Arbeitsgruppe Programmiersprachen und Übersetzerkonstruktion
Institut für Informatik
Christian-Albrechts-Universität zu Kiel

Bachelorarbeit

Erstellen einer Abstimmungssoftware für Vorlesungen

Malte Hecht

11. April 2013

Betreut durch: Prof. Dr. Michael Hanus und M.Sc. Björn Peemöller

Erklärung der Urheberschaft

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Ort, Datum

Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

1 Einleitung

1.1 Motivation

Üblicherweise werden Abstimmungen (z.B. während Vorlesungen) durchgeführt, indem der Dozent den Teilnehmern eine Frage stellt, woraufhin die Teilnehmer durch Handheben oder durch Unterlassen des Handhebens ihre Antwort geben. Falls eine kompliziertere Fragestellung und eine genauere Auswertung gewünscht ist, können Fragebögen o.Ä. verteilt werden. Dies erfordert jedoch mehr Zeit (und Personal) in der Vor- und Nachbereitung. Diese Zeit fehlt dann bei der Vermittlung des Unterrichtsstoffs. Ferner lassen sich schriftliche Abstimmungen aufgrund ihrer Dauer nicht gut in die Veranstaltung integrieren.

Das Handhebesystem hat ebenfalls einige Nachteile. Zum einen gibt es oftmals sehr viele Enthaltungen, da das System nicht anonym ist. Insbesondere bei Fragen, die das Verständnis des Unterrichtsstoffs oder die Bewertung des Dozenten betreffen, ist ein korrektes Ergebnis nicht zu erwarten. Ferner stößt man bei mehr als drei Antwortmöglichkeiten an die Grenzen des Systems, da eine exakte Auswertung mit steigender Anzahl der Antwortmöglichkeiten immer schwieriger wird. Die Vorteile eines Audience Response Systems (ARS) liegen also auf der Hand. Durch die Anonymität des Systems ist gewährleistet, dass die Teilnehmer ehrlichere Antworten abgeben. Des Weiteren sind komplexe Fragestellungen mit vielen Antwortmöglichkeiten möglich, die trotzdem in kurzer Zeit durchgeführt und exakt ausgewertet werden können. Ferner ermöglicht die unmittelbare Auswertung einer Abstimmung es dem Dozenten, direkt auf das Ergebnis einzugehen (z.B. durch erneutes Erklären des Unterrichtsstoffs).

Möglich wird eine solche Software durch die zunehmende Verbreitung von mobilen Internetzugängen. Dadurch entfällt die Anschaffung, Installation und Wartung von Hardware zum Abstimmen. Für Teilnehmer ohne Smartphone oder Notebook kann inzwischen günstige Abstimmungshardware in Form von Smartphones angeschafft werden, die dem Teilnehmer für die Dauer der Veranstaltung zur Verfügung gestellt werden.

1.2 Ziel der Arbeit

Die Arbeit befasst sich mit der Entwicklung eines Audience Response Systems. Die Software soll es ermöglichen, Abstimmungen und kurze Befragungen unter Teilnehmern einer Veranstaltung zügig durchzuführen und auszuwerten. Die Anschaffungskosten für Hard-

ware sollen gering gehalten werden, indem auf bei den Teilnehmern bereits vorhandene Hardware (Smartphones, Notebooks) zurückgegriffen wird.

2 Audience Response Systeme

Audience Response Systeme (ARS) sind Systeme die es ermöglichen Abstimmungen mit einem (großen) Publikum durchzuführen. Hierzu ist zumeist Abstimmungshardware in Form einer Art Fernbedinung notwendig. Diese wird jedem Teilnehmer zur Verfügung gestellt. Eine solche Fernbedienung besitzt eine Anzahl an Knöpfen, mit denen Teilnehmer einer Abstimmung eine Antwortmöglichkeit auswählen können. Zu diesem Zweck müssen die Fragestellung und die Antwortmöglichkeiten den Teilnehmern vorher zugänglich gemacht werden (z.B. mittels eines Beamers).

Inzwischen existieren bereits Lösungen, die versuchen die spezielle Abstimmungshardware zu ersetzen. Hierzu wird auf unterschiedlichem Wege Abstimmungssoftware auf Geräten wie Smartphones, Tablets sowie Note- und Netbooks verfügbar gemacht. Diese Geräte ermöglichen hierdurch eine Teilnahme an im System angebotenen Abstimmungen. Somit ist in einem solchen Fall spezielle Abstimmungshardware nicht mehr notwendig.

2.1 eduVote

eduVote[1] ist eine Software der Buchholz Wengst GbR und wird bereits an einigen deutschen Hochschulen eingesetzt. Um Abstimmungen durchzuführen zu können muss der Nutzer jedoch eine Lizenz erwerben durch die jährliche Kosten entstehen (Einzellizenz: 299 €, Rahmenlizenz: 2800 €). Daraufhin erhält er einen Loginnamen und ein Passwort. Mittels einer nativen Desktop-Anwendung (diese ist für Windows- und Mac-Systeme verfügbar) ist es ihm nun möglich sich am System anzumelden. Hier kann er einzelne Abstimmungen oder Serien von Abstimmungen erstellen. An diesen können Teilnehmer nun teilnehmen. Zu diesem Zweck stehen für verschiedene Systeme native Applikationen bereit (iOS, Android, Weboberfläche). Zur Teilnahme müssen die Teilnehmer den Loginnamen des Dozenten in die jeweilige Anwendung eingeben. Anschließend wählen sie eine von fünf Antwortmöglichkeiten (A, B, C, D, E oder yes, no). Die Komplexität der Fragestellungen ist also durch die Anzahl der Antwortmöglichkeiten stark begrenzt. Die Fragestellung und die Antwortmöglichkeiten sind nicht auf den Geräten der Teilnehmer sichtbar. Daher müssen diese vom Dozenten zur Verfügung gestellt werden (z.B. via Beamer). Mittels eines Sessioncodes lassen sich Abstimmungen mehrfach durchführen und auswerten. Der Sessioncode muss von den Teilnehmern dann ebenfalls bei der Anmeldung am System eingegeben werden. Nach Durchführung der Abstimmung kann der Dozent das Ergebnis über die ihm zu Verfügung gestellte native Anwendung einsehen.

[0] `http://www.eduvote.de`

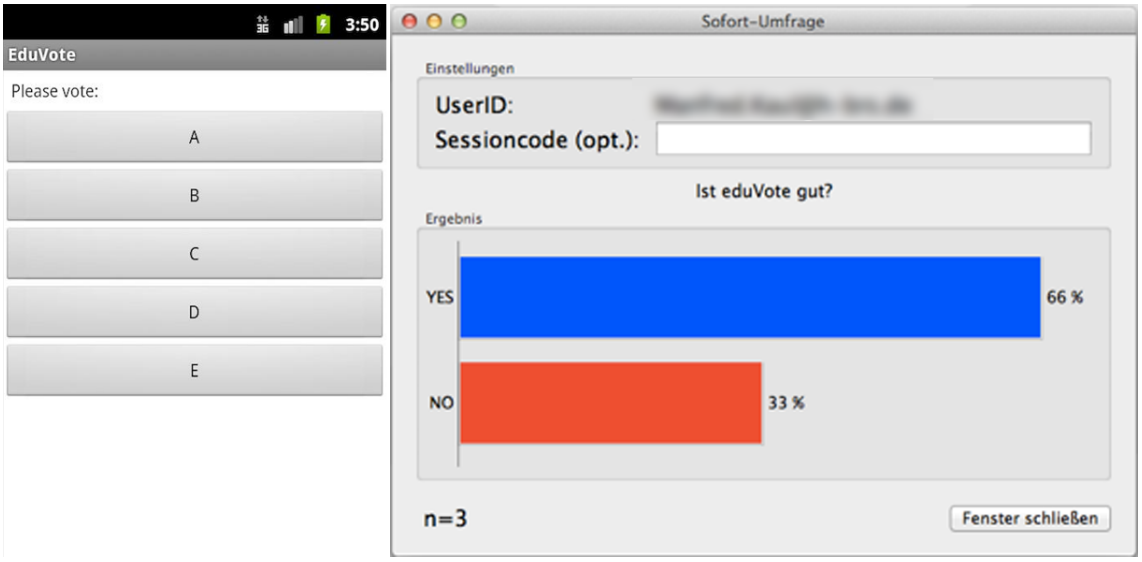


Abbildung 2.1: eduVote Android-App[2] (links) und Dozenten-Anwendung[3] (rechts)

2.2 mQlicker

Das Audience Response System namens mQlicker[4] wird vom australischen Unternehmen Agin3 Pty Ltd.[5] entwickelt. Um Abstimmungen durchzuführen ist es notwendig sich auf der mQlicker-Webseite zu registrieren. Anschließend kann man sich in der sogenannten Presenter-Oberfläche[6] anmelden. Hier lassen sich Interaktionen erstellen, die aus einer oder mehreren Fragestellungen bestehen. Zu diesen lassen sich beliebig viele Antwortmöglichkeiten hinzufügen. Zur Durchführung einer Interaktion wird dieser eine Session hinzugefügt, die sich sich starten, pausieren und beenden lässt. Beim Starten der Session wird ein Sessioncode ausgegeben, den ein Dozent nun seinem Publikum mitteilen kann. Über eine Webseite[7] und durch Eingabe des Sessioncodes erhält das Publikum nun Zugang zur Interaktion des Dozenten. In dieser werden dem Teilnehmer die in der Interaktion enthaltenen Fragen gestellt. Das Ergebnis der Session kann der Dozent in der Presenter-Oberfläche aufrufen. Neben Fragen mit Einfach oder Mehrfachauswahl sind auch Fragen zu denen die Teilnehmer freie Textantworten abgeben können möglich. Rangordnungen von Antwortmöglichkeiten lassen sich jedoch nicht erstellen.

[2] `http://www.androidpit.de/de/android/market/apps/app/de.eduvote.ted/eduVote`
[3] `https://kaul.inf.h-brs.de/wordpress/2012/12/eduvote`
[4] `https://www.mqlicker.com`
[5] Noch keine Webseite vorhanden.
[6] `https://mqlicker.com/user`
[7] `https://respond.cc`

2.3 PPVote / CliKAPAD

PPVote[9] und CliKAPAD[10] sind Marken der Firma Albert Hall Meetings Limited. Bei PPVote handelt es sich um eine Abstimmungssoftware. Die dazugehörige Hardware (verschiedene Audience Reply Keypads und USB-Basisstation) wird unter dem Namen CliKAPAD vertrieben. Eingehende Teilnehmerantworten werden über eine USB-Basisstation empfangen. Die Antworten werden von den Audience Reply Keypads, die den Teilnehmern zuvor ausgehändigt werden, gesendet. Fragestellung und Antwortmöglichkeiten werden den Teilnehmern vom Dozenten zur Verfügung gestellt (z.B. via Beamer). Auf den Keypads wählen die Teilnehmer nun eine Antwortmöglichkeit um teilzunehmen. Das Ergebnis einer Abstimmung lässt sich z.B. als Diagramm in Präsentationen, die mit bekannter Präsentationssoftware erstellt wurden, einbinden. Die Preise für ein Audience Reply Keypad von CliKAPAD liegen zwischen £40 und £110. Eine USB-Basisstation kostet £300. Es ist auch möglich die Hardware zu mieten.

Abbildung 2.2: Keypad[8]

[8] http://www.androidpit.de/de/android/market/apps/app/de.eduvote.ted/eduVote
[9] http://www.ppvote.com
[10] http://www.clikapad.com

3 Verwendete Software und Techniken

3.1 Ruby on Rails

Ruby on Rails ist ein von David Heinemeier Hansson in Ruby geschriebenes, quelloffenes Webframework. Es wurde im Jahre 2004 erstmals veröffentlich. Ursprünglich diente es zur Ent-wicklung der Projektverwaltungssoftware Basecamp. Mit Rails erstellte Anwendungen sind nach dem MVC-Muster aufgebaut. Die Grund-Prinzipien sind don't repeat yourself (DRY) und Konvention über Konfiguration. Das DRY-Prinzip besagt, dass redundante Informationen vermieden werden sollen. Im Vergleich zu anderen Webframeworks ist z.B. eine Angabe der Attribute in den Modell-Klassen nicht notwendig. Das Persistenz-Framework Active Record (siehe 3.1.1) übernimmt diese aus den zugehörigen Tabellen der Datenbank. Das Prinzip der Konvention über Konfiguration fällt insbesondere Rails-Einsteigern positiv auf. Mit nur drei Kommandozeilen lässt sich eine erste lauffähige Rails-Applikation generieren. Grundsätzlich ist in Rails zu Beginn alles so konfiguriert, wie die meisten typischen Webanwendungen es erfordern. Wie der Name vermuten lässt, werden Rails-Anwendungen in Ruby programmiert. [MO12, S. 21-26]

Das in Rails das MVC-Muster eine wichtige Rolle spielt, zeigt auch der Aufbau des Frameworks. Die drei wichtigsten Komponenten heißen Active Record (Model), Action View (View) und Action Controller (Controller). [MO12, S. 24]

3.1.1 Active Record

Active Record ist ein von Martin Fowler in seinem Buch Patterns of Enterprise Application Architecture beschriebenes Architekturmuster. In Rails wird eine Ruby-Implementierung dieses Musters als Persistenz-Framework und ORM-Schicht eingesetzt. Die Bibliothek stellt eine Basisklasse zur Verfügung, von der die konkreten Modellklassen abgeleitet werden. Eine abgeleitete Klasse entspricht einer gleichnamigen, existierenden Tabelle der Datenbank. Instanzen dieser Klasse sind somit Repäsentationen einer Zeile in der Tabelle. [Fow03] [MO12, S. 239]

Listing 3.1: eine eigene ActiveRecord Klasse

```
1 class User < ActiveRecord::Base
2 end
```

Die entsprechende Tabelle in der Datenbank könnte so aussehen:

Listing 3.2: Entsprechende Tabelle

```
1 CREATE TABLE users (
2   id int(11) NOT NULL auto_increment,
3   login varchar(255),
4   password varchar(255),
5   PRIMARY KEY  (id)
6 );
```

Man sieht hier, dass die Spalten der Tabelle als Attribute in der Klasse nicht angegebenen werden. Active Record übernimmt diese automatisch aus der Datenbank. In einer Modell-Klasse können auch Relationen zu anderen Klassen des Modells, Validationsbedingungen für Attribute etc. angegeben werden. [MO12, S. 272-280]

Für den Datenzugriff wird innerhalb der Railsanwendung nur mit den entsprechenden Instanzen einer Modell-Klasse gearbeitet. Die direkten Anfragen an die Datenbank übernimmt Active Record. Es werden verschiedene Datenbanksysteme unterstützt[1]. Die im Zuge dieser Arbeit entwickelte Anwendung nutzt eine SQLite3 Datenbank.

3.1.2 Action View

Das Action View-Modul ist in Ruby on Rails für die Verarbeitung der Templates zuständig. Es kann verschiedene Arten von Templates verarbeiten. Die meisten Webanwendungen nutzen im Zusammenhang mit Rails die ERB-Templates, die an der Dateiendung .erb zu erkennen sind. ERB steht für Embedded Ruby. Die Templates bestehen also z.B. aus einer Auszeichnungssprache wie HTML und beinhalten an einigen Stellen eingebetteten Ruby-Code. [MO12, S. 371-372]

Listing 3.3: ein ERB-Template

```
1 <% if @voting.title %>
2   <h1><%= @voting.title %></h1>
3 <% else %>
4   <p>Es ist kein Titel vorhanden</p>
5 <% end %>
```

Listing 3.3 zeigt zwei Arten von eingebettetem Ruby-Code. Zum einen lässt sich Code mit `<% ... %>` einbetten. Auf diese Art eingebetteter Code wird ausgeführt, mögliche Rückgabewerte werden jedoch nicht in das Template eingebunden. Durch `<%= ... %>` eingebundener Code werden die Rückgabewerte der Code-Zeile in das Template eingebunden.

