Bachelorarbeit

Benjamin Bellmann

Realisierung einer Social-Network-Webanwendung
auf Basis eines MVC-Frameworks und
Ajax-Technologie

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Benjamin Bellmann

Realisierung einer Social-Network-Webanwendung auf Basis eines MVC-Frameworks und Ajax-Technologie

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Gunter Klemke
Zweitgutachter : Prof. Dr. Birgit Wendholt

Abgegeben am 21. November 2008

Benjamin Bellmann

Thema der Bachelorarbeit

Realisierung einer Social-Network-Webanwendung auf Basis eines MVC-Frameworks und Ajax-Technologie

Stichworte

Soziales Netzwerk, Webanwendung, Ajax, MVC, CakePHP

Kurzzusammenfassung

Diese Bachelorarbeit befasst sich mit dem Entwurf und der Realisierung einer Social-Network-Webanwendung. Als Grundgerüst wird das MVC-Framwork CakePHP verwendet. Die Ajax-Technologie wird dabei verwendet, um interaktive Elemente zu erstellen. Das resultierende soziale Netzwerk selbst ist so konzipiert, dass dieses als Grundlage für zukünftige Webanwendungen einsetzbar ist.

Benjamin Bellmannn

Title of the paper

Development of a social network web application, based on a MVC-Framework and ajax technology.

Keywords

social network, web application, ajax, MVC, CakePHP

Abstract

This thesis (Bachelor) deals with the design and development of a social network. This social network is based on the MVC-framework CakePHP. The ajax-technology is used to allow interactivity.The resulting social network is designed to be a base application for further social network web applications.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation

Seit dem Niedergang der New Economy hat sich das Web rasant verändert. Die Art und Weise wie Menschen heute mit dem Web interagieren wäre noch vor wenigen Jahren nicht denkbar gewesen.
Seit der New Economy bis zum Web 2.0 haben sich die Technologien gewandelt und weiterentwickelt. Allem voran hat sich das Umfeld und die Infrastruktur des Webs selbst verändert. Breitbandanschlüsse und Flatrates gehören zu der Grundausstattung der breiten Masse. Diese weiterentwickelte Netzinfrastruktur erlaubt es, ganz andere Webanwendungen, zu erstellen und zu benutzen. Dienste wie Fotoportale, Videoplattformen und Kartendienste wären ohne ausreichende Bandbreite nicht nutzbar.
Das passive Nutzen des Webs gehört der Vergangenheit an. Das Mitmachnetz integriert die Benutzer und fordert sie auf, selbst aktiv zu werden und Inhalte zu publizieren. Die Angebote im Web2.0 bestehen aus dynamischen und interaktiven Elementen und nur noch wenig aus statischen Inhalten.
Eine besondere Rolle im Web 2.0 nehmen die sozialen Netzwerke ein. Soziale Netzwerke zählen zu den Communities. Diese bieten eine Plattform, um es Menschen mit gleichen und ähnlichen Interessen zu ermöglichen, sich zu treffen und auszutauschen. Communities erfreuen sich einer sehr großen Popularität. Viele erfolgreiche Communities bilden den Treffpunkt für Millionen von Benutzern. Bekannte Vertreter dieser Art sind das Wissensportal Wikipedia, die Videoplattform YouTube, das Fotoportal Flickr und das soziale Netzwerk Facebook. Das Besondere an sozialen Netzwerken ist, dass Wissen, Videos oder Fotos nicht im Mittelpunkt stehen, sondern die Benutzer selbst und deren Beziehungen untereinander.
Open Source Software, sowie die gesammelten Erfahrungen der Entwickler mit dem Web, haben die Kosten eines Markteintritts für Firmen bzw. Produkte erheblich reduziert. Die Erfahrungen in Projekten haben jedoch gezeigt, dass die Entwicklung eines „State of the Art“ sozialen Netzwerkes sehr aufwendig sein kann.
Aufgrund der umfangreichen Anforderungen bei der Entwicklung von sozialen Netzwerken ist diese Arbeit notwendig, um Lösungen zu finden, die die Entwicklung vereinfachen.
Der größte Unterschied zu anderen Lösungen, die auf das Gleiche abzielen, liegt in der einfachen Anwendung, der flexiblen Anpassbarkeit und der Übertragbarkeit auf andere Problemstellungen.

1.2 Fragestellung und Zielsetzung

Die Frage dieser Bachelorarbeit lautet: „Wie lässt sich der Entwicklungsaufwand für ein soziales Netzwerk reduzieren?“
Ziel ist es eine Lösung zu finden, um soziale Netzwerke mit reduzierten Entwicklungskosten und Entwicklungszeiten zu entwickeln. Dieses soll jedoch nicht auf Kosten des Funktionsumfangs geschehen. Alle für ein soziales Netzwerk entscheidenden Charakteristiken sollen integriert sein. Als Fallbeispiel wird ein soziales Netzwerk für Motorsportbegeisterte entwickelt. Diese Plattform soll Beziehungs- und Identitätsmanagement ermöglichen.
Diese Bachelorarbeit entsteht im Rahmen der Firma fast networx. fast networx hat in den letzten Jahren mehrere erfolgreiche soziale Netzwerke entwickelt. Zu diesen zählen die Community stadthunde.com für Hundehalter und comeunited.com - Community für Medien und Entertainment. Um den Entwicklungsaufwand für weitere Projekte zu reduzieren, soll eine wiederverwendbare und zukunftstaugliche Lösung gefunden werden.

1.3 Szenario

Dieses Szenario soll den typischen Umgang mit dem Motorsportportal verdeutlichen. Das Motorsportportal gilt dabei als Stellvertreter für eine Vielzahl an denkbaren sozialen Netzwerken, welche alle auf derselben Basis aufsetzten.
Ein Motorsportbegeisterter erfährt von dem Motorsportportal und ruft die Seite in einem Webbrowser auf. Er sieht, dass dort bereits viele Motorsportbegeisterte anwesend sind und es verschiedene Wege gibt sich mit Gleichgesinnten auszutauschen. Um diese Möglichkeiten wahrzunehmen, beschließt er sich beim Motorsportportal zu registrieren. Um sich zu registrieren, gibt er E-Mail-Adresse und Passwort an. Im folgenden wird er als Benutzer A bezeichnet.
Als Erstes wird Benutzer A daraufhin gewiesen, dass es von Vorteil für ihn selbst und andere ist, wenn er sich in einem Benutzerprofil vorstellt. Dieses tut Benutzer A, und trägt einige Informationen über seine Interessen und Hobby in sein Benutzerprofil ein.
Nun kann Benutzer A sich mit anderen Benutzern des Motorsportportals in einem Diskussionsforum austauschen. Benutzer A kann seine Meinung zu bereits existierenden Themen kundtun oder ein eigenes Thema eröffnen, zu welchem dann die anderen Benutzer Kommentare abgeben können.
Ein Gruppensystem erlaubt Benutzer A, seine Zugehörigkeit zu einem bestimmten Sachverhalt auszurücken. Da Benutzer A sich vor allem für Rallye interessiert, tritt er der Gruppe „Rallye-Freunde“ bei. Innerhalb dieser Gruppe kann Benutzer A sich mit anderen Gruppenmitgliedern in einem gruppeninternen Diskussionsforum austauschen.
Benutzer A stellt fest, dass einige interessante Beiträge innerhalb des Diskussionsforums von ein und demselben Benutzer stammen, im folgenden Benutzer B genannt. Benutzer A möchte nun direkt mit Benutzer B kommunizieren.

Dazu benutzt Benutzer A das interne Nachrichtensystem des Motorsportportals, welches den direkten Informationsaustausch zwischen zwei Benutzern ermöglicht. Nach mehreren ausgetauschten Nachrichten stellt Benutzer A fest, dass er sich gut mit Benutzer B versteht. Auf dessen Benutzerprofil sieht Benutzer A, das nicht alle Informationen über Benutzer B abrufbar sind, sondern nur für Freunde des Benutzers B. Daher möchte Benutzer A eine Online-Freundschaft mit Benutzer B eingehen. Dazu stellte Benutzer A eine Kontaktanfrage an Benutzer B. Benutzer B bestätigt die Freundschaftseinladung. Dadurch kann Benutzer A nun alle Benutzerprofildaten des Benutzers B abrufen.

1.4 Vorgehen und Inhalte

In **Kapitel 2** werden die funktionalen und nicht funktionalen Anforderungen an ein soziales Netzwerk beschrieben. Zum besseren Verständnis werden zuvor die Grundlagen von sozialen Netzwerken und Webanwendungen erläutert.

Kapitel 3 behandelt die Marktanalyse unter den Gesichtspunkten der Marktentwicklung und Marktstrukturierung.

Kapitel 4 erläutert die Grundlagen der verwendeten Technologien. Der Schwerpunkt liegt hierbei auf dem MVC-Framework CakePHP und der Ajax-Technologie.

In **Kapitel 5** wird das Systemdesign vorgestellt. Dabei wird eine Webanwendung entworfen, welche als Referenzarchitektur bei der Erstellung von sozialen Netzwerken eingesetzt werden kann.

In **Kapitel 6** wird die Systemrealisierung beschrieben.

Kapitel 7 beinhaltet Zusammenfassung und Ausblick.

2 Anforderungen

In diesem Kapitel werden die funktionalen und nicht funktionalen Anforderungen an eine Webanwendung für soziale Netzwerke beschrieben. Zum besseren Verständnis werden zuvor die Grundlagen von sozialen Netzwerken und Webanwendungen erläutert.

2.1 Social Networks

Mit der Weiterentwicklung von Internetdiensten, welche sich mehr und mehr auf vom Benutzer erzeugte Inhalte fokussieren, kurz Web 2.0, haben auch soziale Netzwerke Einzug ins Internet gehalten. Sie erleichtern dort die Kommunikation und Kollaboration. Sie stellen neben Weblogs und Wikis einen wesentlichen Bestandteil des Web2.0 da.

Soziale Netzwerke sind eine besondere Art der Sozialstrukturen. Ein soziales Netzwerk besteht dabei aus Netzknoten und Relationen zwischen diesen. Jeder Netzknoten stellt ein Individuum da. Die Relationen bilden die Beziehungen zwischen diesen Individuen. Der Begriff soziales Netzwerk bezeichnet also ein Beziehungsgeflecht, welches Menschen mit anderen Menschen verbindet. Die Beziehungen können dabei verschiedenster Art sein wie beispielsweise Verwandtschaft oder Nachbarschaft. Der Sinn an einem sozialen Netzwerk für jeden Einzelnen ergibt sich aus den potenziellen Möglichkeiten, die aus seinen Beziehungen hervorgehen können.

Ein soziales Netzwerk lässt sich anhand seines Umfangs und seiner Dichte bewerten. Der Umfang bezeichnet die Anzahl an involvierten Individuen. Die Dichte gibt Auskunft darüber wie fragmentiert oder verbunden die einzelnen Teilnehmer eines Beziehungsgeflechts sind.

Jedes Mitglied eines solchen sozialen Netzes kann von sich Informationen innerhalb seines eigenen Profils veröffentlichen. Diese Repräsentation des Benutzers durch sein Profil entspricht in etwa den Knoten herkömmlicher sozialer Netzwerke. Auch das Herstellen von Beziehungen ob zu Bekannten oder real nicht Bekannten ist möglich.

Soziale Netzwerke lassen sich hierbei in zwei Kategorien unterteilen. Die erste Kategorie umfasst die themenbezogenen sozialen Netzwerke, welche eine sehr hohe Homogenität der Benutzer aufweisen. Bekannte Vertreter dieser Art sind Facebook und studiVZ, welche sich primär auf Studenten spezialisiert haben und beide mehrere Millionen Nutzer haben. Auf der anderen Seite stehen generische

Plattformen mit sehr breit gefächerten Interessen der Benutzer. Der bekannteste Vertreter dieser Kategorie ist MySpace mit weltweit über 100 Millionen aktiven Nutzern.

2.2 Webanwendungen

Eine Webanwendung ist eine Software mit welcher ein Benutzer ausschließlich mittels eines Webbrowsers interagiert. Ein oder mehrere Webserver stellen diese Anwendung für Zugriffe aus dem Internet oder Intranet zur Verfügung. Der Großteil der Programmausführung findet dabei auf dem Webserver statt. Nur spezielle oder einfache Skripte werden vom Webbrowser auf der Clientseite interpretiert und ausgeführt. Somit lässt sich ein dezentrales Arbeiten gewährleisten, wodurch kein Konfigurationsaufwand auf Clientseite notwendig ist. Diese geringen Anforderungen an die Clientseite stellen einen großen Unterschied zu Client-Server Anwendungen da, bei welcher immer eine komplette Anwendung auf dem Client einzurichten ist. Die Eigenschaft des zentralen Erweiterns und Verwaltens ohne clientseitigen Aufwand ist einer der Hauptgründe für die starke Popularität von Webanwendungen [1].
Ein weiterer wichtiger Punkt der Akzeptanz dieser Art von Anwendungen ist die Plattformunabhängigkeit auf Clientseite. Das Betriebssystem auf Clientseite ist dabei vollkommen irrelevant, da das Programm nur innerhalb eines Webbrowsers operiert. Jedoch ist anzumerken, dass es Unterschiede bei der Interpretation von Webserver Antworten je nach Webbrowser geben kann. Diese Unterschiede beziehen sich sowohl auf die Darstellung als auch auf das Verhalten der Webseiten. Gerade das unterschiedliche Verhalten bei der Behandlung von Ereignissen kann zu Funktionseinschränkungen führen.
Mittels Webstandards soll es ermöglicht werden, dass Webseiten in jeden Webbrowser korrekt dargestellt werden. Die Webstandards sind Richtlinien für die Darstellung von Inhalten, in einem Webbrowser, aber auch Richtlinien für die Webseiten selbst. Diese Webstandards werden vom „World Wide Web Consortium“(W3C) und der „Internet Engineering Task Force“(IETF) festgelegt. Die bekanntesten Webstandards sind HTML, CSS, DOM und ECMAScript. Zur Überprüfung der Einhaltung dieser Standard durch Webbrowser wurde der ACID-Test entwickelt welcher darüber Auskunft gibt ob bzw. in welchem Grad ein Webbrowser den Standards genügt. Um die Webseiten auf Einhaltung der Standards zu überprüfen, wurden Validatoren entwickelt. Einer der bekanntesten Validatoren ist der Markup Validation Service des W3C.
Webbrowser neuerer Generation halten sich zunehmend an die gültigen Webstandards, womit die Probleme unterschiedlicher Interpretation weitestgehend passé sind.

2.2.1 Besonderheiten bei der Entwicklung von Webanwendungen

Webanwendungen stellen ähnliche Anforderungen an den Entwicklungsprozess wie beispielsweise Desktopanwendungen, jedoch unterscheiden sich die Ausprägungen dieser Anforderungen z. T. erheblich. Aufgrund der Tatsache das Webanwendungen einen wenig komplexen Auslieferungsmechanismus haben, ist das schnelle Realisieren von Projekten oftmals ein entscheidender Faktor um sich am Markt zu positionieren. Sowohl inhaltliche als auch technologische Anforderungen an eine Webanwendungen stellen sich häufig erst während der Entwicklung heraus. Auf neue Marktsituationen muss sich schnell eingestellt werden, da die Konkurrenz nur einen Klick entfernt ist.
Aufgrund dieser kurzen Entwicklungsphasen und sich ändernden Anforderungen ist es auch schwierig, die Anforderungen vollständig zu definieren. Die Entwicklung ist ein iterativer, sehr dynamischer Prozess, der die Auslieferung von Zwischenprodukten mittels Prototypen notwendig macht, damit der Auftraggeber frühzeitig auf die Projektumsetzung Einfluss nehmen kann [2].
Um diesen Besonderheiten in der Praxis am besten Rechnung zu tragen, bietet sich die Verwendung eines agilen Vorgehensmodelles an.

2.3 Quelle der Anforderungen

Die im folgenden beschriebenen funktionalen und nicht funktionalen Anforderungen setzen sich aus mehreren Quellen zusammen. Zum einen gilt der Anspruch „State of the Art“ zu sein zum anderen die subjektiven Anforderungen des Unternehmens fast networx.

2.4 Anforderungen an eine Webanwendung für soziale Netzwerke

Die folgenden Anforderungen unterteilen sich in funktionale und nichtfunktionale Anforderungen.
Die funktionalen Anforderungen beschreiben was das Produkt tun soll bzw. wie es sich nach außen verhalten soll. Nichtfunktionale Anforderungen drücken die Beschaffenheit und Eigenschaften des Produktes aus. Mit nichtfunktionalen Anforderungen werden im Allgemeinen Rahmenbedingungen festgelegt, welche das Produkt einhalten muss.

