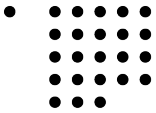

Praxisprojekt
Mobile Echtzeitbewertung von Vortragsveranstaltungen

vorgelegt an der Fachhochschule Köln, Campus Gummersbach
im Studiengang Medieninformatik

ausgearbeitet von:
Manuel Krischer, Matr.-Nr.: 11054020
Erster Prüfer: Prof. Dr. Horst Stenzel
Zweiter Prüfer: Prof. Dr. Hans-Ludwig Stahl
Gummersbach, 1. Juli 2010
Dokumentation V1.0

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Problemstellung

Seminare und Vorträge werden in immer vielfältigeren Umgebungen durchgeführt. Bislang dominiert die Anwesenheit lokaler Zuhörer, doch zeigt der Zukunftstrend immer mehr auf eine gemischte Umgebung mit lokalen und entfernten Zuhören oder sogar einem Publikum nur aus entfernten Personen. Während die Technik zur Übertragung der seminaristischen Veranstaltungen in Echtzeit bereits in einer sehr ausgereiften Form zur Verfügung steht, ist es bislang für die entfernten Zuhörer nur eingeschränkt möglich, in die Veranstaltung mit einer Rückmeldung einzugreifen.

Diese Ausarbeitung richtet sich an die Vortragenden, die eine Echtzeitrückmeldung zu Ihren Vorträgen ermöglichen möchten, und an Zuhörer von Vorträgen, denen es ermöglicht wird, mittels verschiedener Kommunikationskanäle mit der dozierenden Person zu interagieren.

1.1 Umfang dieser Arbeit

In diesem Praxisprojekt sind eine Analyse zu verfügbaren Lösungen am Markt und ein technisches Konzept als Grundlagen für das weitere Vorgehen enthalten. Darauf aufbauend wurde eine Machbarkeitstudie mittels eines Prototypes entwickelt um das Konzept zu validieren. Das Konzept des Usabilty Engineering, die Durchführung der entsprechenden Methoden zum Human Interaction Design und die Evaluierung werden in einer thematisch anschließenden Bachelorthesis behandelt.

2 Motivation

Für die mobilen Echtzeit-Bewertungen von Seminaren und Vorlesungen soll ein System geschaffen werden, welches von einer breiten Nutzergruppe einfach und schnell verwendet werden kann. Es soll Feedback zur laufenden Veranstaltung von den Teilnehmern an den Vortragenden ermöglichen.

2.1 Begriffsdefinitionen

Zum besseren Verständnis dieses Dokuments zu Beginn einige Begriffsdefinitionen.

- **Veranstaltung** Als Veranstaltung wird eine einzelne Vorlesung oder ein Vortrag bezeichnet.
- **Veranstaltungsschlüssel** Für jede Veranstaltung wird im System ein zufälliger Schlüsselwert erzeugt.
- **Vortragender** Der Vortragende ist der Hauptverantwortliche und Durchführende einer Veranstaltung.
- **Benutzer** Alle Hörer einer Veranstaltung sind potentielle Nutzer des Systems.
- **Attribut** Ein Attribut ist eine Bewertungseigenschaft einer Veranstaltung, beziehungsweise des Vortragenden. Es muss sich dabei um bewertbare Eigenschaften handeln.
- **Bewertung** Eine Bewertung ist der von einem Benutzer eingegebene Datensatz mit einer optionalen Bewertung der Attribute, sowie der ebenfalls optionalen Bewertung mittels eines Freitextes.
- **Kommentar** Ein Kommentar bezeichnet eine Bewertung von einem Benutzer, die an eine Veranstaltung gekoppelt ist.
- **Eingabemodul** Eingabemodule dienen zur Transliteration von Eingaben der Benutzer zu einem Kommentar. In welcher Form die Eingabe erfolgt ist unerheblich, sofern das Eingabemodul einen validen Kommentar für das System daraus erzeugen kann.

3 Marktübersicht

Zu Beginn der Arbeiten steht eine Analyse bereits verfügbarer Programme in diesem Anwendungssegment.

3.1 Poll Everywhere

- Interact With Your Audience in Real Time -

So lautet der Untertitel zu "Poll Everywhere", einer Software, die ihren Fokus auf Echtzeit-Umfragen (Miles, 2010) setzt und eine eigenständige Lösung darstellt. Die Kommunikation erfolgt mit den Endgeräten der Zuhörer, das System kann mit Daten durch SMS Text, Twitter oder von einem Webinterface umgehen. Die Auswertungseinheit reagiert auf neu eingehende Antworten und aktualisiert die Ergebnisse in Echtzeit. Zudem ist es möglich, die Umfragen auf eine bestimmte Zuhörergruppe zu beschränken, die eingehenden Nachrichten können vor der Veröffentlichung moderiert werden und die Zuhörer erhalten optional eine automatische Antwort. Die Ergebnisse werden dem Fragesteller zur weiteren Verwendung als CSV[1] Daten oder Powerpoint Folien bereitgestellt.

Die Software kann in verschiedenen Preiskategorien (PollEverywhere, 2010) mit monatlichen Betrag gebucht werden, von 30 teilnehmenden Zuhörern pro Umfrage in der freien Variante bis zu 20.000 Teilnehmern in der Platinumversion. Auch Finanzierungsmodelle für Bildungseinrichtungen werden angeboten.

"Poll Everywhere" ist ein Dienst der auf den Servern der gleichnamigen Firma ausgeführt wird und erlaubt keinen Zugriff auf den Programmcode. Für die Einbindung der Funktionen wird allerdings eine REST[2] API angeboten, die Zugriff auf die Daten in verschiedenen Web-Standardformaten wie JSON, XML, RSS, CSV, und SWF) bietet.

3.2 Vorzeitig ausgeschiedene Systeme

Weitere Echzeit-Abstimmungssysteme, die auf bereits durch ihre Gestaltung oder Konzept den Anforderungen nicht gerecht werden konnten.

[1] Comma Separated Value, Standard ASCII Format für Tabellendaten.
[2] Representational State Transfer, einfache Datenübertragung mittels URI und HTTP Protokoll

3.2.1 Tagga

Ausscheidungskriterium: Nur SMS als Zugangsmöglichkeit vorgesehen.

URL http://www.tagga.com/

3.2.2 Mozes

Ausscheidungskriterium: Nur SMS als Zugangsmöglichkeit vorgesehen.

URL http://www.mozes.com

3.2.3 mFrage

Ausscheidungskriterium: Nur SMS als Zugangsmöglichkeit vorgesehen.

URL http://www.mfrage.de

3.2.4 PowerVote

Ausscheidungskriterium: Eigene Hardwarekomponente zur Abstimmung benötigt.

URL http://www.telerat.de/index.php?id=148#c137

3.2.5 PowerPoint Twitter Tools

Ausscheidungskriterium: Funktioniert nur in der Kombination von Microsoft PowerPoint für Windows und Adobe Flash.

URL http://www.sapweb20.com/blog/powerpoint-twitter-tools/

3.3 Fazit

Poll Everywhere bietet einen Funktionsumfang, der für einen Großteil der Anwendungsfälle ausreichend ist, aber erlaubt keinen direkten Zugriff auf die Software der Umfrageserver. Durch die fehlende Quelloffenheit wird der Hochschule keine Möglichkeit zur Anpassungen oder Weiterentwicklungen oder Einbindung in ein modulares Podcastingsystem geboten.

Alle anderen Lösung bieten leider nur einen beschränkten Eingabeumfang, häufig nur per SMS, oder sind an eine eigene Hardwarelösung gekoppelt. Deshalb empfiehlt sich eine eigenständige Lösung, die allen gestellten Anforderungen gerecht wird.

4 Technisches Konzept

Das technische Konzept sieht eine verteilte Anwendung vor, die folgende grundlegende Kriterien als Randbedingungen erfüllen soll. Hierzu werden zuerst die Anforderungen an das Produkt aufgestellt und anschliessend daraus die grundlegende Funktionsweise der Softwarekomponenten festgelegt.

- **Plattformunabhängikeit** Um eine breite Verwendung finden zu können, sollte das Produkt nicht an eine Ausführungsplattform gekoppelt sein.
- **Quelloffenheit** Um Raum für Erweiterungen und Verbesserungen zu lassen, sollte das Produkt quelloffen entwickelt werden.
- **Multiple Dienstunterstützung** Die Menge und Art der Eingabemedien für die Benutzer sollte nicht durch das Produkt eingeschränkt sein.
- **Erweiterbarkeit** Durch die Verwendung von Standards und definierten Schnittstellen, soll das Produkt gut erweiterbar sein.

4.1 Anforderungsermittlung

Neben den grundlegenden Anforderungen an das Produkt stehen noch die funktionalen und nicht-funktionalen Anforderungen, welche durch Domänenrecherche und Usability Engineering ermittelt werden. Im Rahmen des Praxisprojekts wurden diese Punkte nicht bearbeitet, sondern von Erfahrungswerten aus vergangenen Projekten ausgegangen.

4.1.1 funktionale Anforderungen

Die funktionalen Anforderungen legen fest, was das Produkt machen soll.

Vortragender

- **Erstellung neuer Veranstaltungen** Der Vortragende soll neue Veranstaltungen einrichten können.
- **Erstellung neuer Umfragen** Umfragen innerhalb einer Veranstaltung sollten schnell einsatzfähig gemacht werden können.

- **Beenden der laufenden Veranstaltung** Die Möglichkeit eine Veranstaltung bewerten zu können, sollte einfach abgeschlossen werden können.
- **Überblick der aktuellen Bewertungen** Der aktuelle Bewertungsstand sollte übersichtlich dargestellt werden.
- **Anzeige der Umfrageergebnisse** Die Ergebnisse einer Umfrage müssen einfach einsehbar sein.
- **Projektionsgerät** Die Anzeige der Bewertungen sollte nicht auf dem Projektionsgerät, sondern nur auf dem Primärbildschirm erfolgen. Bei Umfragen sollte eine Wahlmöglichkeit bezüglich der Darstellung angeboten werden.
- **Hilfe** Die Funktionen sollten einfach und übersichtlich erklärt werden.