In Rails existieren drei verschiedene Bereiche für Templates. Layout-Templates, Standard-Templates und Partials. Layout-Templates enthalten meist die Grundstruktur der anzuzeigenden Seite. Das in der Datei `app/layouts/application.html.erb` hinterlegte Template wird standardmäßig als Rahmen für alle zu rendernden Standard-Templates verwendet. Es können weitere Layout-Templates im `layout`-Ordner hinterlegt werden. [MO12, S. 395]

[1] `https://github.com/rails/rails/tree/master/activerecord`

Die Standard-Templates enthalten den Großteil der eigentlichen Oberfläche [MO12, S. 372]. Oftmals enthalten sie Elemente, die auch in anderen Templates benötigt werden oder sich häufig wiederholen. Hierfür hält Rails sogenannte Partials bereit. Mit ihnen lassen sich häufig benötigte Elemente in weitere Dateien auslagern. Den Dateinamen wird ein Unterstrich vorangestellt. Mit dem Befehl:

```
<%= render :partial => "name_des_partials" %>}
```

wird ein Partial in ein Layout-Template, Template oder ein anderes Partial eingebunden. [MO12, S. 141][BCD+13]

3.1.3 Action Controller

Das Action Controller Modul enthält die Logik des Programms. Es verbindet die View und somit die Interaktionen des Benutzers mit dem Modell. Dazu enthält jeder Controller mindestens eine Action. Actions sind öffentliche Methoden in den Controllern. In der Datei `routes.rb` wird zu jeder Action eine Route festgelegt. Die jeweilige Action wird bei Aufruf der Route ausgeführt und rendert abschließend ein Template oder ruft eine weitere Action auf. [MO12, S. 331][api13b]

3.1.4 Action Dispatch

Die Action Dispatch-Bibliothek ist ein weiterer Teil des Rails-Frameworks. Durch sie werden die eingehenden Anfragen an die entsprechenden Actions der Controller weitergeleitet. Hierzu werden in der Routen-Konfigurationsdatei `routes.rb` verschiedene Routen festgelegt. Dazu gibt es mehrere Möglichkeiten. [MO12, S. 24]

Listing 3.4: eine Möglichkeit eine Route anzulegen

```
1    get "users/:id" => "users#show", as: "users_show"
2    put "users/:id" => "users#update"
```

Zeile 1 des Listings 3.4 zeigt wie eine Route definiert werden kann. Zuerst wird die HTTP-Request-Methode festgelegt, in diesem Fall wurde die GET-Methode gewählt. Dann wird mit `users/:id` die eigentlich Route angegeben, wobei `:id` ein zu übergebender Parameter ist. Anschließend folgt mit `users#show` der Teil, in dem Controller und Action angegeben werden [MO12, S. 352]. Im Beispiel wird beim Aufruf dieser Route also im UsersController die show-Action ausgeführt. Das anschließende `as: "show_user"` gibt der Route einen Namen. Dadurch ist es möglich im Controller oder der View die Route zu referenzieren. [MO12, S. 376] Im einem Template könnte so ein Link dargestellt werden:

Listing 3.5: benannten Routen nutzen

```
1  <%= link_to 'Benutzer anzeigen', users_show_path(user.id) %>
```

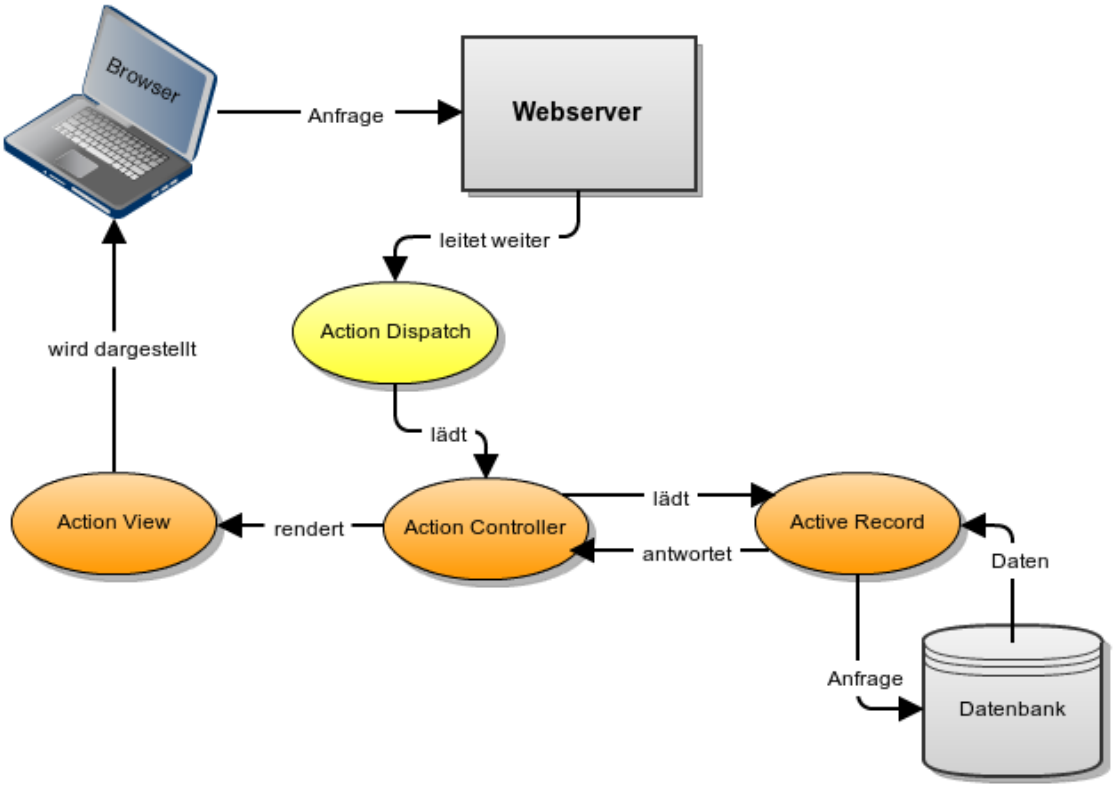


Abbildung 3.1: Aufbau und Funktionsweise einer Ruby on Rails-Applikation

3.2 Ruby

An dieser Stelle soll nur auf einige wenige Aspekte von Ruby eingegangen werden, damit die im Implementierungs-Kapitel vorgestellten Code-Ausschnitte besser nachvollzogen werden können.

Ruby ist eine interpretierte, höhere Programmiersprache, die erstmals 1995 vom Japaner Yukihiro Matsumoto veröffentlicht wurde. Ruby ist stark objektorientiert und ermöglicht imperative sowie funktionale Programmierung. Bis 2004 war Ruby hauptsächlich in Japan verbreitet. Die Bekanntheit stieg jedoch stark durch die Veröffentlichung von Ruby on Rails. Ruby-Code ist aufgrund der geringen Anzahl an Sonderzeichen und der hohen Objektorientierung sehr gut lesbar. [MO12, S. 57-59][rub13b]

3.2.1 Variablen

In Ruby gibt es verschiedene Arten von Variablen. Für das Verständnis der Codebeispiele ist es wichtig lokale Variablen und Instanzvariablen zu unterscheiden. Instanzvariablen

sind an ein Objekt gebunden und innerhalb dieses Objekts gültig. Sie werden durch ein `@`-Zeichen vor dem Variablennamen gekennzeichnet (siehe Listing 3.2). Da der in den Templates enthaltene Ruby-Code in der Instanz eines Controllers gerendert wird, werden sie genutzt um Daten in die View zu übertragen.[2] Lokale Variablen sind dagegen nur innerhalb eines Programmblocks, also z.B. einer Methode gültig und haben kein `@`-Zeichen vor ihrem Namen. [rub13a][MO12, S. 64-65]

3.2.2 Gems

Gems sind Ruby Bibliotheken im gem-Format. RubyGems ist das offizielle Paketsystem für Ruby-Bibliotheken. Sind Ruby und RubyGems installiert, lassen sich über die Kommandozeile schnell Gems installieren. Mit dem Befehl `gem install rails` ließe sich z.B. das Ruby on Rails Framework installieren. Um Gems innerhalb einer Railsanwendung nutzen zu können, müssen die gewünschten Bibliotheken im Gemfile des jeweiligen Rails-Projekts hinzugefügt werden. Hierzu wird dem Gemfile einer Railsanwendung beispielsweise die Zeile `gem 'sqlite3'` hinzugefügt. Ruft man nun im Rootverzeichnis der entsprechenden Railsanwendung den Befehl `bundle install` auf, wird die aktuellste Version der SQLite3 Datenbankanbindung heruntergeladen und installiert. Einige Gems wie z.B. das SQLite3-Gem sind von Beginn an im Gemfile einer Railsanwendung eingetragen [MO12, S.202-204]. Im Zuge dieser Bachelorarbeit wird auch auf weitere Gems zurückgegriffen:

bcrypt

Das bcrypt-ruby-Gem wird zum Verschlüsseln der Passwörter eingesetzt. Es sorgt dafür, dass Passwörter sicher in der Datenbank verwahrt werden, selbst wenn Dritte Zugang zu dieser erlangen. [MO12, S. 144]

nbayes

Das nbayes-Gem ist eine Ruby-Implementierung des naiven Bayes-Klassifikators[3]. Dieser wurde aus dem Bayestheorem des Mathematikers Thomas Bayes abgeleitet. Das Gem wird benutzt um abgegebene Textantworten zu klassifizieren. [Wik13b]

3.3 Javascript

Javascript ist eine Scriptsprache, die heutzutage hauptsächlich für die Manipulation einer HTML-Seite in Baumdarstellung (siehe Document Object Model 3.3.1) genutzt wird. Sie wurde 1995 unter dem Namen Livescript als Teil des Netscape Navigators veröffentlicht

[2] http://stackoverflow.com/questions/8528411/how-are-instance-variables...
[3] `https://github.com/oasic/nbayes`

und sollte die Einflussnahme auf HTML-Seiten während der Darstellung ermöglichen. Der Javascript-Sprachkern ist durch die Firma Ecma International unter dem Namen ECMAScript [ECM11] standardisiert. Javascript bietet die Möglichkeit objektorientiert, prozedural oder funktional zu programmieren. Die Objektorientierung wird durch klassenlose Objektorientierung mit Hilfe von Prototypen realisiert. Es wird also auf Klassen als Sprachelement verzichtet. Ferner ist Javascript dynamisch typisiert. [Wen01]

3.3.1 Document Object Model

Das Document Object Model (DOM) ist eine durch das W3 Konsortium (W3C) spezifizierte Schnittstelle [KGHM12] für den Zugriff auf HTML- und XML-Dokumente. Diese lassen sich aufgrund ihrer Struktur als Knoten-Baum darstellen. Die spezifizierte Schnittstelle ermöglicht es solche Dokumente einzulesen und daraus ein Dokumenten-Objekt zu erstellen. Die Methoden der Schnittstelle ermöglichen es innerhalb des Baumes zu navigieren und so ausgewählte Knoten zu bearbeiten. Anschließend wird aus dem Dokumenten-Objekt durch Serialisation wieder ein gültiges HTML- bzw. XML-Dokument, welches die am Objekt durchgeführten Änderungen enthält. [BV10, S. 563-567][LHWW09]

Inzwischen implementieren alle aktuellen Browser das DOM mehr oder weniger vollständig [Sch07]. Mit Javascript nutzen wir die Schnittstelle um die Struktur unseres HTML-Dokuments zu verändern und zu erweitern.

3.3.2 AJAX / XML-HttpRequest

XML-HttpRequest ist eine inzwischen ebenfalls durch das W3C spezifizierte Schnittstelle, welche die Übertragung beliebiger Daten via HTTP ermöglicht. Die XMLHttpRequest-Technik wird von aktuellen Browsern unter verschiedenen Namen implementiert[4][5][6]. Die in dieser Arbeit vorgestellte Software nutzt diese Schnittstelle um Daten z.B. nachzuladen. Die empfangenen Daten werden dann durch Manipulation des DOM dem HTML-Dokument hinzugefügt. Dies hat den Vorteil, dass nicht die gesamte Seite in einer HTTP-Anfrage erneut übermittelt werden muss. Um dies zu erreichen wird in Javascript ein neues XMLHttpRequest-Objekt erzeugt. Über dieses Objekt wird eine Anfrage an den Server gesendet. Ist die Antwort des Servers eingegangen, können die empfangenen Daten durch Javascript weiter verarbeitet und dem DOM hinzugefügt werden. [ASRMSK12]

Für den eben beschriebenen Vorgang wird im Allgemeinen häufig der Begriff AJAX (Asynchronous JavaScript and XML) verwendet. Damit ist die asynchrone (XML-) Datenübetragung zwischen Browser und Server gemeint, die mit Hilfe von Javascript stattfindet. Statt XML lassen sich auch andere Daten übermitteln. Häufig wird das JSON-Format gewählt, da der Datenoverhead verglichen mit dem XML-Format geringer ist.

[4] `http://msdn.microsoft.com/de-de/library/ms760305.aspx`
[5] `https://developer.mozilla.org/en-US/docs/DOM/XMLHttpRequest?redirectlocale=en-US&redirectslug=XMLHttpRequest`
[6] `https://developer.apple.com/devcenter/safari/index.action`

[Wik13a]

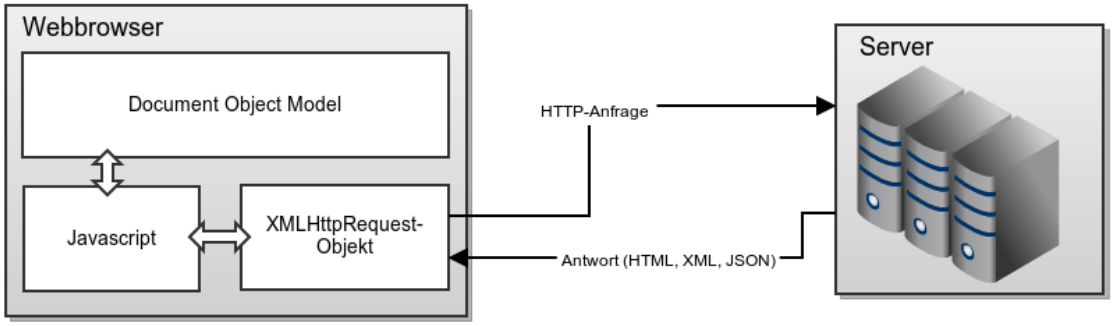


Abbildung 3.2: Ablauf einer AJAX-Anfrage

AJAX in Rails

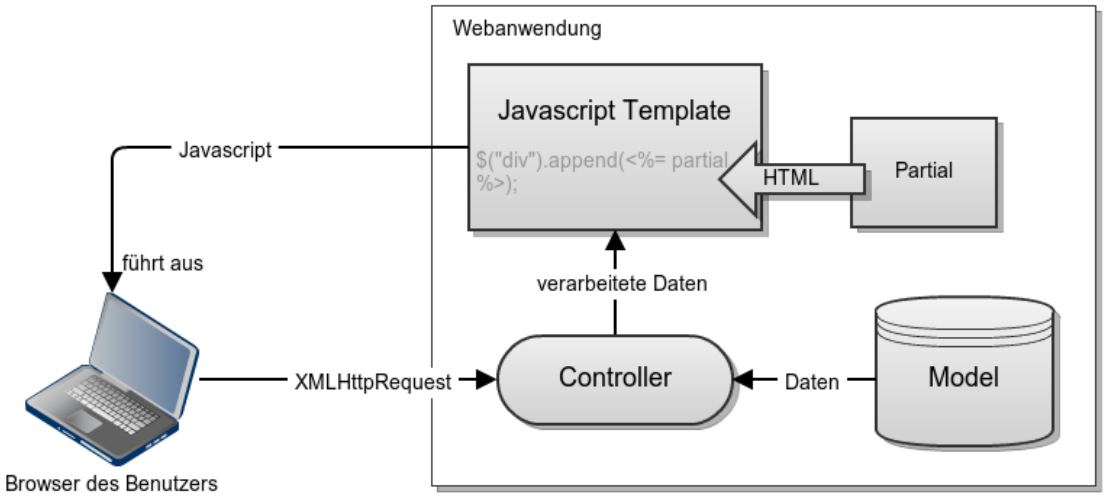


Abbildung 3.3: AJAX in Ruby on Rails

Die Abbildung 3.3 zeigt wie AJAX in Ruby on Rails umgesetzt wurde. Beim Speichern einer Abstimmung wird über das XMLHttpRequest-Objekt des Browsers ein XMLHttpRequest an die Webanwendung geschickt. Der Controller erkennt diese Anfrage als solche und lädt die verarbeiteten Daten daher in ein Javascript Template, welches Javascript enthält. Soll HTML-Code per Javascript z.B. in ein Objekt des DOM geladen werden, wird dieser idealerweise seperat in einem Partial (siehe Action View 3.1.2) abgelegt. Die Besonderheit ist nun, dass die Antwort an den Browser des Benutzers das gerenderte Javascript enthält (und keine Daten im XML- oder JSON-Format). Dieses wird vom

Browser ausgeführt.

3.3.3 JQuery

JQuery ist ein Javascript-Framework. Seit der Ruby on Rails Version 3.1 ist es standardmäßig Bestandteil einer Railsanwendung [MO12, S. 28] und löste das vorher eingesetzte Framework Prototype ab. JQuery vereinfach die Javascript Programmierung, indem es häufig benötigte Funktionalitäten bereitstellt. Diese sind in allen gängigen Browsern funktionstüchtig. Daher müssen mit JQuery selten an einzelne Browser angepasste Lösungen entwickelt werden. [BV10, S. 15-16] Als Beispiel bietet sich eine AJAX-Anfrage an[api13a]:

Listing 3.6: AJAX mit JQuery

```
1 $.ajax({
2   url: "daten.html",
3   context: document.body
4 }).done(function(data) {
5   alert("Empfangen: " + data);
6 });
```

Das Listing 3.6 zeigt wie mit JQuery eine AJAX-Anfrage ausgeführt wird. Die Instanzierung des dem Browser entsprechenden XML-HttpRequest-Objekts übernimmt hierbei das Framework.