2.4.1 Funktionale Anforderungen an eine Webanwendung zum Erstellen von sozialen Netzwerken

Es soll eine Lösung gefunden werden, um den Entwicklungsaufwand für soziale Netzwerke zu reduzieren. Die Lösung soll alle integralen Bestandteile eines sozialen Netzwerkes enthalten. Das Verwenden dieser Bestandteile soll ohne bzw. mit geringst möglichem Entwicklungsaufwand durchführbar sein. Zu diesen Bestandteilen zählen die im folgenden Kapitel aufgezählten Anforderungen. Jedoch soll es Wege geben diese Grundfunktionalitäten zu verändern und zu erweitern.
Funktionen, die eigentlich außerhalb des Fokus eines sozialen Netzwerks liegen, sollen sich jedoch auch an das System anbinden lassen.

2.4.2 Funktionale Anforderungen an ein soziales Netzwerk

Identitätsmanagement: Jeder Benutzer verfügt über sein eigenes Benutzerprofil, welches eine Repräsentation des jeweiligen Benutzers darstellt. Die auf diesem Benutzerprofil verfügbaren Personendaten werden von dem Benutzer des Netzwerks selbst eingepflegt.

Beziehungsmanagement: Haben sich zwei Benutzer mit gemeinsamen Interessen gefunden sollen sie die Möglichkeit haben diese Kontaktbeziehung auch innerhalb der Plattform abzubilden. Diese Verbindung erlaubt besseren Informationsfluss und einfacheren Zugriff untereinander.

Diskussionsforum: Der Austausch zu Interessen stellt eine weitere Möglichkeit da, um in Kontakt mit anderen Benutzern zu treten. Ein Diskussionsforum soll es den Benutzern erlauben, zu verschiedenen Themen, Meinungen und Erfahrungen auszutauschen.

Gruppen: Gruppen sollen eine Gemeinschaft von Benutzern mit sehr ähnlichen Interessen abdecken. Die Homogenität der Benutzer in einer Gruppe ist aufgrund der geringen thematischen Ausbreitung um ein Vielfaches höher, als in einem allgemeinen Diskussionsforum. Mittels einer Gruppenmitgliedschaft kann ein Benutzer seine Zugehörigkeit oder Überzeugung zu einem bestimmten Sachverhalt ausdrücken.

Nachrichtensystem: Der direkte Informationsaustausch zwischen Benutzern soll mittels eines internen Nachrichtensystems möglich sein. Dieses Nachrichtensystem funktioniert im Prinzip wie der Austausch von E-Mails, jedoch nur innerhalb des sozialen Netzwerkes.

Veranstaltungen: Im Motorsport dreht sich vieles um Veranstaltungen, wie beispielsweise Rennen oder Autoshows. Aus diesem Grund soll die Plattform das Veröffentlichen und Eintragen von Veranstaltungen erlauben.

Geoinformationen: Als Feature soll es eine Routine geben, die einem Interessenten anhand seiner eigenen Adressdaten, die metrische Entfernung zur Veranstaltung ermittelt. Ein Benutzer erhält dadurch die Möglichkeit einfacher abzuschätzen, ob eine Veranstaltung für ihn von Interesse ist oder nicht.

Schnittstelle CMS: Eine Schnittstelle für Content Management Systeme soll den bidirektionalen Austausch von Inhalten ermöglichen. Damit können Inhalte aus der Plattform in anderen Webseiten partiell eingebunden werden, um als Werbung für das Motorsportportal zu dienen. Die Importschnittstelle wird verwendet, um redaktionell gepflegte Inhalte in die Plattform einzubinden.

Suchmaschinenoptimierung: Suchmaschinenoptimierung kurz SEO genannt, ist entscheidend um eine große Anzahl potenzieller Nutzer zu erreichen. Daher soll die Anwendung von vornherein auf Suchmaschinenoptimierung ausgelegt werden.

2.4.3 Nichtfunktionale Anforderungen

Benutzbarkeit: Die User sind vom Web wenig komplexe Anwendungen gewohnt. Auch wenn die Anwendungen eine gewisse Komplexität mitbringen, sollen sie zumindest so aussehen als seien sie wenig komplex. Die Handhabung der Webanwendung soll durch Interaktivität, wie von Desktopanwendungen bekannt, vereinfacht werden.

Leistung und Effizienz: Die Bereitstellung bzw. Darstellung der angeforderten Informationen soll scheinbar unmittelbar erfolgen. Auch für Benutzer mit geringer Bandbreitenkapazität soll die Ladezeit kaum vorhanden sein.

Verfügbarkeit: Verfügbarkeit ist eine sehr wichtige Anforderung für ein soziales Netzwerk. Die Benutzer müssen immer in der Lage sein ihre Bekannten zu erreichen und mit ihnen zu kommunizieren.

Erweiterbarkeit und Änderbarkeit: Dies beides sind entscheiden Anforderungen, um auf neue Marktsituationen reagieren zu können. Im Web 2.0 Jargon spricht man auch von der „Ewigen Beta“ was ausdrücken soll, dass eine Software nie fertig ist, sonder immer weiterentwickelt wird.

Vertraulichkeit: Die Benutzer eines sozialen Netzwerkes geben in der Regel sehr viele Informationen von sich Preis. Ein Schutz dieser Daten ist sehr wichtig, da aufgrund von Missbrauch das Vertrauen in die gesamte Plattform sehr schnell schwinden kann. Ein Login- und Rechteverwaltungsmechanismus muss vorhanden sein, um einen unautorisierten Zugriff auszuschließen.

Wiederverwendbarkeit: Einige Funktionalitäten sind Bestandteil eines jeden sozialen Netzwerkes. Auch verschiedenste Ausprägungen besitzen einen immer gleichen Grundstock an Funktionen. Diese Grundfunktionen sollen so aufgebaut sein, dass mit geringem Entwicklungsaufwand neue soziale Netzwerke erstellt werden können.

Mehrsprachigkeit: Es soll möglich sein, die Anwendung Nutzern aus verschiedensten Ländern anzubieten. Dafür müssen sich vor allem die textuellen Informationen an die länderspezifischen Sprachen anpassen lassen.

Testbarkeit: Es sollen Mechanismen existieren, welche die Erfüllung und Qualität der umgesetzten Anforderungen verifizieren können.

3 Marktanalyse

Innerhalb dieses Kapitels wird versucht herauszufinden ob es für das Unternehmen fast networx von Vorteil ist eine externe Lösung zu verwenden oder ob eine Eigenentwicklung realisiert werden muss. Um die Entscheidung treffen zu können, wird eine Marktanalyse durchgeführt.
Das Kapitel Marktentwicklung gibt darüber Auskunft, in welchem Entwicklungszustand, Wachsen, Stagnieren oder Schrumpfen, sich der Markt für soziale Netzwerke befindet.
Die Marktstrukturierung zeigt, in welche Produkte sich der Markt für Lösungen zum Erstellen von sozialen Netzwerken gliedert.

3.1 Marktentwicklung

Seit dem Niedergang der New Economy sind einige sehr erfolgreiche Unternehmen im Web-Umfeld entstanden. Die Umsätze und Erträge sind höher und beständiger, als vor dem Platzen der Dotcom-Blase. In nur wenigen Jahren sind Plattformen entstanden, die einen Marktwert von mehreren Milliarden Dollar besitzen.
Zu diesen erfolgreichen Unternehmen zählen z. B. die sozialen Netzwerke Facebook und MySpace. Das Unternehmen Facebook existiert erst seit vier Jahren. Inzwischen sind bereits über 100 Millionen Nutzer angemeldet und das Unternehmen hat einen Marktwert von 15 Milliarden Dollar [3][4]. MySpace wurde im Jahre 2003 gegründet und konnte seinen 100 millionsten Benutzer schon drei Jahre später feiern.
Beide sozialen Netzwerke sind immer noch im Wachstum begriffen und verzeichnen im Schnitt 250.000 neue Nutzer pro Tag. Das Marktforschungsunternehmen Forrester Research schätzt den gesamten Web2.0-Markt als sehr wachstumsstark ein.
„Enterprise spending on Web 2.0 technologies will grow strongly over the next five years, reaching $4.6 billion globally by 2013, with social networking, mashups, and RSS capturing the greatest share. “[5]

3.2 Marktstrukturierung

Die auf dem Markt erhältlichen Lösungen, um ein eigenes soziales Netzwerk zu betreiben, werden zumeist als „White Label Social Networks“ bezeichnet. Der Begriff

„White Label“ bezieht sich darauf, dass die Anbieter eine Anwendung zur Verfügung stellen, welche sich im Aussehen so anpassen lässt, dass es dem eigenen Corporate Identity entspricht.
Diese Produkte lassen sich grob in zwei Bereiche aufteilen. Die „Hoster“, welche die komplette Anwendung auf einen ihrer Server betreiben. Eine solche Community ist dann meist unter einer Subdomain des Hosters erreichbar und kann bequem online administriert werden. Besondere Kenntnisse sind meist nicht erforderlich um die Anwendung zu individualisieren, jedoch ist dies nur in einem fest gegebenen Rahmen der Fall.
Auf der anderen Seite stehen Lösungen, welche sich auf eigenen Servern einsetzen lassen. Ein wichtiger Vorteil ist, dass die Anwendungsdaten komplett zur Verfügung stehen, womit diese leicht in andere bereits bestehende Unternehmensanwendungen eingebunden werden können. Darüber hinaus lassen sich selbst gehostete Anwendungen leichter an die eigenen Bedürfnisse anpassen. Folglich sind aber spezielle Kenntnisse und einer höherer Konfigurationsaufwand nötig.
Für beide Produktgruppen existieren sehr viele Anbieter am Markt. Einer der bekanntesten Vertreter ist Ning. Ning wurde im Jahre 2005 von Netscape-Mitgründer Marc Andreessen entwickelt und zählt zu denn Hostern. Etwa 275.000 Netzwerke werden zurzeit von Ning gehostet. Zu den angebotenen Funktionen von Ning gehören Videos, Fotos, Blogs, Foren, Gruppen, Import von Facebook-Applications und RSS Im/Export um nur einige zu nennen.
Der bekannteste Vertreter der zweiten Kategorie ist KickApps. Das Produkt KickApps existiert seit 2005 und wird bisher von 40.000 Kunden eingesetzt. Der Vorteil von KickApps liegt in den umfangreichen Anpassungsmöglichkeiten. KickApps stellt Entwicklern sowohl eine API als auch Entwicklungswerkzeuge zur Verfügung um KickApps individueller gestalten zu können

3.3 Marktbewertung und Fazit

Der Markt für Anwendungen zum Kreieren eigener sozialer Netzwerke ist sehr jung und vielfältig. Viele Unternehmen sind dabei, Lösungen zu entwickeln, welche sich nur wenig von der Konkurrenz unterscheiden. Diese Mannigfaltigkeit ist ähnlich wie einst bei Content Management Systemen. Früher existierten sehr viele Anbieter für Content Management Systeme, von denen sich nur wenige am Markt behauptet haben. Daher ist es schwierig abzuschätzen welche „White Label Social Networks“ Angebote eine Zukunft haben und welche nicht.
Ein starker Nachteil ist, dass Erweiterbarkeit und Anpassbarkeit immer nur in meist engen vorgegebenen Rahmen möglich sind. Auch ist die Lizenzpolitik einiger Anbieter nicht immer ideal, beispielsweise müssen oftmals die Werbeeinnahmen mit dem Anbieter geteilt werden. Außerdem würde die Benutzung eines solchen Systems das Unternahmen fast networx stark abhängig von dem jeweiligen Anbieter machen. Aus diesen genannten Gründen wird eine eigene Lösung entwickelt.

4 Grundlagen

Im Folgenden werden die technologischen Grundlagen und Konzepte für die Entwicklung einer erweiterbaren Webanwendung für Social Software vorgestellt werden. Dies geschieht exemplarisch mit der Vorstellung des Frameworks CakePHP. CakePHP ist ein in PHP geschriebenes MVC-Framework und wird seit dem Jahre 2005 entwickelt. Es ist unter der MIT-Lizenz freigegeben und ist somit Open Source. Nach der wachsenden Popularität von Ruby on Rails wuchs auch der Wunsch nach einem ähnlichen System für PHP. Aber nicht nur das MVC-Entwurfsmuster wurde als Grundlage von Ruby on Rails übernommen sondern auch Prinzipien wie Don't Repeat Yourself und „Convention over Configuration“.
Nachfolgend sind die einzelnen technischen Grundlagen erläutert.
In dem Teilkapteil MVC wird das Entwurfsmuster Model-View-Controller allgemein erläutert. Nachfolgend werden die einzelnen MVC-Aspekte, Model, View und Controller, in eigenen Teilkapiteln unter den CakePHP spezifischen Eigenschaften erläutert.
Im Teilkapitel „Ablauf eines Requests“ wird ein typischer Request-Response-Ablauf in CakePHP beschrieben.
Unter der Überschrift „Convention over Configuration„ wird das Prinzip erläutert, wie durch festgelegt Konventionen der Entwicklungsaufwand verringert werden kann.
Der Abschnitt Access Control List beinhaltet den Aufbau der Rechtevergabe und Rechteverwaltung von CakePHP.
Im Teilkapitel Lokalisierung und Internationalisierung werden die von CakePHP angebotenen Funktionalitäten erläutert, um die erstellte Webanwendung, kulturellen und sprachlichen Gegebenheiten anzupassen.
Im Abschnitt Testen werden die Möglichkeiten aufgezeigt, mit welcher eine CakePHPbasierte Webanwendung, nach Erfüllung und Qualität der Anforderungen, überprüft werden kann.
Der Abschnitt Ajax gibt einen allgemeinen Überblick zur Ajax-Technologie und zeigt die Unterschiede zu klassischen Webanwendungen.
Die Verwendung des MVC-Frameworks CakePHP ist Vorgabe von fast networx.

4.1 MVC

Mit dem Entwurfsmuster MVC als Systemgrundlage versucht CakePHP den Anforderungen nach Einfachheit, Wiederverwendbarkeit, Erweiterbarkeit und

Veränderbarkeit nachzukommen. Bei MVC wird die Anwendung in drei Schichten unterteilt, Model, View und Controller.
Das **Model** kapselt die Anwendungsdaten und stellt Funktionen für den Datenzugriff bereit.
Die **View** übernimmt die Darstellung der Daten.
Der **Controller** steuert die Anwendung durch die Behandlung von Ereignissen.
Dies hat den Vorteil, das eine Änderung in einer Schicht nicht unbedingt zur Folge hat, dass eine andere Schicht auch geändert werden muss. Bei einem Umstieg auf ein anderes Datenbanksystem beispielsweise wären keine Änderungen in View und Controller durchzuführen.
Durch diese Entkopplung von Benutzerschnittstelle, Geschäftslogik und Daten, ist es auch möglich, die Kompetenzen der Entwickler besser zur Geltung zu bringen. Auch braucht nicht jeder alles wissen, sondern kann sich auf seinen Teil konzentrieren. So kann ein Webdesigner sich rein um die Darstellung kümmern und muss nicht auch Programmcode verstehen [6].

4.1.1 Model

Die Art und Weise wie Models in CakePHP benutzt werden können, ist eines der mächtigsten Features von CakePHP. In erster Linie sind Models Repräsentationen einer Datenbanktabelle bzw. eines Datensatzes. Mittels des Models können alle CRUD-Operationen auf einer Tabelle durchgeführt werden. CakePHP kapselt die Aufrufe zur Datenbank hin durch eigene Methoden zur Datenmanipulation. Diese speziellen Methoden ermöglichen vermeintlich komplexe Datenbankabfragen, in einfacher Form zu beschreiben.
Im Model selbst können spezielle Bedingungen definiert werden, unter welcher das Model mit seiner entsprechenden Tabelle arbeitet. So kann beispielsweise festgelegt werden welche Attribute ausgelesen werden sollen oder in welcher Ordnung die Resultate zurückgegeben werden. Das Framework erlaubt auch die Assoziation mehrerer Models. Durch diese Assoziation ist es möglich, mit nur einem Aufruf, mehr als nur eine Tabelle abzufragen [7]. Tabelle 1 zeigt die verschiedenen Arten der Assoziation, mittels eines Vergleichs der UML und CakePHP Assoziation.

UML - Assoziation	CakePHP - Assoziation	Beispiel
one to one	hasOne	Ein Benutzer hat ein Gästebuch.
one to many	hasMany	Ein Gästebuch hat mehrere Einträge.
one to many	belongsTo	Ein Gästebucheintrag gehört zu einem Gästebuch.
many to many	hasAndBelongsToMany	Ein Benutzer ist in mehreren Gruppen.

Tabelle 1: Model - Assoziationen

Auch die Definitionen der Assoziation zu anderen Models innerhalb eines Models können spezielle Bedingungen erhalten. So kann das Attribut „dependent“ dafür sorgen, dass alle Kindobjekte gelöscht werden sollen, sobald das Elternobjekt gelöscht wird. Dadurch würden beim Löschen eines Gästebuchs automatisch alle verbundenen Gästebucheinträge mit aus der Datenbank entfernt.
Bei jeder Abfrage kann optional ein Rekursionslevel angegeben werden. Dieser Wert stellt die Reichweite der Assoziation da. So kann es gewollt sein zu einem Benutzer sein Gästebuch und alle Gästebucheinträge zu erhalten, aber ohne die Benutzerdaten der Verfasser dieser Gästebucheinträge.
Zu jedem Model können Validierungskriterien definiert werden. Diese Sorgen dafür, dass nur gültige Werte die Datenbank erreichen. Sollte ein Kriterium nicht erfüllt worden sein, wird der gesamte Schreibvorgang abgebrochen. CakePHP bietet dem Entwickler eine große Palette von Validierungskriterien von Emailvalidierung bis hin zur Einzigartigkeitsprüfung eines Feldes. Aber auch das Selbstverfassen eigener Validierungsregeln ist möglich. Zudem können auch mehrere Validierungskriterien aggregiert werden. Somit kann ein komplexes Regelwerk für jedes einzelne Feld definiert werden.