Benutzer

- **Bewertungsmöglichkeit** Die Benutzer sollen Veranstaltungen bewerten können.
- **Keine Anmeldung an neuem System** Um Veranstaltungen bewerten zu können, sollte keine Anmeldung an einem neuen System nötig sein, sondern die bereits vorhandenen Kommunikationswege der Benutzer verwendet werden.
- **Geringfügige Änderungen** Zwischen Veranstaltungen von einer Institution oder Veranstaltungsreihe sollten keine großen Änderungen bei der Syntax einer Bewertung vorkommen.
- **Rückmeldung vom System** Das System soll eine Rückmeldung zu den Benutzereingaben anzeigen.
- **Hilfe** Es sollte eine einfache und übersichtliche Hilfe zu den Bewertungsmöglichkeiten zur Verfügung stehen.

Administration

- **Aktualisierungsfähigkeit** Es sollte möglich sein verbesserte oder erweiterte Versionen zentral und einfach einzuspielen. Kompatibilität zu früheren Versionen muss gegeben sein.
- **zentrale Verwaltung** Es sollte möglich sein von einer zentralen Stelle Wartungsarbeiten durchzuführen, ohne zum Kunden fahren zu müssen.
- **Datensicherung** Datenbestände sollten durch einfache Datensicherungs-Methoden gesichert werden können.
- **Verwaltungsoberfläche** Es sollte ein administratives Tool zur Verfügung stehen in dem die anfallenden Arbeiten erledigt werden können.
- **Dokumentation** Eine ausführliche Dokumentation über die Funktionen und Schnittstellen sollte zu Verfügung gestellt werden.

- **Hilfe** Es sollte eine Hilfe zur Bedienung zur Verfügung stehen.

4.1.2 nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen stellen die gewünschten Eigenschaften des Produktsystems heraus.

- **Verfügbarkeit/Zuverlässigkeit** Bei verteilten Systemen ist besonders darauf zu achten, dass benötigte Daten durch Datensicherungs-Mechanismen konsistent, sicher gespeichert und steht für die Nutzer zu Verfügung stehen. Weitere Sicherungsmechanismen sollten bei der Übertragung und Überprüfung von Daten existieren, um die Korrektheit der Daten zu überwachen.
- **Erweiterbarkeit** Eine moderne Softwareanwendung sollte auf ein offenes Erweiterungsmodell beinhalten. Ein modularer Aufbau, sowie Dokumentation und Beschreibung der Schnittstellen sind hierzu von Nöten. Unabhängigkeit von Plattformen und Sprachen sollte ermöglicht werden.
- **Sicherheit und Integrität** Nutzerdaten müssen sicher aufbewahrt werden und vor unberechtigten Zugriffen geschützt werden. Bei der Übertragung von persönlichen Daten müssen durch eine angemessene Verschlüsselung und/oder Übertragungsart das Risiko eines erfolgreichen Abhörvorgangs minimiert werden.
- **Skalierbarkeit** Spätere Anpassung im Umfang sowie die Erweiterung der Leistung eines Softwareproduktes fordern die Skalierbarkeit dieser. Die Vermeidung von "Bottlenecks" kann durch mitwachsende Verbindungsmöglichkeiten und Erweiterungen verhindert werden. Vorherige Analyseszenarien helfen von Beginn an "Bauteile" sinnvoll zu proportionieren.
- **Gebrauchstauglichkeit** Gerade für technisch weniger versierte Nutzer aber auch generell um eine ordentliche Software zu entwerfen ist dem Punkt "Usability" ein hohes Maß an Aufmerksamkeit zu widmen. Nutzer und ihre verschiedenen Bedürfnisse im Bezug auf das System sollten frühzeitig ermittelt werden und zu einer sicheren, effektiveren und effzienteren Softwarelösung führen. Hierfür stehen diverse Modelle zur Verfügung.

4.2 Systemarchitektur

Bei der pragmatischen Herangehensweise an die Prototypentwicklung wurde ein klassisches Client-Server Modell entwickelt. Der zentrale Server verwaltet alle Veranstaltungen und übermittelt alle Daten nur den betroffenen Clients der Benutzer oder Vortragenden. Dieses Vorgehen verhindert ein übermäßiges Datenvolumen bei den Übetragungen und schränkt Inkonsistenzen in der Datenhaltung ein.

Die Eingabemodule arbeiten unabhängig und liefern die Daten der Benutzer an den Server. Die Eingabemodule wandeln die dienstabhängigen Nachrichten in dienstunabhängige Nachrichten des Systems um (Abschnitt 4.3). Nach der Auswertung werden die Daten an den entsprechenden Client des betroffenen Vortragenden weitergeleitet.

In der Implementierung kommunizieren Server und Client über einen XMPP-Server miteinander, statt des Clients soll aber auch eine Weboberfläche, die mittels REST Webservice mit dem Server kommuniziert, zur Verwaltung von Veranstaltungen genutzt werden können.

4.3 Nachrichtenaufbau

Um mit dem System neue Nachrichten eines Benutzers zu einem Kommentar umzuwandeln, sind folgende Bestandteile von den jeweiligen Eingabemodulen zu übermitteln. Wie diese Bestandteile ermittelt werden, ist modulabhängig.

4.3.1 Bestandteile eines Kommentars

Zeitstempel Datum und Uhrzeit der Erstellung. Wird vom System generiert.

Dienstname Name des übermittelnden Dienstes. Muss vom Eingabemodul geliefert werden

Schlüssel Veranstaltungsschlüssel der Veranstaltung, an den dieser Kommentar gerichtet ist. Muss vom Eingabemodul geliefert werden

Absenderkennung Eindeutige Identifikation des sendenden Benutzers. Muss vom Eingabemodul geliefert werden.

Absendername Klarname des sendenden Benutzers. Wenn nicht vorhanden, fromID verwenden. Muss vom Eingabemodul geliefert werden.

Freitext Textkommentar. Muss vom Eingabemodul geliefert werden.

Attributbewertungen Benötigt “key“ und “value“. Bewertung eines Attributes mit dem Buchstabenschlüssel und Wert als Integer von 1 bis 9. Kann keinmal, einmal oder mehrmals verwendet werden. Bei gleichen Buchstabenschlüsseln wird das erste Vorkommen ausgewertet. Muss vom Eingabemodul geliefert werden.

Umfrage Wert für eine Umfrage. Kann keinmal oder einmal verwendet werden. Muss vom Eingabemodul geliefert werden.

5 Architektur des Prototypen

Um das aufgestellte Konzept zu validieren, wird unter dem Namen **LectuRate** im weiteren Verlauf eine protoypische Software konzipiert und implementiert. Hierzu muss die konkrete Systemarchitektur und weitere Implementierungsdetails wie die Wahl der Programmiersprache und das Format der ausgetauschten Nachrichten festgelegt werden.

5.1 Implementierungsplattform

Um der Anforderung einer plattformunabhängigen Lösung gerecht zu werden, sind einige Programmiersprachen und Umgebungen denkbar. Auf Grund des Verbreitungsgrades, der größtenteils überarbeitungsfreien Portierungen auf neue Zielplattformen, der Standardunterstützung und nicht zuletzt auch auf Grund der breiten Erweiterungsmöglichkeiten wurde der Prototyp in Java entwickelt.

Neben dem Sprachumfang von Java 1.6 wurden auch Erweiterungen wie Javamail (Oracle (2009)), Smack XMPP (JiveSoftware (2008), Saint-Andre (2004)), Jersey (Oracle (2010b)), TwitterResponse (Matt Harris (2010)) und JUnit (Kent Beck (2010)) verwendet.

Die enge Verzahnung und die vorhandenen Kenntnisse des Java Swing Frameworks (Oracle (2010a)) mit der Java Umgebung gaben den Ausschlag für die Verwendung dieses grafischen Toolkits, wobei die Erfüllung aller Anforderungen damit nicht erreicht werden kann.

5.2 Architektur

In Abbildung 5.1 ist dargestellt, wie die Architektur aus dem in Abschnitt 4.2 beschriebenen Konzept in der prototypischen Implementierung umgesetzt wurde. Der zentrale Server ist flankiert von den Eingabemodulen und Modulen zur Visualisierung der Veranstaltung.

