] Siehe: http://www.eclipse.org/rap/

4.1 AJAX-Technologie

Mit der AJAX-Technologie[5] ist es möglich desktopähnliche Anwendungen zu programmieren, die im Browser lauffähig sind. Der Programmcode kann dabei zur Laufzeit vom Webserver generiert und gesendet werden (Code-on-Demand). Beispiele für AJAX-baiserte Anwendungen finden sich im vorigen Kapitel.

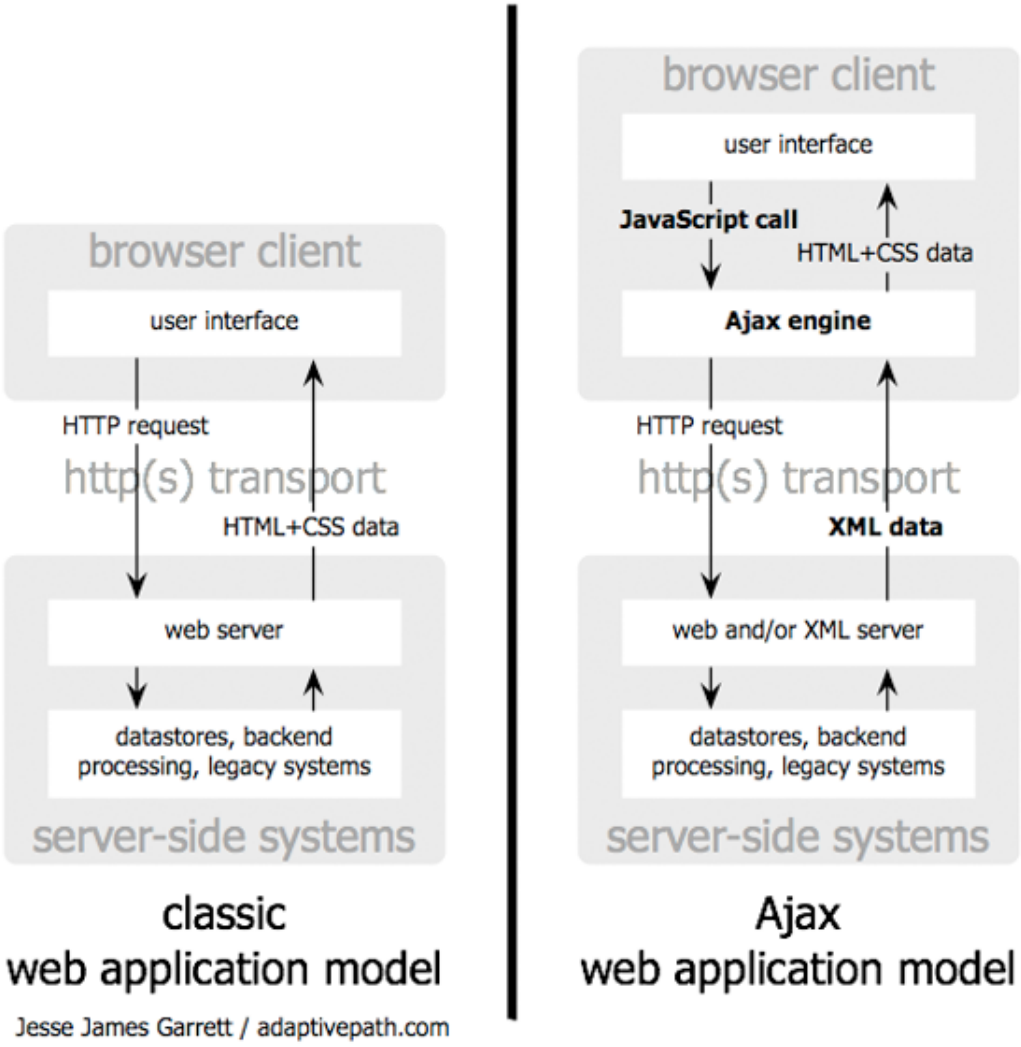

Abbildung 4.1: AJAX im Vergleich zu herkömmlichem HTML [Garrett, 2005]