Mit sogenannten Model-Behaviors ist es möglich, das Verhalten eines Models zu beeinflussen. In einem Behavior kann bestimmt werden, für welche Operationen sich das Model anders als gewohnt verhalten soll. Auch stellen sie eine Art Bibliothek da, da sie in mehreren Models verwendet werden können. So könnte man ein Behavior erstellen um alle Änderungen eines Datensatzes, zu protokollieren. Bei jeder Save-Operation würden dann zusätzliche Informationen über die Historie des Datensatzes mit gespeichert.

4.1.2 View

Die Views enthalten die gesamte Darstellungslogik, um die vom Controller übergebenen Daten zu formatieren und dem User zu präsentieren. Sie stellen die

Vorlagen für die Ausgabe da. Die Views in CakePHP lassen sich in vier Kategorien unterteilen.

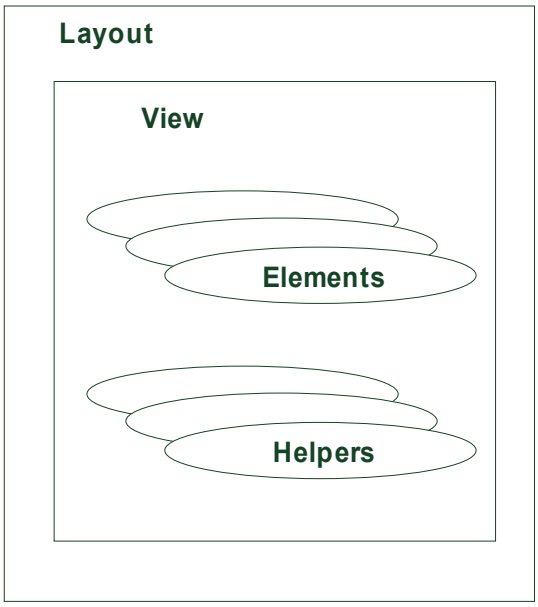


Abbildung 1: View Zusammensetzung

Layout: Layouts stellen den Rahmen der View da. In ihnen ist das Grundgerüst der Webseite festgelegt, Seitenbereiche wie Header, Footer und Platzhalter für den eigentlichen Inhalt. Normale Views werden fast ausschließlich innerhalb von Layouts dargestellt. Ausnahme sind nur die Ajax Request, welche kein Layout benötigen.

Elemente: Sind kleine, wiederverwendbare Präsentationsschnipsel. Sie werden innerhalb von Views oder des Layouts eingebunden. Eine Loginbox beispielsweise wäre prädestiniert für ein Element, da diese auf vielen Seiten angezeigt wird, solange der User nicht eingeloggt ist.

Helper: Helper enthalten Präsentationslogik, welche in Layouts, Elements und Views benutzt werden können. Sie stellen Hilfsmittel für den Bau von HTML-Seiten dar. So lassen sich mit einem Formular-Helper auf einfache Art und mit einer gewissen Zeitersparnis HTML-Formulare erstellen.

View: Die View selbst stellt die seitenspezifischen Inhalte dar. Die View wird aus einem Controller heraus aufgerufen und erhält vom Selbigen die darzustellenden Inhalte.

4.1.3 Controller

Ein Controller enthält die Geschäftslogik für einen Bereich der Anwendung. Die Schnittstellen, die der Controller nach außen anbietet, werden Actions genannt und sind normale PHP-Methoden. Die Controller-Actions sind die Einstiegspunkte einer HTTP-Anfrage. So würde die URL www.example.com/guestbooks/view/1 an den Controller guestbooks geleitet und die Methode View mit dem Parameter 1 aufgerufen. Jeder spezifische Controller wird abgeleitet vom Application-Controller. Dieser Application-Controller wird benutzt, um anwendungsweit für alle Controller gemeinsame Methoden zu definieren. Die Controller selbst sind standardmäßig mit einem Model verbunden. Jede der Controller-Actions ist mit einer View verbunden. Der Controller stellt also das Bindeglied zwischen Model und View da. Die Abbildung 2 zeigt den Quelltext eines Controllers welcher die Zusammenarbeit der einzelnen MVC-Aspekte verdeutlichen soll.

```
class GuestbooksController extends AppController {
      // Controller-Action
      function view($id) {
          // Zugriff auf das Model Guestbook
          $guestbook = $this->Guestbook->find('id'=>$id);
          // Der View die Daten des guestbook zu Verfügung stellen
          $this->set('guestbook',$guestbook);
      }
}
```

Abbildung 2: Zusammenwirken von MVC (Quelltext)

Controller bieten Callbackroutinen, um in unterschiedlichen Verarbeitungsphasen Aktionen auszuführen. Hier sollen die Wichtigsten benannt sein.

BeforeFilter: Diese Funktion wird vor jeder Controller-Action ausgeführt. Dies ist ein idealer Zeitpunkt, um beispielsweise Zugriffsrechte zu bestimmen.

AfterFilter: Wird nach jeder Controller-Action ausgeführt. Hier können neben den schon vom Controller an die View übergebenen Werten noch zusätzliche Werte übergeben werden. Dies sind dann allgemeine Informationen, welche Seiten übergreifend im Layout eingebunden werden.

Um auch in den Controllern dem Prinzip von „Dont Repeat Yourself“ entgegen zukommen, können häufig verwendete Funktionen in Components ausgelagert werden. Diese Components können von jedem beliebigen Controller aus eingebunden werden und können somit als eine Art Funktionsbibliothek angesehen werden. Durch diese Kapselung ist es auch möglich, Components projektübergreifend einzusetzen. CakePHP bietet dem Entwickler schon einige vorgefertigte Components an. Session-Handling, Emails und Authentication sind nur einige der vorgefertigten Components.

4.2 Ablauf eines Requests

Ein typischer CakePHP-Request gliedert sich in mehrere Schritte. Jeder Request hat immer denselben Einstiegspunkt. Die Anfrage wird vom Dispatcher entgegengenommen und ausgewertet. Zunächst wird die URL in einzelne Abschnitte zerlegt. Der Router ermittelt nun anhand dieser Abschnitte den zugehörigen Controller und die Action.
Der Aufruf des Controllers selbst ist in mehrere Phasen gegliedert. In der Phase unmittelbar vor dem Aufruf der konkreten Action werden allgemeine Abfragen definiert wie Authentisierung und Autorisierung. Die zweite Phase beinhaltet das Ausführen der eigentlichen angeforderten Businesslogik. Dazu kann mittels der Models auf die Datenbank zugegriffen werden, um angeforderte Daten auszuliefern oder zu verändern. Am Ende der Controller-Action werden der Präsentationsschicht die ermittelten Anzeigedaten übergeben. Anschließend wird die zur Controller-Action zugehörige View aufgerufen und dem Benutzer im Webbrowser dargestellt [8].

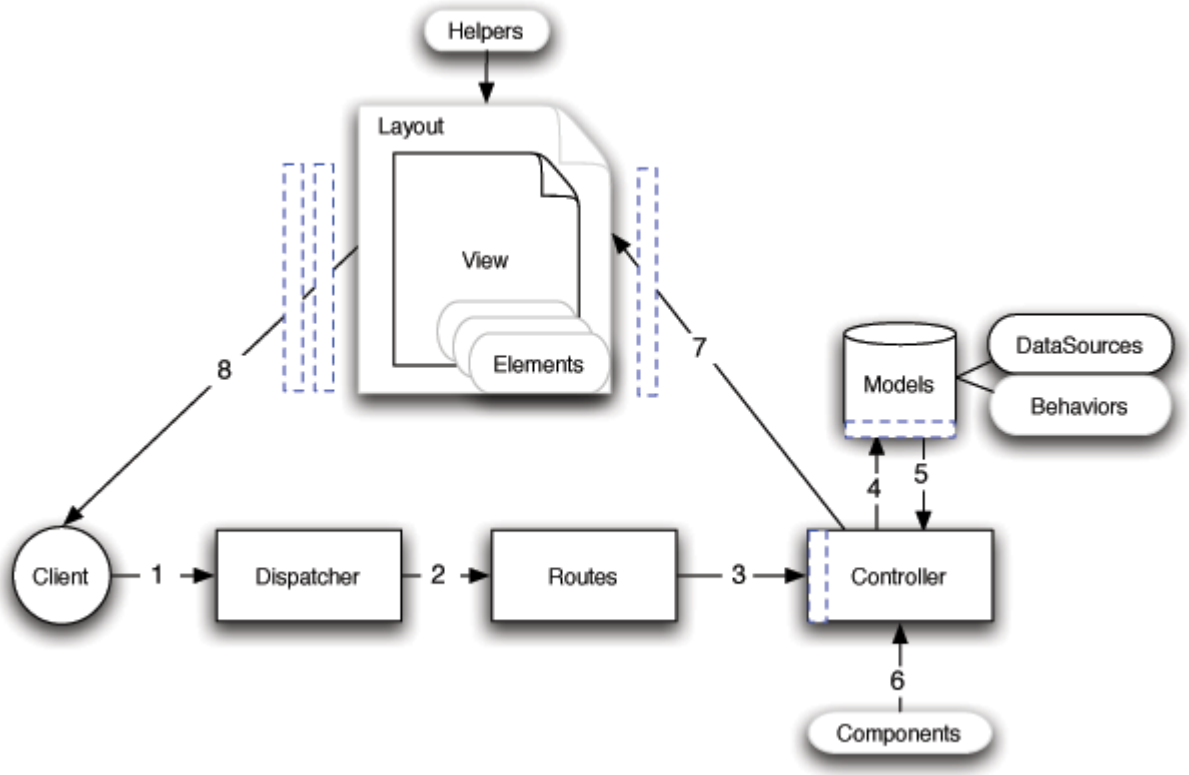


Abbildung 3: Typischer Request-Ablauf

Quelle: http://manual.cakephp.org/view/21/A-Typical-CakePHP-Request

4.3 Convention over Configuration

CakePHP folgt dem Paradigma „Convention over Configuration“. Mittels „Convention over Configuration“ wird versucht, durch klare Konventionen den Konfigurationsaufwand zu verringern. Die Art und Weise wie „Convention over Configuration“ in CakePHP umgesetzt ist entspricht in etwa dem von Ruby on Rails. Dies bedeutet, dass statt einer variablen Konfiguration festgelegte Konventionen bestehen.
Das Framework CakePHP bietet Vorgaben von Dateinamen bis hin zu Tabellennamen. Hält sich der Entwickler an diese Namenskonventionen, kann ein großer Teil an Konfigurationsaufwand eingespart werden. Auch Mappings werden dadurch automatisch aufgelöst, da durch die Konventionen festgelegt ist, zu welchem Model welche Tabelle gehört. Der Entwickler muss nur in Ausnahmefällen selber diese Auflösung vornehmen. Durch die Namensvorgaben ergibt sich das Zusammenspiel zwischen einzelnen Objekten automatisch, so ist beispielsweise bei gegebenen Controllernamen das zu verwendende Model schon bestimmt. Auch bietet es dem Entwickler eine vordefinierte Verzeichnisstruktur an. Alle Aspekte (Model, View, Controller usw.) sind somit schon auf Dateiebene strukturiert.

4.4 Access Control List

Die von CakePHP von Haus aus mitgelieferte Möglichkeit um Zugriffe auf Objekte zu verwalten ist in der Component ACL realisiert [9]. ACL steht für Access Control List, zu Deutsch Zugriffssteuerungsliste, und ist eine bekannte Technik der Zugriffsverwaltung. Somit wurde das Rad nicht neu erfunden sondern auf Bewährtes zurückgegriffen. Die ACL lässt sich grob in zwei Bereiche unterteilen. Zum einen alle „Dinge“ die auf etwas Zugreifen wollen. Diese sind die Initiatoren der Handlung und werden von CakePHP als ARO (Access Request Object) bezeichnet. Der Gegenstand der Handlung, alle „Dinge“ auf welche zugegriffen werden kann, wird als ACO (Access Control Object) bezeichnet. Hierbei ist anzumerken das ein ARO durchaus auch ein ACO sein kann. Dies ist z. B. der Fall, wenn ein User auf Daten eines Users zugreift. Welche ARO nun was mit einem ACO tun darf, ist im Zugriffsrecht festgelegt.

4.4.1 Berechtigungsmatrix

Diese Zugriffsrechte sind in einer Matrix angeordnet. Die Matrix enthält alle AROs, ACOs und die möglichen Operationen auf den ACOs. Die Standardoperationen, welche auf ein ACO angewandt werden können, sind die von Datenbanken bekannten CRUD-Operationen **C**reate, **R**ead, **U**pdate, **D**elete, Tabelle 2 veranschaulicht dies. Auf das ACO Blog1 darf ARO User2 nur lesend zugreifen.

Wohingegen User1 den Eigentümer des Blogs darstellt und somit zusätzlich die Berechtigungen zum editieren und löschen besitzt.

ARO	ACO	Create	Read	Update	Delete
User1	Blog1	0	1	1	1
User2	Blog1	0	1	0	0

Tabelle 2: Berechtigungsmatrix

4.4.2 Hierarchie

Sowohl AROs als auch ACOs können eine Vererbungsstruktur abbilden. Dies hat den Vorteil, dass man nicht jedes Recht für eine Operation explizit angeben muss. Somit kann man die Rechte des Eltern-Objektes übernehmen und gegebenenfalls einzelne Rechte überschreiben. Die hierfür zugrunde liegende Technik heißt Modifier Preorder Tree Traversal (MPTT). Abbildung 4 zeigt den MPTT für eine mögliche ARO-Struktur.

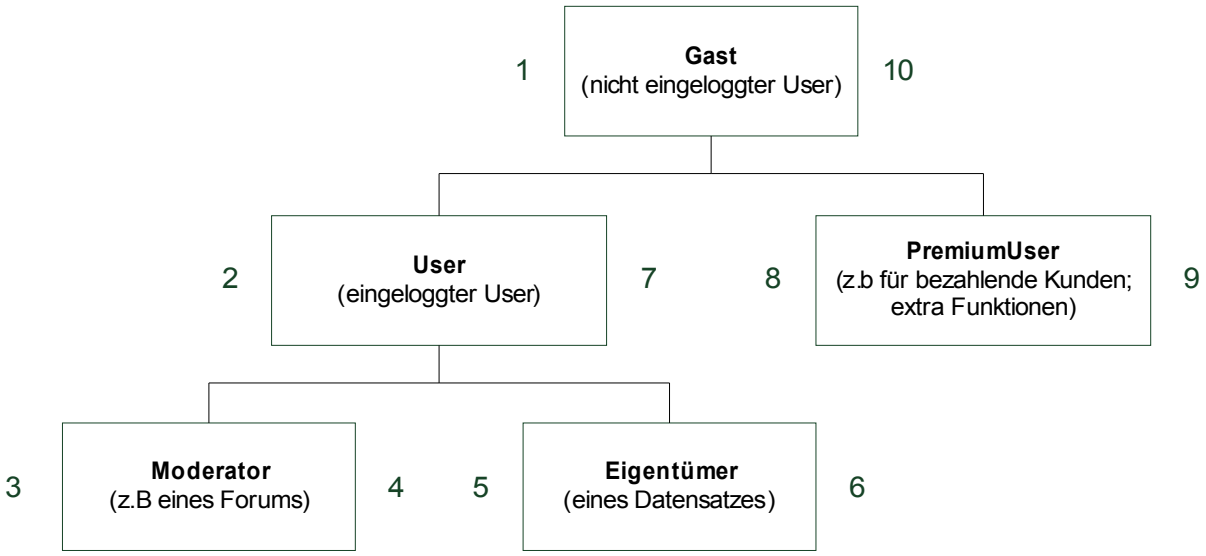


Abbildung 4: Modified Preorder Tree Traversal

4.5 Lokalisierung und Internationalisierung

CakePHP bietet Möglichkeiten, eine Anwendung kulturellen und sprachlichen Gegebenheiten anzupassen. Diese Anpassbarkeit wird mit Lokalisierung bezeichnet. Ein weiterer Aspekt der Lokalisierung oder auch Internationalisierung eines Produktes ist die Parametrisierung der Texte über die verwendete Sprachumgebung. Dazu wird für jeden zu verwenden Sprachraum eine eigene Datei angelegt. In dieser werden Bezeichner und die konkreten länderspezifischen Texte hinterlegt. Aus einer View heraus kann nun einfach eine Routine aufgerufen werden, welche zu einem Bezeichner den entsprechenden Text in korrekter Sprache ausgibt.