5.3 Nachrichtenaustausch

Um Informationen zwischen den einzelnen Komponenten austauschen zu können, ist ein definiertes Nachrichtenformat nötig. Hierzu bieten sich einige textbasierte oder binäre Standards auf der Java-Plattform an. Die Entscheidung für Nachrichten auf XML-Basis wurde

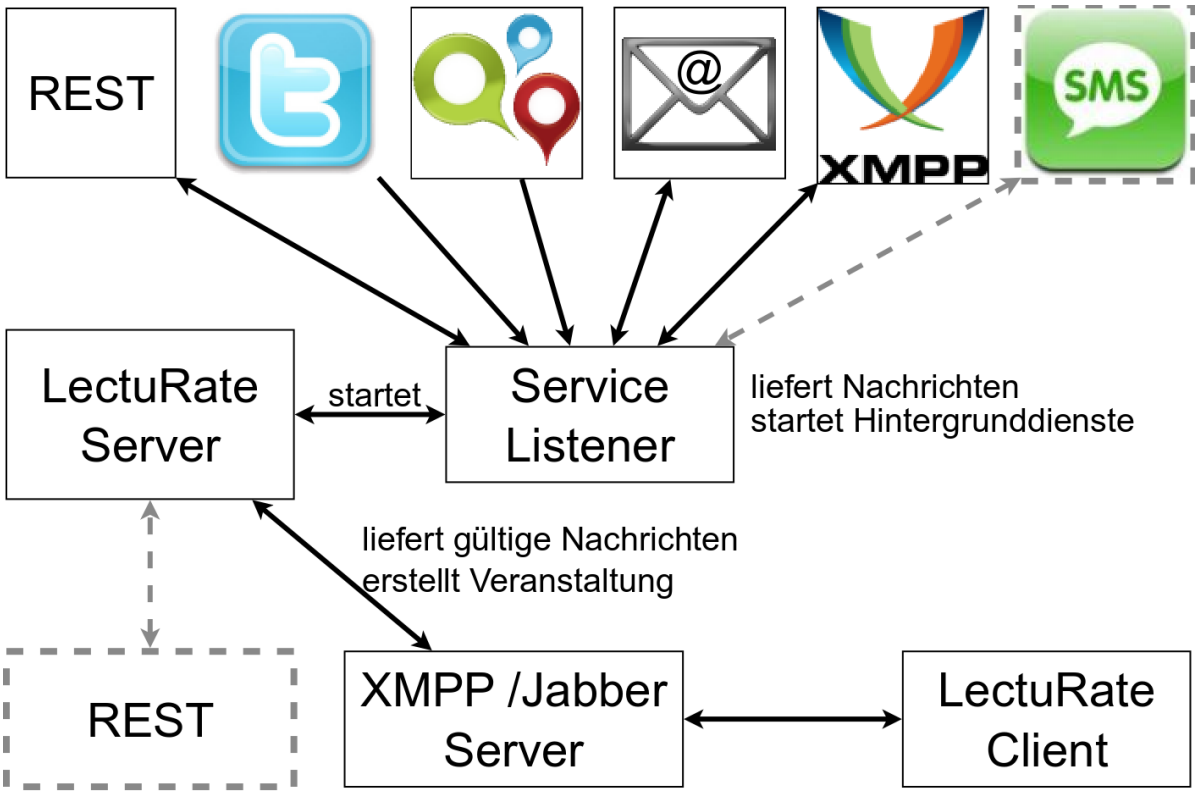

Abbildung 5.1: Systemüberblick des LectuRate Systems.
Grau gestrichelte Elemente gehören zum Konzept, aber sind nicht im Prototyp implementiert.

auf Grund der Verbreitung und der Lesbarkeit für Menschen getroffen. Bei der Fehlersuche sind Nachrichten, die ohne Hilfsmittel gelesen und ausgewertet werden können, eine große Erleichterung.

Nachrichten, wie sie zwischen den Programmkomponenten ausgetauscht werden, finden sich im Anhang im Abschnitt XML-Stanzas (10.7).

5.4 Kommentaraufbau

Nachfolgend sind die Bestandteile eines Kommentars aufgeführt, die von den Benutzern an das System übermittelt werden müssen. Die Art der Eingabe ist abhängig von dem genutzten Eingabemodul. Da für alle Eingabemöglichkeiten die selben Vorraussetzungen gelten sollen, wurde als kleinster gemeinsamer Nenner eine maximale Textlänge von 140 Zeichen und ein einfacher Zeichencode für textbasierte Eingabemodule vereinbart.

5.4.1 Bestandteile

Die Kommentare im LectuRate System gliedern sich in verschiedene Teile auf, die je nach eingesetztem Eingabemodul in ihrer Eingabe variieren können. Ausser dem obligatorischen Veranstaltungsschlüssel, hier als "#LECTUREKEY" bezeichnet, sind alle anderen Bestandteile je nach Eingabemodul und Verwendung optional.

```
@lecturate-account #LECTUREKEY *$ID0-9 *$POLL1-? Freitext
```

- **@lecturate-account** wird zur Identifizierung bei den Eingabemodulen eingesetzt, bei denen keine andere Adressierung der Nachrichten vorgesehen ist. Im Prototypen wird dieser Nachrichtenteil nur bei den Eingabemodulen für Twitter (siehe Abschnitt 5.5.1) und Identi.ca (siehe Abschnitt 5.5.1) benötigt.
- **#LECTUREKEY** ist ein programmgenerierter Veranstaltungsschlüssel mit 2-? Zeichen der für die jeweilige Veranstaltung erstellt wird und zur Identifikation dieser dient. Die Zufälligkeit eines bei Veranstaltungsbeginn generierten Schlüssels sorgt zudem dafür, dass nur Personen, denen der Schlüssel im Rahmen der laufenden Veranstaltung mitgeteilt wird, an der Bewertung teilnehmen können. Dieses Schlüsselwort ist obligatorisch und sollte vor der Übermittlung an den Server vom Eingabemodul validiert werden. Nachrichten ohne oder mit einem falschen Veranstaltungsschlüssel werden vom System verworfen.
- ***$ID0-9** $ID steht für einen Buchstabencode, der für jede Installation von LectuRate festgelegt werden kann, aber sich innerhalb der Veranstaltungen einer Installation nicht unterscheidet. Der Buchstabencode steht für eines der Attribute, die zu jeder Installation definiert wurden und in den Veranstaltungen bewertet werden können.

  Im Beispiel aus dem dem deutschen Prototyp würde zum Beispiel der Buchstabe "G" oder "g" mit einer Ziffer im Bereich von 0 bis 9 im direkten Anschluss das Attribut

"Geschwindigkeit des Vortrags" bewerten. Eine komplette Wertung würde in diesem Fall zum Beispiel "*G5" lauten.

Die Ziffern stehen für die beiden Extrema des jeweiligen Attributes am Ende des Ziffernbereichs und für eine neutrale Wertung mit den Ziffern im Zentrum. Oben genanntes Beispiel steht für eine absolut neutrale Wertung der Vortragsgeschwindigkeit, während die Ziffern in Richtung des Extrema 1 für eine zu langsame Vortragsgeschwindigkeit und Ziffern in Richtung des Extrema 9 für eine zu schnelle Vortragsgeschwindigkeit stehen.

Die Codebuchstaben gelten zur besseren Gebrauchstauglichkeit für die Benutzer jeweils für eine Installation, in der protoypischen Implementierung von LectuRate sind drei Attribute vorgesehen, die bewertet werden können. Die Anzahl der Attribute ist im Serverprogramm nicht beschränkt und hängt von der Implementierung im Client ab. Obwohl keine technische Limitierung der Codebuchstabenlänge vorliegt, sollte im Hinblick auf Systeme mit einer Eingabebeschränkung, diese möglichst kurz gehalten werden. Die Verwendung dieses Schlüsselwortes ist optional und kann für jeden definierten Attributecode verwendet werden. Bei doppelten Vorkommen wird das Erste ausgewertet.

- ***$POLL** Ein weiteres Schlüsselwort, welches für jede LectuRate Installation festzulegen gil, ist für die Umfragen innerhalb der Veranstaltungen vorgesehen. Hierbei wird im direkten Anschluss eine Ziffer oder Zahl größer 1 angehängt, welche die gewünschte Wahlmöglichkeit repräsentiert. Dieses Schlüsselwort ist optional und es wird nur das erste Vorkommen ausgewertet.

5.5 Server

Das Server System gliedert sich in mehrere Bestandteile auf. Der Server verwaltet eine Warteschlange für eingehende Nachrichten und übermittelt diese an die zugehörige Veranstaltung. Er verwaltet alle laufenden Veranstaltungen sowie die Benutzer, die zur Erstellung von Veranstaltungen berechtigt sind. Zu Beginn startet er den Eingabemoduldienst, der sich um die Verwaltung der Eingabemodule kümmert.

5.5.1 Eingabemodule

Die Eingabe von Anmerkungen und Bewertungen an eine laufenden Veranstaltung soll über eine nicht beschränkte Anzahl von Diensten ermöglicht sein. Für jeden der unterstützten Dienste muss innerhalb des Serversystems eine Serviceklasse vorhanden sein, welche die jeweilige Kommunikation mit dem Dienst übernimmt und eingehende Nachrichten an den Server weiterleitet. In der Regel erfordern die einzelnen Dienste ein Nutzerkonto bei dem Dienstanbieter für jede LectuRate Installation. Die Eingabemodule sollten die in dem Abschnitt Nachrichtenaufbau 5.4.1 beschriebenen Elemente unterstützen. Als kleinster gemeinsamer Nenner ist zudem die Länge der Textnachricht auf 140 Zeichen beschränkt.

XMPP/Jabber

Um Kommentare über das XMPP Protokoll wie es in Saint-Andre (2004) zu ermöglichen, ist es erforderlich, je LectuRate Serverinstallation ein Nutzerkonto auf einem beliebigen XMPP Server zu erstellen. Es handelt sich nicht um das in Abschnitt (Client Kommunikation) beschriebene XMPP Konto. Im weiteren Abschnitt wird es Serveraccount genannt.

Der Serveraccount muss allen Nutzern des Systems vor der Verwendung bekannt sein. Die Nutzer schicken eine XMPP Nachricht an den Serveraccount und notieren innerhalb des Textes den Veranstaltungsschlüssel mit einem vorangestellten "#"-Zeichen. Optionale weitere Elemente sind die Schlüsselbegriffe für Umfragen oder für die Attributbewertungen. Text ohne eines der Schlüsselelemente wird als Kommentartext betrachtet. Die Zeichenbeschränkung reduziert den Kommentartext auf eine definierte Länge.

Das Modul antwortet dem Sender nach einer Überprüfung der Nachricht ob diese erfolgreich an den Server weitergeleitet wurde. Nachrichten ohne oder mit einem nicht vorhandenen Veranstaltungsschlüssel werden nicht an den Server weitergeleitet. Nachrichten, die in einer zu hohen Frequenz von einem Absender eingehen, werden wegen Spamverdacht abgelehnt.