Abbildung 4.1 zeigt die Kommunikationswege einer klassischen Internetanwendung (links) im Vergleich zu einer AJAX-basierten Web2.0-Anwendung (rechts). Beim klassischen Ansatz

[5] Asynchronous JavaScript and XML. Siehe dazu [Doernhoefer, 2006] und [Garrett, 2005].

wird die aktuelle Internetseite komplett vom Server auf den Client übertragen. Folgt daraufhin eine Interaktion des Nutzers, so wird eine HTTP-Anfrage an den Server geschickt, der daraufhin mit einer neuen HTML-Seite antwortet. Das Backend (der jeweils unterste Kasten des Server-Systems in der Grafik) ist dabei bei beiden Ansätzen gleich. Im Vergleich zum klassischen Ansatz ist es mit der AJAX-Technologie möglich von der Web-Programmiersprache JavaScript Gebrauch zu machen. So können jederzeit und unabhängig von Nutzerinteraktionen Daten mit dem Server ausgetauscht werden.

Der Benutzer löst durch seine Interaktion die Ausführung einer JavaScript-Aktion aus. Daraufhin kann der Browser entweder lokalen Code ausführen oder eine HTTP-Anfrage an den Webserver senden. Diese Anfragen können asynchron erfolgen, d.h. sie unterbrechen den Anwendungsfluss nicht, sondern werden (nicht-blockierend) „im Hintergrund“ verarbeitet, sodass der Benutzer nichts davon mitbekommt. Als Antwort erhält der Client entweder eine HTML-Seite, XML-Daten oder JavaScript-Code mit neuer Funktionalität. Der Benutzer bekommt entweder neuen Seiteninhalt zu sehen oder es wird mittels JavaScript eine Funktion ausgeführt die etwas am Zustand der Anwendung ändert. Genau diese Funktionalitäten ermöglichen den Bau von dynamischen Internetseiten, die weit mehr können, als bloß (statischen) Inhalt darstellen.

4.2 Client/Server Kommunikationsmodell

Dieser Abschnitt behandelt die Einstufung der – für den Anwender transparenten – Trennung von Client und Server im Kommunikationsmodell.

Nach [Tanenbaum, 2003] gibt es in Verteilten Systemen verschiedene Schnitte für die Auftrennung zwischen Client und Server. Diese Trennung verläuft fließend von Kategorie *(a)* bis *(e)*, wie Abbildung 4.2 zeigt. Eine typische AJAX-Anwendung kann hier in *(c)* eingeordnet werden, da die Anwendungsschicht sowohl auf den Client als auch auf den Server verteilt ist. Die Meinungen mögen an diesem Punkt ein wenig auseinander gehen, ob nicht auch *(b)* oder *(d)* eine gültige Einordnung wären. Beides ist denkbar, hängt aber ganz von der jeweiligen Anwendung ab. Kategorie *(b)* könnte etwa eine Anwendung beschreiben, bei der clientseitig nur als Anfragen an den Server gesendet werden und alle nötigen Berechnungen dort stattfinden. Im Gegensatz dazu könnte Kategorie *(d)* eine Anwendung beschreiben, die den Programmcode vollständig auf den Client lädt und dort ausführt. Beides sind sicherlich Sonderfälle, die so in der Realität selten auftauchen werden bzw. z.T. nicht ohne Weiteres umsetzbar sind. Die Kategorien *(a)* und *(e)* sollte man bei einer normalen Web-Anwendung hingegen nicht wiederfinden. Denn das Benutzerinterface sollte eindeutig auf dem Client und die (systemweite) Datenbank eindeutig auf dem Server lokalisiert sein.