4.6 Testen

Bei komplexeren Anwendungen ist das Testen der Produktqualität während und nach der Produktion von großer Bedeutung. Das Testen soll sicherstellen, dass eine Anwendung die an sie gestellten Anforderungen in vorgegebener Qualität erfüllt.
CakePHP bietet dem Entwickler verschiedene Vorgehensweisen an, um Tests durchzuführen. Dabei wurde keine eigene Lösung entwickelt sondern auf das Testframework simpletest zurückgegriffen [10]. Durch den Einsatz des simpletest-Testframeworks ist es möglich, Unit- und Webtests durchzuführen.

4.6.1 Unit - Test

Mittels der Unit-Tests können einzelne Programmteile getestet werden. Dieses können unter anderem Methoden innerhalb einer Component, eines Models oder die Actions eines Controllers sein.

4.6.2 Web - Test

Web-Tests testen die Anwendung aus Usersicht. Mit solchen Testfällen werden also eher Anwendungsfälle abgedeckt. Mittels dieser Web-Tests kann durch die Anwendung navigiert werden. Formulare können automatisch ausgefüllt und abgeschickt werden.

4.7 Ajax

4.7.1 Hintergrund

Mit Ajax ist es möglich Webanwendungen zu entwickeln, die eine ähnliche Interaktivität zulassen, wie sie von Desktopanwendungen bekannt ist. Dies wir erreicht durch eine im Hintergrund laufende asynchrone Verbindung zwischen dem Webbrowser und dem Webserver. Mittels Ajax können einzelne Teile einer Webseite nachgeladen werden. Ohne Ajax ist dies nur mit einem kompletten neuen Seitenaufruf möglich. Durch Ajax wird auch ein Stück weit die Anforderung nach Benutzerfreundlichkeit erfüllt.

4.7.2 DHTML / Dom Scripting

Einfache HTML-Webseiten sind rein statisch und bieten keine Möglichkeiten der Interaktivität. Dies hat sich mit der Einführung des DHTML-Standards geändert. Das „D“ steht für Dynamic. Es lässt sich dadurch also „Bewegung“ in Webseiten bringen. Durch DHTML können einzelne HTML-Elemente manipuliert werden [11]. Die HTML-Elemente werden dazu beim Seitenaufruf in eine Baumstruktur geladen. Mittels dieser Struktur auch DOM-Baum genannt kann auf jedes einzelnes Element zugegriffen werden. Jede Änderung des DOM-Baums resultiert in einer sofortigen Änderung der Webseite im Browser. Um DOM Scripting nutzen zu können muss der Webbrowser JavaScript interpretieren können. Der Nachteil von Dom Scripting ist allerdings, dass nur Seiteninhalte manipuliert werden können die beim Laden der Seite auch ausgeliefert wurden [12]. Ajax schließt nun genau diese Lücke.

4.7.3 Herkömmlicher Seitenaufruf

Im klassischen Modell einer Webanwendung erfolgt bei jeder Aktion des Benutzers eine Anforderung nach einer neuen Seite. Dieser HTTP-Request wird nun vom Webserver verarbeitet. Als Ergebnis wird der HTTP-Response an den Browser gesendet. Dieser Vorgang geschieht meist mit einer sichtbaren Verzögerung. Abbildung 5 veranschaulicht den Request-Response-Vorgang [13].

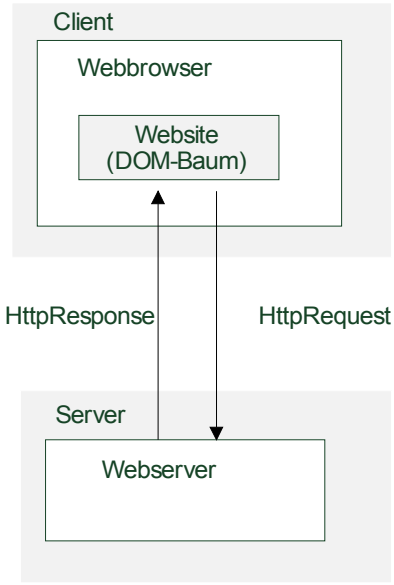


Abbildung 5: Request-Response-Vorgang einer herkömmlichen Webanwendung

4.7.4 XML-HTTP-Request

Ajax ermöglicht das Nachladen von Seiteninhalten, ohne dass der Benutzer direkt einen Seitenaufruf tätigen muss. Dies geschieht mittels des JavaScript Objektes XMLHttpRequest. Es dient dem Datenaustausch mittels HTTP-Protokoll. Der XMLHttpRequest erfolgt asynchron, somit muss das aufrufende Skript nicht exklusiv auf Antwort warten und kann stattdessen in der Zeit andere Aufgaben bearbeiten. Der Request muss nicht zwangsläufig in XML erfolgen, es ist möglich jede Art von Text zu verwenden.

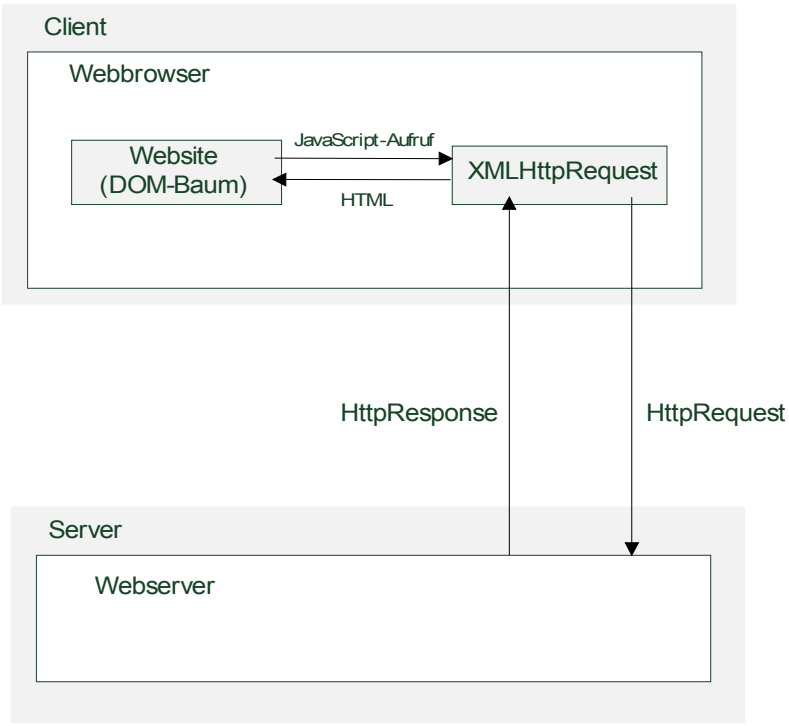


Abbildung 6: Request-Response-Vorgang mittels XMLHttpRequest

4.7.5 Prototype & script.aculo.us

Das JavaScript-Framework Prototype wurde mit dem Ziel entwickelt, die Entwicklung dynamischer Webanwendungen zu unterstützen. Es stellt dem Entwickler viele Funktionen zur Verfügung, die die Arbeit mit JavaScript, DOM und vor allem Ajax vereinfachen und beschleunigen. Prototype ist ein sehr komplexes Framework, trotzdem ist es relativ gut strukturiert. Es wird häufig in MVC-Frameworks eingebunden und bildet selbst die Grundlage für andere JavaScript Frameworks und Bibliotheken. Script.aculo.us ist beispielsweise eine JavaScript-Bibliothek, welche auf Prototype aufbaut. Es erweitert das Framework um dynamische visuelle Effekte und Elemente zur Benutzerführung [14].
Sowohl Prototype als auch script.aculo.us sind in vollem Umfang in CakePHP integriert und lassen sich einfach benutzen.
Beide Frameworks stehen, wie auch CakePHP, unter der MIT-Lizenz und sind damit kostenfrei nutzbar.

4.8 Zusammenfassung

Dieses Kapitel hat gezeigt, dass CakePHP für die Entwicklung von Webanwendungen gut geeignet ist. CakePHP bildet eine Umgebung und stellt Hilfsmittel bereit, um die Entwicklung von datenbankbasierten Webanwendungen zu vereinfachen und zu beschleunigen. Die grundlegenden Ziele sind Einfachheit, Wiederverwendbarkeit, Testbarkeit und Erweiterbarkeit. Womit nur durch die Verwendung von CakePHP schon einige der nicht funktionalen Anforderungen abgedeckt werden können.

5 Systemdesign

Um den Entwicklungsaufwand eines Produktes zu reduzieren, stehen mehrere Möglichkeiten zur Auswahl. Hier sollen aber nur solche betrachtet sein, welche direkt die technische Umsetzung betreffen und nicht die Rahmenbedingung, wie etwa der zugrunde liegende Entwicklungsprozess.
Eine Methode ist das Zusammenfügen von bereits bestehenden Komponenten bzw. Teilprodukten zu einem neuen Produkt. Aufgaben für Komponenten sind im Falle einer Social-Network-Webanwendung z. B. Benutzerverwaltung, Beziehungsmanagement, Zugriffkontrolle, Nachrichtenaustausch etc.
In einem neuen Projekt können diese Komponenten einzeln eingebunden werden, was zu einer deutlichen Verringerung des Entwicklungsaufwandes führt.
Das Einbinden und Konfigurieren der Komponenten ist jedoch selbst mit einem gewissen Aufwand verbunden. Um nun auch den Aufwand für das Einbinden und Konfigurieren der Komponenten zu minimieren, wird eine Basisarchitektur entwickelt, in welcher sämtliche Komponenten bereits komplett integriert sind. Die einzelnen Komponenten sind also schon alle in dieser Basisarchitektur enthalten und müsse nicht einzeln integriert werden.
Die Basisarchitektur soll aber nicht nur als Sammlung von Komponenten betrachtet werden. Vielmehr ist die Basisarchitektur als eine komplett einsatzbereite Webanwendung für soziale Netzwerke zu verstehen. In der Basisarchitektur ist auch der gesamte Request-Ablauf enthalten, welcher unter anderem die Berechtigungsauswertung und Sprachermittlung beinhaltet. Ein konkretes Projekt wie das Motorsportportal muss nur die Basisarchitektur einbinden, um komplett alle Funktionen eines sozialen Netzwerkes anbieten zu können. Lediglich die darstellenden Elemente, wie Views und das Layout müssen angepasst werden.

5.1 Übersicht

Abbildung 7 soll die Abgrenzung von der Basisarchitektur zu einem konkreten Projekt, wie dem Motorsportportal, verdeutlichen. Das System lässt sich grob in zwei Bereiche unterteilen. Zum einen die Klassen der Basisarchitektur zum anderen das konkrete Projekt mit spezifischen Anforderungen.
Die Basisklassen bilden die Grundlage für konkrete Projekte. So ist es möglich eine neue Webanwendung zu entwickeln, welche die Basisklassen benutzt oder erweitert. Der Idealfall ist, dass für ein neues Projekt nur die Views und Layouts verändert werden müssen, die Programmlogik hingegen aber unverändert bleibt.

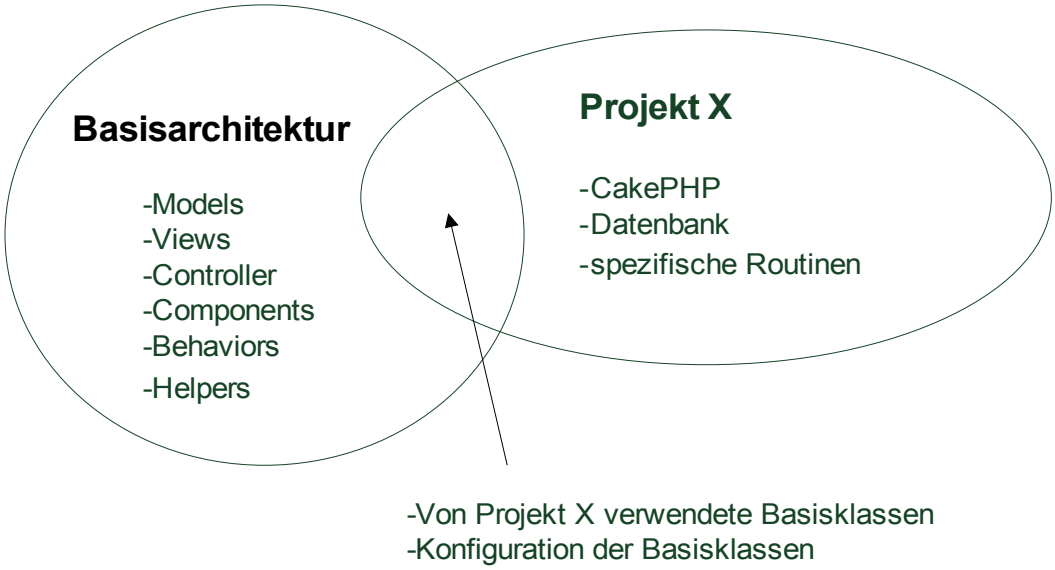


Abbildung 7: Gesamtarchitektur

Basisarchitektur: Die Basisarchitektur stellt sowohl eine Art Klassenbibliothek als auch eine komplette Webanwendung da.

ProjektX: ProjektX ist ein konkretes Projekt, wie das Motorsportportal, welches Klassen aus der Basisarchitektur verwendet.

CakePHP: Die Anwendung CakePHP liegt im Projekt, so dass der Programmablauf beim Projekt selbst liegt.

Datenbank: Die Datenbankanbindung geschieht über das Projekt selbst, da nur das Projekt Tabellen enthält. Die Basisarchitektur gibt nur die verwendbaren Models vor.

Spezifische Routinen: Sind solche, die sich aus den speziellen Projektanforderungen ergeben und nicht durch Klassen aus der Basisarchitektur abgedeckt werden.

Von Projekt X verwendete Basisklassen: Basisklassen, die von ProjektX eingebunden werden. Diese Klassen können komplett durch Vererbung übernommen werden und/oder partiell dessen Logik durch Methodenüberschreibung verändert werden.

Konfiguration der Basisklassen: Anpassen der Basisklassen

Für die Umsetzung wird das im vorherigen Kapitel beschriebenen MVC-Framework CakePHP verwendet. Denn mit diesem wurden schon umfangreiche Erfahrungen in vorherigen Projekten gemacht. Außerdem ist der Unterschied zu anderen Produkten dieser Art, wie Ruby on Rails und Zend-Framework, nur marginal.

5.1.1 Einbinden der Basisklassen

Das Einbinden der Basisklassen lässt sich durch Vererbung am besten realisieren. So kann alles wie es ist verwendet werden, oder einzelne Methoden überschrieben werden, um deren Funktionalität zu verändern. In einem konkreten Projekt können die Basisklassen eingebunden werden und die konkreten Klassen von diesen ableiten. Abbildung 8 veranschaulicht dies.

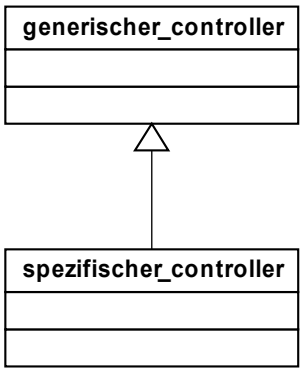


Abbildung 8: Einbinden der Basisklassen (UML)

```
//Datei: "basis_controller.php"
//Die Basisklasse
class BasisController extends GenericAppController {}


//Datei: "specific_controller.php"
//Einbinden der Basisklasse
require_once(BASIS_CONTROLLERS."basis_controller.php");
//Die spezifische Klasse
class SpecificController extends BasisController {}
```

Abbildung 9: Einbinden der Basisklassen (Quelltext)

Dieses Verfahren hat zur Folge, dass wenn eine Klasse aus der Basisarchitektur im Projekt verwendet werden soll, eine extra Klasse im Projekt angelegt werden muss, welche von der generischen Klasse ableitet. Dieses Vorgehen lässt sich für Controller, Models, Components, Behaviors, Helpers so verwenden.
Für Views, Elements, Layouts ist das Ableiten nicht möglich, da diese selbst keine Klassen sind, sondern nur reine Templatedatein. Auch das partielle Verändern der Views ist nicht möglich. Denn es ist nicht möglich, wie bei Klassen einzelne „Bereiche“ zu überschreiben. Weil sich jedoch die Views von Projekt zu Projekt meist komplett unterscheiden, ist dies nicht problematisch.
Um dennoch nicht auf die Views aus der Basisarchitektur verzichten zu müssen, lässt sich ein Mechanismus integrieren der das Vorhandensein von konkreten Views überprüft. Sollte keine konkrete View existieren, wird stattdessen die View aus der

Basisarchitektur verwendet. Dieser Mechanismus stellt während des Entwicklungsprozesses sogar einen Vorteil da.
Um eine möglichst hohe Wiederverwendbarkeit gewährleisten zu können, ist der Anspruch vorhanden, die Basisklassen möglichst abstrakt zu halten, um dadurch ein hohes Maß an Konfigurierbarkeit zu ermöglichen. Die einzelnen Methoden sollten nur für grundlegende, meist den Ablauf betreffende, Änderungen überschrieben werden müssen.