Ferner erleichtert JQuery den Zugriff auf das DOM, indem es die sogenannte Factory-Funktion `$()` bereitstellt. Sie ist das Kernelement des Frameworks. Über Selektoren, die an CSS-Selektoren angelehnt sind, sind alle Knoten des DOM leicht zu erreichen. Um alle div-Ebenen im DOM zu erhalten, kann mit JQuery statt `document.getElementsByTag('div')`, die Factory-Funktion [BV10, S. 42-43] verwendet werden: `$('div')`. Durch den Aufruf von `$('div')` werden die Knoten als JQuery-Objekte gekapselt zurückgegeben. Die JQuery-Objekte stellen Methoden zur Verfügung, welche die weitere Verarbeitung des Elements oder der Elemente erleichtern. Es sind natürlich auch komplexere Selektoren möglich wie `$('div.nav > a[href^="http"]')`. Hiermit lassen sich alle `a`-Elemente, die in `div`-Ebenen der Klasse `nav` enthalten sind und deren `href`-Attribut mit `http` beginnt, auswählen. Etwas Äquivalentes in reinem Javascript wäre deutlich komplexer. [BV10, S. 67-83][api13a]

JQuery UI

JQuery UI ist eine auf JQuery aufbauende Erweiterung des Frameworks. Sie bietet Lösungen zur Gestaltung der Benutzeroberfläche. Diese sind unterteilt in:

Interaktionen Funktionen, die eine Interaktion des Nutzers mit Objekten aus dem DOM ermöglichen. Für diese Arbeit werden Sortables verwendet, die es dem Anwender ermöglichen ausgewählte Objekte zu sortieren.

Effekte Eine Auswahl von Funktionen für visuelle Effekte wie Überblendungen und Farbanimationen (etc.). Für das in dieser Arbeit beschriebene Audience Response System werden insbesondere Farbanimationen genutzt, um dem Anwender Fortschritt und Erfolg bzw. Misserfolg von Aktionen anzuzeigen.

Widgets Widgetss sind vorgefertigte Elemente für die Oberfläche (Buttons, Dialoge etc.). Statt der von JQuery UI angebotenen Widgets wird auf die Elemente der Bootstrap-Bibliothek zurückgegriffen.

Themes Es lassen sich CSS-Stylesheets generieren, die das Aussehen der Widgets und Interaktionen bestimmen. Da keine JQuery UI-Widgets zum Einsatz kommen, wird auf Themes verzichtet. Das Layout der benötigten Interaktionen wird den CSS-Stylesheets der Railsanwendung von Hand hinzugefügt.

Neben JQuery UI werden noch weiter auf JQuery aufbauende Erweiterungen eingesetzt. Das jCountdown-Plugin ermöglicht es Countdowns zu erstellen. Die Countdowns werden eingesetzt um die Restdauer aktiver Abstimmungen anzuzeigen.[7] Für die Benutzerverwaltung werden alle im System registrierten Benutzer in einer Tabelle dargestellt. Zu übersichtlicheren Gestaltung dieser kommt das DataTables-Plugin zum Einsatz. Es ermöglicht die Sortierung der Tabelle durch Anklicken einer Spaltenüberschrift. Des Weiteren bietet es Funktionen zur Paginierung und zum Durchsuchen der in der Tabelle enthaltenen Daten.[8] Schlussendlich wurde für die grafische Darstellung der Abstimmungsergebnisse die Highcharts-Erweiterung gewählt. Sie stellt mehrere Diagrammtypen zur Verfügung mit denen sich Ergebnisse grafisch darstellen lassen.[9]

3.4 Bootstrap

Bootstrap ist ein sogenanntes Front-End-Framework und wurde für die Anwendung Twitter[10] entwickelt. Für die Oberflächen zur Verwaltung des Dienstes wurden ursprünglich verschiedenste Bibliotheken verwendet. Dieser Zustand sollte mit dem Einsatz von Bootstrap als einheitliche Bibliothek geändert werden. Da sich das Framework bewährte, wurde es im August 2011 der Öffentlichkeit als Open-Source-Projekt zugänglich gemacht. Seitdem wurde es bereits vielfach eingesetzt.

Zur Oberflächengestaltung bietet Bootstrap verschiedene Elemente. Als Grundgerüst bietet es ein Grid-System an, mit dem sich die grundlegende Struktur einer Seite gestalten lässt. Es lassen sich mit diesem System sogenannte Responsive-Layouts erstellen. Diese passen ihre Darstelllung der Bildschirmauflösung des zum Anzeigen der Seite verwendeten Gerätes an. Ferner werden grundlegende CSS-Styles bereitgestellt, die eine einheitliche Formatierung von Text, Formularen, Tabellen etc. ermöglicht. Außerdem beinhaltet das Framework zahlreiche Komponenten wie Buttons, Tabs etc. zur Bedienung der Oberflä-

[7] `http://www.webmuse.co.uk/projects/jcountdown-jquery-plugin`
[8] `http://www.datatables.net`
[9] `http://www.highcharts.com`
[10] `http://www.twitter.com`

che. Schlussendlich werden auch Komponenten angeboten die auf Javascript zurückgreifen. Der zu diesen Elementen gehörende Javascript-Code nutzt das Javascript-Framework JQuery.[11]

3.5 Android

Android ist eine offene, auf dem Linux-Kernel basierende Plattform für mobile Geräte. Sie wird von der Open Handset Alliance[12] entwickelt. Die Open Handset Alliance ist ein Konsortium aus 84 Software-Firmen, Netzbetreibern, Chip- und Mobiltelefonherstellern und Marketingunternehmen und wurde 2007 von Google mit 33 Partnern geründet. Im selben Jahr wurde auch eine erste Vorabversion des Android Software Development Kits (Android SDK) veröffentlicht. Ein Jahr später folgte mit dem HTC Dream (T-mobile G1) das erste Gerät mit dem Android-System. Inzwischen läuft Android auf einer Vielzahl von Geräten, wie Smartphones, Tablets und Netbooks und ist sehr verbreitet. Teil des Android Systems ist eine auf mobile Geräte angepasste Java virtuelle Maschiene namens Dalvik. Sie führt alle mit Java und dem Android SDK entwickelten Applikationen (Apps) aus, nachdem diese in das Dalvik Executable Format konvertiert wurden. Die mit dem Android SDK entwickelten Apps bauen auf dem Android Application Framework auf. Es ermöglicht Entwicklern den Zugriff auf verfügbare Hardware und gibt eine Anwendungsstruktur vor. [Kü11, S.19-23][Wik13c][13]

Im Folgenden wird auf Teile des Application Frameworks eingegangen, die für das in dieser Arbeit beschriebene Audience Response System von Wichtigkeit sind.

3.5.1 Activities

Activities bestehen in der Regel aus einer Java-Klasse und einer zugehörigen View. Sie können als Aktionen gesehen werden, die vom Benutzer durchgeführt werden können. In unserem Fall sind es Activities, die die Aktionen "zur Veranstaltung anmelden", "an einer Abstimmung teilnehmen" und "erneut nach Abstimmungen schauen" repräsentieren. Von der Activity-Klasse werden einige Methoden geerbt, die den Lebenszyklus einer Activity beschreiben. Diese Methoden werden überschrieben um das gewünschte Ergebnis zu erzeugen. In der Regel wird zumindest die `onCreate`-Methode überschrieben. Sie wird aufgerufen, wenn die Activity gestartet wird. In der `onCreate`-Methode wird dann eine View durch Aufruf der geerbten `setContentView`-Methode geladen. Eine Activity wird beim Starten einer App aufgerufen, wenn sie als Launcher-Actitvity festgelegt wurde. Ansonsten werden Activities durch den Aufruf der Methode `startActvity` gestartet. Ihr wird ein Intent übergeben, der näher beschreibt, welche Activity gestartet werden soll. [Kü11, S.85-86][14]

[11] `http://twitter.github.io/bootstrap`
[12] `http://www.openhandsetalliance.com`
[13] `http://developer.android.com`
[14] `http://developer.android.com/reference/android/app/Activity.html`

3.5.2 Intent

Intents sind abstrakte Beschreibungen von durchzuführenden Aktionen. Sie werden für den Nachrichtenaustausch verwendet. Explizite Intents benennen ihr Ziel, indem ihnen Paket- und Klassenname übergeben werden. Sie werden meist intern für den Nachrichtenaustausch zwischen den einzelnen Activities verwendet. Implizite Intents benennen kein direktes Ziel sondern eine Kategorie. Mit impliziten Intents werden meist Komponenten anderer Apps gestartet, die die angegebene Kategorie verarbeiten können. Die in dieser Arbeit vorgestellte Android-Applikation verwendet ausschließlich explizite Intents. Einem Intent können mit der `putExtra`-Methode primitive Daten übergeben werden, die der Empfänger des Intents auslesen kann. Dies wird z.B. verwendet, um einer Activity die Daten zum Anzeigen einer Abstimmung zu übergeben. [Kü11, S.99-103][15]

3.5.3 AsyncTasks

In Android-Applikationen werden Programmlogik und das Zeichnen der Benutzeroberfläche in einem Thread ausgeführt. Daher wird dieser Thread oft als UI- oder Main-Thread bezeichnet. Nur in diesem Thread ist der Zugriff auf Komponenten der Benutzeroberfläche möglich. Wird die Java-Implementierung von Threads direkt verwendet um Nebenläufigkeit zu realisieren, werden Handler nötig um Änderungen an der Benutzeroberfläche durchzuführen. Aus diesem Grund werden vom Application Framework sogenante AsyncTasks bereitgestellt. Sie bilden eine Helferklasse für die in Java implementierten Threads und Handler. AsyncTasks sind für Operationen gedacht die nicht länger als ein paar Sekunden andauern. Daher eignen sie sich gut um z.B. HTTP-Anfragen auszuführen.[16]

Um einen eigenen AsyncTasks zu erstellen muss eine Klasse von der AsyncTasks-Klasse erben. Von dieser werden die Methoden `onPreExecute(), doInBackground(Params...), onProgressUpdate(Progress...) und onPostExecute(Result)` geerbt. Die Methode `doInBackground` wird überschrieben und enthält den Programmteil, der in einem eigenen Thread ausgeführt werden soll. Die anderen Methoden werden im UI-Thread ausgeführt und haben somit Zugriff auf die Benutzeroberfläche. Die Methode `onPreExecute` wird unmittelbar vor dem Starten des Threads ausgeführt. Sie wird meist genutzt um z.B. eine Statusleiste in der View sichtbar zu machen. Die `onProgressUpdate`-Methode wird aufgerufen wenn Aktualisierungen des Status vorliegen. Die Methode `onPostExecute` wird unmittelbar nach Beenden des Threads ausgeführt, ihr wird als Parameter der Rückgabewert der `doInBackground`-Methode übergeben. Somit wird sie meist zur weiteren Verarbeitung des Ergebnisses und zum Ausblenden der Statusanzeige genutzt. Die Instanz eines AsyncTasks wird gestartet indem die `execute`-Methode des Objektes aufgerufen wird.[17]

[15] `http://developer.android.com/reference/android/content/Intent.html`
[16] `http://developer.android.com/reference/android/os/AsyncTask.html`
[17] `http://developer.android.com/reference/android/os/AsyncTask.html`

3.5.4 Dragsort Listview

Die Dragsort Listview implementiert durch "drag and drop" sortierbare Listen für Android-Applikationen.[18]

[18] https://github.com/bauerca/drag-sort-listview

4 Systementwurf

In diesem Kapitel werden zunächst die Anforderungen an die Software ermittelt. Aus diesen wird dann die Struktur der geplanten Software abgeleitet. Zuerst werden funktionale und nicht-funktionale Anforderungen aufgeführt. Anschließend wird erklärt wieso bei der Entwicklung die in Kapitel 2 vorgestellte Software zur Entwicklung der Abstimmungssoftware verwendet wird. Danach werden Struktur und Funktionsweise des Systems unter Zuhilfenahme von Grafiken und UML-Diagrammen erläutert und ausgewählte Benutzeroberflächen vorgestellt.

4.1 Anforderungen

4.1.1 Funktionale Anforderungen

Aufgabe ist es ein System zu entwerfen, mit dem Teilnehmer einer Veranstaltung an Abstimmungen, die der Veranstaltung zugeordnet sind, teilnehmen können. Die Teilnahme erfolgt über eine Android-Applikation, für Teilnehmer ohne Android-Smartphone wird eine Weboberfläche bereitgestellt, über die ebenfalls abgestimmt werden kann. Jeder Teilnehmer kann nur einmal an einer Abstimmung teilnehmen.

Mit eigenen Benutzerdaten melden sich Dozenten auf einer separaten Weboberfläche an. Nach erfolgreicher Anmeldung besteht die Möglichkeit, Veranstaltungen anzulegen und zu bearbeiten. Zu den Veranstaltungen können Abstimmungen erstellt werden. Diese werden vom Dozenten dauerhaft oder für einen bestimmten Zeitraum aktiviert und können jederzeit deaktiviert werden. Dozenten können Abstimmungsergebnisse zurücksetzen um die gleiche Abstimmung für nachfolgende Abstimmungsgruppen zu verwenden. Über einen eindeutigen Zugangscode (im folgenden Veranstaltungspasswort genannt), den jede angelegte Veranstaltung besitzt, werden Teilnehmer zu den vom Dozenten aktivierten Abstimmungen geleitet. Es gibt vier Arten von Abstimmungen:

1. Der Teilnehmer wählt eine von mehreren Antwortmöglichkeiten als Antwort aus (Einfachauswahl).
2. Es können auch mehrere Antwortmöglichkeiten ausgewählt werden, aber mindestens eine (Mehrfachauswahl).
3. Der Teilnehmer soll die Antwortmöglichkeiten sortieren, wobei diese absteigend gewichtet werden (Rangordnung).
4. Die Teilnehmer können einen freien Text als Antwort schreiben (Freitext).

Die Ergebnisse der Abstimmungen kann ein Dozent über seinen Zugang einsehen. Die Ergebnisse der ersten drei Abstimmungstypen werden grafisch in Diagrammen dargestellt. Beim vierten Typ werden die Textantworten ausgegeben und auf Wunsch klassifiziert. Nach der Installation der Software existiert genau ein Dozent mit Administratorrechten. Somit ist es ihm möglich weitere Benutzer (Dozenten) anzulegen und vorhandene zu bearbeiten. Diese können wiederum ebenfalls Administratorrechte besitzen.

4.1.2 Nicht-funktionale Anforderungen

Die Weboberfläche soll auch auf Geräten mit Touchmonitoren gut zu bedienen sein. Dazu ist es erforderlich, dass sich die Weboberflächen automatisch an unterschiedliche Bildschirmauflösungen anpassen.

Auf Interaktionen des Benutzers mit der Oberfläche sollen unmittelbar visuelle Rückmeldungen erfolgen. Ferner sollen Ladezeiten möglichst gering ausfallen um eine zügige Durchführung und Auswertung der Abstimmungen zu ermöglichen.

4.2 Aufbau der Software

Abbildung 4.1 zeigt die grobe Funktionsweise der Software. Zu sehen sind drei Benutzergruppen. Zu beachten ist, dass der Administrator nur ein Dozent mit dem Recht zur Benutzerverwaltung ist.

Kern des Systems ist die Webanwendung, die mit Hilfe des Ruby on Rails Frameworks entwickelt wird. Sie bezieht alle notwendigen Daten, wie Veranstaltungen, Abstimmungen, abgegebene Stimmen etc. via Active Record (Model, siehe Active Record 3.1.1) aus der Datenbank . Ferner sind hier die Weboberflächen für Teilnehmer, Dozenten und Administratoren (Views) zu finden und schließlich auch die Datenverarbeitung (Controller). Des Weiteren beinhaltet sie eine REST-Schnittstelle über die aktive Abstimmungen abgerufen werden können. Die Schnittstelle ist für die Anbindung von nativen Anwendungen auf Smartphones und Tablets gedacht.

Der zweite Teil des Systems ist die Android-Applikation. Diese wird in der Grafik von einem Teilnehmer mit Smartphone dargestellt (rechts). Sie nutzt die eben erwähnte REST-Schnittstelle zur Kommunikation mit der Webanwendung.