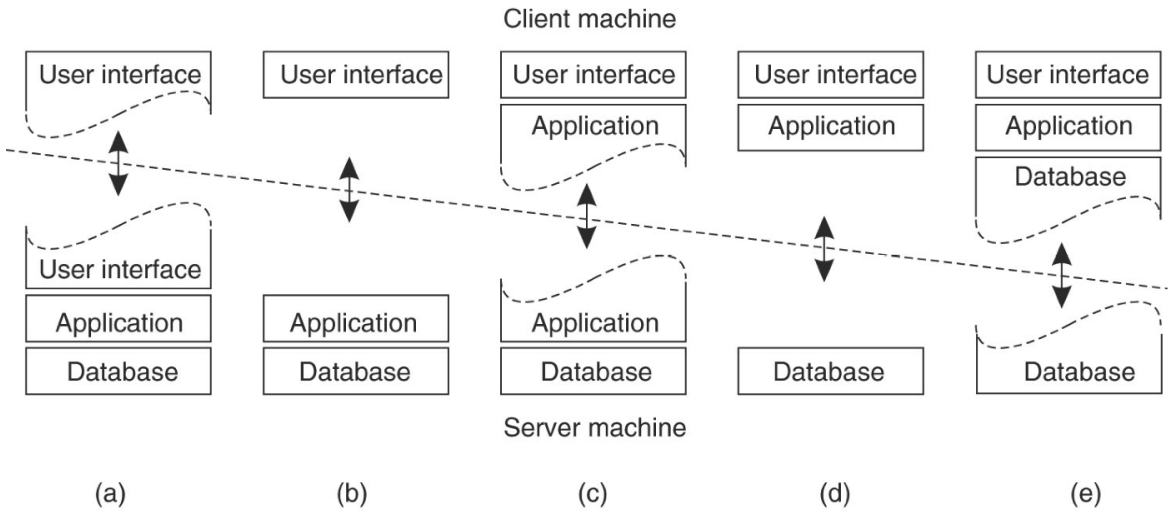

Abbildung 4.2: Das Client/Server-Modell nach A.Tanenbaum. [Tanenbaum, 2003]

Im Falle von Maps on Rails findet ein Großteil der Berechnungen auf dem Server statt. Die komplette Anwendungslogik (in Ruby) ist dort verankert. Clientseitig wird für die Benutzeroberfläche und für die Steuerung der Kommunikation mit dem Server Javascript-Code eingesetzt. Der Server generiert z.B. zur Laufzeit JavaScript, das anschließend auf dem Client ausgeführt wird. So kann Maps on Rails also zwischen *(b)* und *(c)* eingeordnet werden. Diese stark serverlastige Ausrichtung liegt zum einen daran, dass so kostenintensive Berechnungen nur einmal (serverseitig) ausgeführt werden müssen und ist zum anderen in dem Aufbau einer Web-Anwendung mit einem synchronen Datenbestand zwischen mehreren Benutzern begründet.

4.3 Ruby on Rails

Rails ist ein Framework für die Programmiersprache Ruby, das wiederum auf weiteren Frameworks (z.B. Action Pack und Active Record) basiert. Alle dabei integrierten Komponenten kommen aus einer Hand und sind auf einander abgestimmt, sodass ein verhältnismäßig unkomplizierter Start möglich ist.

Das Haupteinsatzgebiet von Rails ist die Erstellung von Web-Anwendungen mit einer relationalen Datenbank. Rails stellt – wenn man so möchte – eine (integrierte) Domänenspezifische Sprache für die Web-Entwicklung dar, die auf den vollen Umfang der Sprache Ruby zurückgreifen kann. Nähere Informationen zur Programmiersprache Ruby – auf die an dieser Stelle nicht weiter eingegangen wird – und dem Rails Framework finden sich in [Thomas u. a., 2005], [Thomas und Hannson, 2006], [Tate, 2005] und [Schempp, 2006].