E-Mail

Um eine E-Mail zur Eingabe von Kommentaren in das LectuRate System zu verwenden, ist es erforderlich, dass je LectuRate Installation ein E-Mail Eingangskonto vorhanden ist. Zur besseren Kommunikation mit den Benutzern ist ein Versandserver nach dem SMTP Standard empfehlenswert. Im weiteren Abschnitt wird dieses E-Mail Konto als Serverkonto bezeichnet.

Das Serverkonto muss allen Nutzern des Systems vor der Verwendung bekannt sein. Die Nutzer schicken eine E-Mail von einem beliebigen Konto an das Serverkonto und notieren innerhalb der E-Mail den Veranstaltungsschlüssel mit einem vorangestellten "#"-Zeichen. Optionale weitere Elemente sind die Schlüsselbegriffe für Umfragen oder für die Attributbewertungen. Text ohne eines der Schlüsselelemente wird als Kommentartext betrachtet. Die Zeichenbeschränkung reduziert den Kommentartext auf eine definierte Länge.

Das Serverkonto wird in einem festen Zeitintervall überprüft und die eingehenden Nachrichten ausgewertet. Je nach Nachrichteninhalt generiert das System eine E-Mail Antwort und sendet diese an den Urheber der eingehenden Nachricht zurück.

Nachrichten ohne oder mit einem nicht vorhandenen Veranstaltungsschlüssel werden nicht an den Server weitergeleitet und erzeugen eine entsprechende E-Mail Antwort. Nachrichten, die einer Veranstaltung zugeordnet werden konnten, werden an den Server weitergeleitet und erzeugen eine E-Mail Antwort mit einer Bestätigung des Eingangs. E-Mail Nachrichten, die in einer zu hohen Frequenz von einem Absender eingehen, werden wegen Spamverdacht abgelehnt.

Twitter.com

Twitter ist ein Microblogsystem, bei dem Benutzer über eine kurze Nachricht mit maximal 140 Zeichen einen aktuellen Status mitteilen können. Jeder Benutzer kann anderen Benutzern des Systems "folgen", d.h. er bekommt eine Benachrichtigung über Aktualisierungen des jeweiligen Benutzers. Ebenso können andere Nutzer den eigenen Aktualisierungen "folgen". Um andere Benutzer in seinen eigenen Aktualisierungen anzusprechen oder zu markieren, wird dem jeweiligen Benutzernamen ein "@"-Symbol vorangestellt. Nachrichten, in denen ein Benutzer in dieser Weise erwähnt wird, werden dem Angesprochenen in einer eigenen Kategorie angezeigt.

Um eine Nachricht an das LectuRate-System zu senden, ist es erforderlich, dass je LectuRate Serverinstallation ein eigenes Nutzerkonto bei Twitter existiert. Dieser Benutzername wird im weiteren Abschnitt als Serveraccount bezeichnet. Der Serveraccount muss bei allen Nutzern, die mit dieser LectuRate Installation interagieren möchten, bekannt sein. Jeder Nutzer benötigt zusätzlich ein eigenes Konto bei Twitter.

Um einen Kommentar zur laufenden Veranstaltung abgegeben zu können, wird nun von dem Konto der Nutzer eine Nachricht geschrieben, die zwingend durch ein vorangestelltes "@" den Serveraccount anspricht und den Veranstaltungsschlüssel mit einem vorangestellten "#"-Zeichen enthält. Optionale weitere Elemente sind die Schlüsselbegriffe für Umfragen oder für die Attributbewertungen. Text ohne eines der Schlüsselelemente wird als Kommentartext betrachtet.

Der Serveraccount wird in einem festen Zeitintervall überprüft und die eingehenden Nachrichten ausgewertet. Nachrichten, die in einer zu hohen Frequenz von einem Absender eingehen, werden wegen Spamverdacht abgelehnt. Auf Grund von dienstseitigen Zugriffsbeschränkungen werden keine Antworten an die Absender generiert. Die dienstseitige Nachrichtenlängenbeschränkung von 140 Zeichen erfordert keine weitere Reduktion vom LectuRate System.

Identi.ca / Status.net

Wie bei Twitter handelt es sich bei Status.net beziehungsweise der Referenzinstallation Identi.ca um ein sogenanntes Microblogsystem. Die API wurde der Twitter-API nachempfunden, dementsprechend funktioniert dieses Modul genau wie sein in Abschnitt 5.5.1 beschriebener Pendant.

Representational State Transfer (REST)

Beim Representational State Transfer[1], im weiteren Verlauf als REST bezeichnet, handelt es sich um eine Architekturstil für verteilte System, bei dem jede Resource mittels einer URL

[1] REST, beschrieben in der Dissertation von Roy Fielding aus dem Jahr 2000, http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

angesprochen werden kann. Hierzu werden in der Regel die Methoden GET, POST, PUT und DELETE aus dem HTTP-Standard verwendet.

Das REST Modul aus dem LectuRate Client ermöglicht in der jetzigen Form die Übertragung eines Kommentars, die Prüfung eines Veranstaltungsschlüssels und die Rückgabe einer Liste der Veranstaltungsschlüssel aller laufenden Veranstaltungen.

Die genauen URLs zur Verwendung der REST-Schnittstelle finden sich im Anhang in Abschnitt 10.6.

5.6 Client

Der entwickelte LectuRate-Client ist eng an den Entwürfen des Paper-Based-Prototyping aus Abschnitt 10.2 angelehnt.

5.6.1 Attributsymbole

Bei der prototypischen Umsetzung des LectuRate Systems wurden für die drei exemplarisch verwendeten Attribute, Symbole zur einfachen Identifizierung entwickelt.

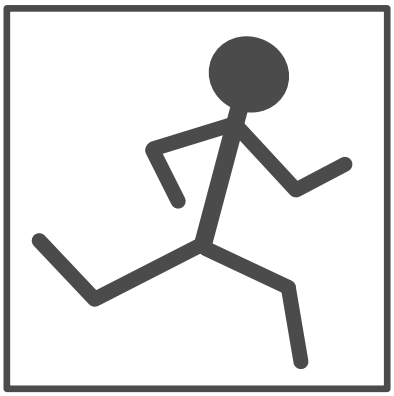

Abbildung 5.2: Vortragsgeschwindigkeit

Abbildung 5.3: Vortragslautstärke

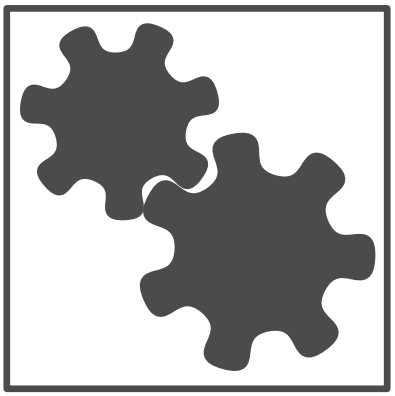

Abbildung 5.4: Vortragsverständlichkeit

6 Abgrenzung des Prototypen

In Abgrenzung des Prototyps zu einer marktreifen Softwarelösung sind einige Funktionen aus dem Konzept des Systems nicht implementiert worden.

6.1 Server

Im Servermodul gelten folgende Einschränkungen bezüglich der konzeptuellen Funktionalitäten und der tatsächlich implementierten Funktionalitäten.

6.1.1 Short Message Service (SMS)

Die SMS gehört von der Verbreitung (QUELLE) und Zugangsmöglichkeit zu der am weitesten verbreiteten Eingabemethode und wurde trotzdem nicht im Prototypen implementiert, da es sich um die einzige kostenverursachende Methode handelte.

6.1.2 REST-Anbindung zum Client

Die Möglichkeit, statt des LectuRate Clients über eine REST-Schnittstelle mit dem Server zu kommunizeren wurde im Prototypen nicht umgesetzt.

6.1.3 REST verwendet HTTP GET statt POST

Obwohl der HTTP POST Befehls nach Definition des RESTful Webservice bei einer Datenübermittlung verwendet werden soll, wurde aus Aufgrund von technischen Problemen die Übermittlung eines neuen Kommentars mittels HTTP GET durgeführt. Näheres zu möglichen Problemen in diesem Zusammenhang findet sich im Anhang zu des REST-Ressourcen (10.6).

6.1.4 Persistierung von Veranstaltungen

Sämtliche erstellte Veranstaltungen gelten nur für die Dauer der Ausführung einer Serverinstanz. Es ist keine Möglichkeit zur Persistierung der Daten vorhanden.

6.1.5 Einschränkung des E-Mail Eingabemoduls

Das E-Mail Modul erfordert im Prototypen die Zugangsmöglichkeit mittels IMAP Protokoll, POP3 wird nicht unterstützt. Es ist nicht möglich, Nachrichten in den BODY-Teil einer E-Mail zu schreiben, es wird nur der Text im Subject-Teil als Nachricht ausgewertet.

6.2 Client

Im Client wurden die nachfolgenden Funktionalitäten aus dem Konzept nicht implementiert.

6.2.1 Vollbildoberfläche

Die Vollbildoberfläche wurde durch die limitierten technischen Möglichkeiten der SWING Oberflächenflächebibliothek im Protoypen unmöglich gemacht. Es ist keine Möglichkeit zur getrennten Ansteuerung vom Bildschirm des Vortragenden und des Projektionsgerätes vorhanden.

6.2.2 Umfragewerkzeug

Die Einrichtung einer oder mehrerer Umfragen zu einer Veranstaltung wurde nicht implementiert.

6.2.3 Persistierung der Veranstaltungen

Es ist keine Möglichkeit vorhanden, eine laufende Veranstaltung oder deren Ergebnisse innerhalb des Clients abzuspeichern oder abgespeicherte Veranstaltungen erneut zu öffnen.

6.2.4 Statistiken

Es existiert keine statistische Auswertung zu den laufenden Veranstaltungen mit genaueren Informationen über die eingehenden Ereignisse.