4.2.1 Teilnehmer

An einer aktiven Abstimmung wird via Browser oder Android-Applikation teilgenommen. Für die Teilnahme ist es zunächst notwendig sich mit dem vom Dozenten herausgegebenen Passwort für die Abstimmung einzuloggen. Das Passwort ist gleichzeitig Schlüssel der Veranstaltung. Wird eine Veranstaltung mit entsprechendem Passwort in der Datenbank gefunden ist der Teilnehmer erfolgreich eingeloggt. Nach erfolgreicher Anmeldung

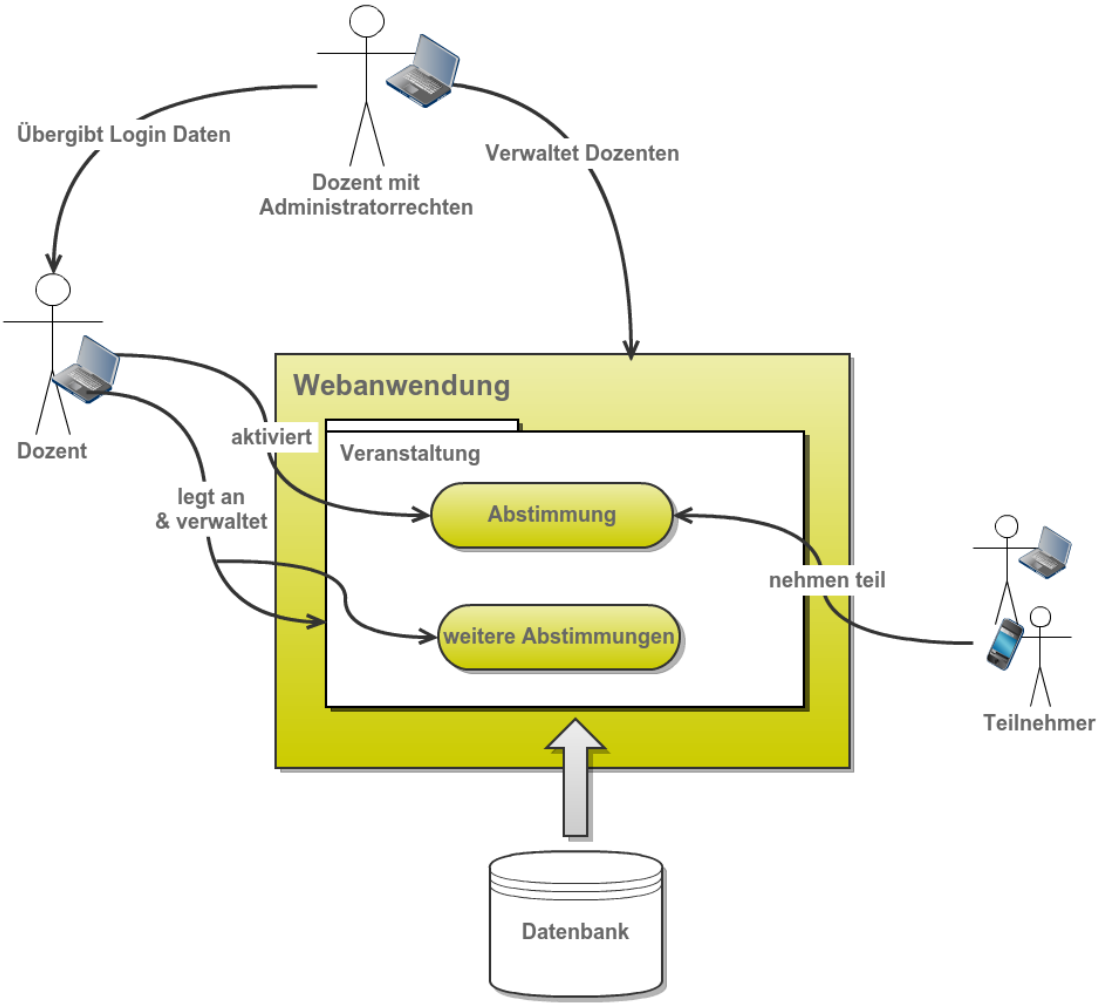


Abbildung 4.1: Funktionsweise der Software

werden nur Abstimmungen angezeigt, die aktiv sind und an denen der Teilnehmer noch nicht teilgenommen hat. Falls mehrere Abstimmungen der Veranstaltung gleichzeitig aktiv sind, wird die Abstimmung angezeigt, deren Ende der Aktivität am nähsten ist. Sind keine Abstimmungen mehr aktiv oder hat der Teilnehmer bereits an allen aktiven Abstimmungen teilgenommen, wird ein Button zur Aktualisierung angezeigt. Durch Klicken des Buttons werden erneut die aktiven Abstimmungen abgerufen.

Wird die Weboberfläche aufgerufen, werden die vom Controller verarbeiteten Daten in das entsprechende Template (View) geladen. Diese wird dann im Browser des Teilnehmers angezeigt. Erfolgt die Teilnahme über ein Android-Smartphone, werden die vom Controller verarbeiteten Daten in JSON übermittelt. Die Android-Applikation lädt empfangene Daten in entsprechende Templates, die dann auf dem Bildschirm des Teilnehmers erscheinen. Die vier unterschiedlichen Abstimmungstypen unterscheiden sich hierbei in

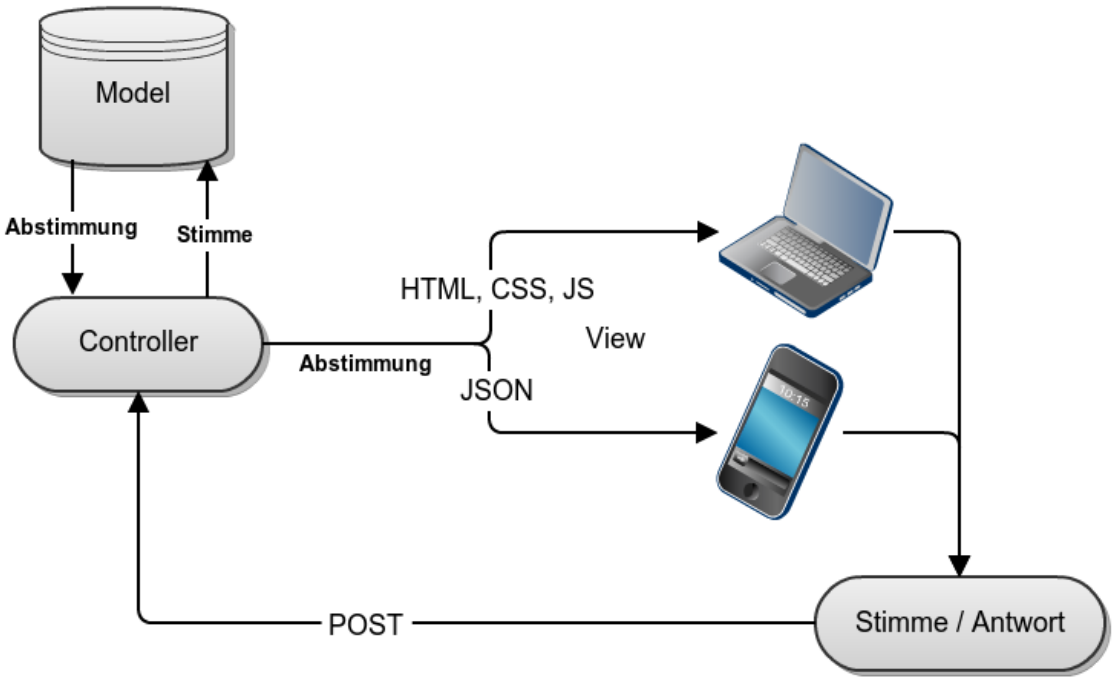


Abbildung 4.2: Ablauf der Stimmabgabe

der Darstellung:

Typ 1 Optionsboxen für die Auswahl einer Antwort

Typ 2 Checkboxen für die Auswahl mehrerer Antworten

Typ 3 Sortierbare Liste (JQuery UI Sortables/Dragsort Listview)

Typ 4 Textbox für Texteingabe

Hat der Teilnehmer seine Antwort eingegeben, kann er diese durch Klicken eines Buttons absenden, wodurch wird die Antwort übertragen wird. Der Controller verarbeitet die eingehende Antwort und prüft, ob diese gültig ist. Ist dies der Fall, so wird sie gespeichert und es werden weitere aktive Abstimmungen angezeigt, falls vorhanden. Ist die Antwort ungültig wird eine Fehlermeldung via Toast (Android) oder Flashmessage (Ruby on Rails) ausgegeben. Eine Antwort ist dann valide wenn folgende Bedingungen erfüllt sind:

1. der Benutzer ist eingeloggt
2. der Benutzer hat zuvor noch nicht teilgenommen
3. die Abstimmung existiert
4. die Abstimmung ist aktiv
5. die Abstimmung gehört zur Veranstaltung für die der Benutzer eingeloggt ist

6. es wurden Antwortmöglichkeiten gewählt/sortiert oder ein Text wurde als Antwort eingegeben

Um zu verhindern, dass Teilnehmer mehrmals an einer Abstimmung teilnehmen, wird für die Weboberfläche auf Cookies zurückgegriffen. Im Cookie werden die Primärschlüssel der Abstimmungen gespeichert, an denen bereits teilgenommen wurde. Verwendet der Teilnehmer die Android-Applikation, wird im HTTP-Post ein das Gerät identifizierender String (z.B. die International Mobile Station Equipment Identity, kurz IMEI) übermittelt und in der Datenbank mit der abgegebenen Stimme gespeichert.

Für den Fall, dass Teilnehmer noch für eine vergangene Veranstaltung angemeldet sind, jedoch bereits an einer anderen Veranstaltung mit aktiven Abstimmungen teilnehmen, muss die Möglichkeit bestehen sich abzumelden. In der Webanwendung wird zu diesem Zweck ein Abmelde-Button angezeigt. Durch Klicken des Buttons wird die Session des Teilnehmers gelöscht und die Anmeldeseite wird erneut angezeigt. Für die Android-Applikation ist dies nicht notwendig. Durch den Zurück-Button des Android-Smartphones kann der Anmeldebildschirm erneut aufgerufen werden.

4.2.2 Dozenten

Im Folgenden wird der Teil der Anwendung beschrieben, der von Dozenten bedient wird. Wie Abbildung 4.1 zeigt, werden Dozenten von Dozenten mit Administratorrechten angelegt. Hat ein Dozent seine Benutzerdaten erhalten, kann er sich mit ihnen an der Weboberfläche für Dozenten anmelden. Ist dies erfolgreich geschehen, kann er neue Veranstaltungen und Abstimmungen anlegen und bestehende verwalten. Dozenten können nur Veranstaltungen und Abstimmungen verwalten, die sie auch selbst angelegt haben. Ferner können Sie ihre eigenen Benutzerdaten ändern, wobei es ihnen nicht möglich ist, ihren Loginnamen zu ändern oder sich Administratorrechte zuzuweisen. Dozenten können sich durch Klicken eines Buttons von System abmelden.

Veranstaltungen

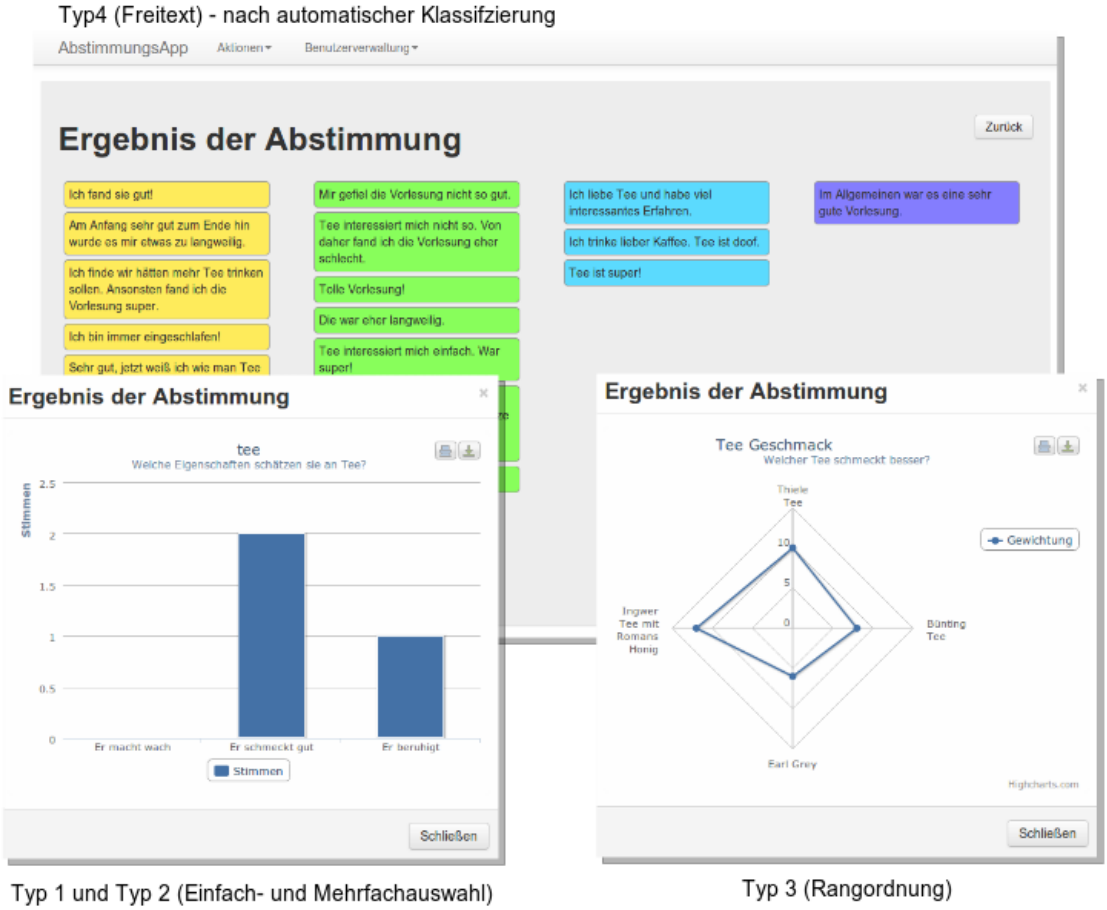
Beim Anlegen einer neuen Veranstaltung müssen ein Titel und ein Veranstaltungspasswort vergeben werden. Das Veranstaltungspasswort darf nur einmal im System vorkommen, da es als Schlüssel der Veranstaltung fungiert. So können Teilnehmer über es zur richtigen Veranstaltung geleitet werden.

Abstimmungen

Wurden Veranstaltungen angelegt können ihnen Abstimmungen hinzugefügt werden. Abstimmungen bestehen aus einem Titel, einer Fragestellung, einem Ablaufdatum bis zu dem sie aktiv sind und einem Abstimmungstyp. Liegt das Ablaufdatum in der Vergangenheit, ist die Abstimmung inaktiv und an ihr kann nicht teilgenommen werden. Das

Ablaufdatum wird nicht als Datum angegeben, sondern als Zeitraum (z.B. fünf Minuten). Beim Speichern der Abstimmung ist diese ab dem Moment des Speicherns dann für z.B. fünf Minuten aktiv. Ferner kann zu einer Abstimmung eine beliebige Anzahl an Antwortmöglichkeiten hinzugefügt werden. Antwortmöglichkeiten werden nur gespeichert wenn sie nicht leer sind. Auf diese Weise lassen sich bereits erstellte Antwortmöglichkeiten auch leicht aus der Abstimmung entfernen. Ist die Abstimmung vom Typ 4 (Textantwort), lassen sich Antwortmöglichkeiten erstellen, ansonsten werden sie jedoch ignoriert.

Auswertung der Abstimmungen



Typ 1 und Typ 2 (Einfach- und Mehrfachauswahl)

Typ 3 (Rangordnung)

Abbildung 4.3: Darstellungen der Abstimmungsergebnisse je nach Abstimmungstyp

Der Dozent kann zu jeder Zeit die Ergebnisse seiner Abstimmungen einsehen. Je nach Abstimmungstyp werden die Ergebnisse auf drei Arten dargestellt. Ergebnisse von Abstimmungen des ersten oder zweiten Typs (Einfachauswahl und Mehrfachauswahl) werden

in einem Balkendiagramm dargestellt. Ergebnisse von Abstimmungen des dritten Typs (Sortierung) werden in einem Kiviatdiagramm dargestellt. Die Ergebnisse der Abstimmungen des vierten Typs (Textantwort) können nicht ohne weiteres in einem Diagramm ausgewertet werden.

Statt einer Darstellung im Diagramm werden beim vierten Abstimmungstyp alle abgegebenen Antworten ausgegeben. Diese sind aus Gründen der Übersicht in drei Spalten aufgeteilt. Es besteht nun die Möglichkeit die Antworten automatisch klassifizieren zu lassen. Somit ist eine inhaltliche Trennung der Antworten in bis zu fünf Klassen möglich, ohne dass vorher alle abgegebenen Antworten gelesen werden müssen. Um dies zu erreichen wird das NBayes Gem (siehe nbayes 3.2.2), eingesetzt. Jede Antwort befindet sich in einer eigenen Box. Durch (mehrmaliges) Anklicken der Box lässt sich die Antwort einer von fünf Klassen zuordnen. So wird eine Vorauswahl von Antworten klassifiziert. Anschließend können die übrigen Antworten automatisch durch die Webanwendung klassifiziert werden. Hierzu werden die Primärschlüssel der durch den Dozenten klassifizierten Antworten an den entsprechenden Controller der Webanwendung übermittelt. Dieser "füttert"den Algorithmus mit den zuvor ausgewählten Antworten und lässt die übrigen Antworten durch den Algorithmus klassifizieren. Die nun vollständig klassifizierten Antworten werden an die View übergeben. Hier werden sie, je nach Anzahl der verwendeten Klassen, in bis zu fünf Spalten dargestellt.