5.1.2 Konfigurieren der Basisklassen

5.1.2.1 Konfiguration per Überschreibung von Klassenvariablen

Eine einfache Möglichkeit um Einfluss auf die Basisklassen zu nehmen, ist das Überschreiben von Klassenvariablen. Abbildung 10 zeigt das Vorgehen, anhand eines Controllers. Für Models, Components, etc. ist das Vorgehen identisch.

```
class BasisUsersController extends BasisAppController{
	...
	var $limit = 10;
	...
	function index(){
		...
		//UserDatensätze aus der DB holen, mit Limit 10;
		$users = $this->User->findAll(null,$this->limit);
		...
	}
	...
}

class UsersController extends BasisUsersController{
	//Das Limit aus dem GenericUsersController überschreiben
	var $limit = 25;
}
```

Abbildung 10: Konfigurieren der Basisklassen 1 (Quelltext)

5.1.2.2 Konfiguration mittels globaler Konfiguration

Eine andere Möglichkeit, Einfluss auf die Basisklassen zunehmen, ist das Verwenden einer zentralen Konfiguration. Die Basisarchitektur hat eine eigene Konfigurationsdatei, welche von einer Spezifischen überschrieben bzw. angepasst werden kann, somit ist in den Klassen selbst dieses nicht mehr nötig.

```
class BasisAppController{
      // zentrale Konfiguration
      var $conf = array(
                  'index'=>
                        array('limit'=>10)
                  );
      ...
}

class BasisUsersController extends BasisAppController{
      ...
      var $limit = $this->conf['index']['limit'];
      ...
      function index(){
            ...
            //UserDatensätze aus der DB holen, mit Limit;
            $users = $this->User->findAll(null,$this->limit);
            ...
      }
      ...
}

class UsersController extends BasisUsersController{
      // hier ist nichts zu tun
}
```

Abbildung 11: Konfigurieren der Basisklassen 2 (Quelltext)

Anstatt eine Konfigurationsdatei zu verwenden oder die Konfigurationen z. B. im Application-Controller abzulegen, kann auch eine Datenbanktabelle dafür benutzt werden. Diese Tabelle könnte zur Laufzeit in einem Konfigurationsarray vorgehalten werden, damit nicht für jede einzelne Konfigurationsabfrage ein Datenbankaufruf nötig ist.

5.2 Basisarchitektur

5.2.1 Model

Das nachfolgende UML-Klassendiagramm, Abbildung 12, zeigt die verwendeten Models. Jedes dieser Models entspricht einer Tabelle in der Datenbank. Zur besseren Darstellung sind nicht alle Felder jeder Tabelle angegeben, sondern nur solche, welche obligatorisch oder vielmehr später von Bedeutung sind.

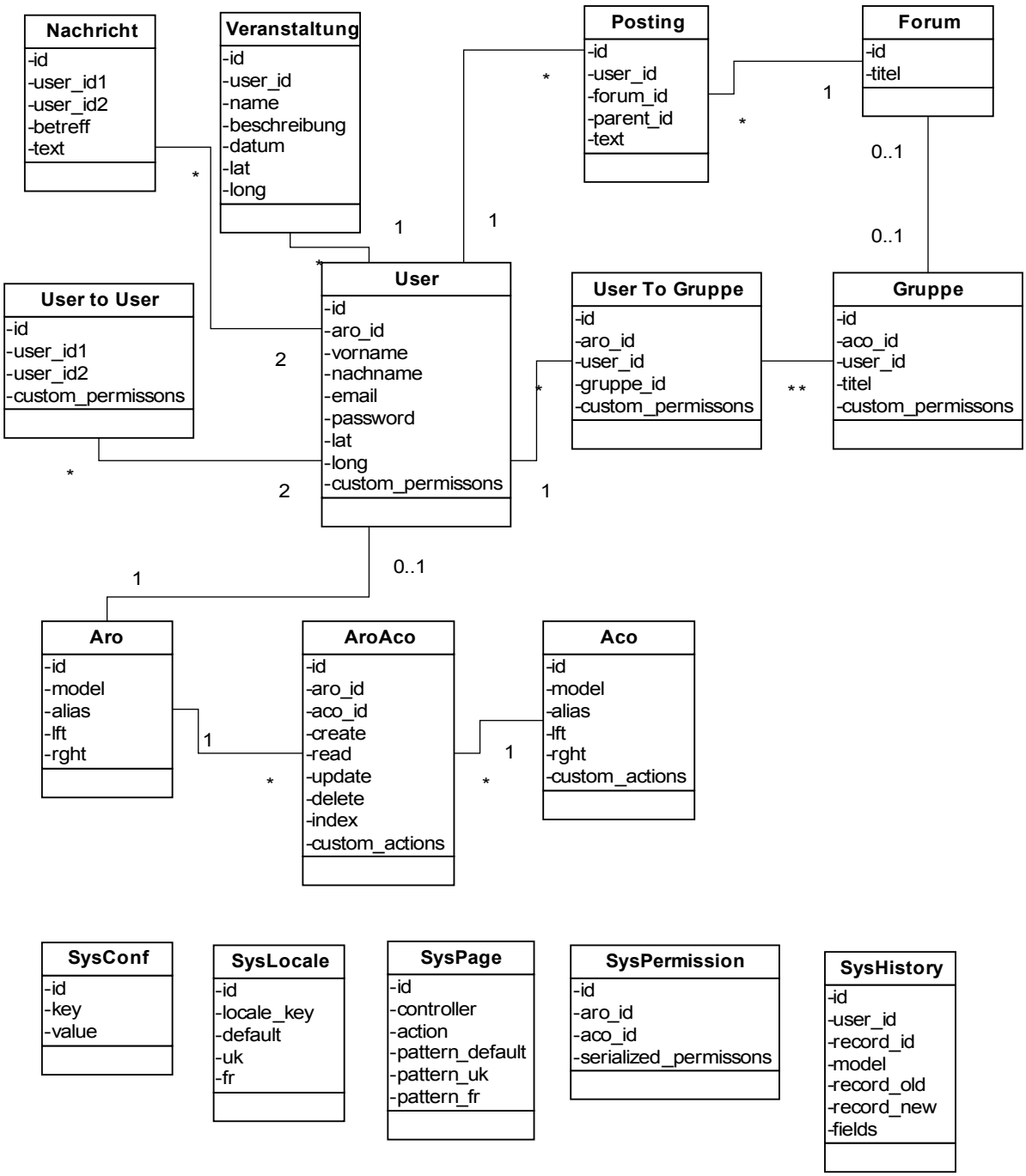


Abbildung 12: Model-Diagramm

Diese Models lassen sich in zwei Kategorien einordnen. Zum einen all jene welche für „User Generated Content“ verwendet werden und solche die nur Systeminformation enthalten. Nachfolgend sind die Models für „User Generated Content“ kurz erläutert.

User: Das User-Model repräsentiert die Profildaten eines Benutzers.

User To User: Abbilden von Freundschaften/Kontakten und deren möglichen Status, wie „Kontaktanfrage gestellt“ und „Kontakt angenommen“. Auch das Ignorieren von Benutzern lässt sich auf diese Art realisieren.

Nachricht: Enthält eine Nachricht von einem Benutzer zu einem anderem.

Gruppe: Interessengemeinschaften.

User To Gruppe: Bildet die Zugehörigkeit zu einer Gruppe ab. Enthält auch den Zustand der Mitgliedschaft wie Mitglied, Moderator und der Gleichen.

Forum: Diskussionsforum.

Posting: Das Posting-Model stellt einen Beitrag innerhalb eines Diskussionsforums dar.

Veranstaltung: Repräsentation einer Veranstaltung.

Nachfolgend sind die Models für Systeminformation kurz erläutert.

Aro: Enthält die möglichen Rollen eines Users. Zu diesen möglichen Rollen zählen zum einen die „Standard“ Rollen wie „User“, „Premium-User“ und der Gleichen. Dazu kommen noch die Rollen in Abhängigkeit des angeforderten Objektes, wie beispielsweise „Eigentümer“ oder „Moderator“.

Aco: die Objekte, genauer die Funktionen auf Objekte, für welche ein Benutzer Zugriffsrechte erhalten kann.

AroAco: Repräsentiert die Zugriffsmatrix von AROs auf ACOs. Für jedes ARO kann festgelegt werden, ob eine Funktion auf ein ACO erlaubt ist oder nicht.

SysPermission: Enthält das Kreuzprodukt der Berechtigungen. Für jede einzelne ARO ist festgehalten, was diese mit jeder einzelnen ACO tun darf. Wird erst verwendet, wenn alle AROs und ACOs fix definiert sind. Ist für den Live-Betrieb zu verwenden, weil die Vererbungsstruktur schon aufgelöst ist und dadurch performantere Abfragen möglich sind.

SysHistory: Historie über vorgenommene Datenänderungen von Usern.

SysLocale: Ausgabetexte in verschiedenen Sprachen.

SysConf: Das SysConf-Model enthält allgemeine Konfigurationsinformationen.

SysPage: In dem SysPage-Model kann zu einer Controller-Action eine frei definierbare URL festgelegt werden.

5.2.1.1 Abbildung von Beziehungen

Der eigentlich Schwerpunkt von sozialen Netzwerken liegt in der Abbildung von Beziehungen, daher auch oft Beziehungsnetzwerke genannt. Hat ein Benutzer einen anderen Benutzer, mit dem er real bekannt ist, oder nur aus dem Interesse heraus im Netz in Kontakt zu stehen gefunden, kann er diesen in den Kreis seiner Online-Bekanntschaften aufnehmen. Diese Relation zwischen zwei Benutzern lässt sich wie in Abbildung 13 dargestellt realisieren.

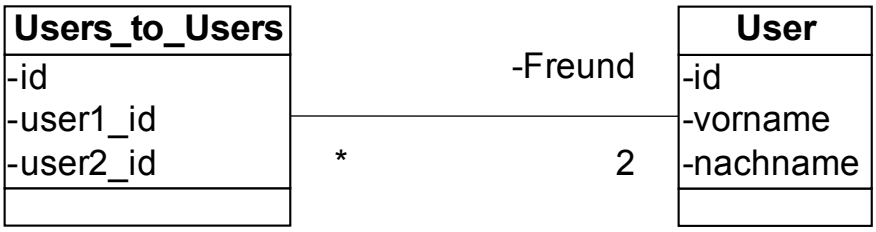


Abbildung 13: Abbildung von Beziehungen (UML)

In CakePHP lässt sich dies durch eine hasMany – Assoziation im User Model realisieren.

```
class User extends AppModel {
    var $name = 'User';

    var $hasMany = array(
        'Freunde' => array(
            'className'  => 'UserToUser',
            'foreignKey' => 'user1_id',
        )
    );
}
```

Abbildung 14: Abbildung von Beziehungen (Quelltext)

Hierbei wird davon ausgegangen, dass das Hinzufügen eines Bekannten erst nach dessen Zustimmung erfolgen kann. Beim Vorgang der Zustimmung wird einer zweiter Eintrag in Users_to_Users vorgenommen, um genau die andere Richtung der Beziehung abzubilden. Dieses Anlegen von zwei Datensätzen, für nur eine Beziehung, hat den Vorteil, dass für jede Richtung extra Informationen mit gespeichert werden können. So kann beispielsweise feingranular die Sichtbarkeit von Informationen kontaktbezogen festgelegt werden.

5.2.1.2 Abbildung von Zugehörigkeiten

Eine häufige Aufgabe ist das Abbilden von Zugehörigkeiten. Dies ist überall dort der Fall, wo ein Benutzer Teilnehmer, Mitglied oder dergleichen sein kann. In Abbildung 15 ist die Zugehörigkeit von Benutzern zu Gruppen dargestellt. Eine Gruppe kann man sich hierbei als eine Gemeinschaft von Personen mit ähnlichen Interessen vorstellen.

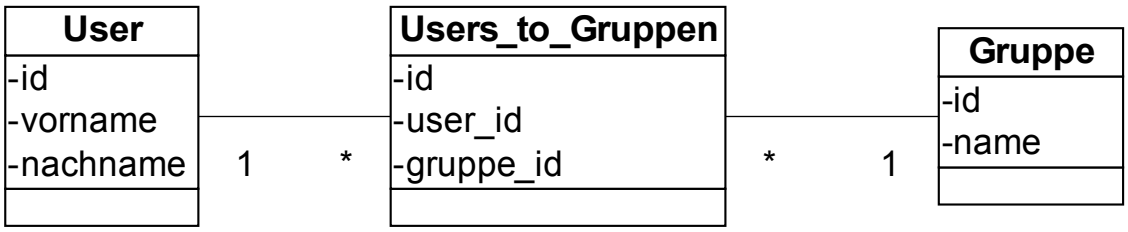


Abbildung 15: Abbildung von Zugehörigkeiten (UML)

```
class Gruppe extends AppModel {
      var $name = 'Gruppe';
      var $hasAndBelongsToMany = array(
            'Mitglied' => array(
                  'className'  => 'User',
            )
      );
}
```

Abbildung 16: Abbildung von Zugehörigkeiten (Quelltext)

Der in Abbildung 16 gezeigte Quelltext veranschaulicht die Implementierung in CakePHP. Dabei wird von der Gruppe eine Relation zum User aufgebaut, welche durch den Assoziationstyp „hasAndBelongstoMany“ realisiert ist. Dadurch stehen bei einem Abruf einer Gruppe automatisch alle Daten der Mitglieder mit zur Verfügung.

5.3 Controller

Abbildung 17 zeigt die in der Basisarchitektur verfügbaren Controller.

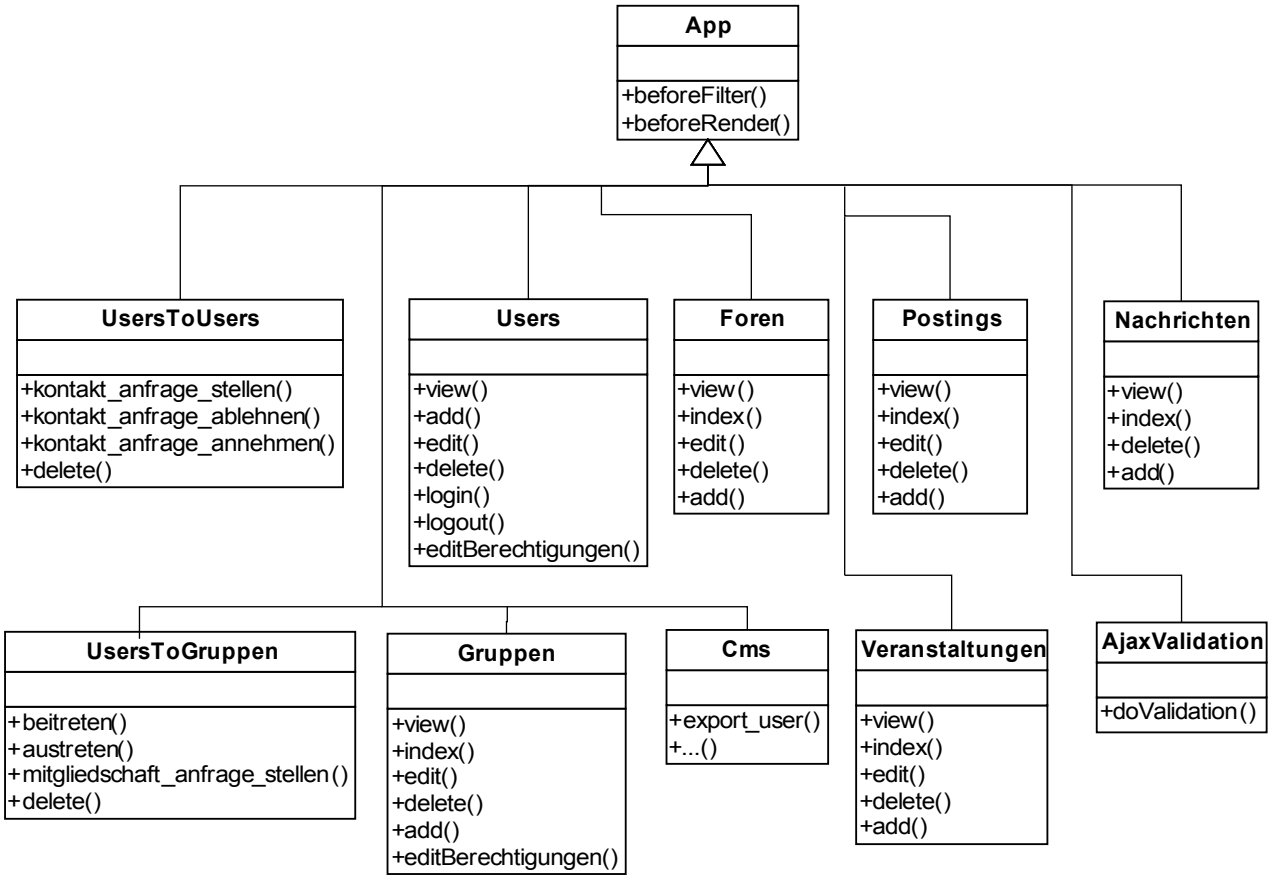


Abbildung 17: Controller-Übersicht

Als obligatorische Funktionen eines Controllers gelten die view, add, edit, delete und index Methoden. Im Folgenden sind nur die Besonderheiten der einzelnen Controller genannt.