7 Realisierung

Das Kapitel Realisierung befasst sich mit den Problemen bei der Erstellung des Prototypes.

7.1 Probleme

Folgende Probleme konnten für den Prototyp innerhalb des gesteckten Zeitrahmens nicht zufriedenstellend gelöst werden und benötigten weitere zeitliche Resourcen oder fremdes Fachwissen zum Thema.

7.1.1 Vollbildoberfläche

Die Vollbildoberfläche wurde durch die limitierten technischen Möglichkeiten der Java Swing Oberflächenflächebibliothek im Protoypen unmöglich gemacht. Es ist keine Möglichkeit zur getrennten Ansteuerung vom Bildschirm des Vortragenden und des Projektionsgerätes vorhanden.

Lösungsstrategie

Es muss eine andere Oberflächebibliothek verwendet werden, welche die benötigten Fähigkeiten mitbringt.

7.1.2 Übermittlung neuer Bewertungen

Bislang wird eine neue Attributbewertung sowohl im Client als auch auf dem Server von der Veranstaltungsklasse berechnet. Die Berechnungsformel ist somit redundant in zwei Komponenten vorhanden. Das kann je nach Umsetzung und Versionierung zu inkonsistenten Attributwerten führen.

Lösungsstrategie

Die Berechnung sollte nur in dem Veranstaltungsobjekt auf dem Server erfolgen. Dort muss dann eine Wertaktualisierung für die Clients angestossen werden. Bei der aktuellen Client Lösung hieße das, eine zusätzliche XML Nachricht mit jeweils den absoluten Werten zu

verschicken und im Client darzustellen. Hierzu müsste die bisherige Übermittlung des in XML serialisierten Kommentars überarbeitet werden.

7.1.3 Datenübermittlung mittels HTTP POST

Trotz umfangreicher Recherche war es im Entwicklungsprozess mit dem eingesetzten Java Standardframework für JAX-RS Oracle (2010b) nicht möglich, mittels HTTP POST übermittelte Nutzdaten innerhalb des Programms zu verarbeiten, während die Kopfdaten problemlos verwertet werden konnten.

Lösungsstrategie

Weitere Recherchen zum Thema oder die Verwendung einer anderen JAX-RS Implementierung.

7.1.4 Verbindung zum XMPP Server

Obwohl die Verbindung über das gewählte XMPP Toolkit Smack (JiveSoftware (2008)) mittels vielfältiger Einstellungsmöglichkeiten mit unterschiedlichsten XMPP/Jabberserver hergestellt werden kann, war es in den Versuchen nicht möglich, eine Verbindung mit einem Server der 1und1 Internet AG aufzubauen. Mit neueren Versionen der dort eingesetzten Serversoftware und anderer Software funktioniert die Verbindung allerdings.

Lösungsstrategie

Da alle getesteten XMPP/Jabber Messenger die Verbindung zu den Servern herstellen können, bleibt zu ermitteln, ob die Problematik dem verwendeten Smack Framework geschuldet ist oder durch eine andere Konfiguration gelöst werden kann.

7.1.5 Serversoftware als .jar-Archiv verbreiten

Auf Grund der eigenwilligen Konfiguration des Twitter/Identi.ca Moduls ist es bislang nicht möglich, den Server korrekt aus einer komprimierten .jar-Datei zu benutzen. Das Modul versucht die Anmeldedaten aus zwei .property Dateien einzulesen, die vom Server bei der Initialisierung mit den Daten aus der Konfigurationsdatei geschrieben werden. Dieser Schreibvorgang funktioniert nicht bei einer Ausführung mit dem .jar-Archiv

Lösungsstrategie

Das Modul überarbeiten, so dass die Anmeldedaten direkt aus der Konfiguration des Servers gelesen werden. Alternativ wäre eine Dekomprimierung, Modifizierung und anschliessende, erneute Komprimierung der Daten denkbar.

8 Ausblick

Neben einer vollständigen Implementierung der Konzeptbestandsteile, die im Prototyp nicht berücksichtigt wurden, bietet die Thematik auch weitere Ansatzpunkte, die nicht weiter im Rahmen dieser Arbeit vertieft wurden.

8.1 Nicht erfüllte Anforderungen des Prototyps

Die folgenden Anforderungen konnten durch die prototypische Implementierung nicht erfüllt werden und sind deßhalb ein erster Ansatzpunkt für weiterführende Aufgabenstellungen.

- Die Ergebnisse einer Umfrage können nicht eingesehen werden.
- Die Darstellung kann nicht nur an den Primärbildschirm gekoppelt werden.
- Es gibt keine Hilfefunktion.
- Nicht alle Eingabemodule geben den Benutzern eine Rückmeldung.
- Die Verwaltungsoberfläche umfasst nur eine Auswahl von Aufgaben.
- Es gibt keine Endbenutzerdokumentation der Schnittstellen.
- Eine Verschlüsselung der Übertragung ist noch nicht implementiert.

9 Evaluation

Da die theoretische Abhandlung des Themas erst in der anschließenden Bachelorthesis behandelt wird, fällt auch die Evaluation zum Praxisprojekt kurz aus.

9.1 Evaluation des Prototypen

Die Evaluationsphase beinhaltete offene Vorträge mit einer Vorstellung des Prototypen im Umfeld der Medieninformatik sowie eine detailierte Vorstellungen des Projekts bei den Projektbetreuern und Privatpersonen. Innerhalb dieser Phase wurden die folgenden Punkte oder Details der Implementierung ermittelt, die Spielraum für verbesserte Lösung beinhalten.

- Bewertungsmöglichkeiten von Attributen, die sich nicht mit einer Skala ausdrücken lassen.
- Algorithmus zur zeitabhängigen Gewichtung der eingehenden Bewertung in der Visualisierung.

Literaturverzeichnis

[JiveSoftware 2008] JiveSoftware: *Smack Documentation*. 2008. – URL http://www.igniterealtime.org/builds/smack/docs/latest/documentation/

[Kent Beck 2010] Kent Beck, et a.: *JUnit Java Test-Framework*. 2010. – URL http://www.junit.org/

[Matt Harris 2010] Matt Harris, et a.: *Twitter API Wiki*. 2010. – URL http://apiwiki.twitter.com/

[Miles 2010] Miles, Stephanie: *Poll Everywhere - Interact With Your Audience in Real Time*. März 2010. – URL http://www.appvita.com/2010/03/19/poll-everywhere-interact-with-your-audience-in-real-time/

[Oracle 2009] Oracle, Sun: *JavaMail API Specifications*. 2009. – URL http://java.sun.com/products/javamail/

[Oracle 2010a] Oracle, Sun: *Creating a GUI With JFC/Swing*. 2010. – URL http://java.sun.com/docs/books/tutorial/uiswing/index.html

[Oracle 2010b] Oracle, Sun: *JAX-RS: Java API for RESTful Web Services*. 2010. – URL https://jsr311.dev.java.net/

[PollEverywhere 2010] PollEverywhere: *Poll Everywhere Website*. März 2010. – URL http://www.polleverywhere.com/

[Saint-Andre 2004] Saint-Andre, Peter: *RFC 3920: Extensible Messaging and Presence Protocol (XMPP)*. Internet Engineering Task Force (Veranst.), Oktober 2004. – URL http://www.ietf.org/rfc/rfc3920.txt

10 Anhang

Im Anhang befinden sich neben der Projektplanung auch unterschiedliche Artefakte aus der Entwicklung des Prototypen, sowie eine Installationsanweisung.

10.1 Projektplan

Ausgehend von 15 Creditpoints wird in der vorläufige Projektplanung mit einem Zeitaufwand von 450 Stunden (30 Std/1Credit) kalkuliert.

Meilenstein I: Ideenfindung/Vorbereitung

Typ	Beginn	Ende	Zeit[1]	Erfüllungsgrad
Vorbereitung der Arbeitsumgebung	23.03.10	23.03.10	4	100%
Erstellung Expose	23.03.10	28.03.10	3	100%
Anpassung Idee	23.03.10	30.03.10	6	100%
Projektplanung Praxisprojekt	23.03.10	29.06.10	7	100%

Meilenstein II: Markterkundung

Typ	Beginn	Ende	Zeit[1]	Erfüllungsgrad
Internetrecherche	30.03.10	03.04.10	16	100%
Hochschulinterne Recherche	30.03.10	05.04.10	4	100%

Meilenstein III: Systemdesign

Typ	Beginn	Ende	Zeit[1]	Erfüllungsgrad
Festlegung der Architektur	25.03.10	28.03.10	19	100%
Konzept-/Screendesign Client	14.04.10	11.05.10	24	100%
Konzept Server	30.03.10	28.04.10	27	100%
Entwurf XML Stanzas	30.03.10	20.05.10	12	100%
Entwurf Datenklassen	30.03.10	15.05.10	22	100%
Entwurf der Nachrichten	30.03.10	02.04.10	8	100%

Meilenstein IV: Prototyp

Typ	Beginn	Ende	Zeit[1]	Erfüllungsgrad
Anbindung der Dienste	05.04.10	27.04.10	44	100%
Eingabemodul Daemon	16.04.10	27.04.10	13	100%
Nachrichtenwarteschlange	18.04.10	21.04.10	4	100%
Spamfilter	17.04.10	22.04.10	7	100%
Kommunikation Server-Client	20.04.10	02.05.10	9	100%
GUI Programmierung	05.04.10	27.04.10	23	100%
Fehlerbehebungen	25.04.10	30.05.10	38	100%
Webseite	18.05.10	20.05.10	14	100%

Meilenstein V: Technisches Konzept

Typ	Beginn	Ende	Zeit[1]	Erfüllungsgrad
Festlegung der Anforderungen	07.04.10	20.04.10	18	100%
Ausformulierung des Konzepts	17.05.10	20.06.10	22	100%
Beschreibung der Systemarchitektur	28.05.10	20.06.10	13	

Meilenstein VI: Praxistest, Evaluierung

Typ	Beginn	Ende	Zeit[1]	Erfüllungsgrad
Entwurf Umfragen	06.06.10	15.06.10	9	100%
Durchführung und Auswertung	17.06.10	07.07.10	15	100%
Vorbereitung u. Durchführung Probelauf	02.06.10	02.06.10	5	100%

Meilenstein VII: Dokumentation

Typ	Beginn	Ende	Zeit[1]	Erfüllungsgrad
Erstellung der Dokumentstruktur	02.06.10	06.06.10	3	100%
Festlegung des Inhalts	08.06.10	08.06.10	5	100%
Dokumentation	08.06.10	29.06.10	55	100%
Präsentation	25.06.10	06.07.10	7	100%

[1]Der kalkulierte Zeitaufwand für diese Tätigkeit in Stunden

10.2 Paper Based Prototyping

Paper Based Prototyping ist eine schnelle und effiziente Methode, um Benutzerschnittstellen ohne Programmieraufwand zu modellieren und mit echten Nutzern zu testen.