4.2.3 Dozenten mit Administratorrechten

Dozenten mit Administratorrechten sind, wie der Name vermuten lässt, Dozenten mit erweiterten Rechten. Im Vergleich zu normalen Dozenten sind sie befugt sich alle im System registrierten Dozenten in einer Übersicht auflisten zu lassen. Außerdem können sie regstrierte Dozenten auch über Titel, Nachname, Vorname und Login suchen. Des Weiteren sind sie berechtigt Dozenten und Dozenten mit Administratorrechten anzulegen, zu bearbeiten und zu löschen. Wie Abbildung 4.1 zeigt, sind sie für die Übergabe der Benutzerdaten verantwortlich. Ob ein Dozent Administratorrechte besitzt, wird über einen boolschen Wert im Datenmodell (siehe Model 5.1.1) festgelegt. Ist dieser Wert wahr, wird er bei erfolgreicher Anmeldung am System in der Session des Dozenten mit Administratorrechten gespeichert. Bei allen Aktionen die einem Dozenten mit Administratorrechten vorbehalten sind, wird zuerst geprüft ob der Wert in der Session vorhanden und wahr ist. Um den Dozenten mit Administratorrechten vom System abzumelden wird seine Session gelöscht.

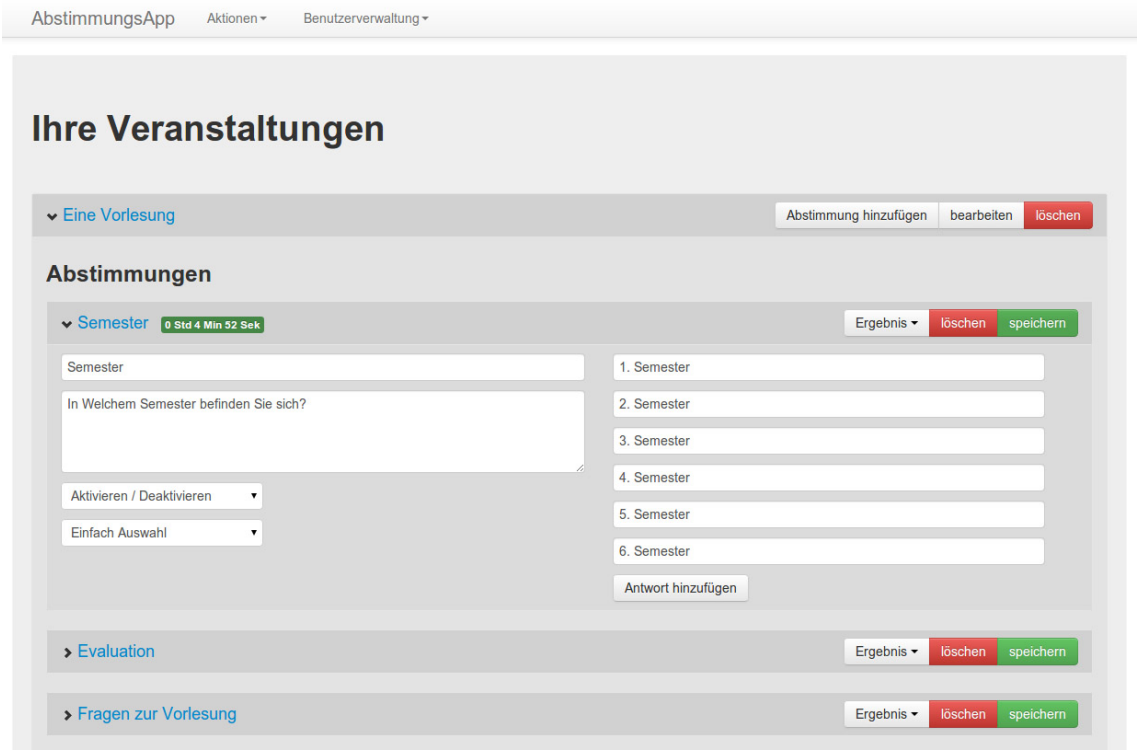


Abbildung 4.4: Weboberfläche für Dozenten (mit Administratorrechten)

4.2.4 Struktur der Android-Applikation

Abbildung 4.5 zeigt den Ablauf der Teilnahme bei Benutzung der Android Applikation. Zuerst wird das Veranstaltungspasswort eingeben. Durch Klicken des Anmeldebuttons wird angefragt, ob eine Veranstaltung mit dem eingegebenen Veranstaltungspasswort existiert. Ist dies der Fall, wird die aktuellste aktive Abstimmung, an der noch nicht teilgenommen wurde, angezeigt. Durch einen Klick auf den Abstimmen-Button, wird die Antwort abgeschickt. War die Teilnahme erfolgreich, wird erneut geschaut ob für die Veranstaltung eine weitere aktive Abstimmung existiert, an der der Teilnehmer nicht bereits teilgenommen hat. Falls eine weitere aktive Abstimmung vorhanden ist, wird sie abgerufen und dargestellt. Falls keine weiteren Abstimmungen mehr vorhanden sind, so erscheint die dritte Oberfläche (in Abbildung 4.5 ganz rechts). Sie zeigt an, dass keine weiteren aktiven Abstimmungen vorliegen. Über einen Button kann erneut nach aktiven Abstimmungen gesucht werden.

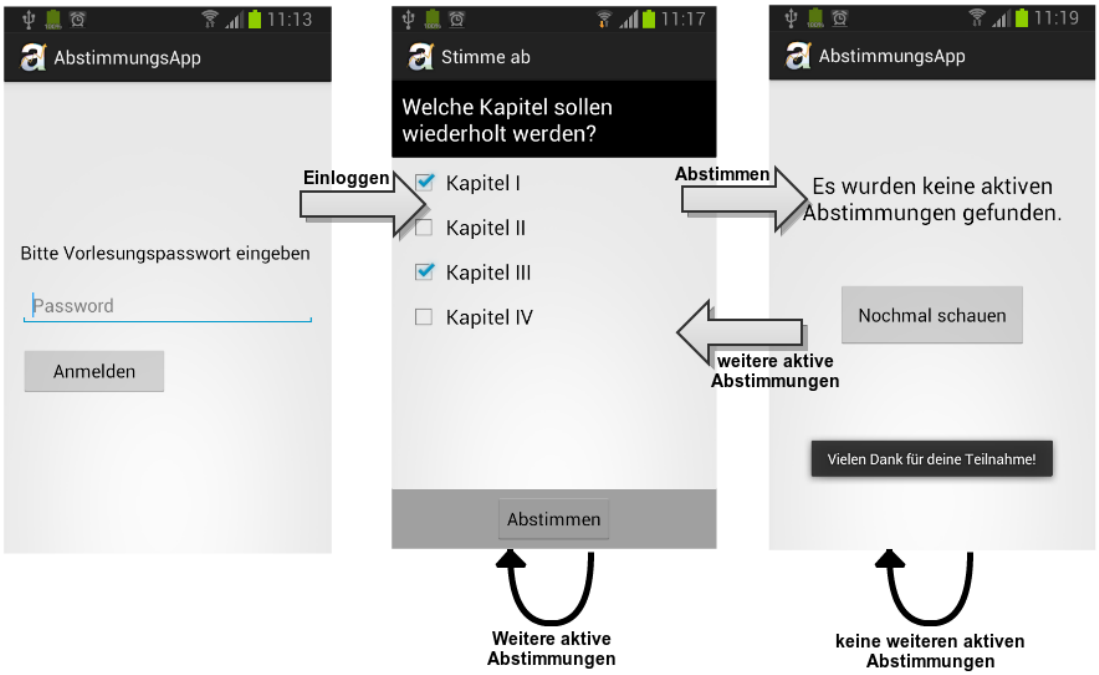


Abbildung 4.5: Oberfläche und Ablauf der Android-Applikation

5 Implementierung

5.1 Webservice

5.1.1 Model

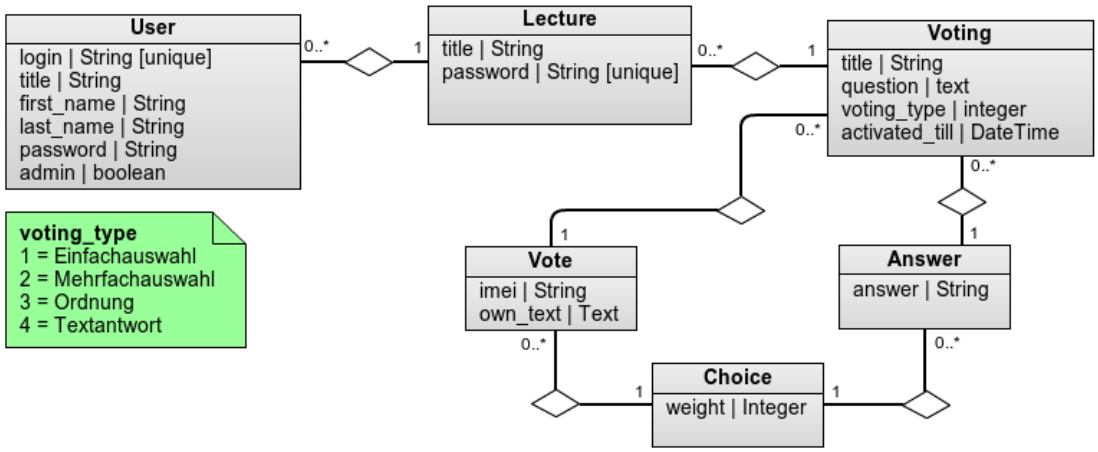


Abbildung 5.1: Entity-Relationship-Modell

User

Die User-Entität repräsentiert die Dozenten und Dozenten mit Administratorrechten. Zu jedem Benutzer (`User`) gehört ein eindeutiger Login-Name und ein Passwort, mit denen er sich am System anmelden kann. Die Verschlüsselung der Passwörter und das Speichern der dazugehörigen Salt-Werte werden vom Gem bcrypt-ruby übernommen. Der boolsche-Wert `admin` legt fest, ob der Benutzer Administratorrechte besitzt oder nicht. Des Weiteren gehören zu einem Benutzer Titel, Vorname und Nachname, wobei der Titel optional ist. Jeder Benutzer kann in Relation zu beliebig vielen Veranstaltungen (`Lectures`) stehen.

Lecture

Die `Lectures` im Model sind Veranstaltungen in denen die Abstimmungen organisiert werden. Jede Veranstaltungen hat genau eine Relation zu einem Benutzer (`User`). Im `title`-Attribut einer Veranstaltung wird der Veranstaltungstitel gespeichert. Jede Veranstaltung kann mit beliebig vielen Abstimmungen (`Votings`) in Relation stehen. Das `password`-Attribut ist eindeutig, da über das Veranstaltungspasswort die Zuordnung der Teilnehmer zu den aktiven Abstimmungen einer Veranstaltung erfolgt.

Voting

Die Votings im Model sind unsere Abstimmungen. Jede Abstimmung ist genau einer Veranstaltung (`Lecture`) zugeordnet und über diese auch genau einem Benutzer (`User`). Der Titel eines Votings beschreibt kurz den Inhalt der Abstimmung. Er wird nur für eine bessere Ordnung der Übersichtsseiten in der Weboberfläche für Dozenten benötigt, da die Fragestellungen (`question`-Attribut) sehr lang sein dürfen. Das `voting_type`-Attribut ist vom Typ Integer und legt den Abstimmungstyp fest. Alle Abstimmungstypen sind in der grünen Notiz in Abbildung 5.1 zu sehen. Das `activated_till`-Attribut ist vom Typ DateTime. Es speichert ein Datum und eine Uhrzeit. Liegen Datum und Uhrzeit in der Zukunft, ist die Abstimmung noch bis zum gespeicherten Zeitpunkt aktiv. Somit ist es Teilnehmern, die das Passwort der Veranstaltung kennen, zu dem die Abstimmung gehört, möglich teilzunehmen. Liegen Datum und Zeit in der Vergangenheit, ist die Abstimmung inaktiv und somit für Teilnehmer nicht erreichbar. Zu jeder Abstimmung gehören beliebig viele Stimmen (`Votes`) und beliebig viele Antwortmöglichkeiten (`Answers`).

Answer

Eine `Answer` ist eine Antwortmöglichkeit. Sie ist genau einer Abstimmung zugeordnet und beinhaltet nur den Text der Antwortmöglichkeit (`answer`-Attribut). Über die `Choice`-Entität steht jede Antwortmöglichkeit mit beliebig vielen Stimmen (`Votes`) in Relation.

Vote

Die `Vote`-Entität repräsentiert die abgegebenen Stimmen der Teilnehmer. Sie beinhaltet die Attribute `imei` und `own_text`. Die IMEI wird nur gespeichert, falls der Teilnehmer über ein Smartphone oder Tablet abgestimmt hat. Über sie wird der Teilnehmer identifiziert um zu verhindern, dass er ein weiteres Mal an der Abstimmung teilnimmt. Das `own_text`-Attribut wird nur für den vierten Abstimmungstyp benötigt. Es bietet Platz für den freien Text, den ein Teilnehmer beim vierten Abstimmungstyp anstelle einer Antwortmöglichkeit eingibt. Jede Stimme (`Vote`) steht über die `Choice`-Entität mit beliebig vielen Antwortmöglichkeiten (`Answers`) in Relation. Diese Relationen verknüpfen die Stimme mit den vom Teilnehmer gewählten Antwortmöglichkeiten.

Choice

Abbildung 5.2: Gewichtung der Antwortmöglichkeiten

Die Choice-Entität ist als Attribut der Relation zwischen Vote- und Answer-Entität zu sehen und wird ausschließlich für den dritten Abstimmungstyp (Ordnung) benötigt. Das `weight`-Attribut speichert die Gewichtung der Antwortmöglchkeiten. Sei n die Anzahl der Antwortmöglichkeiten. So ist die Antwortmöglichkeit, die der Teilnehmer an die oberste Postition geschoben hat (Abbildung 5.2), mit $n-1$ gewichtet, die Antwortmöglichkeit an zweiter Stelle mit $n-2$ und die letzte Postion wird mit 0 gewichtet.

Ruby-Umsetzung des Models

Das ER-Modell aus Abbildung 5.1 wurde in Rails mit Hilfe des Active Record-Persistenz-Frameworks umgesetzt. Hierzu wurde für jede Entität aus dem Model eine entsprechende Klasse angelegt. Als Datenbank wird eine SQLite3-Datenbank verwendet. Aufgrund der Verwendung von Active Record lässt sich diese jedoch schnell durch eine andere, gängige relationale Datenbanksoftware austauschen. Die Entitäten des Models verteilen sich auf folgende Dateien:

1. `user.rb`
2. `lecture.rb`
3. `voting.rb`

4. `answer.rb`
5. `vote.rb`
6. `choice.rb`

In diesen Dateien werden die möglichen Relationen zwischen den Entitäten festgelegt. Ferner werden die Attribute festgelegt auf die von außen zugegriffen werden darf. Für jedes Attribut lassen sich auch Bedingungen festlegen (sog. Validierungen), die erfüllt sein müssen, damit ein Objekt in die Datenbank übernommen wird. Sind eine oder mehrere Bedinungen nicht erfüllt, wird eine Fehlermeldung erzeugt, die dem Benutzer angezeigt werden kann. Jede Klasse des Modells erbt von der Klasse ActiveRecord::Base. Mit Hilfe der geerbten Methoden erfolgt der Zugriff auf die Datenbank. Die folgenden Auszüge aus dem Ruby-Code des Modells zeigen wie die Relationen und Validierungen umgesetzt wurden.

Listing 5.1: app/models/vote.rb (Ausschnitt)

```
1  class Vote < ActiveRecord::Base
2    attr_accessible :imei, :own_text
3    belongs_to :voting
4    has_many :choices, :dependent => :destroy
5    has_many :answers, :through => :choices
```

Mit Hilfe von `attr_accessible` werden die Attribute des Modells beschrieben, auf die von außen zugegriffen werden darf. Da es sich im Beispiel um die Stimmen der Teilnehmer handelt, muss auf die Attribute `imei`, und `own_text` zugegriffen werden.

Anschließend werden die möglichen Relationen zu anderen Entitäten beschrieben. Die Zeile `belongs_to :voting` legt fest, dass jede Stimme genau zu einer Abstimmung gehört. In Zeile 4 und 5 wird festgelegt, dass eine Stimme mit beliebig vielen Antwortmöglichkeiten verknüpft sein kann. Die Relation hat das Attribut `weight` zur Gewichtung der gewählten Antwortmöglichkeit. Das Attribut der Relation wurde, wie auch im ER-Modell dargestellt, über eine weitere Entität (Choices) realisiert. Mit dem Befehl `:through => :choices` wird festgelegt, dass die Relation zwischen Stimme und Antwort über die Choices-Entität erfolgt. Der Befehl `dependent: :destroy` legt fest dass, falls eine Stimme gelöscht wird, auch alle mit dieser Stimme in Relation stehenden Choices aus der Datenbank entfernt werden. Dies ist notwendig, da diese Daten ohne eine zugehörige abgegeben Stimme nutzlos sind.