Users: Der Users-Controller bietet Methoden für Login und Logout eines Nutzers. Auch hat der Nutzer die Möglichkeit, Rechte für Besucher seines Profils zu vergeben.

Users To Users: Verwalten der Kontakte zu anderen Nutzern. Kontakte kommen erst nach einem Bestätigungsprozess zustande.

Gruppen: Enthält alle Methoden, die für das Bearbeiten und Darstellen von Gruppen benötigt werden. Der Gruppengründer hat zusätzliche Optionen zur Auswahl, um Benutzungsrechte für die Gruppe festzulegen.

Users To Gruppen Verwalten der Gruppenzugehörigkeiten.

AjaxValidation: Nimmt Ajax Requests zur Eingabedaten-Validierung entgegen.

Cms: Dient zum Export von Inhalten nach Content-Management-Systemen.

App: Der Application-Controller ist die Oberklasse aller Controller. Der Application-Controller stellt allgemeine Funktionen zur Verfügung.

5.4 Components

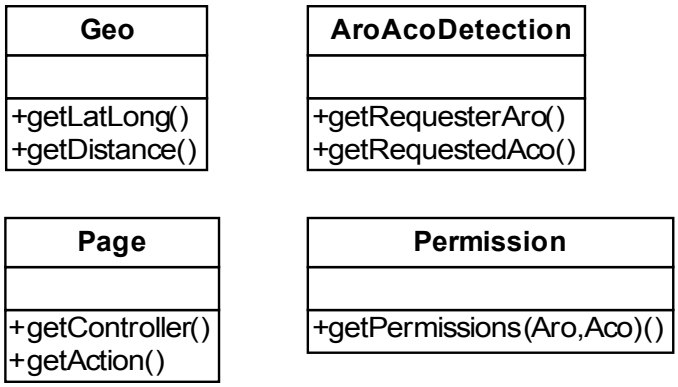


Abbildung 18: Component-Übersicht

Geo: Stellt Funktionen aus dem Bereich der Geokodierung zur Verfügung. Die Methode getLangLong ermittelt zu einer Adresse den Längen- und Breitengrad. Die Methode getDistance berechnet den Abstand zwischen zwei Längen- und Breitengraden.

Page: Die Page-Component ermittelt zu einer URL die zuständige Controller-Action.

AroAcoDetection: Stellt die Rolle des Users, die ARO, fest. Auch das Objekt auf welches zugegriffen wird, die ACO, wird ermittelt.

Permission: Zu gegebenem ARO und ACO werden die Zugriffsrechte ermittelt.

5.4.1 Geokodierung

Geokodierung enthält das Prüfen und Lokalisieren von Adressen. Eine geokodierte Adresse ist die genaueste Möglichkeit, Standorte zu lokalisieren. Aufgrund von Geokodierung werden Zusammenhänge ersichtlich die ohne räumlichen Bezug nicht möglich wären. Dadurch ist es möglich, zusätzliche Informationen aus der geografischen Lage zu ermitteln, wie etwa die Berechnung der Entfernung zwischen zwei Punkten. Eine solche geokodierte Adresse besteht dann als Ergebnis nur noch aus Längen- und Breitengrad.

5.4.1.1 Geokodierung mit der Google Maps API

Google bietet neben dem bekannten Suchdienst eine Menge weitere Dienste an. Google Maps ist einer dieser kostenfreien Dienste. Google Maps beinhaltet, wie sich aus dem Namen schon ableiten lässt, vieles, was mit geografischen Aufgaben zu tun hat. Es lassen sich damit Landkarten unter verschiedenen Gesichtspunkten betrachten, wie etwa Satelliten- und Luftbilder oder Straßenkarten.

Darüber hinaus bietet die Google Maps API die Möglichkeit kostenlos Adressen zu geokodieren [15].

Abbildung 19 stellt den Ablauf einer Anfrage an die Google Maps Api da.

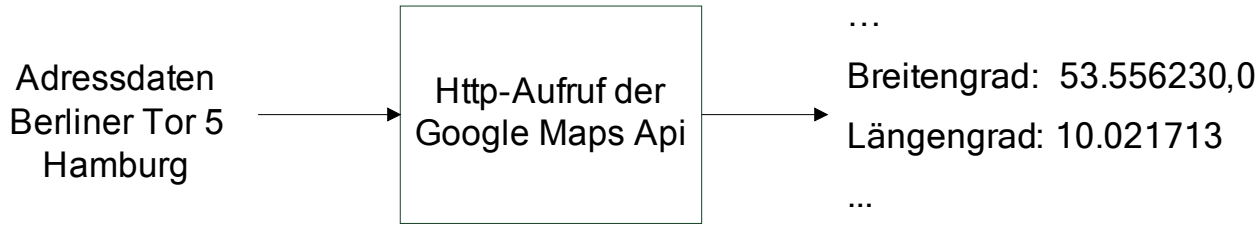


Abbildung 19: Ablauf der Geokodierung mittels der Google Maps Api

Der Aufruf der Google Maps API erfolgt durch den Aufruf einer bestimmten URL. Diese URL enthält die Adresse, hier mit Straße und Ort. Ein zusätzlicher Parameter gibt das gewünschte Antwort-Format an, beispielsweise XML.

```
http://maps.google.com/maps/geo?q=Berliner Tor 5,hamburg,&output=xml
```

Dieser Aufruf würde als Resultat, die geokodierten Adressdaten, im XML-Format enthalten. Die Antwort enthält nicht nur die Längen- und Breitengrade sondern auch weitere Informationen zur Adresse, wie Postleitzahl und Namen des Stadtteils.

5.4.1.2 Distanzermittlung

Um die Entfernungen zwischen zwei, oder mehreren Orten ermitteln zu können, müssen von diesen die Längen- und Breitengrade bekannt sein. Eine für die meisten Anwendungen ausreichende Genauigkeit für die Entfernung ist die Berechnung anhand einer Einheitskugel. Hierbei wird die Luftlinie, also der direkte Weg zwischen zwei Orten ermittelt. Ein Ermitteln der befahrbaren Verkehrswege ist damit nicht realisierbar. Im folgenden Quelltext ist der Algorithmus für die Distanzermittlung beschrieben.

```
// Geodaten HAW
      $Breitengrad1  = 10.021713;
      $Laengengrad1  = 53.556230;
// Geodaten University of Shanghai for Science and Technology
      $Breitengrad2 =  31.12318;
      $Laengengrad2  = 121.254257;
// Kreiszahl Pi
      $pi = pi();

// Umrechnung der Gradzahl in RAD:
      $breite1  = $Breitengrad1 / 180 * $pi ;
      $länge1  = $Laengengrad1 / 180 * $pi ;
      $breite2  = $Breitengrad2 / 180 * $pi ;
      $länge2  = $Laengengrad2 / 180 * $pi ;

// Die Formel zur Entfernungsberechnung bedient sich einer Einheitskugel:
$e = acos( sin($breite1)*sin($breite2) +
cos($breite1)*cos($breite2)*cos($länge2-$länge1) );
$entfernung = $e * 6378.137;
```

Abbildung 20: Distanzermittlung

5.4.2 CMS-Schnittstelle

Durch eine CMS-Schnittstelle soll der bidirektionale Datenaustausch zwischen Content-Management-Systemen und dem Motorsportportal ermöglicht werden. Als Content-Management-System oder Inhaltsverwaltungssystem gilt eine Software, die das Erstellen und Bearbeiten von Inhalten wie Texten und Multimediaelementen ermöglicht und organisiert. Durch diese Schnittstelle soll es möglich sein, Inhalte mit anderen Systemen auszutauschen.

5.4.2.1 Import

Um redaktionell gepflegte Inhalte eines CMS einbinden zu können, müssen die Inhalte zunächst aus dem CMS ausgelesen werden. Dazu bieten viele CMS eine Exportschnittstelle auf Basis des Dokumentenformates XML an. Eine andere Möglichkeit um Inhalte eines CMS einbinden zu können, stellt der direkte Seitenaufruf dar. Durch direkten Seitenaufruf wird das CMS veranlasst, eine Webseite mit den gewünschten Inhalten darzustellen. Die dargestellten Inhalte können dann leicht eingelesen und weiterverarbeitet werden. Diese Methode ist meist langsamer als der Austausch per XML, dafür funktioniert dieses Vorgehen bei jedem Web-Content-Management-System.

5.4.2.2 Export

Für den Export von Inhalten aus dem Motorsportportal nach Content-Management-Systemen wird ein eigener Controller bereitgestellt. Dieser Controller enthält Methoden welcher die Anfragen eines CMS bedient. Solche Methoden können z. B. eine Liste der neuesten Benutzer, oder die neusten Forumsbeiträge liefern.

5.4.3 Zugriffskontrolle

Ein soziales Netzwerk stellt besondere Ansprüche an das eingesetzte Rechtesystem, weil im Web 2.0 vorwiegend Gebrauch von „User Generated Content“ gemacht wird. Was zur Folge hat, dass Rechte festgelegt werden müssen für Ressourcen die erst im laufenden Betrieb der Anwendung entstehen. Das heißt also, es werden nicht nur Rechte für statische sondern auch dynamische „Ressourcen“ benötigt.
Das von CakePHP mitgelieferte ACL-System lässt sich für diese Anforderungen nicht gut einsetzen. Dieses sieht vor, dass für jedes einzelne Objekt immer auch eine ACO definiert werden muss, und zusätzlich die Rechte auf dieses ACO in der Zugriffsmatrix. Die Rechteabfrage ist auch nicht mit einem Zugriff auf die Zugriffsmatrix erledigt, da sowohl ARO- also auch ACO-seitig Möglichkeiten zur Vererbung geboten werden. Dies hat zur Folge, dass unter Umständen die gesamte Hierarchie durchlaufen werden muss, um die Zugriffsrechte zu erhalten. Für wenige Rechteabfragen pro Website mag dies noch gut gehen. Sollen aber auf einer Website die Zugriffsrechte für eine ganze Liste von Daten abgefragt werden, ist dieses sehr rechenintensiv.

Eine Möglichkeit, die Rechteabfragen performanter zu gestalten liegt in der Vorhaltung aller Rechte zu einem ARO-ACO-Paar, sodass Rechte direkt aus einer Tabellenzeile ausgelesen werden können. Um diese Tabelle aufzubauen, muss das Kartesische (Kreuz) Produkt auf alle AROs und ACOs gebildet werden. Da sich jedoch die AROs und ACOs zur Laufzeit verändern, müsste dieses Kreuzprodukt immer neu gebildet werden.
Eine Lösung ist nun die AROs und ACOs nur für Gruppen zu verwenden. Das heißt in der ARO werden nicht mehr einzelne konkrete User gespeichert sondern nur noch Gruppen von Usern, für die ACOs gilt Entsprechendes. So werden keine AROs und ACOs zur Laufzeit hinzugefügt. Somit lassen sich die vorgehaltenen Rechte benutzen, was eine Rechteermittlung mit nur einer Abfrage erlaubt.

5.4.3.1 Authentifizierung

Mittels der Authentifizierung kann die behauptete Identität eines Endbenutzers überprüft werden. Zu diesem Zweck werden bei einer Anmeldung am System die Anmeldeinformationen ausgewertet. In den Anmeldeinformationen sind meist Benutzername und Passwort enthalten. Entsprechen diese Angaben einen im System bekannten Benutzer, ist die Authentifizierung erfolgreich abgeschlossen.
Aus Sicherheitsgründen werden die Passwörter nicht im Klartext in der Datenbank abgelegt, sondern zuvor MD5 Hash verschlüsselt. Die MD5-Verschlüsselung zählt zu den Einwegverschlüsselungen. Ein beim Login eingegebenes Passwort wird auch MD5 verschlüsselt und mit dem Passwort aus der Datenbank verglichen.

5.4.3.2 Autorisierung

Die Autorisierung entscheidet über die Berechtigungen des Benutzers innerhalb der Anwendung. Die Autorisierung erfolgt nach einer erfolgreichen Authentifizierung. Um die Berechtigung ermittelt zu können muss bekannt sein, wer die Handlung ausführt und was das Objekt der Handlung ist.
Das hier beschriebene Verfahren zur Autorisierung ist in den Components AroAcoDetection und Permission realisiert.

5.4.3.3 ACO – Ermittlung des Gegenstands einer Handlung

Bei jedem Zugriff auf ein Objekt muss zunächst ermittelt werden, um welches ACO es sich handelt. In den meisten Fällen entspricht der Objektname dem Namen des ACOs und somit kann das zugehörige ACO direkt ermittelt werden. Dennoch lassen sich damit nicht alle Situationen abdecken. Existieren für ein ACO mehrere Ausprägungen funktioniert diese direkte Zuordnung nicht. Dies ist beispielsweise der Fall bei Gruppen, welche in verschiedene Arten die Mitgliedschaft zulassen. So werden Gruppen in öffentliche und geschlossene Gruppen unterschieden. In die erst Genannten könnten Benutzer sich selbst zum Mitglied machen, in geschlossenen Gruppen hingegen erst nach einem Anmeldeprozess. Um nun das korrekte ACO zu ermitteln, muss zu jedem Objekt explizit angegeben werden, welches ACO benutzt werden soll. Das Objekt enthält in diesem Fall selbst die Information, welches ACO zutreffend ist.

5.4.3.4 ARO – Ermittlung des Initiators einer Handlung

Bei jedem Zugriff auf ein Objekt muss ermittelt werden, in welcher Rolle sich der User für eben diesen Zugriff befindet. Hierbei können sich drei Situationen ergeben.

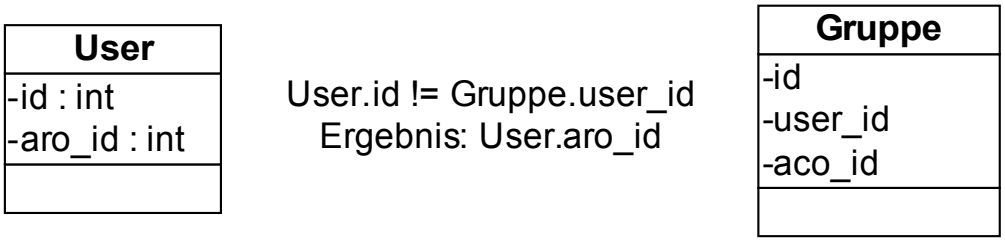


Abbildung 21: User-Objekt: Keine Verbindung

Es besteht keine Verbindung zwischen dem User und dem angeforderten Objekt. Daher wird für den User seine Standardrolle verwendet welche in User.aro_id definiert ist.

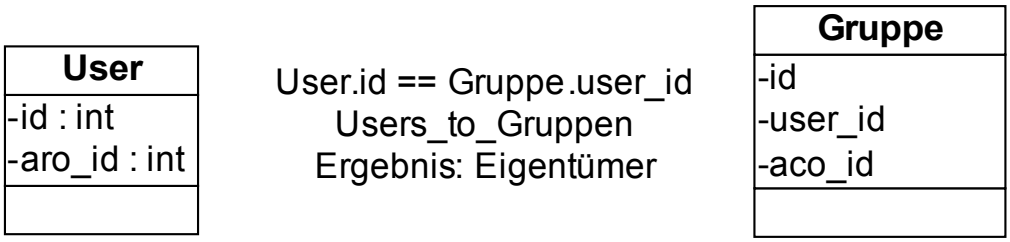
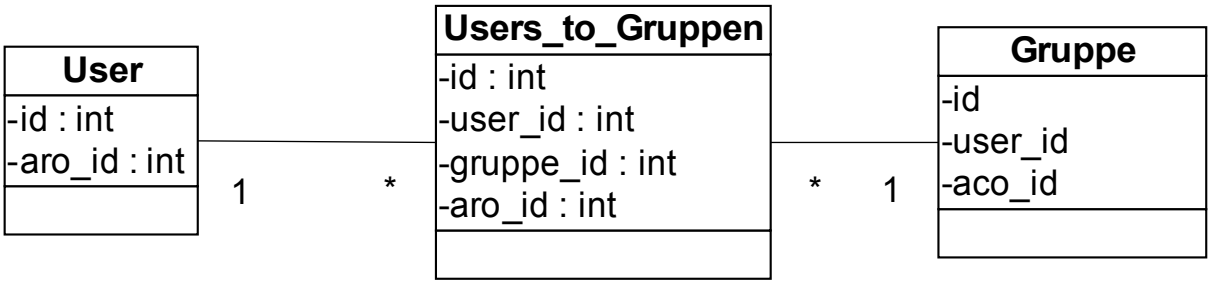


Abbildung 22: User-Objekt: Eigentümer

Der User ist der Eigentümer des angeforderten Objekts. Das Objekt Feld user_id entspricht der eigenen User.id.