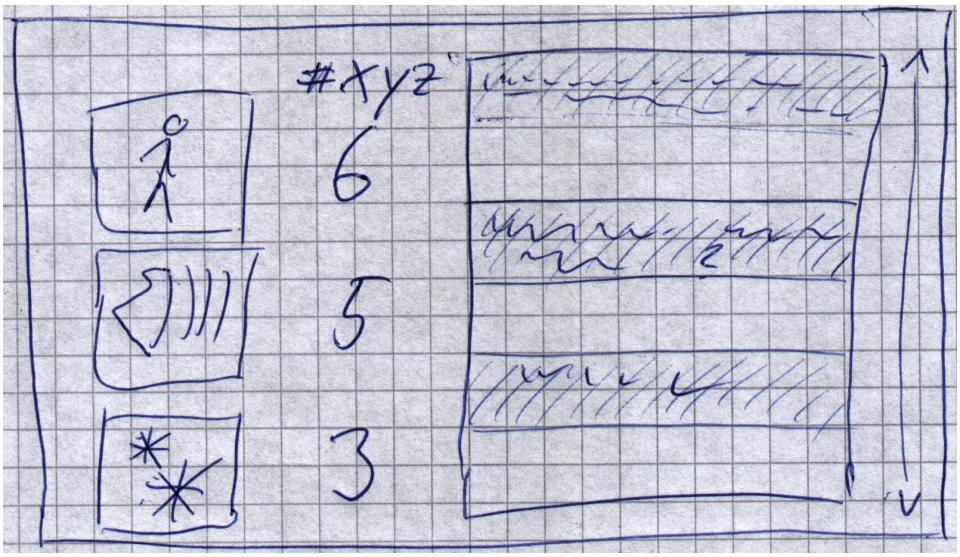

Abbildung 10.1: Konzept: Papierentwurf der Client-Oberfläche

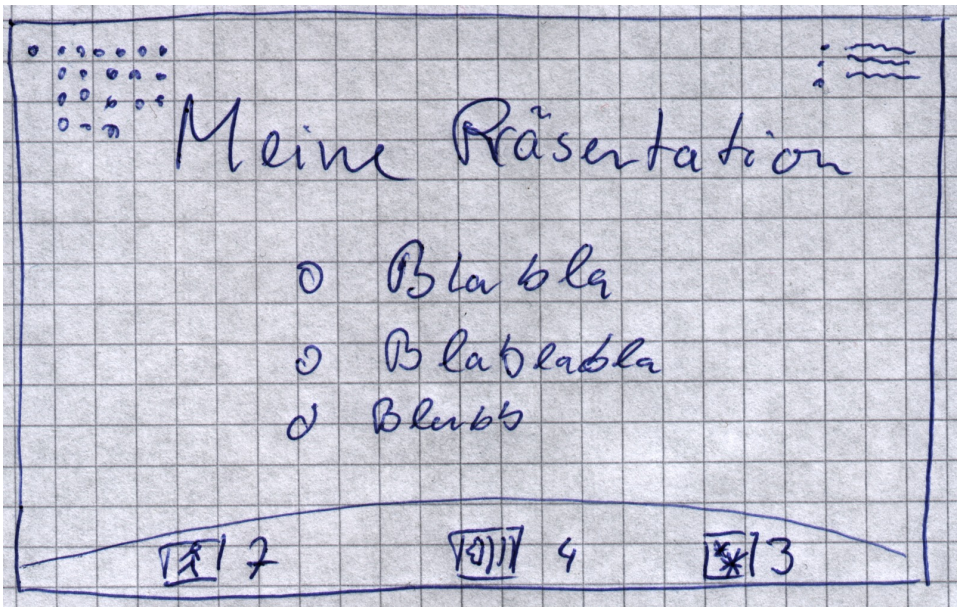

Abbildung 10.2: Konzept: Papierentwurf der Client-Vollbildoberfläche

10.3 Bildschirmfotos

Im Bereich der Bildschirmfotos sind einerseits die wichtigsten Fenster des Clientsystems für die Dozierenden zu sehen und andererseits auch die wichtigsten Eingabemodule für die Benutzer dargestellt.

10.3.1 Client

Der Prototyp-Client besteht aus dem Hauptfenster, einem Einstellungsdialog, einem Fenster mit Loggingausgaben zur Funktionskontrolle, sowie einigen Dialogfenstern.