Rails ist nach dem Prinzip „Convention over Configuration“ aufgebaut. D.h. es sind bestimmte Einstellungen per Konvention festgelegt, die nicht extra konfiguriert werden müssen. Dies ermöglicht es, am Anfang eines Projekts schnelle Ergebnisse zu produzieren, während es dem Entwickler aber trotzdem frei steht, diese Konfiguration zu verändern. Die beiden Frameworks Active Record und Action Pack bilden elegante Lösungen um eine Anbindung an eine Datenbank herzustellen und die Anwendung nach dem MVC-Architekturmuster (siehe [Gamma u. a., 1995]) aufzubauen. Interessant sind auch die in Rails integrierten Möglichkeiten für das Unit-Testing sowie für das Testen des Anwendungscontrollers ohne die Verwendung eines Web-Servers.

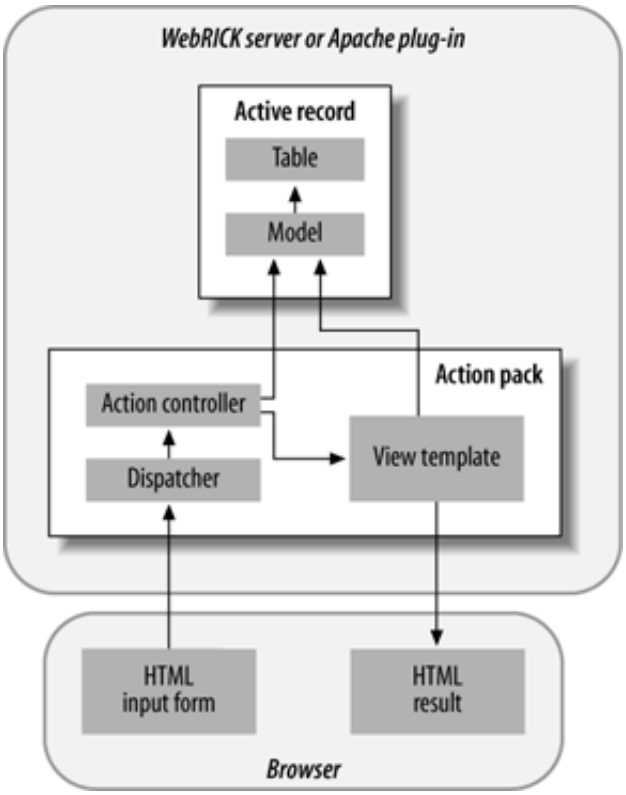

Abbildung 4.3: Der schematische Aufbau des Ruby on Rails Frameworks. [Tate, 2005]

Abbildung 4.3 zeigt den Aufbau von Ruby on Rails mit den beiden Frameworks Active Record und Action Pack, die ihre Arbeit auf Seite des Servers verrichten. Auf Clientseite läuft nur ein Internetbrowser. Während Active Record für die Abbildung des Anwendungsmodells auf die Datenbank zuständig ist, steuert Action Pack den Ablauf der Anwendung. Im Dispatcher wird auf Grund der vom Client aufgerufenen URL entschieden, welche Aktion des zugehörigen Controllers ausgeführt wird. Das Ergebnis wird dem Nutzer durch das Rendern des View Templates präsentiert, das der entsprechenden Aktion zugeordnet ist. Während

dieser Ausführungsphase stehen die Anwendungsdaten durch Active Record als Objekte zur Verfügung.

Rails unterstützt sogenannte Helper, die die Integration zusätzlicher Funktionalität ermöglichen, diese aber so kapselt, dass der eigentliche Anwendungscode davon unberührt bleibt. Ein solcher, bereits mitgelieferter, Helper bietet beispielsweise in der View die Möglichkeit HTML und JavaScript[6] zu generieren, so dass der Entwickler den Code nicht selbst schreiben muss. Rails bringt also von Haus aus einige hilfreiche Fähigkeiten mit, die es dem Entwickler ermöglichen, sich mit dem Kerngebiet seiner Anwendung zu beschäftigen und sich möglichst wenig mit Nebensächlichkeiten zu beschäftigen. Dies bedeutet aber nicht, dass der Entwickler von der Materie (z.B. Javascript, HTML) keine Ahnung mehr haben muss, nur weil er sich nicht unmittelbar damit beschäftigt.