Listing 5.2: app/models/user.rb (Ausschnitt)

```
1  class User < ActiveRecord::Base
2    has_secure_password
3    attr_accessible :admin, :firstname, :lastname, :login,
4                    :password, :password_confirmation, :title
5    has_many  :lectures, :dependent => :destroy
6    validates :login, :uniqueness => true
7    validates :firstname, :lastname, :presence => true
8    validates :password, :confirmation => true,
```

```
9               :length => { :within => 5..40 }
```

Listing 5.2 zeigt in Zeile 2 wie mit `has_secure_password` die vom bcrypt-Gem bereitgestellten Methoden in die User-Klasse eingebunden werden. Dazu zählen Methoden zum Speichern und Validieren des Passwortes und zur Authentifizierung von Benutzern. Die Befehle in Zeile 3 bis 5 wurden bereits erläutert. Anschließend folgen die Validierungsbedingungen für die Attribute der User-Entität. Zuerst wird festgelegt, dass der Login-Name eines Dozenten nur einmal in der User-Tabelle vorkommen darf. Zeile 4 erreicht, dass Vor- und Nachname eines Dozenten angegeben werden müssen. Danach wird bestimmt, dass zur Bestätigung eines Passwortes die Attribute `password` und `password_confirmation` den gleichen Inhalt haben müssen. Hierbei ist `password_confirmation` nur ein Hilfsattribut für die Validierung und wird nicht in der Datenbank gespeichert. Außerdem muss ein Passwort mindestens aus fünf Zeichen bestehen und darf maximal aus 40 Zeichen bestehen um gültig zu sein.

Methoden des Modells

Um den Controllern einen einfachen Zugriff auf die Daten aus der Datenbank zu ermöglichen, wurden die Klassen der Modelle noch um einige Methoden erweitert. Auf diese Weise werden komplexere Datenbankanfragen in die Modelle ausgelagert. Anhand eines Beispiels wird im Folgenden der Aufbau einer solchen Anfrage erläutert. Für alle in den Modellen zum Einsatz kommenden Anfragen werden die vom Active Record-Framework bereitgestellten Methoden eingesetzt. Dadurch werden Sicherheitslücken in den Anfragen verhindert, die z.B. SQL-Injections ermöglichen.

Listing 5.3: app/models/voting.rb (Ausschnitt)

```
1 def self.active_by_id_and_password ( id, password,
2   participated=[0] )
3
4   return Voting.joins(:lecture).includes(:answers).where(
5     "votings.id = ? AND activated_till > ? AND
6     lectures.password = ? AND votings.id not in (?)",
7     id, DateTime.now, password, participated ).first
8
9 end
```

Die in Listing 5.3 dargestellte Methode gibt eine einzelne Abstimmung inklusive aller zugehöriger Antwortmöglichkeiten zurück. Voraussetzung ist, dass die Abstimmung aktiv ist und zu der Veranstaltung mit dem der Methode übergebenen Veranstaltungspasswort gehört. Ferner muss die ID der Abstimmung gleich der übergebenen ID sein. Außerdem kann ein Array an Abstimmungs-IDs übergeben werden, an denen bereits teilgenommen wurde.

Die Methode wird verwendet um zu gewährleisten, dass Teilnehmer nur Abstimmungen aufrufen können, die zu der Veranstaltung gehören für die sie das Veranstaltungspasswort

besitzen, die aktiv sind und an denen sie noch nicht teilgenommen haben. Um das zu erreichen, werden mit dem Befehl `joins(:lecture)` die Abstimmungen zusammen mit den zugehörigen Veranstaltungen ermittelt. Dies ermöglicht die Abfrage des Veranstaltungspassworts in der `where`-Klausel. Die zu der Abstimmung zugehörigen Antwortmöglichkeiten werden mit Hilfe des `includes(:answers)`-Ausdrucks dem Ergebnis als Array von Objekten hinzugefügt. Der `join`-Befehl hingegen fügt dem Ergebnis nichts hinzu, sondern ermöglicht nur den Zugriff auf die Attribute der gejointen Tabelle. Durch den abschließenden first-Befehl wird nur die erste gefundene Abstimmung zurückgegeben.

5.1.2 View

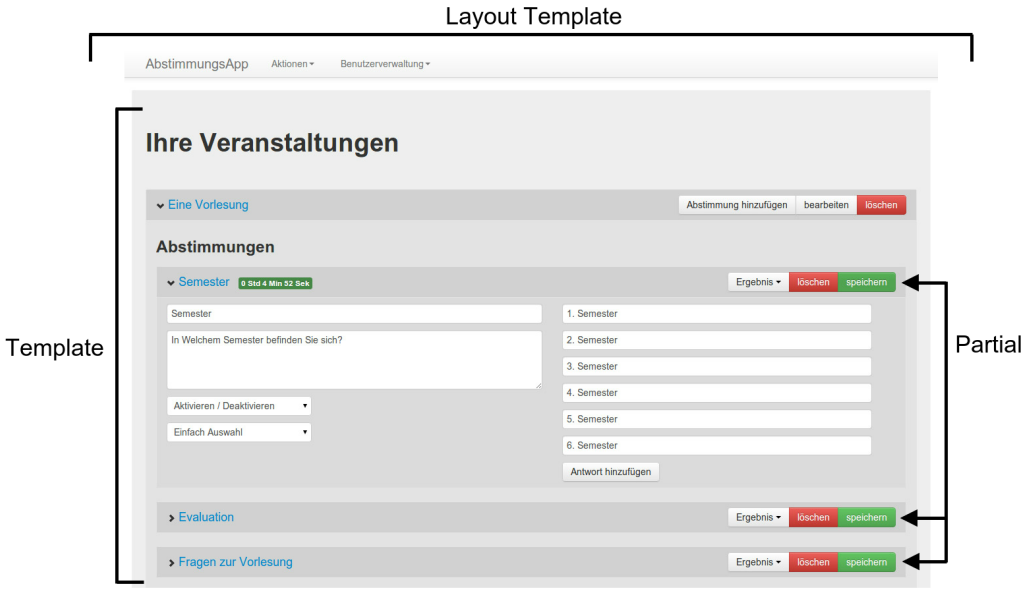


Abbildung 5.3: Bestandteile der View

Wie in Abbildung 5.3 zu sehen ist, setzen sich die Weboberflächen aus verschiedenen Teilen zusammen (siehe Action View 3.1.2).

Für die Abstimmungssoftware wurden zwei Layouts verwendet. Ein Layout für die Weboberfläche der Teilnehmer und eines für die Weboberfläche der Dozenten und Dozenten mit Administratorrechten. Die Layouts beinhalten jeweils die Navigationsleiste und einen Container für den Inhalt in Form einer `div`-Ebene. Innerhalb dieser div-Ebene befindet sich der `yield`-Befehl.

Die Templates beeinhalten die eigentlichen Oberflächen. Somit existiert für jede Oberfläche (z.B. die Veranstaltungsübersicht) ein Template. Dynamische Inhalte werden durch eingebetteten Ruby-Code geladen. Sich wiederholende oder in mehreren Templates benötigte Teile der Oberfläche werden in Partials ausgelagert. Diese Partials werden durch den `render`-Befehl in die Templates geladen. Übergibt man dabei eine Collection, also

eine Sammlung von Instanzen einer Klasse, wird automatisch über die Instanzen der Sammlung iteriert. Für jede Instanz der Sammlung wird das Partial dabei einmal gerendert.

Um die nicht-funktionalen Anforderungen zu erfüllen, wird für die Oberflächen viel auf Javascript und AJAX zurückgegriffen. Beim Anlegen, Bearbeiten und Löschen von Abstimmungen werden nur die erforderlichen Bereiche der Oberfläche aktualisiert. Um die Änderungen für den Benutzer verständlicher zu machen werden die visuellen Transitionen von JQuery UI (Überblendungen, Aufleuchten etc., siehe JQuery UI 3.3.3) eingesetzt.

Beispiel

Listing 5.4: app/views/lectures/index.html.erb (Ausschnitt)

```
1  <h1>Ihre Veranstaltungen</h1>
2  <% @lectures.each do |lecture| %>
3  <div id="lecture_<%= lecture.id %>">
4    <div class="lecture_title"><%= lecture.title %></div>
5    <div class="btn-group">
6      <%= link_to 'Abstimmung hinzufuegen', new_voting_path() %>
7      <%= link_to 'bearbeiten', edit_lecture_path(lecture) %>
8      <%= link_to 'loeschen', lecture %>
9    </div>
10   <div id="lecture_body_<%= lecture.id %>">
11     <h3>Abstimmungen</h3>
12     <div id="lecture_<%= lecture.id %>_votings">
13       <%= render :partial => 'votings/voting',
14         :collection => lecture.votings %>
15     </div>
16   </div>
17 </div>
18 <% end %>
```

Das Listing 5.4 zeigt einen aus Gründen der Leserlichkeit vereinfachten Auszug aus dem Template für die Abstimmungübersicht der Dozenten. Die Abstimmungsübersicht ist in Abbildung 5.3 zu sehen. Jede Veranstaltung besitzt eine Kopfzeile. Diese enthält den Veranstaltungstitel und drei Buttons zum Verwalten der Veranstaltung (Abstimmung hinzufügen, Veranstaltung löschen und bearbeiten). Im Rumpf der Veranstaltung werden die dazugehörigen Abstimmungen angezeigt, die auch jeweils aus Kopfzeile und Rumpf bestehen.

Die mit `%`-Zeichen beginnenden Tags sind Ruby-Code, alle anderen Tags sind Html-Code. In Zeile 2 wird über die in der Instanzvariablen `@lecture` (siehe Variablen 3.2.1) enthaltenen Instanzen von Veranstaltungen iteriert. Die Variable `@lectures` wird in der `index`-Methode des LecturesController belegt (siehe Abschnitt 5.1.3, LecturesController). Nun steht in Zeile 2 bis 18 die lokale Variable `lecture` zur Verfügung. Diese enthält jeweils die Instanz einer Veranstaltung und die zur Veranstaltung in Relation stehenden

Antwortmöglichkeiten. In Zeile 3 bis 9 werden Veranstaltungstitel und die drei Buttons zum Verwalten der Veranstaltung in die Kopfzeile der Veranstaltung eingesetzt. Dann folgt der Rumpf, der alle zugehörigen Abstimmungen enthält. Das Html-Gerüst einer Abstimmung liegt in einem Partial. Dieses wird durch den Befehl:

```
render :partial => 'votings/voting', :collection => lecture.votings}
```

in den Zeilen 13-14 gerendert. Die der Veranstaltung zugehörigen Abstimmungen werden als Collection übergeben.

5.1.3 Controller

In den Unterkapiteln 4.2.1 Teilnehmer, 4.2.2 Dozenten und 4.2.3 Dozenten mit Administratorrechten wurden die benötigen Funktionalitäten aller Bereiche des Systems erläutert. Aus ihnen lässt sich folgendes Klassendiagramm für einen Aufbau der Controller der Rails-Anwendung ableiten. Die Controller erben hierbei vom ApplicationController, der Methoden enthalten kann, die von jedem Controller benötigt werden. Dieser wiederum erbt vom ActionController (siehe Action Controller 3.1.3), der Teil des Ruby on Rails-Frameworks ist und häufig benötigte Funktionalitäten eines Controllers implementiert.

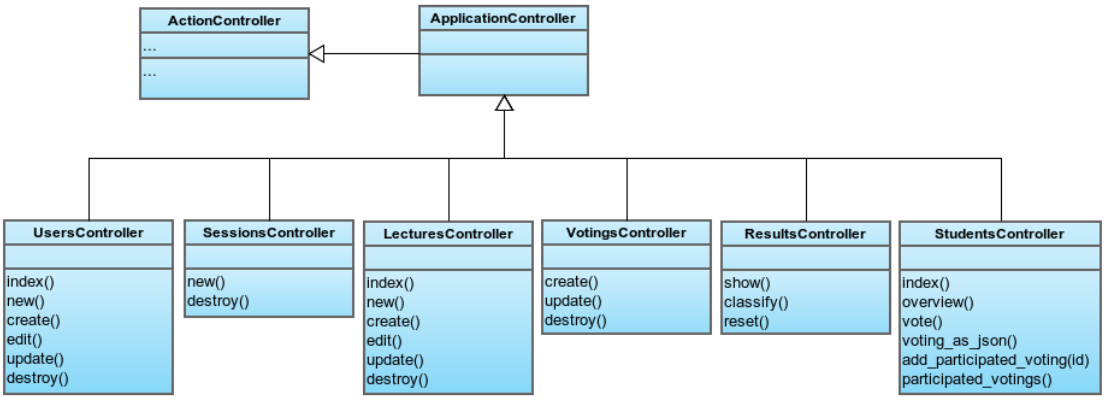


Abbildung 5.4: Die Controller der Webanwendung

Die Methoden `index`, `new`, `create`, `edit`, `update` und `destroy` sind in mehreren Controllern zu finden. Die `index`-Methode lädt die Daten, die der Controller behandelt und übergibt sie an eine View, um so eine Übersicht über diese Daten zu rendern (z.B. Übersicht aller Dozenten). Die Methode erstellt ein neues Objekt (z.B. eine Instanz eines neu anzulegenden Dozenten) und rendert dazu ein leeres Formular, welches der User mit Daten füllt. Nach Abschicken dieses Formulares werden die eingebenen Daten von der `create`-Methode in eine leere Instanz der User-Modellklasse übertragen und gespeichert. Ähnlich verhält es sich mit der `edit`- und `update`-Methode, nur dass diese die Instanz eines bestehenden Dozenten aus der Datenbank laden. Das durch die `edit`-Methode gerenderte Formular enthält die geladenen Daten, die nun geändert werden können. Die geänderten Daten werden wiederum von der `update`-Methode in ein dem zu ändernden

User entsprechendes Objekt übertragen und gespeichert. Die `destroy`-Methode löscht einen Datensatz aus der Datenbank unter Verwendung des Persistenz-Frameworks Active Record (siehe Active Record 3.1.1).

Im folgenden wird genauer auf die Implementierung der wichtigsten Controller eingegangen.

StudentsController

Der `StudentsController` beeinhaltet Methoden zur Darstellung der Abstimmungen und zum Empfangen der Teilnehmerantworten. Er sorgt auch dafür, das jeder Teilnehmer nur einmal an einer Abstimmung teilnehmen kann. Des Weiteren beinhaltet er die REST-Schnittstelle zur Anbindung der Android-Applikation. Die `index`-Methode rendert das Login-Formular, in welches das Veranstaltungspasswort eingegeben wird. Ist der Teilnehmer bereits angemeldet, wird die overview-Methode auferufen. Sie prüft nochmals ob der Nutzer bereits angemeldet ist. Ist dies der Fall, werden aktive Abstimmungen der Veranstaltung in die Instanzvariable `@voting` geladen. Hierzu wird die in Listing 5.3 vorgestellte Methode des Voting-Modells aufgerufen.

Anschließend wird geprüft ob es sich um eine AJAX-Anfrage handelt (siehe AJAX / XML-HttpRequest 3.3.2). Ist dies der Fall, wird ein Javascript-Template gerendert. Das gerenderte Javascript wird vom Teilnehmer-Browser ausgeführt und ersetzt einen Teil der Oberfläche des Teilnehmers durch die geladene Abstimmung. Der Html-Code der Abstimmung, wird direkt in das Javascipt-Template durch den Befehl `<%= j render :partial => "voting"%>` eingefügt. Das "j"vor dem render-Kommando ist ein Alias für die `escape_javascript`-Methode. Sie sorgt dafür, dass Zeichen im Html-Code wie Hochkommata etc. maskiert werden, damit ein gültiges Javascript entsteht.

Handelt es sich nicht um eine AJAX-Anfrage, wird folgendes Html-Template gerendert:

Listing 5.5: app/views/students/overview.html.erb (Ausschnitt)

```
1 <% if @voting.nil? %>
2   <%= render :partial => 'refresh' %>
3 <% else %>
4   <%= render :partial => 'voting' %>
5 <% end %>
```

Es enthält (wie auch das Javascript-Template) eine Fallunterscheidung. Ist die Instanzvariable `@voting` Null, wird ein Partial gerendert, welches Html-Code für einen aktualisieren Button enthält. Ist `@voting` nicht Null, wird das Partial gerendert, welches den Html-Code zur Darstellung einer Abstimmung beeinhaltet.

Die Unterscheidung der Anfragen im Controller ermöglicht auch Teilnehmern mit Browsern ohne Javascript-Unterstützung die Teilnahme an Abstimmungen. Ausgenommen ist hier der dritte Abstimmungstyp (siehe 4.2.1 Abstimmungstypen).

Die Methoden `add_participated_voting` und `participated_votings` werden zur Ver-

waltung der Cookies genutzt. Erstere legt ein Cookie an und speichert darin die der Methode übergebene ID einer Abstimmung. Ist das Cookie bereits vorhanden, wird die Abstimmungs-ID an den Cookie-Wert durch Kommata getrennt angehängt.

Die Methode `participated_votings` liest das Cookie aus und gibt ein Array von Abstimmungs-IDs zurück, an denen der Teilnehmer bereits teilgenommen hat. Dieses Array kann direkt der in Listing 5.3 vorgestellten Methode übergeben werden.