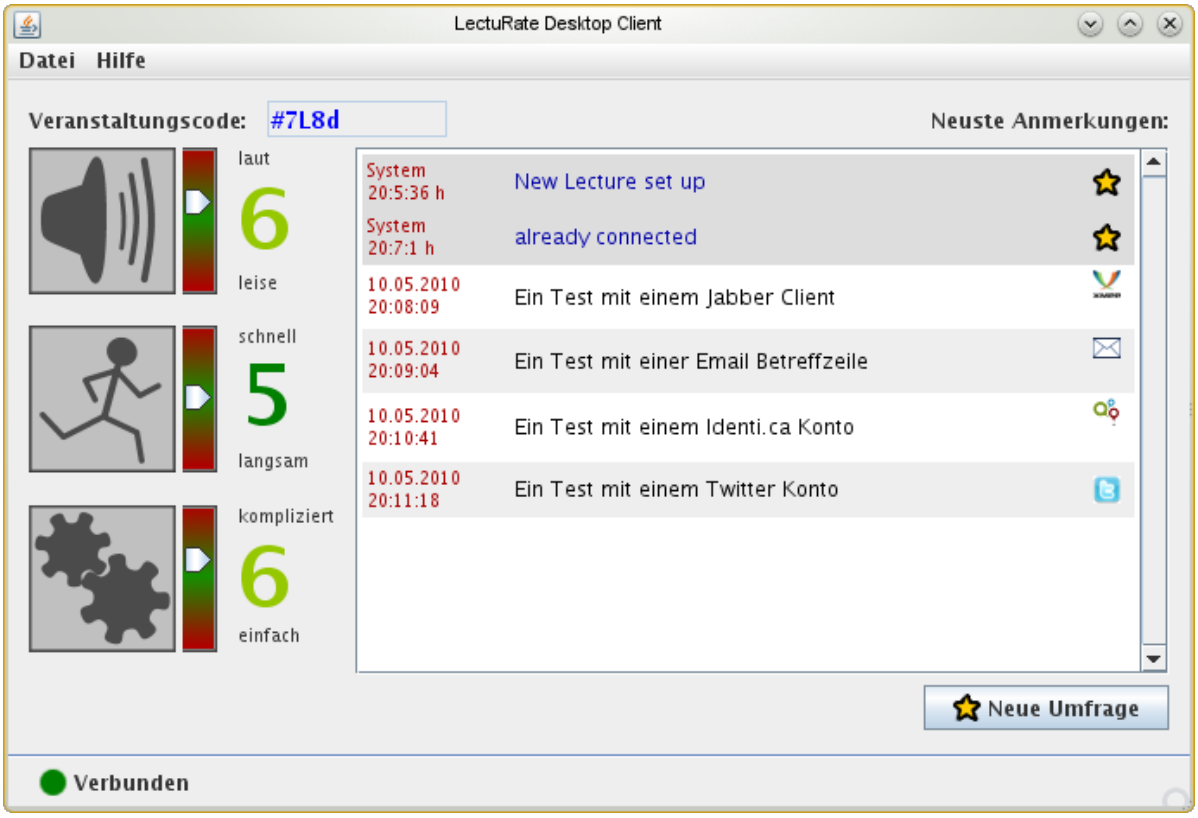

Abbildung 10.3: LectuRate-Client: Hauptprogrammfenster

Abbildung 10.4: LectuRate-Client: Einstellungsdialog

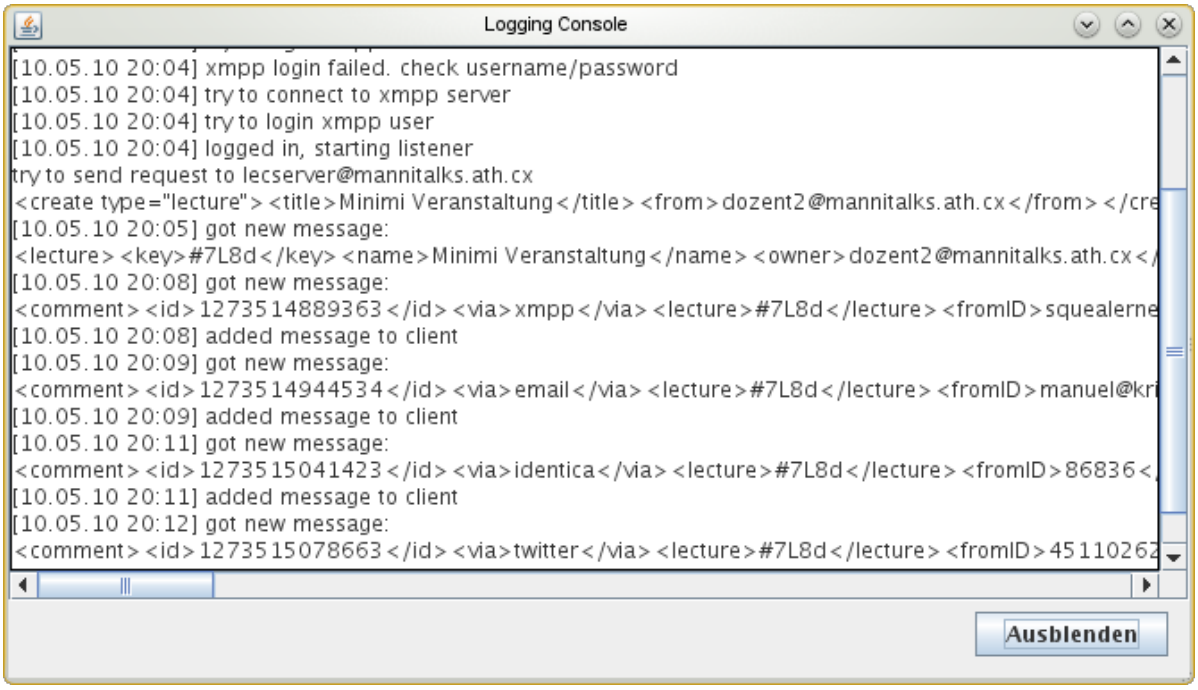

Abbildung 10.5: LectuRate-Client: Programmmeldungsfenster

10.3.2 Eingabemodule

Bei den Eingabemodulen finden sich die wichtigsten Implementierungen des Prototyps zur Eingabe für die Benutzer in einer beispielhaften Anwendung.

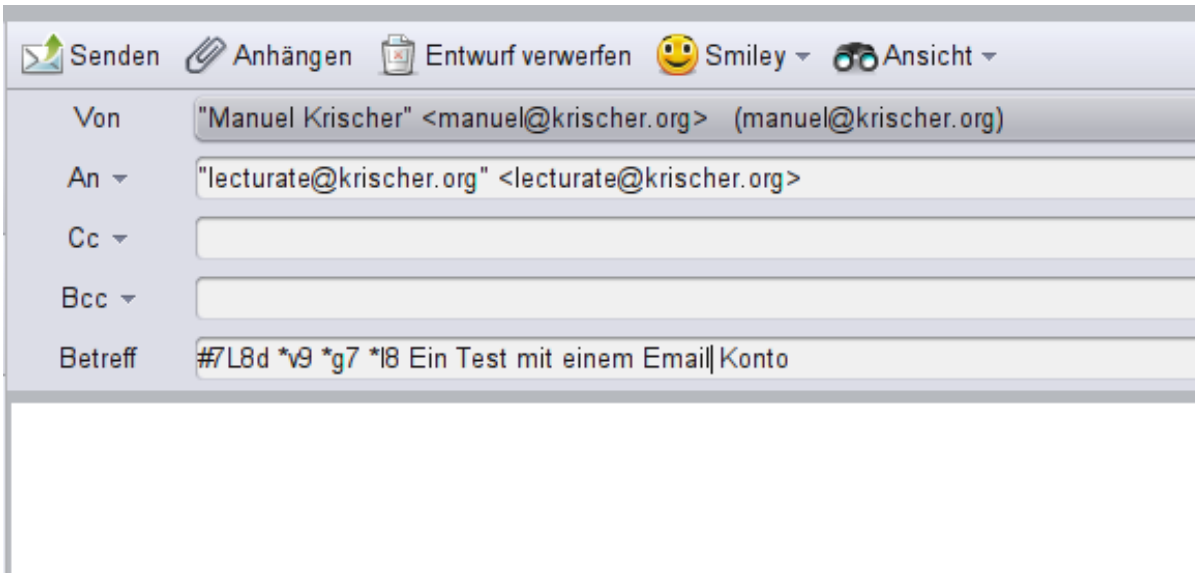

Abbildung 10.6: LectuRate Eingabe: E-Mail Nachricht

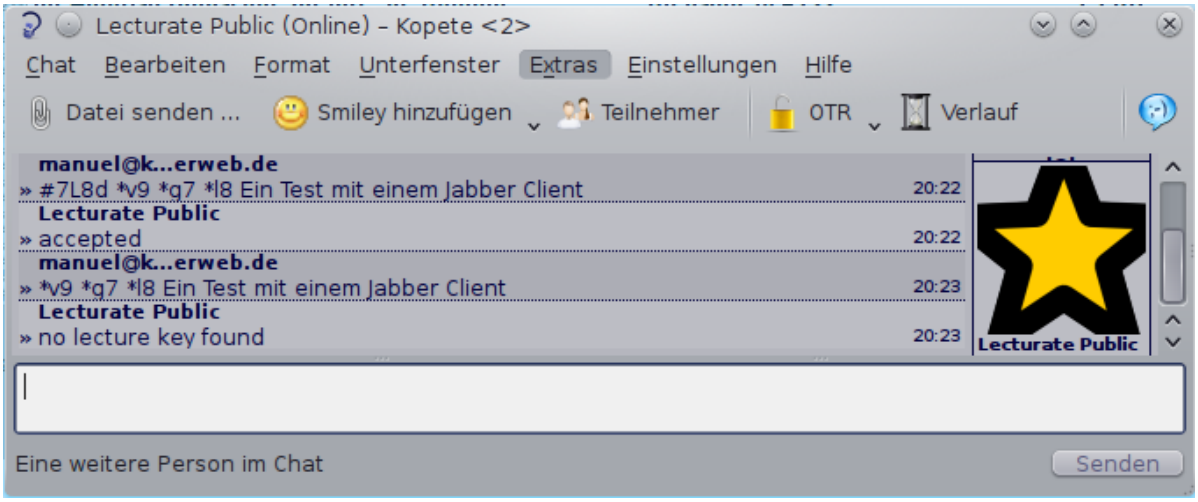

Abbildung 10.7: LectuRate Eingabe: XMPP/Jabber Nachricht

10.4 Installationsanleitung

Die Installationsanleitung enthält alle nötigen Informationen zur Inbetriebnahme des LectuRate Software Prototypen.

Abbildung 10.8: LectuRate Eingabe: Twitter Nachricht

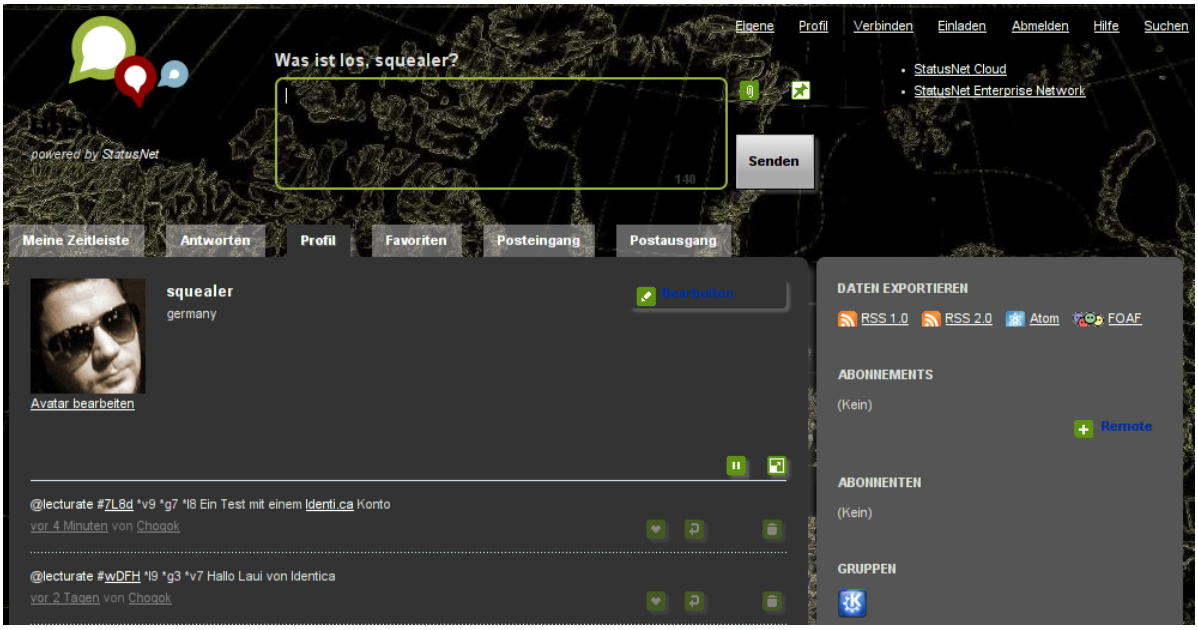

Abbildung 10.9: LectuRate Eingabe: Identi.ca Nachricht

10.4.1 Server

Der Server wird in einem Zip-Archiv geliefert, das nach der Dekomprimierung in einen Ordner die folgenden Bestandteile aufweisen sollte:

- /lib
  - TwitterResponse.jar
  - activation.jar
  - asm-X.X.jar
  - jackson-core-asl-X.X.X.jar
  - jaxb-api.jar
  - jersey-client-X.X.X.X.jar
  - jersey-core-X.X.X.X.jar
  - jersey-json-X.X.X.X.jar
  - jersey-server-X.X.X.X.jar
  - jersey-spring-X.X.X.X.jar
  - jettison-X.X.jar
  - jsr173_api.jar
  - jsr311-api-X.X.X.jar
  - junit-X.X.jar
  - mail.jar
  - oauth-client-X.X.X.X.jar
  - oauth-signature-X.X.X.X.jar
  - smack.jar
  - smackx.jar
- LecturateServer.jar
- README.TXT
- config-server.xml

Wenn die Datei `config-server.xml` nicht existiert, wird der Programmstart verweigert. Zuerst muss eine Datei mit dem Inhalt der in Abschnitt 10.5.1 dargestellten Konfiguration erstellt werden. Für jedes Eingabemodul muss ein Benutzerkonto bei dem entsprechenden Dienst vorhanden sein und die Benutzerdaten vorliegen. Zusätzlich ist es nötig, ein XMPP/Jabber Konto[1] für die Kommunikation mit den Clientprogrammen einzurichten.