4.4 Blackboard Architektur & iROS

In der vorangegangenen Ausarbeitung zu Anwendungen I wird die Blackboard-Technologie als ein für dieses Szenario mögliches Kommunikationsmodell erwähnt. Ebenso naheliegend ist daher der Einsatz der iROS Middleware, die ein Blackboard zur Kommunikation im Collaborative Workplace einsetzt. Soweit dies lose gekoppelte Systeme betrifft ist dies auch richtig. Jedoch gibt es Gründe, warum die Blackboard-Architektur für die Umsetzung für dieses Szenario nur bedingt bzw. nicht geeignet ist.

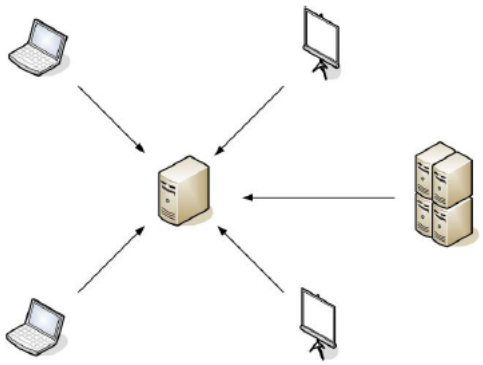

Abbildung 4.4: Zugriff der Teilnehmer auf das Blackboard. [Burfeindt, 2006]

Abbildung 4.4 zeigt das Blackboard als zentralen Nachrichtenserver in der Mitte der Grafik. Rund herum sind die Clients bzw. Server angeordnet. Prinzipiell ist es dabei egal ob ein Kommunikationsteilnehmer eines Blackboards Client oder Server ist, da der Zugriff nach dem gleichen Schema erfolgt.

[6] Mit Javascript ist hier auch die Kommunikation mit der AJAX-Technologie gemeint.

Da es sich bei Maps on Rails um eine Web-Anwendung handelt, die Daten per HTTP über eine Punkt-zu-Punkt-Verbindung austauscht, stehen die beiden Modelle im Konflikt zueinander. Ein Nachrichtenaustausch über das lose gekoppelte Blackboard-Modell zwischen Client und Web-Server wäre so nur möglich, wenn beide einen eigenen Agenten verwenden würden, der den Nachrichtenaustausch über das Blackboard übernimmt und für sie transparent macht. Dies wäre für Maps on Rails ein in keinem Verhältnis stehender Aufwand. Daher wird für die prototypische Umsetzung auf das Blackboard bzw. die Nutzung von iROS verzichtet und der direkte Weg der Client/Server-Kommunikation für übliche Web-Anwendungen gewählt.

Dennoch kann das Blackboard-Modell bei der Verteilung von (identischen) Nachrichten an mehrere Empfänger bei mobilen (lose gekoppelten) Systemen von Vorteil sein. Es macht die Verteilung der Nachrichten für den Server an alle Clients transparent und steigert die Fehlertoleranz des Systems beispielsweise bei Verbindungsabbrüchen. Als ein sinnvolles Einsatzgebiet im Collaborative Workplace an der Hochschule für Angewandte Wissenschaften Hamburg sei hier auf das Projekt „smart:shelf“ ([Hollatz, 2008], [Meißner, 2008] und [Urich, 2008]) hingewiesen.

5 Zusammenfassung

In dieser Ausarbeitung wurde das Szenario der Einsatzleitzentrale für Maps on Rails erneut vorgestellt. In Kapitel 3 wurden verschiedene Web-Anwendungen vorgestellt, die eine entweder thematische oder technische Verwandtschaft zu Maps on Rails aufweisen. Nachfolgend wurden in Kapitel 4 Technologien und Frameworks vorgestellt, die in Maps on Rails Verwendung finden und auch in den verwandten Anwendungen eingesetzt werden. Diese dienen als Ergänzung zu den Ausarbeitungen von Seminar und Anwendungen I, in denen näher auf die Anforderungen des Szenarios eingegangen wird.