User.id == User_toGruppen.user_id
&& Gruppe.id == Users_to_Gruppen.gruppe_id
Ergebnis: Users_to_Gruppen.aro_id

Abbildung 23: User-Objekt: Zugehörigkeit

Der User steht in einer besonderen Relation zum Objekt. In dem Fall einer Gruppe z.b Gruppenmoderator oder Gruppenmitglied.

5.4.3.5 Permissions

Die eigentlichen Berechtigungen werden nach der Feststellung von ARO und ACO ermittelt. Dieser Vorgang lässt sich in mehrere Phasen unterteilen.

1. **Ermittlung der möglichen Rechte:** Für jede ACO sind unterschiedliche Rechte definiert. Zu den obligatorischen CRUD-Operationen kommen noch Spezifische hinzu. Diese spezifischen Rechte sind abhängig von den dazugehörigen Controller-Actions.
2. **Auslesen der Rechte:** Für jedes ermittelte mögliche Rechte wird die Berechtigung ermittelt. Dies geschieht entweder aus der AroAco-Tabelle bzw. im Live-Betrieb anhand der SysPermission.
3. **Spezielle Rechte auslesen:** Das bisherige ARO-ACO System lässt es nur zu, Objektgruppen in den AROs und ACOs zu definieren und keine konkreten Ausprägungen. So beinhaltet die ACO-Tabelle z. B. nur Posting und nicht Posting yxz. Oft ist es jedoch der Fall, das ein Benutzer Rechte auf seine Datensätze selbst verwalten möchte, also gewähren oder verbieten möchte. Um dies zu ermöglichen, erhält der Benutzer Rechte, welche er selber definieren kann. Dies kann er allgemein für unterschiedliche Benutzertypen wie „User“ oder „Mitglied/Freund“ festlegen. Diese Rechte werden im Userdatensatz selbst festgehalten. Um Rechte für einzelne User festzulegen, wird die beziehungsabbildende Tabelle um ein extra Rechtefeld erweitert.
   Dieser Mechanismus lässt sich 1 zu 1 auch auf andere Situationen übertragen. Das Gruppensystem wäre eine solche Situation, in welcher die Rechte für jeden einzelnen Gruppenmoderator vom Gruppengründer feingranular konfiguriert werden können.

5.4.4 URLs

Die URLs in CakePHP folgen einer starren vorgegebenen Struktur.
www.example.org/ControllerName/ActionName/Param1/Param2/...
Oft soll es aber möglich sein diese URLs komplett frei aufzubauen. So sollen wichtige Bezeichner enthalten sein, welche von Suchmaschinen gefunden werden. Auch sollen diese Bezeichner länderspezifisch sein, was sowohl für die User als auch für die Suchmaschinenoptimierung vorteilhaft ist. Daraus folgt, dass jeder Teil der URL nach der Toplevel-Domain, also der Pfad frei konfigurierbar sein muss.
Was der Pfad aber auf jeden Fall enthalten muss, sind die Parameter für die Controller-Actions. Wenn beispielsweise ein Datensatz angezeigt werden soll, muss natürlich bekannt sein welcher Datensatz. Standardmäßig wird der Datensatz der verwendet werden soll aus der URL extrahiert. In der URL ist dieser dann direkt als Id enthalten, unter welcher er auch in der Datenbank zu finden ist.
Diese freie URL-Gestaltung hat zur Folge, dass nicht mehr direkt von einer URL auf eine Controller-Action geschlossen werden kann.

Diese verlorene URL-Controller-Action Kopplung lässt sich wiederherstellen in dem man statt der URL ein URL-Muster verwendet. Eine eingehende URL wird dann mit mehreren Mustern aus der Datenbank verglichen. Wird ein passendes Muster gefunden, ist auch die zugehörige Controller-Action gefunden. Tabelle 3 zeigt wie diese Datenbankeinträge aussehen können.

Default-URL-Muster	UK-URL-Muster	Controller	Action	Parameter
/Produkte/#0	/products/#0	products	view	#0

Tabelle 3: URL-Struktur

Die einzelnen möglichen Funktionsparameter werden in den Mustern mit einer # gekennzeichnet. In diesem durch # dargestellten Platzhaltern befinden sich dann die für Suchmaschinen relevanten Daten des Datensatzes. Wird beispielsweise die URL Produkte/Markenname-Produktname-Produktkategorie_456 aufgerufen, ist das passende Muster /Produkte/#0, was zu Folge hat, dass die Controller-Action products/view ausgeführt wird.

5.5 Lokalisierung und Internationalisierung

Für Webanwendungen ist Mehrsprachigkeit häufig aufgrund der Tatsache, dass diese von überall auf der Welt erreichbar sind ein Thema. Bei einer Desktopanwendung, welche nur für ein bestimmtes Land entwickelt wurde, ist diese dann meist auch nicht in anderen Ländern aufzufinden. Soll eine Anwendung mehrsprachig ausgelegt sein, sollte dies schon so früh wie möglich bekannt sein. Denn die nachträgliche Erweiterung um die Mehrsprachfenähigkeit ist mit einigem Mehraufwand verbunden.
Um Mehrsprachigkeit zu ermöglichen, darf in der gesamten Anwendung kein Ausgabetext direkt im Quelltext stehen. Es müssen Platzhalter existieren, welche in Abhängigkeit der Sprache des Users zur Laufzeit ersetzt werden.
Der von CakePHP dafür vorgesehene Weg ist jedoch nicht ausreichend. Dieser Weg ist nur auf Sprachdatein beschränkt und bietet keine Möglichkeit die Sprachen über eine Datenbank zu verwalten. Auch reicht es meist nicht nur einen Text in der richtigen Sprache auszugeben. Einige Texte müssen auch bestimmte Werte enthalten. Es müssen Platzhalter innerhalb der Sprachtexte existieren, welche dynamisch ersetzt werden können.

5.5.1 Sprachermittlung

Um dem Benutzer die Anwendung in der „richtigen“ Sprache zu präsentieren, muss diese zuvor ermittelt werden. Hierfür bieten sich mehrere Möglichkeiten an.

Das Analysieren der aufgerufenen URL kann darüber Aufschluss geben welche Sprache der Benutzer bevorzugt. Je nach Aufbau der Anwendung kann die Position der Sprachinformation innerhalb der URL unterschiedlich sein. Tabelle 4 zeigt die in diesem Fall signifikanten Teile der URL.

www.example.**de**	Top-Level-Domain
www.**de**.example.com	Subdomain
www.example.com/**de**	Teil des Verzeichnispfades

Tabelle 4: URL-Spracherkennung

Eine andere Art die Sprache des Benutzers zu ermitteln ist das Abfragen der Browserinformationen. Jeder Browser übermittelt bei einem Seitenaufruf die Sprache des Benutzers. Diese Information ist zugänglich über die PHP-Variable $_SERVER['HTTP_ACCEPT_LANGUAGE'].

5.6 Sicherheit

Sicherheit und alle damit verbundenen Aspekte sind für Webanwendungen ein wichtiges Thema. Sollten Userdaten manipuliert worden sein, oder der Server nicht erreichbar sein, hinterlässt dies einen schlechten Eindruck bei den Benutzern. Das Vertrauen der User ist aber gerade bei sozialen Netzwerken, in welchen die User meist viele Informationen von sich preisgeben, ein entscheidender Faktor.

5.6.1 Bedrohung durch Benutzereingaben

Eine der größten Gefahrenpotentiale bieten die von Usern vorgenommen Eingaben. Zwei bekannte Methoden um fehlende oder unzureichende Eingabedatenvalidierung zu missbrauchen sind SQL-Injection und XSS.
SQL-Injections zielen darauf ab, die Datenbank zu kompromittieren. Dazu werden den Eingabedaten SQL-Queries untergeschoben. Findet nun eine mangelhafte Überprüfung statt, oder werden die Daten direkt an den SQL-Interpreter weitergegeben, ergeben sich daraus vielfältige Möglichkeiten für den Angreifer. Dadurch kann auf Daten zugegriffen werden, auf die der Benutzer sonst keine Zugriffsmöglichkeiten hat. Das Manipulieren von Daten ist dadurch möglich. Im schlimmsten Fall gelingt dem Angreifer sogar das erlangen des Root-Status und damit Vollzugriff auf den Webserver.
XSS steht für Cross-Site Scripting und zielt im Gegensatz zu SQL-Injections nicht auf einen Server ab. Vielmehr stehen bei XSS die den Server benutzenden Clients im

Vordergrund. Es wird versucht Schadcode in eine Webseite einzubringen, welcher dann clientseitig beim Endbenutzer ausgeführt wird. Ziel ist meist das Ausspähen bzw. übernehmen von Benutzerkonten.
Beide Angriffsarten können durch CakePHP-Bordmittel weitestgehend ausgeschlossen werden. Dazu wird die von CakePHP mitgelieferte Sanitize-Klasse verwendet die Funktionen bereitstellt, welche die Validierung von Eingabedaten durchführen. Der ideale Zeitpunkt für die Durchführung der Validierung ist der BeforeFilter des Application-Controller, dieser wird vor der Ausführung der eigentlichen Action ausgeführt. Damit ist sichergestellt, dass alle Eingabedaten zentral überprüft werden.

5.6.2 Mensch oder Maschine

Es reicht nicht aus die Eingabedaten an sich zu überprüfen, sondern es muss auch sicher gestellt sein das die Eingaben von einem Menschen getätigt werden. Dies ist wichtig um beispielsweise das automatische Versenden von Nachrichten zu unterbinden, da diese häufig, wie von Email bekannt, für unerwünschte Werbung missbraucht werden. Mit sogenannten CAPTCHAs ist es weitestgehend möglich einen Menschen von einer Maschine zu unterscheiden. CAPTCHA ist ein Akronym für „Completely Automated Public Turing test to tell Computers and Humans Apart“. Was so viel bedeutet wie „Vollautomatischer öffentlicher Turing-Test, um Computer und Menschen zu unterscheiden“.

5.7 History

Von Benutzern erzeugte Inhalte können unerwünschte Texte und Aussage enthalten. Gerade Inhalte welche rechtlich bedenklich sind, sind zumeist unerwünscht. Daher muss eine Möglichkeit existieren, um Streitfälle oder Rechtsverstöße nachvollziehen zu können. Es dürfen keine gemachten Eingaben eines Benutzer durch ihn selbst oder andere endgültig gelöscht oder überschrieben werden kann. Dies lässt sich mit einer Änderungshistorie für jeden Datensatz realisieren. Davon ist nicht nur der Nachrichtenaustausch zwischen Benutzern betroffen. Alle vorgenommenen Eingaben, wie etwa innerhalb eines Benutzerprofils oder eines Forums sind davon betroffen.
Für die Umsetzung einer solchen Änderungshistorie bietet es sich an ein Behavior zu verwenden. Beim Abspeichern von Daten wird dieses Behavior aktiv. Es vergleicht die bestehenden Daten mit den eingegebenen Daten. Sollte ein Unterschied bestehen, wird das in der Tabelle SysHistory vermerkt. Der Eintrag in der SysHistory enthält die Namen und Daten der Felder, welche sich verändert haben, die Datensatz-Id und die Id des Users, welcher die Änderung vorgenommen hat.

5.8 Ajax

5.8.1 Eingabedaten-Validierung

Überall wo Benutzer Eingaben tätigen können muss es eine Möglichkeit geben um diese Eingaben zu überprüfen. Diese Überprüfung untersucht die eingegebenen Daten mittels eines vorher definierten Regelwerkes auf ihre Gültigkeit. Sollte die Eingabe nicht den Regeln genügen, erhält der User eine Fehlermeldung. In klassischen Webanwendungen verhält es sich so, dass diese Überprüfung erst nach dem Abschicken aller Eingabedaten erfolgt. Nach diesem Ereignis werden alle Eingaben geprüft und bei Misserfolg eine oder mehrere entsprechende Fehlermeldungen dem User angezeigt.
Unter Usability-Aspekten ist es jedoch angenehmer ein sofortiges, unmittelbares Feedback auf jede gemachte Eingabe zu erhalten. Und genau dies ist mit den klassischen Webanwendungen nicht ohne Einschränkungen möglich. Einfache Abfragen (mittels JavaScript) wie: besteht der Name nur aus Buchstaben oder ist das Datum gültig können mit den klassischen Webanwendungen durchaus gelöst werden. Aber bei einer Abfrage wie „ist der Benutzername schon vergeben“ reicht dieser Ansatz nicht aus.
Mit Ajax lässt sich diese Anforderung erfüllen. CakePHP bietet extra für diesen Fall eigene Methoden an, welche die Ajax-Aufrufe kapseln und z. T. vereinfachen. Für jedes Eingabefeld wird ein Observer eingerichtet. Dieser ruft bei einem bestimmten Ereignis auf genau dieses Eingabefeld eine serverseitige Funktion auf. Ein solches Ereignis kann beispielsweise das Verlassen eines Eingabefeldes sein. Die im Hintergrund aufgerufene Funktion führt dann die Validierung durch und gibt dem Benutzer unmittelbar Auskunft über den Erfolg bzw. Misserfolg der Überprüfung.

5.8.2 Selektives Nachladen von Informationen

Oft ist es nötig dem Benutzer Eingabemöglichkeiten zu bieten die untereinander eine spezielle Relation besitzen. Eine solche Relation ist die zwischen Land und Bundesland. Wählt der Benutzer in einer Auswahlbox ein Land aus, sollen in einer anderen Auswahlbox nur die relevanten Bundesländer zu Auswahl stehen. Mittels Ajax kann nach Wahl des Landes eine Funktion aufgerufen werden welche alle relevanten Bundesländer ermittelt und nachlädt. Somit können dem User nur die Daten gegeben werden welche er wirklich benötigt. Dies hat positiv Auswirkungen auf die Usability und auf die Performance.

5.9 Ablauf eines Requests

Aufgrund der Tatsache, dass das von CakePHP bereitgestellte Verfahren zur Routenermittlung nicht verwendet wird, muss der Request Ablauf den neuen Gegebenheiten angepasst werden. Mit der CakePHP-Lösung zur Routenermittlung war es möglich direkt von einer URL auf Controller und Action zu schließen, dies ist aufgrund der freien URL-Struktur nun nicht mehr möglich. Zudem werden die Rechteabfragen, Eingabedaten-Validierung und Spracherkennung zu einem integralen Bestandteil eines jeden Seitenaufrufes. Abbildung 24 veranschaulicht den modifizierten Teil eines Request-Ablaufs.

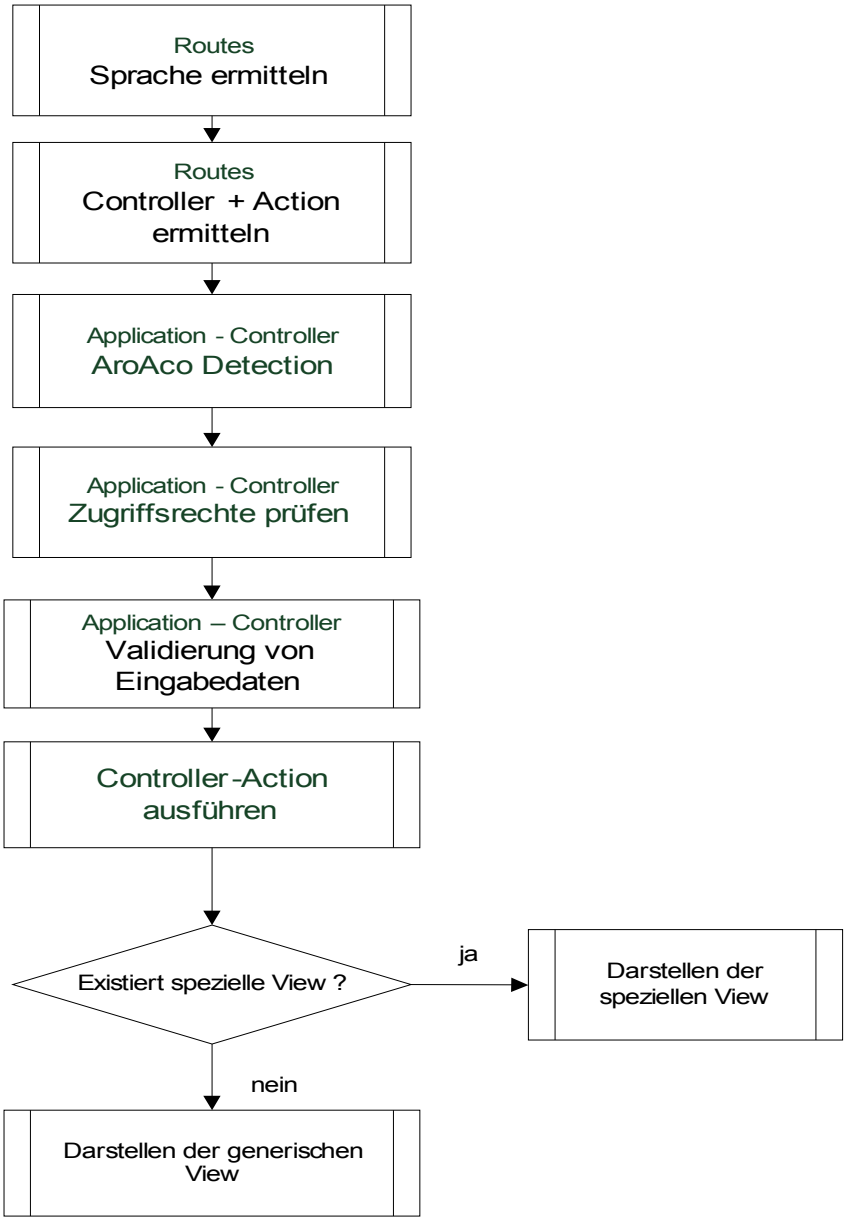


Abbildung 24: Request-Ablauf

1. Sprache des Benutzers ermitteln. Wie in Kapitel 5.5.1 beschrieben.
2. Die Controller-Action ermitteln. Dies geschieht anhand des speziellen Behandelns der URL. Dazu wird zu der aufgerufenen URL die Controller-Action ermittelt. Siehe Kapitel 5.4.4.
3. Festellen „Wer“ auf „Was“ zugreift. Siehe Kapitel 5.4.3.3 und5.4.3.4.
4. Festellen „Wer“ mit „Was“ was darf. Siehe Kapitel 5.4.3.5.
5. Validierung der Eingabedaten. Siehe Kapitel 5.6.1.
6. Ausführen der eigentlichen „angeforderten“ Logik.
7. Dem Benutzer die View darstellen. Die generischen Views werden nur zur einfacheren Entwicklung und schnellem Prototyping benötigt. Siehe Kapitel 5.1.1.