Den größten Teil des StudentsControllers nimmt die `vote`-Methode ein. Sie besteht aus einer Fallunterscheidung, die die eingehenden Antworten je nach Abstimmungstyp validiert und speichert. Das folgende Listing zeigt den Code-Auschnitt, der das Validieren und Speichern des dritten Abstimmungstyps übernimmt:

Listing 5.6: app/controller/students.rb (Ausschnitt)

```
1  voting = Voting.active_by_id_and_password( params[:voting_id],
2    session[:loggedin] )
3  #...
4  vote = Vote.new
5  vote.imei = params[:imei]
6  voting.votes << vote
7
8  if vote.save
9    answers = Answer.find( params[:choice] )
10   case voting.voting_type
11   #...
12   when 3
13     if params[:choice].instance_of? Array
14
15       vote.answers << answers
16       choices = vote.choices
17
18       params[:choice].reverse.each_with_index do |answer_id, index|
19         choice = choices.find_by_answer_id(answer_id)
20         choice.weight = index
21         choice.save
22       end
23     else
24       error = "Fehler bei der Auswertung der Antwort!"
25     end
26   end
27 end
```

Zuerst wird mit Hilfe des in der Session gespeicherten Passwortes und der im POST enthaltenen Abstimmungs-ID die Abstimmung, an der teilgenommen wurde, geladen. Anschließend wird eine neue Instanz einer Stimme erzeugt und mit der Abstimmung verknüpft (Zeilen 4 bis 6). Wird die Stimme erfolgreich in der Datenbank gespeichert, werden die vom Teilnehmer gewählten Antwortmöglichkeiten aus der Datenbank geladen (Zeile 9). Im Falle des dritten Abstimmungstyps sind das alle mit der Abstimmung

in Relation stehenden Antwortmöglichkeiten. Das `params[:choice]`-Array enthält die IDs aller Antwortmöglichkeiten der Abstimmung in der Reihenfolge, wie sie vom Teilnehmer sortiert wurden. Daher werden die Antwortmöglichkeiten in der gleichen Reihenfolge geladen. Danach findet die eigentliche Fallunterscheidung statt. Im Falle des dritten Abstimmungstyps wird geprüft, ob die im POST übermittelte Antwort auch zum Abstimmungstyp passt. Ist dies der Fall, werden zuerst alle Antwortmöglichkeiten mit der Stimme in Relation gebracht (Zeile 15). Da diese Relation über eine Ausprägung der `choices`-Entität funktionieren, wird pro Relation automatisch eine Ausprägung angelegt. In der folgenden Zeile werden nun die angelegten Ausprägungen als Collection geladen. In Zeilen 18 bis 23 wird die Gewichtung gespeichert. Damit die Gewichtung stimmt, wird das `params[:choice]`-Array umgedreht (Zeile 18). In Zeile 19 wird nun eine choice-Ausprägung aus der choices-Collection aufgerufen. Bei der ersten Iteration ist es die Ausprägung, die mit der Antwortmöglichkeit in Relation steht, die der Teilnehmer auf die am geringsten gewichtetste Position geschoben hat. Bei der zweiten Iteration ist es die am zweitgeringsten gewichtetste (usw.). Die folgende Zeile legt das Gewicht der Ausprägung fest. Hierzu wird der Index im `params[:choice]`-Array verwendet. Daher wurde dieses auch umgedreht. Abschließend wird die Gewichtung gespeichert.

Falls die im POST übermittelten Daten nicht zum Abstimmungstyp passen, wird in der `error`-Variable ein Fehler-String gespeichert.

REST-Schnittstelle

Neben den gerade erläuterten Methoden enthält der StudentsController auch die REST-Schnittstelle zur Anbindung der Android-Applikation. In Ruby on Rails kann über die `render`-Methode ein Objekt direkt in JSON konvertiert werden. Die `voting_as_json`-Methode wird über eine eigene Route per HTTP-GET aufgerufen. In der URL werden das Veranstaltungspasswort und die IMEI verschlüsselt übergeben. Mit Hilfe dieser Parameter wird in der `voting_as_json`-Methode eine aktive Abstimmung, an der die übergebene IMEI noch nicht beteiligt war, aus der Datenbank geladen. Die geladenen Daten werden über die `render`-Methode im JSON-Format ausgegeben. Wird keine Abstimmung gefunden, wird ein entsprechender Status im JSON-Format zurückgegeben.

ResultsController

Der ResultsController fasst alle notwendigen Daten für die Darstellung der Ergebnisse zusammen und rendert je nach Abstimmungstyp die entsprechende View. Ferner beinhaltet er eine Methode zum Zurücksetzen einer Abstimmung und eine zur Klassifizierung von Textantworten (vierter Abstimmungstyp, siehe 4.2.1).

Für die Darstellung der Ergebnisse ist die `show`-Methode zuständig. Ihr wird über die URL die ID einer Abstimmung übergeben, die zu Beginn aus der Datenbank geladen wird. Eine Fallunterscheidung wie auch im Studentscontroller lädt je nach Abstimmungstyp das Ergebnis aus der Datenbank und das zum Abstimmungstyp gehörende Template für die

Darstellung des Ergebnisses (4.3 Darstellungen der Abstimmungsergebnisse je nach Abstimmungstyp). Da für die ersten drei Abstimmungstypen auf eine grafische Darstellung des Ergebnisses zurückgegriffen wird, wird ein Javascript-Template gerendert. Die Daten der Abstimmung werden als Javascript-Array eingebunden. Hierzu werden im Controller die Daten auf zwei Arrays aufgeteilt. Ein Array enthält die Antwortmöglichkeiten, das andere enthält die Stimmen bzw. Gewichtungen je Antwortmöglichkeit. Das Javascript-Template enthält Javascript um ein Highchart-Diagramm zu erzeugen und in eine div-Ebene zu laden. Die Antwortmöglichkeiten und Stimmen werden mit dem Befehl `<%= raw( @answers.to_s ) %>` und `<%= raw( @votes.to_s ) %>` in Javascript konvertiert und dann im Browser des Dozenten von der Highcharts-Bibliothek weiter verarbeitet. Für den vierten Abstimmungstyp wird ein Html-Template gerendert. Jede Antwort wird in einem eigenen Kasten dargestellt. In das Template wird Javascript eingebunden. Dieses ermöglicht es, bestimmte Antworten einer von fünf Klassen zuzuordnen. Klickt der Benutzer nun auf den Klassifizieren-Button, liest das Javascript die vergebenen Klassen aus und schickt die IDs der klassifizierten Klassen per HTTP-POST an die `classify`-Methode des Result-Controllers.

In der `classify`-Methode des ResultsControllers werden nun die klassifizierten Textantworten geladen. Anschließend wird mit dem Befehl `nbayes = NBayes::Base.new` eine Instanz des NBayes-Classifiers erzeugt. Die klassifizierten Textantworten werden dem Classifier mit einem Aufruf von:

```
nbayes.train( "Text...".split(/\textbackslash s+/), "class\_i" )
```

($i = 1..5$) übergeben. Wurden alle vom Nutzer klassifizierten Textantworten der `train`-Methode übergeben, kann durch einen Aufruf von:

```
nbayes.classify( "nicht klassifizierte Textantwort..."
.split(/\textbackslash s+/) ).max\_class
```

die Klasse ermittelt werden, in welche die übergebene Textantwort mit höchster Wahrscheinlichkeit gehört. So werden alle nicht-klassifizierten Textantworten einer Klasse zugeordnet und abschließend erneut gerendert.

Die `reset`-Methode des ResultsControllers ermöglicht es, das Abstimmungsergebnis einer Abstimmung zurückzusetzen. Hierzu wird die Abstimmung, deren Ergebnis zurückgesetzt werden soll, in die Instanzvariable `@voting` geladen. Mit dem Aufruf von `@voting.votes.destroy_all` werden anschließend alle mit der Abstimmung in Relation stehenden Stimmen gelöscht. Durch anschließendes Rendern eines Javascript-Templates werden entsprechende visuelle Rückmeldungen im Browser des Anwenders erzeugt.

LecturesController

Im LecturesController befinden sich alle Methoden zum Anzeigen, Anlegen, Bearbeiten und Löschen von Veranstaltungen. Die `index`-Methode rendert die View, welche die Übersicht über alle Veranstaltungen und Abstimmungen des angemeldeten Dozenten liefert (Abbildung 4.2.3).

VotingsController

Mit Hilfe des VotingsControllers werden Abstimmungen angelegt, geändert und gelöscht.

Die standardmäßig vorhandenen Methoden `index`, `new` und `edit` fehlen hier. Sie werden nicht benötigt, da die Übersicht über alle Veranstaltungen und Abstimmungen des jeweiligen Dozenten schon in der `index`-Methode des LecturesControllers gerendert wird. Das Formular zum Anlegen und Bearbeiten der Abstimmungen wird hierbei als Partial in diese Übersicht eingebunden.

UsersController

Der UsersController beinhaltet alle nötigen Funktionen zum Anlegen, Bearbeiten und Löschen von Dozenten und Dozenten mit Administratorrechten.

SessionsController

Der SessionsController ist für die Anmeldung der Teilnehmer und Dozenten am System und deren Abmeldung zuständig. Hierzu nutzt er die durch das bcrypt-Gem (siehe bcrypt 3.2.2) bereitgestellte `authenticate`-Methode zur Authentifizierung der Dozenten und Dozenten mit Administratorrechten. Zur Anmeldung der Teilnehmer wird das bcrypt-Gem nicht verwendet, da es zur sicheren Verschlüsselung der Passwörter einen salt-Wert verwendet. Somit wäre die Zuordnung zu einer Veranstaltung über das vom Teilnehmer eingegebene Passwort nicht mehr möglich. Daher werden Veranstaltungspasswörter ohne salt-Wert gehasht. Ein Teilnehmer ist dann erfolgreich angemeldet, wenn eine Datenbankanfrage mit dem (gehashten) eingebenem Passwort eine Veranstaltung zurückliefert.

5.2 Android-Applikation

5.2.1 Aufbau

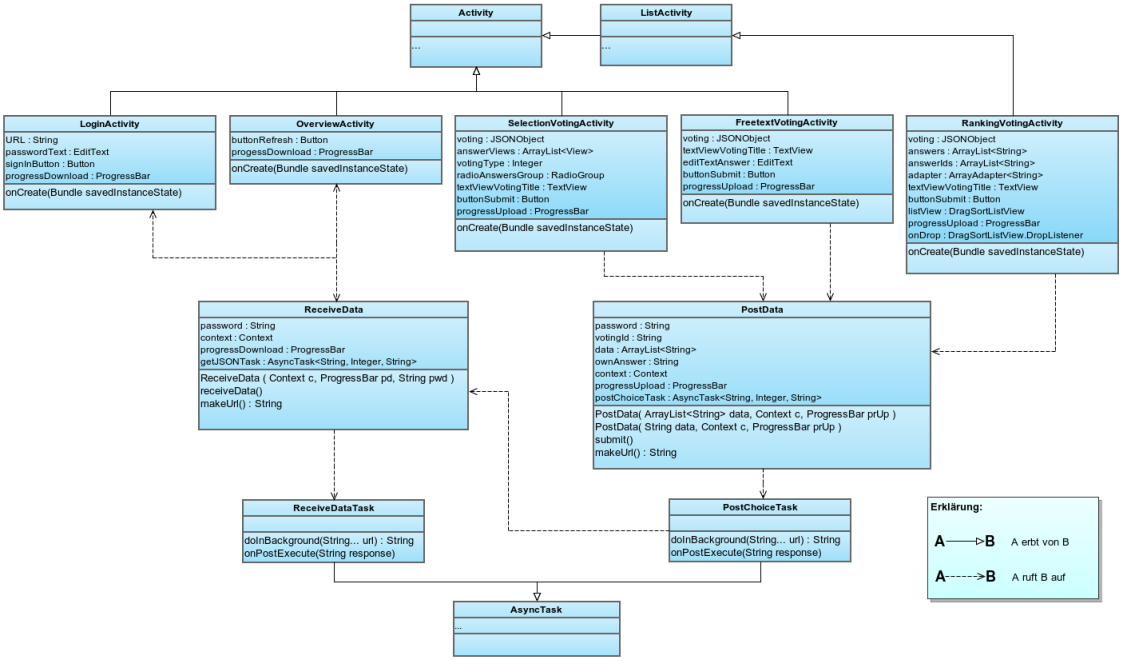


Abbildung 5.5: Klassendiagramm der Android-Applikation

Das Klassendiagramm in Abbildug 5.5 lässt sich in fünf Ebenen unterteilen. Die oberste Ebene enthält zwei Klassen, die vom Android-SDK bereitgestellt werden, die Activity-Klasse (siehe Activities 3.5.1) und die davon erbende ListActivity-Klasse. Beide Klassen enthalten Methoden, welche die Klassen der zweiten Ebene durch Vererbung erhalten. Die Klassen der zweiten Ebene stehen jeweils für eine Benutzer-Interaktion und besitzen je eine zugehörige View. Die SelectionVotingActivity ist, wie der Name vermuten lässt, für zwei Abstimmungstypen zuständig. Dieser Weg wurde gewählt, da der erste und zweite Abstimmungstyp eine ähnliche View benötigen.

In der dritten Ebene der Grafik befinden sich zwei Klassen, die zur Kommunikation mit der REST-Schnittstelle der Webanwendung genutzt werden. Zum Abrufen und Senden der Daten starten diese zwei Threads in Form von AsyncTasks (siehe AsyncTasks 3.5.3), die in separaten Klassen angelegt sind (vierte Ebene). Die AsyncTasks erben wiederum von einer vom Android-SDK bereitgestellten Klasse (fünfte Ebene).

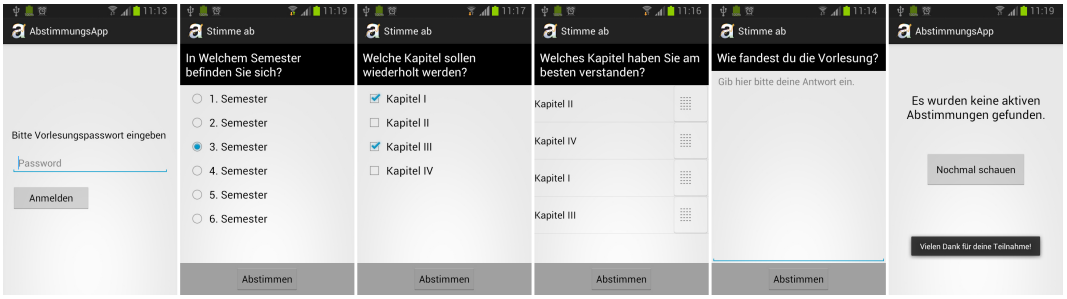


Abbildung 5.6: Die zu den Activities gehörenden Oberflächen

5.2.2 Activities

LoginActivity und OverviewActivity

Die LoginActivity ist hauptsächlich für die Darstellung der Anmelde-Oberfläche zuständig. Diese besteht aus einem Textfeld, einem Login-Button und einem versteckten ProgressDialog um anzuzeigen, ob gerade nach aktiven Abstimmungen gesucht wird. Beim Betätigen des Login-Buttons wird eine Instanz der ReceiveData-Klasse erstellt, falls noch keine existiert. Anschließend wird die `receiveData`-Methode der Klasse aufgerufen.

Die OverviewActivity ähnelt vom Aufbau stark der LoginActivity. Die Oberfläche besitzt ebenfalls einen ProgressDialog, um anzuzeigen, ob gerade nach aktiven Abstimmungen gesucht wird und einen Aktualisieren-Button. Ein Textfeld für die Passworteingabe ist nicht nötig, da das Passwort bei erfolgreicher Anmeldung an eine Veranstaltung in den App-Einstellungen bis zum erneuten Aufruf der LoginActivity zwischengespeichert wird. Durch Betätigen des Buttons wird erneut eine Instanz der ReceiveData-Klasse erstellt und die `receiveData`-Methode der Klasse aufgerufen.

SelectionVotingActivity

Die SelectionVotingActivity ist für die Darstellung der Abstimmungen des ersten und zweiten Abstimmungstyps gedacht. Die Daten der Abstimmung werden aus dem Intent, mit dem die Activity gestartet wurde, geladen und in ein JSON-Objekt konvertiert. Die darin enthaltenen Daten werden in die View übertragen. Die View besteht aus einer TextView zur Darstellung der Fragestellung. Des Weiteren werden die Antwortmöglichkeiten dargestellt: Je nach Abstimmungstyp als Options- oder Checkbox. Beim Betätigen des Abstimmen-Buttons wird eine Instanz der PostData-Klasse erzeugt und deren `submit`-Methode aufgerufen.

Die Funktionsweise der RankingVotingActivity und der FreetextVotingActivity ist nahezu identisch, daher wird auf eine Erläuterung an dieser Stelle verzichtet.

5.2.3 Anbindung an die REST-Schnittstelle

Die Anbindung an die REST-Schnittstelle erfolgt duch die Klassen ReceiveData und PostData. Je nach Internetverbindung des Gerätes kann eine HTTP-Anfrage einige Sekunden in Anspruch nehmen. Daher werden die HTTP-Anfragen in eigenen AsyncTask ausgeführt um ein Einfrieren der Benutzeroberfläche zu verhindern.