5.1 Fazit

Die vorgestellte Technik scheint den Anforderungen des Szenarios gewachsen zu sein. Mit Blick auf die Masterarbeit wird diese Behauptung durch das Projekt in der Praxis untersucht. Für die Umsetzung von Maps on Rails kann auf das bereits vorhandene Bildmaterial der Kartendienste von Yahoo! bzw. Google zurückgegriffen werden. Ziel ist die prototypische Erstellung eines interaktiven Informationssystems mit Ruby on Rails.

Der Einsatz von Ruby on Rails ist dabei nur ein möglicher Weg zur Konstruktion von Web-Anwendungen bzw. Rich Internet Application. Das Echo Framework und das Google Web Toolkit stellen andere bzw. konkurrierende Ansätze mit der Programmiersprache Java dar.

5.2 Ausblick

Durch die Umsetzung des Prototypen Maps on Rails soll eine Bewertung des gewählten Ansatzes zur Erstellung von Web-Anwendungen erstellt werden. Diese ist durch das Szenario bedingt auf die Integration von Kartendiensten ausgelegt. Mehr über die Integration eines Kartendienstes in eine Ruby on Rails Anwendung ist im Projektbericht nachzulesen.

Die Umsetzung von Maps on Rails ist für das folgende Semester im Rahmen der Masterarbeit geplant. Dabei geht es neben der Realisierung des Systems auch um die Evaluation der

Sprache Ruby und des Ruby on Rails Frameworks im praktischen Einsatz und in Verbindung mit anderen Techniken in einem Verteilten System.

Literaturverzeichnis

[Burfeindt 2006] BURFEINDT, Lars: *Konstruktion einer Middleware für computergestützte Gruppenarbeit in ubiquitärer Systemumgebung*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2006. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/diplom/burfeindt.pdf

[Carfagno 2007] CARFAGNO, Virginio: *Evaluation des Google Web Toolkits durch Entwicklung einer ajaxbasierten Mind-Mapping-Anwendung*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/diplom/carfagno.zip

[Doernhoefer 2006] DOERNHOEFER, Mark: JavaScript. In: *SIGSOFT Softw. Eng. Notes* 31 (2006), Nr. 4, S. 16–24. – ISSN 0163-5948

[eyeOS Project 2007] EYEOS PROJECT: *EyeOS*. 2007. – URL http://www.eyeos.org/. – Zugriffsdatum: 02.12.2007

[Gamma u. a. 1995] GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph ; VLISSIDES, John: *Design Patterns*. Addison Wesley Professional, 1995. – ISBN 0-210-63361-2

[Garrett 2005] GARRETT, Jesse J.: *Ajax: A New Approach to Web Applications*. 2005. – URL http://adaptivepath.com/ideas/essays/archives/000385.php

[Google Inc. 2007a] GOOGLE INC.: *Google Docs & Spreadsheets*. 2007. – URL http://docs.google.com/. – Zugriffsdatum: 11.05.2007

[Google Inc. 2007b] GOOGLE INC.: *Google Maps*. 2007. – URL http://maps.google.de/. – Zugriffsdatum: 11.11.2007

[Google Inc. 2007c] GOOGLE INC.: *Google Maps API*. 2007. – URL http://www.google.com/apis/maps/. – Zugriffsdatum: 26.11.2007

[Hartmann 2008] HARTMANN, Leif: *Rich Internet Applications - Technologien*. 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master07-08-aw/hartmann/bericht.pdf. – In Vorbereitung.

[Hollatz 2008] HOLLATZ, Dennis: *Projektbericht: smart:shelf*. 2008. – In Vorbereitung.

[Köckritz 2006] KÖCKRITZ, Oliver: *Geschichte und Konzept von Collaborative Workspaces*. 2006. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2006/koeckritz/abstract.pdf

[Lewis 2007] LEWIS, Andrew: *Google Maps Applications with Rails and Ajax*. Apress, 2007

[Meißner 2008] MEISSNER, Stefan: *Projektbericht: smart:shelf*. 2008. – In Vorbereitung.