6 Systemrealisierung

6.1 Arbeitsumgebung

Das entwickelte System gliedert sich wie im Kapitel zuvor beschrieben in zwei Abschnitte.
Die Basisarchitektur stellt Klassen für Controller, Models, Components etc. zur Verfügung.
Das Motorsportportal beinhaltet die Anwendung CakePHP und Klassenrümpfe, welche von den Klassen aus der Basisarchitektur ableiten.
Abbildung zeigt 25 die Basisarchitektur als Eclipse-Projekt. Abbildung 26 zeigt das Motorsportportal als Eclipse-Projekt. Beide Abbildungen sind stark gekürzt. Es werden nur die den User betreffenden Controller, Models und Views gezeigt.

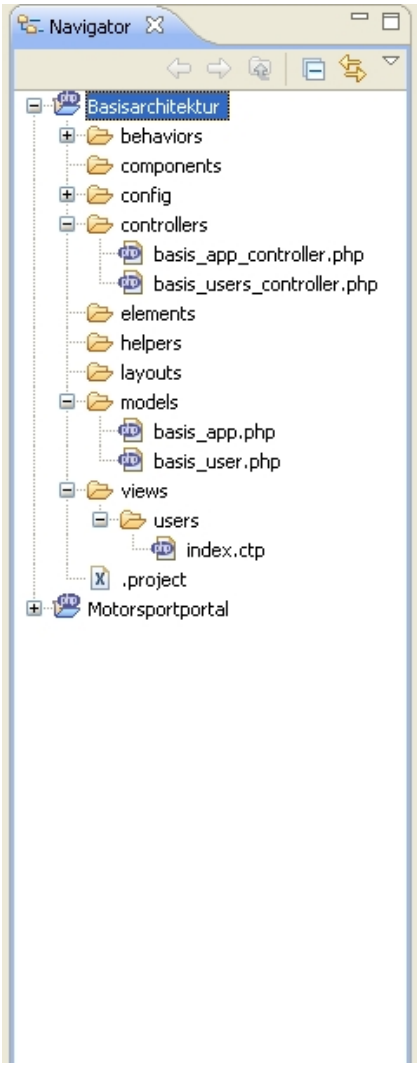


Abbildung 25: Basisarchitektur in Eclipse

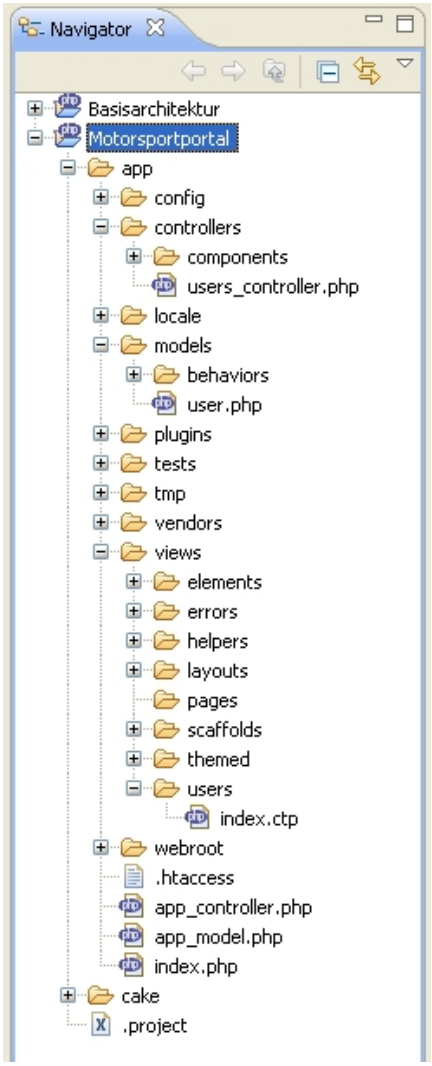


Abbildung 26: Motorsportportal in Eclipse

Alle Klassen des gesamten Motorsportportals beinhalten nur das Ableiten der Basisklassen. Abbildung 27 veranschaulicht dies für den User-Controller(users_controller.php) auf Seiten des Motorsportportals.

```
require_once(BASIS_CONTROLLERS."basis_users_controller.php");
class UsersController extends BasisUsersController {}
```

Abbildung 27: Einbinden der Basisklassen im Motorsportportal(Quelltext)

6.2 CAPTCHA

Die im vorherigen Kapitel beschriebene Funktion zur Unterscheidung von Mensch und Maschine begegnet dem Benutzer wie in Abbildung 28 dargestellt. Zur Realisierung wurde auf die PHP-Bibliothek „PhpCaptcha - A visual and audio CAPTCHA generation library“ zurückgegriffen [16].

Abbildung 28: CAPTCHA

6.3 Ajax-Formulare

Nachfolgend ist die realisierte Eingabedaten-Validierung mittels Ajax anhand eines Beispiels erläutert. Abbildung 29 zeigt ein Email-Eingabefeld. Die unter dem Eingabefeld zu sehende Fehlermeldung ist die Antwort eines Ajax-Request zur Eingabedaten-Validierung.

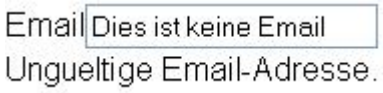


Abbildung 29: E-Mail Eingabefeld

Abbildung 30 zeigt den Programmcode der View. Zu sehen ist die Definition eines Eingabeformulars inklusive eines Texteingabefelds für eine E-Mail-Adresse.
Mittels des Observe-Field-Aufrufes wird für das Eingabefeld ein Listener eingerichtet. Dieser Listener wird bei Verlassen des Eingabefeldes ausgelöst und führt die Eingabedatenvalidierung durch. Das Ergebnis der Eingabedatenvalidierung wird in der View im div-Container „Fehlermeldung“ dargestellt.

```
<?
//Formular erstellen
echo $form->create('User');
echo $form->input('email',array('id'=>'email'));
echo $form->end();
//Ajax Observe-Field auf das Email-Eingabe-Feld setzen.
echo $ajax->observeField(
               "email",
               array(
                  "update"=>"Fehlermeldung",
                  "url"=>"/AjaxValidation/doValidation/"
               )
         )
?>

<div id="Fehlermeldung"></div>
```

Abbildung 30: View (Quelltext)

Abbildung 31 zeigt eine vereinfachte Form des AjaxValidation-Controllers. Die Methode doValidation wird aus der View mittels des eingerichteten Observe-Fields aufgerufen. Die Eingabedaten werden gegen die im Model definierten Validierungsregeln geprüft. Zusätzlich wird die E-Mail-Adresse auf Einzigartigkeit in der User-Tabelle überprüft.

```
<?php
class AjaxValidationController extends AppController
{
      //Eingabedaten auf Gültigkeit prüfen.
      function doValidation(){
         //Dem Model die Eingabedaten übergeben
         $this->User->set($this->data);
         //Durchführen der Validierung
         $this->User->validates();
         if(!empty($this->User->validationErrors)){
            echo  "Ungueltige Email-Adresse.";
          }else{
             if($this->User->hasAny(
                          array('email'=>$this->data['User']['email'])))
{
                 echo  "Email-Adresse schon vergeben.";
              }
          }
      }
}
?>
```

Abbildung 31: AjaxValidation-Controller (Quelltext)

In Abbildung 32 ist das User-Model dargestellt. Als einzige Regel ist festgelegt, dass das Feld email auch eine gültige E-Mail-Adresse enthalten muss.

```
<?php
class User extends AppModel{
	var $name = "User";

	//Validierungs-Regeln
	var $validate = array(
		'email' => array(
		'rule' => array('email', true),
		)
	);
}
?>
```

Abbildung 32: User-Model (Quelltext)

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung

In dieser Arbeit wurde eine Basisarchitektur für soziale Netzwerke entwickelt. Durch dessen Einsatz kann der Entwicklungsaufwand für zukünftige Projekte erheblich reduziert werden. Neu zu erstellende soziale Netzwerke können durch das Einbinden der Klassen aus der Basisarchitektur sofort einsatzbereit sein. Einzig das Layout muss immer den projektspezifischen Anforderungen angepasst werden. Das Verhalten der Basisarchitektur lässt sich, falls erforderlich, durch mehrere Wege flexibel anpassen.
Das erstellte System basiert auf dem quelloffenen MVC-Framework CakePHP.
Die Basisarchitektur deckt wie gefordert alle Aspekte eines sozialen Netzwerkes ab. Das Abbilden von Beziehungen ist für Benutzer leicht anwendbar. Zum Austauschen von Interessen und Erfahrungen stehen dem Benutzer Diskussionsforen und Gruppen zu Verfügung.
Durch den Einsatz von Ajax wird die Benutzung interaktiver und damit für den Benutzer erleichtert, so wie dieser es von Desktopanwendungen gewohnt ist.
Mit dem entwickelten Rechtesystem lassen sich feinstufig Zugriffsrechte definieren und festlegen. Für jedes Objekt kann bestimmt werden, was Benutzer damit anfangen können. Auch die Benutzer können für ihre erzeugten Inhalte die Zugriffsrechte selbst bestimmen.
Durch strikte Eingabedaten-Validierung wird ein guter Schutz gegen SQL-Injections und Cross-Site Scripting gewährleistet.
Mittels einer komplett frei definierbaren URL-Struktur ist ein wichtiger Punkt im Sinne der Suchmaschinenoptimierung erfüllt.

7.2 Resümee

Der große Vorteil der erstellten Basisarchitektur für soziale Netzwerke liegt in der einfachen Verwendung. In konkreten Projekten kann diese Basisarchitektur leicht eingebunden und erweitert werden. Das umgesetzte System beinhaltet alle Charakteristiken eines sozialen Netzwerkes. Das Einbinden von multimedialen Inhalten, wie Profilfotos, Fotoalben und Videos, wurde jedoch nicht berücksichtigt.

Das in der Basisarchitektur enthaltene Rechtesystem ist performanter als das Pendant von CakePHP. Jedoch ist aufgrund der mehrstufigen Berechtigungsermittlung der Weg zum Auslesen einer Berechtigung komplexer.
CakePHP hat viele häufig wiederkehrende Aufgaben abgenommen und erleichtert. Durch die Trennung von Präsentations- Business- und Persistenzschicht ist eine vereinfachte Wiederverwendbarkeit und Erweiterbarkeit gewährleistet.
Um frei definierbare URLs zu ermöglichen, musste sich von der CakePHP internen Routenermittlung getrennt werden, da diese nicht genügend Flexibilität bot.
Aufgrund der engen Verzahnung der Komponenten innerhalb der Basisarchitektur bieten sich wie erwähnt Vorteile in der zügigen Einbindung in neue Projekte. Ein Nachteil dieser engen Verzahnung ist jedoch, dass sich einzelne Komponenten nicht direkt in andere Anwendungen integrieren lassen.
Ajax – Funktionalitäten ließen sich ohne Probleme in die Basisarchitektur integrieren.

7.3 Ausblick

Es sind noch zahlreiche Funktionen denkbar, die in die erstellte Basisarchitektur aufgenommen werden könnten. Bisher bestehen keine Funktionen, die es Benutzern erlauben Dateien ins soziale Netzwerk einzubinden. Das wäre vor allem für Bilder und Videos interessant. Eine Oberfläche zum Verwalten der Konfiguration, Sprachtexte, Berechtigungen und Links wäre eine sinnvolle Erweiterung.
Das Verfahren zum Erstellen der Basisarchitektur für soziale Netzwerke lässt sich auch auf andere Anwendungsgebiete übertragen.

Glossar

ARO: Ein ARO ist der Initiator einer Handlung. ARO ist das Akronym für „Access Request Object“. Mögliche AROs sind z. B. User, Eigentümer, Mitglied etc.

ACO: Ein ACO ist das Subjekt einer Handlung. ACO steht für „Access Control Object“. Mögliche ACOs sind z. B. User, Gruppe, Forum etc.

AROACO: Zugriffsrecht eines AROs auf ein ACO.

Application-Controller: Oberklasse aller Controller. Stellt allgemeine Funktionen zur Verfügung. Bietet Möglichkeiten um den Requestablauf zu beeinflussen.

Controller-Action: Enthält die Businesslogik, welche für einen Seitenaufruf zuständig ist.

Basisarchitektur: Die Basisarchitektur stellt sowohl eine Art Klassenbibliothek als auch eine komplette Webanwendung da.

Basisklassen: Als Basisklassen werden alle Klassen innerhalb der Basisarchitektur bezeichnet. In konkreten Projekten können diese konfiguriert und erweitert werden.

Behavior: Mit Behaviors ist es möglich, das Verhalten eines Models zu beeinflussen. In einem Behavior kann bestimmt werden, für welche Operationen sich das Model anders als gewohnt verhalten soll.

Controller: Ein Controller enthält die Geschäftslogik für einen Bereich der Anwendung.

Model: Models sind die Repräsentationen einer Datenbanktabelle bzw. eines Datensatzes.

View: Die Views enthalten die gesamte Darstellungslogik, um die vom Controller übergebenen Daten zu formatieren und dem User zu präsentieren.

Literaturverzeichnis

1: Thomas Walter, Kompendium der Web-Programmierung, Springer, 2007

2: Gerti Kappel, Birgit Pröll, Siegfried Reich und Werner Retschitzegger, Web Engineering. Systematische Entwicklung von Webanwendungen, Dpunkt Verlag, 2003

3: Jeremiah Owyang, Social Network Stats: Facebook, MySpace, Reunion (Jan, 2008) , 11.11.2008, http://www.web-strategist.com/blog/2008/01/09/social-network-stats-facebook-myspace-reunion-jan-2008/

4: Facebook, Facebook-Statistiken, 13.11.2008, http://www.facebook.com/press/info.php?statistics

5: Forrester Research, Global Enterprise Web 2.0 Market Forecast: 2007 To 2013, 11.11.2008, http://www.forrester.com/Research/Document/Excerpt/0,7211,43850,00.html

6: Michael Bächle, Paul Kirchberg, Informatik Spektrum - Frameworks für das Web 2.0, Springer Verlag, 2007, Band 30

7: Cake Software Foundation, Associations: Linking Models Together, 11.11.2008, http://manual.cakephp.org/view/78/Associations-Linking-Models-Together

8: Cake Software Foundation, Typical CakePHP Request, 11.11.2008, http://manual.cakephp.org/view/21/A-Typical-CakePHP-Request

9: Cake Software Foundation, Access Control Lists, 11.11.2008, http://book.cakephp.org/view/171/access-control-lists

10: Cake Software Foundation, Testing, 11.11.2008, http://book.cakephp.org/view/160/Testing

11: Christian Wenz, JavaScript und Ajax, Galileo Press, 2006

12: Stefan Münz, Wolfgang Nefzger, HTML Handbuch, Franzis, 2005

13: Frank Thiesing, Sebastian Kortemeyer, Informatik Spektrum - Entwicklung moderner Web-Anwendungen mit Open-Source-Bausteinen, Springer Verlag, 2008, Band 31

14: Prototype Core Team, , 11.11.2008, http://www.prototypejs.org/2008/8/11/practical-prototype-and-scriptaculous

15: Google Inc., Google Maps API, 11.11.2008, http://code.google.com/apis/maps/documentation/

16: Ed Eliot, Visual and Audio PHP CAPTCHA Generation Class, 11.11.2008, http://www.ejeliot.com/pages/2

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 21. November 2008
Ort, Datum

Unterschrift