ReceiveData

Beim Instanzieren der ReceiveData-Klasse wird ihr der Anwendungs-Kontext, ein ProgressDialog und das Veranstaltungspasswort übergeben. Durch Aufruf der `receiveData`-Methode wird der ProgressDialog in der View sichtbar gemacht, um dem Benutzer anzuzeigen, dass die Anfrage gestartet wurde. Außerdem wird ein Thread gestartet, dem die URL zur REST-Schnittstelle der Webanwendung übergeben wird. In dieser URL werden das verschlüsselte Veranstaltungspasswort und ein das Gerät identifizierender String (z.B. die IMEI des Gerätes) als Parameter übergeben. Der gestartete Thread instanziert den in Java zu Verfügung stehenden Apache HTTP-Clienten. Dieser nutzt die HTTP-GET-Methode um eine Anfrage zur übergebenen URL zu starten. Ist die Anfrage nicht erfolgreich, wird dem Benutzer eine Fehlermeldung angezeigt. Ansonsten wird aus der übergebenen Antwort ein JSON-Objekt erzeugt. Enthält das Objekt eine Abstimmung, wird die dem Abstimmungstyp entsprechende Activity gestartet. Die Abstimmungsdaten werden dem Intent (siehe Intent 3.5.2) zum Starten der jeweiligen Activity beigefügt.

Enthält das JSON-Objekt den Status `noactive` war die Anmeldung erfolgreich. Es liegen jedoch keine aktiven Abstimmungen vor, an denen der Teilnehmer noch nicht teilgenommen hat. Daher wird die OverviewActivity gestartet.

Enthält das Objekt eine Fehlermeldung wird diese dem Teilnehmer angezeigt. Vor dem Beenden des AsyncTasks wird der ProgressDialog wieder versteckt, da die Anfrage durchgeführt wurde.

PostData

Dem Konstruktor der PostData-Klasse wird ebenfalls der Anwendungs-Kontext und ein ProgressDialog übergeben. Die zu übermittelnden Daten können in Form einer ArrayList aus Strings oder als einzelner String übergeben werden. Letzteres wird für den vierten Abstimmungtyp genutzt. Der String enthält dann die durch den Benutzer eingegebene Textantwort. Wird nun die `submit`-Methode aufgerufen, wird zunächst wieder der übergebene ProgressDialog sichtbar gemacht, um dem Nutzer anzuzeigen, dass seine Antwort übermittelt wird. Anschließend wird ein AsyncTask gestartet. Dieser instanziert einen Apache HTTP-Clienten und nutzt die HTTP-POST-Methode um die beim Instanzieren der PostData-Klasse übergebenen Daten zu übermitteln. Wurde der HTTP-POST erfolgreich durchgeführt, wird der Inhalt der Antwort der `onPostExecute`-Methode als String übergeben. Ist die Antwort null wird eine Fehlermeldung ausgegeben, ansonsten wird ein

JSON-Objekt aus dem String erzeugt. Enthält dieses den Status `success`, wird eine Instanz der ReceiveData-Klasse erzeugt, um erneut nach weiteren aktiven Abstimmungen zu schauen. Enthält es eine Fehlermeldung wird diese dem Nutzer angezeigt.

Die folgende Abbildung 5.7 verdeutlicht nochmals den Ablauf der Android-Applikation.

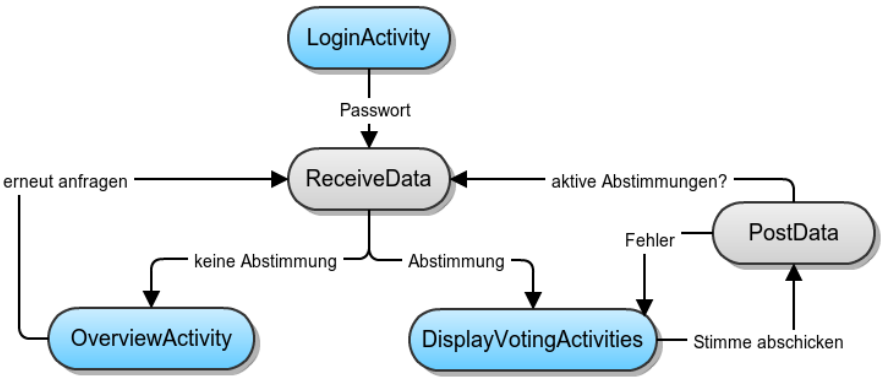


Abbildung 5.7: Ablauf der Android-Applikation auf Programmebene

6 Fazit

Mit Hilfe des in dieser Arbeit vorgestellten und erläuterten Audience Response Systems ist es möglich Abstimmungen unter zahlreichen Teilnehmern zügig durchzuführen und auszuwerten. Hierzu wird zum Abstimmen auf die Hardware der Teilnehmer zurückgegriffen. Die Bedienung der Software ist intuitiv und schnell, um die Durchführung und Auswertung von Abstimmungen zu beschleunigen. Hierzu wurde auf die in Kapitel 3 vorgestellte Software verwendet. Als gute Wahl erwies sich der Einsatz des Webframeworks Ruby on Rails. Der Ansatz die Konvention über die Konfiguration zu stellen, wie auch die gute Dokumentation des Frameworks erleichtert den Einstieg. Als ebenfalls gute Wahl erwies sich das Bootstrap-Framework. Zu Beginn der Implementierung wurde zunächst JQuery UI für die Gestaltung der Oberflächen eingesetzt. Nach dem Wechsel auf Bootstrap zeigte sich jedoch, dass eine einheitliche Struktur der Oberfläche hiermit leichter zu realisieren ist. Für visuelle Effekte oder sortierbare Elemente wurde aber weiter auf JQuery UI zurückgegriffen.

Man kann sagen, dass alle an die Software gestellten Anforderungen bis jetzt erfüllt wurden. Von einer fertigen Software lässt sich jedoch noch nicht sprechen. Vor einem ersten richtigen Einsatz wäre eine Testphase wichtig. Außerdem sind noch kleine strukturelle Änderungen an der Ruby on Rails-Anwendung zu erledigen. Auch die Benutzeroberflächen sollte mit weiteren Hinweisen für den Anwender versehen werden um die Bedienung zu erläutern. Nicht zuletzt fehlen noch Applikationen für weitere Smartphone-Betriebssysteme (iOS, BlackBerry etc.) um möglichst allen Teilnehmern eine native Anwendung zum Abstimmen anbieten zu können.

Im Vergleich zu Anderen Audience Response Systemen steht die in dieser Arbeit vorgestellte Software trotzdem gut da. Die Entscheidung, dass Teilnehmer sich zu einer Veranstaltung anmelden, um zu den darin in enthaltenen, aktiven Abstimmungen zu gelangen, war gut. So müssen Teilnehmer nur einmal das Veranstungspasswort eingegeben, um Zugriff auf die im Laufe der Veranstaltung aktivierten Abstimmungen zu erhalten. Auch die sortierbaren Antwortmöglichkeiten werden zur Zeit von keinem anderen System angeboten. Es lässt sich also sagen, dass das vorgestellte System eine bessere Integration von Abstimmungen in Veranstaltungen gewährleisten kann als andere Systeme.

Neben den genannten noch zu erledigenden Aufgaben, wurden nach der ersten Präsentation der Software noch weitere Verbesserungsvorschläge gemacht. Sinnvoll wäre es den Dozenten anzuzeigen, wie viele Teilnehmer gerade für eine Veranstaltung angemeldet sind, und wie viele an den aktiven Abstimmungen bereits teilgenommen haben. Für den Einsatz in Vorlesungen sollten sich Abstimmungen innerhalb der Veranstaltung in (Vorlesungs-)Termine gruppieren lassen. Auch eine mehrfache Durchführung von Ab-

stimmungen, ohne deren Ergebnis zurücksetzen zu müssen, wäre wünschenswert, um Ergebnisse direkt miteinander vergleichen zu können.

Neben den gerade aufgeführten Verbesserungsvorschlägen sind aber auch einige Konzepte der in Kapitel 2 vorgestellten Systeme sehr interessant. mQlicker und eduVote ermöglichen die mehrfache Durchführung einzelner Abstimmung, indem sich Sessions einer Abstimmung starten lassen. Diese Sessions enthalten nach der Durchführung das Abstimmungsergebnis und ermöglichen daher einen direkten Vergleich dessen. Ein großer Nachteil des PPVote Systems ist auch gleichzeitig sein Vorteil. Durch die Anschaffung der Abstimmungshardware ist sichergestellt, dass jeder Teilnehmer an den angebotenen Abstimmungen teilnehmen kann und nicht mangels Hardware ausgeschlossen wird. Die zunehmende Verbreitung von Smartphones etc. lässt jedoch erwarten das dieses Problem in den kommenden Jahren noch geringer wird.

Abschließend ist festzustellen, dass die Entwicklung des Audience Response Systems viel Zeit gekostet, aber auch Freude bereitet hat. Ein Großteil der Arbeit floß hierbei in die Planung und Implementierung der Webanwendung. Die relativ große Auswahl an verschiedener Software, die hierbei zum Einsatz kam, erforderte Einarbeitungszeit und Recherche. Der kleinere Teil der Arbeit war die Android-Applikation, was als positiv empfunden wurde. Der zunehmende Trend, Anwendungen in das Internet auszulagern, wird meinerseits weiterhin gespannt verfolgt werden. Und damit einhergehend auch der Fortschritt der zur Entwicklung benötigten Software.

7 Anhang

7.1 Installationsvoraussetzungen

Im Folgenden wird die Installation des in dieser Arbeit vorgestellten Audience Response Systems erläutert. Im Beispiel wurde hierzu auf die Linux Distribution Debian in der Version 6.0.7 zurückgegriffen. Für die Android-Applikation wird die Eclipse IDE vorausgesetzt. Diese Installationsanleitung dient nur zum Ausprobieren der Software, da der verwendete WEBrick Server nicht für den produktiven Einsatz gedacht ist.

7.2 Installation

7.2.1 Ruby on Rails-Anwendung

Zunächst wird die Versionsverwaltungssoftware Git installiert:

```
sudo apt-get install git
```

Anschließend wird ein neues Verzeichnis angelegt und in das Verzeichnis gewechselt:

```
mkdir abstimmungsapp
cd abstimmungsapp
```

Nun wird ein Git-Repository in diesem Verzeichnis initialisiert. Danach wird ein Remote-Repository angegeben. Mit dem pull-Befehl wird dann die Software aus dem Repository geladen. Die Zugriffsberechtigung für das Remote-Repository wird vorausgesetzt.

```
git init
git remote add origin
ssh://git@git-ps.informatik.uni-kiel.de:55055/theses/2013-hecht-ba.git
git pull origin master
```

Anschließend wird zur Ausführung der Anwendung benötigte Software installiert. Um die aktuellste Ruby-Version zu erhalten, kann diese auch auf der Ruby-Homepage[1] heruntergeladen werden.

```
sudo apt-get install ruby
sudo apt-get install libsqlite3-dev
sudo apt-get install nodejs
```

[1] http://www.ruby-lang.org

Bei aktuellen Ruby-Versionen ist das zugehörige Paketverwaltungssystem RubyGems[2] enthalten. Dieses wird nun genutzt um das Ruby on Rails-Gem zu installieren.

```
sudo gem install rails
```

Anschließend wechseln wir in das Verzeichnis der Ruby on Rails-Anwendung. Der Befehl `bundle install` installiert alle für die Rails-Anwendung benötigten Gems via RubyGems.

```
cd rails/abstimmungsapp
bundle install
```

Nach der Installation der erforderlichen Ruby-Gems lässt sich der WEBrick-Server starten.

```
rails s
```

7.2.2 Android-Applikation

Um die Android-Applikation nutzen zu können wird die Eclipse IDE vorausgesetzt. Außerdem wird das Android SDK[3] benötigt. Um das Android SDK in Eclipse nutzen zu können sind die Android Developer Tools (ADT) notwendig. Bei den ADT handelt es sich um ein Eclipse Plugin. Dieses lässt sich über ein Eclipse-Repository[4] installieren.

Sind alle Voraussetzungen erfüllt, kann das Eclipse-Projekt importiert werden. Es befindet sich im Ordner "`android/AbstimmungsApp`". des Git-Repositories[5]. Des Weiteren muss noch das Dragsort Listview-Projekt importiert werden, es befindet sich im Ordner "`android/library`". Nun lässt sich die Anwendung als Android-Applikation ausführen. Hierzu kann im Android Virtual Device Manager (der direkt in Eclipse augerufen werden kann) ein Android-System emuliert werden. Diese Lösung bietet sich an, falls keine Android-Hardware zur Verfügung steht. Es empfiehlt sich vorher das Intel x86 Atom System Image über den in Eclipse aufrufbaren Android SDK-Manager zu installieren. Das Image kann daraufhin beim Anlegen eines virtuellen Gerätes im Android Virtual Device Manager unter CPU/ABI ausgewählt werden. Das emulierte Android-System läuft so meist deutlich schneller. Ist Android-Hardware vorhanden, kann diese per USB-Kabel an den Rechner angeschlossen werden. Um die Android-Applikation auf dem Gerät ausführen zu können, muss die Option USB-Debugging in den Entstellungen des Gerätes aktiviert sein. Ist dies der Fall kann nun (installierte Treiber vorausgesetzt) die Android-Applikation auf dem Gerät ausgeführt werden.

[2] `http://www.rubygems.org`
[3] `http://developer.android.com/sdk/index.html`
[4] `https://dl-ssl.google.com/android/eclipse`
[5] `ssh://git@git-ps.informatik.uni-kiel.de:55055/theses/2013-hecht-ba.git`

Literaturverzeichnis

[api13a] *jQuery API Documentation.* `http://api.jquery.com`. Version: 2013. – [Online; Stand 7. April 2013]

[api13b] *Ruby on Rails Documentation.* `http://api.rubyonrails.org`. Version: 2013. – [Online; Stand 7. April 2013]

[ASRMSK12] Aubourg, Julian ; Song, Jungkee ; R. M. Steen, Hallvord ; Kesteren, Anne van: *XMLHttpRequest - W3C Working Draft 6 December 2012.* `http://www.w3.org/TR/XMLHttpRequest`. Version: 2012. – [Online; Stand 7. April 2013]

[BCD+13] Bigg, Ryan ; Cheung, Frederick ; Darell, Tore ; Dean, Jeff ; Gunderloy, Mike ; Lindsaar, Mikel ; Marques, Cássio ; Miller, James ; Naik, Pratik ; Tagua, Emilio ; Webers, Heiko: *Layouts and Rendering in Rails.* `http://guides.rubyonrails.org/layouts_and_rendering.html`. Version: 2013. – [Online; Stand 7. April 2013]

[BV10] Bongers, Frank ; Vollendorf, Maximilian: *jQuery - Das Praxisbuch.* Galileo Press, 2010

[ECM11] *ECMAScript Language Specification.* `http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf`. Version: Juni 2011

[Fow03] Fowler, Martin: *Patterns of enterprise application architecture.* Addison-Wesley, 2003

[KGHM12] Kesteren, Anne van ; Gregor, Aryeh ; Hunt, Lachlan ; Ms2ger: *DOM4 - W3C Working Draft 6 December 2012.* `http://www.w3.org/TR/dom`. Version: 2012. – [Online; Stand 7. April 2013]

[Kü11] Künneth, Thomas: *Android 3 - Apps entwickeln mit dem Android SDK.* Galileo Press, 2011

[LHWW09] Le Hégaret, Philippe ; Whitmer, Ray ; Wood, Lauren: *Document Object Model (DOM).* `http://www.w3.org/DOM`. Version: 2009. – [Online; Stand 7. April 2013]

[MO12] Morsy, Hussein ; Otto, Tanja: *Ruby on Rails 3.1 - Das Entwickler Handbuch.* Galileo Press, 2012

[rub13a] *Programming Ruby - The Pragmatic Programmer's Guide.* `http://www.ruby-doc.org`. Version: 2013. – [Online; Stand 7. April 2013]

[rub13b] *Ruby a Programmer's Best Friend - Über Ruby*. http://www.ruby-lang.org/de/about. Version: 2013. – [Online; Stand 7. April 2013]

[Sch07] Schäfer, Mathias: *Das Document Object Model (DOM)*. http://de.selfhtml.org/dhtml/modelle/dom.htm. Version: 2007. – [Online; Stand 7. April 2013]

[Wen01] Wenz, Christian: *JavaScript - Browserübergreifende Lösungen*. Galileo Press, 2001 http://openbook.galileocomputing.de/javascript/javascript01.htm. – E-Book

[Wik13a] Wikipedia: *Ajax (Programmierung) — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/w/index.php?title=Ajax_(Programmierung)&oldid=115876052. Version: 2013. – [Online; Stand 7. April 2013]

[Wik13b] Wikipedia: *Bayes-Klassifikator — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/w/index.php?title=Bayes-Klassifikator&oldid=116313358. Version: 2013. – [Online; Stand 7. April 2013]

[Wik13c] Wikipedia: *Open Handset Alliance — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/w/index.php?title=Open_Handset_Alliance&oldid=116787178. Version: 2013. – [Online; Stand 7. April 2013]