[O'Reilly 2005] O'REILLY, Tim: *What Is Web 2.0*. 2005. – URL http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

[Paulson Oct. 2005] PAULSON, L.D.: Building rich web applications with Ajax. In: *Computer* 38 (Oct. 2005), Nr. 10, S. 14–17. – ISSN 0018-9162

[Piening 2006] PIENING, Andreas: *RESCUE: Leitstand für Desaster-Szenarien*. 2006. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2006/piening/abstract.pdf

[Piening 2007] PIENING, Andreas: *RESCUE: Leitstand für Desaster-Szenarien*. 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master06-07-proj/piening/report.pdf

[Schempp 2006] SCHEMPP, Ruben: *Aktuelle Entwicklungen im Bereich von Programmiersprachen*, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, 2006. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/bachelor/schempp.pdf

[Schempp 2007] SCHEMPP, Ruben: *Anwendungen I: Verteilte Web-Anwendungen mit Ruby*. 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2007/schempp/bericht.pdf. – Zugriffsdatum: 07.02.2008

[Schempp 2008a] SCHEMPP, Ruben: *Projektbericht: Maps on Rails*. 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master06-07-proj/schempp/bericht.pdf. – Zugriffsdatum: 29.02.2008

[Schempp 2008b] SCHEMPP, Ruben: *Seminar: Maps on Rails*. 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master07-08/schempp/bericht.pdf. – Zugriffsdatum: 29.02.2008

[Tanenbaum 2003] TANENBAUM, Andrew S.: *Computer Networks*. Fourth Edition. Upper Saddle River, New Jersey : Prentice Hall, 2003. – ISBN 0-13-038488-7

[Tate 2005] TATE, Bruce: *Beyond Java*. O'Reilly, 2005

[Teranishi u. a. 2006] TERANISHI, Y. ; KAMAHARA, J. ; SHIMOJO, S.: MapWiki: a ubiquitous collaboration environment on shared maps. In: *Applications and the Internet Workshops, 2006. SAINT Workshops 2006. International Symposium on* (2006), Jan, S. 4 pp.–

[Thomas u. a. 2005] THOMAS, Dave ; FOWLER, Chad ; HUNT, Andy: *Programming Ruby: The Pragmatic Programmers' Guide*. Second Edition. The Pragmatic Programmers, 2005. – ISBN 0-9745140-5-5

[Thomas und Hannson 2006] THOMAS, Dave ; HANNSON, David H.: *Agile Web-Development with Rails*. Pragmatic Bookshelf, 2006

[Urich 2008] URICH, Jaroslaw: *Projektbericht: smart:shelf*. 2008. – In Vorbereitung.

[Vinoski 2006] VINOSKI, Steve: Enterprise Integration with Ruby. In: *Internet Computing, IEEE* 10 (2006), July-Aug., S. 91–95. – URL http://ieeexplore.ieee.org/iel5/4236/35969/01704762.pdf?isnumber=35969?=JNL&arnumber=1704762&arnumber=1704762&arSt=+91&ared=+95&arAuthor=+Vinoski%2C+S. – Issue 4. – ISSN 1089-7801

[Winograd 2001] WINOGRAD, Terry: *Architectures for Context*. 2001. – URL http://hci.stanford.edu/~winograd/papers/context/context.pdf. – Zugriffsdatum: 26.07.2007

[Yahoo! Inc. 2007] YAHOO! INC.: *Yahoo! Maps*. 2007. – URL http://maps.yahoo.com/. – Zugriffsdatum: 18.12.2007

[Yahoo! Inc. 2008] YAHOO! INC.: *Yahoo! Maps API*. 2008. – URL http://developer.yahoo.com/maps/ajax/V3.7/reference.html. – Zugriffsdatum: 07.02.2008