[1] Siehe dazu die Hinweise im Abschnitt 7.1.4

Gestartet wird der Server mit dem Befehl `java -jar LecturateServer.jar` auf einer Eingabeaufforderung des Betriebssystem, oder über die automatische Zuordnung des Dateitypes .jar mit der Java-Ausführungsumgebung. Für den Betrieb wird eine Ausführungsumgebung von Java 1.6 oder neuer empfohlen. Für die Eingabemodule "Twitter" und "Identi.ca" gelten die Einschränkungen, die unter Abschnitt 7.1.5 beschrieben sind.

10.4.2 Client

Der Client wird in einem Zip-Archiv geliefert, das nach der Dekomprimierung in einen Ordner die folgenden Bestandteile aufweisen sollte:

- /lib
  - appframework-X.X.X.jar
  - smack.jar
  - smackx.jar
  - swing-worker-X.X.jar
- README.TXT
- LecturateClient.jar
- clientConfig.xml [Optional]

Wenn die Datei `clientConfig.xml` nicht existiert, wird sie beim ersten Start des Programms erstellt und muss dann über die Konfigurationsoberfläche oder direkt mit einem Texteditor bearbeitet werden. Der Inhalt der Datei ist in Abschnitt 10.5.2 dargestellt.

Benötigt für die Verwendung wird ein vom Programm unterstützter XMPP/Jabber Account [2], der im Server als authorisierter Account eingetragen ist, sowie die Kenntnis des XMPP/Jabber Accounts, der für die Kommunikation mit der Server konfiguriert ist.

Gestartet wird der Client mit dem Befehl `java -jar LecturateClient.jar` auf einer Eingabeaufforderung des Betriebssystem, oder über die automatische Zuordnung des Dateitypes .jar mit der Java-Ausführungsumgebung. Für den Betrieb wird eine Ausführungsumgebung von Java 1.6 oder neuer empfohlen.

[2] Siehe dazu die Hinweise im Abschnitt 7.1.4

10.5 Konfiguration

Die Konfiguration der LectuRate Komponenten erfolgt über eine XML Struktur. Während der Server nur über die direkte Bearbeitung einer Datei konfiguriert werden kann, bietet der Client auch eine grafische Oberfläche zur Unterstützung der Benutzer.

10.5.1 Server Konfiguration

Die Serverkonfiguration beinhaltet die Benutzerkennungen für die einzelnen Eingabemodule sowie grundlegende Einstellungen wie die Länge eines Veranstaltungsschlüssels oder die Attributkonfiguration.

```
<?xml version="1.0" encoding="UTF-8"?>
<lecturate>
    <service type="microblog" status="denabled">
        <name>Twitter</name>
        <url>http://www.twitter.com</url>
        <username>username</username>
        <password>passwort</password>
        <ssl>boolean</ssl>
    </service>
    <service type="microblog" status="denabled">
        <name>Identi.ca</name>
        <url>http://www.identi.ca/api/</url>
        <username>username</username>
        <password>passwort</password>
        <ssl>boolean</ssl>
    </service>
    <service type="rest" status="enabled">
        <url>localnetworkaddrees</url>
        <port>port</port>
        <path>/lecturate</path>
    </service>
    <service type="email" status="enabled">
        <name>mailaddress</name>
        <imapserver>imapserver</imapserver>
        <imapport>imapport</imapport>
        <imapauth>authformat</imapauth>
        <imapuser>imapuser</imapuser>
        <imappass>imappasswort</imappass>
        <imaptls>boolean</imaptls>
        <smtpserver>smtpserver</smtpserver>
        <smtpport>smtpport</smtpport>
```

```
32          <smtpauth>authformat</smtpauth>
33          <smtpuser>smtpusername</smtpuser>
34          <smtppass>smtppasswort</smtppass>
35          <smtptls>boolean</smtptls>
36      </service>
37      <service type="xmpp" status="enabled">
38          <name>accountname</name>
39          <username>xmppusername</username>
40          <password>xmpppasswort</password>
41          <url>xmppurl</url>
42          <port>xmppport</port>
43      </service>
44      <attribute id="1">
45          <key>l</key>
46          <name>Lautstaerke</name>
47          <high>zu laut</high>
48          <low>zu Leise</low>
49          <init>5</init>
50      </attribute>
51      <attribute id="2">
52          <key>g</key>
53          <name>Geschwindigkeit</name>
54          <high>zu schnell</high>
55          <low>zu langsam</low>
56          <init>5</init>
57      </attribute>
58      <attribute id="3">
59          <key>v</key>
60          <name>Verstaendlichkeit</name>
61          <high>zu komplex</high>
62          <low>zu simpel</low>
63          <init>5</init>
64      </attribute>
65      <config>
66          <hashkey length="Integer" />
67          <frequency poll="Integer" />
68          <xmpp>
69              <username>xmppaccount</username>
70              <password>xmpppassword</password>
71              <url>xmppurl</url>
72              <port>xmppport</port>
73          </xmpp>
74      </config>
75  </lecturate>
```

10.5.2 Client Konfiguration

Die Clientkonfigration enthält die Zugangsdaten des XMPP-Zugangs, den XMPP Kontakt zur Kommunikation mit dem Server und einfache Farbcodes für die graphische Oberfläche.

```
<lecturate>
  <account>
    <name>accountname</name>
    <username>xmppusername</username>
    <password>xmpppasswort</password>
    <url>xmppurl</url>
    <port>5222</port>
  </account>
  <server>
    <xmppID>xmppid</xmppID>
  </server>
  <interface>
    <color id="5">#008000</color>
    <color id="6">#96c800</color>
    <color id="7">#c88000</color>
    <color id="8">#c87d00</color>
    <color id="9">#ff0000</color>
  </interface>
</lecturate>
```

10.6 RESTful Schnittstellenabfrage

An dieser Stelle finden sich die Abfrage-URLs des RESTful Eingabemoduls sowie deren Antwort im XML Format.

10.6.1 Abfrage gültiger Veranstaltung

Da es bei der Eingabe des Veranstaltungsschlüssels mit dem Raute-Zeichen "#" innerhalb einer URL Probleme geben kann, ist es möglich, den Schlüssel ohne führendes "#" abzufragen. Alternativ muss das Zeichen in einer gültigen Umschreibung für HTTP (%23) angegeben werden.

`$SERVERNAME:$PORT:/$PFAD/key/$SCHLÜSSEL`

Rückgabewert des System

Der Wert des Elements "exist" beinhaltet "true" oder "false", je nach Vorhandensein.

```
1 <lecture>
2 <key>#SCHLUESSEL</key>
3 <exist>boolean</exist>
4 </lecture>
```

10.6.2 Eingabe eines Kommentars

`$SERVERNAME:$PORT:/$PFAD/comment/add?$STRING`

Der $STRING setzt sich aus folgenden Parametern zuzüglich den jeweiligen Werten zusammen und muss in URL üblicher Weise mittels einer "key=value" Notierung erfolgen:

lk Veranstaltungsschlüssel

un Benutzername

uid BenutzerID

txt Mitteiltungstext

a1 Attributwert 1

a2 Attributwert 2

a3 Attributwert 3

Rückgabewert des System

Bei erfolgreicher Übermittlung eines Kommentars

```
<lecturate>
  <code>200</code>
  <text>comment submitted</text>
</lecturate>
```

Bei falschem Veranstaltungsschlüssel

```
<error>
  <code>404</code>
  <text>lecture key unkown</text>
</error>
```

Bei fehlenden benötigten Daten

```
<error>
  <code>406</code>
  <text>could not build comment</text>
  <detail>missing parameter</detail>
</error>
```

Bei sonstigen Problemen

```
<error>
  <code>500</code>
  <text>could not build comment</text>
</error>
```

Beim Aufruf der URL mittels einer nicht unterstützten HTTP Methode (hier POST)

```
<error>
  <code>405</code>
  <text>POST not supported</text>
</error>
```

Beim Aufruf einer URL ohne Resource

```
<error>
  <code>501</code>
  <id>GET with unknown path</id>
  <text>GET $path</text>
</error>
```

Mögliche Probleme

Problematisch ist bei der Verwendung des HTTP GET Methode die URL Länge, die durch diese Anfrage erzeugt wird. Während die meisten Browser bis ca. 2000 Zeichen keine Probleme bereiten, sind manche Proxies und Webserver mit URLs länger als 255 Zeichen überfordert.

Ein Kommentar mit Ausnutzung aller Zeichen im Kommentar zuzüglich der Attribute würde die Gesamtlänge von 255 Zeichen überschreiten. Zu einer weiteren Vergrößerung kann es durch längere Benutzernamen, Benutzer-IDs, Serverpfade oder Veranstaltungsschlüssellänge kommen. Auch alle weiteren Sonderzeichen verlängern die URL durch ihre HTML-konforme Umschreibung. Da die Zahl der möglichen betroffenen Geräte und Softwareversionen immer weiter schrumpft und der RFC 2616 (Hypertext Transfer Protocol HTTP/1.1) section 3.2.1 keine maximale Länge für URLs vorschreibt, wird die Funktion nicht eingeschränkt.

10.7 XML Stanzas

Nachfolgend die Nachrichtenfragmente in XML Form, die zur Kommunikation zwischen den LectuRate Komponenten verwendet werden.

10.7.1 Kommentar

Das folgende Fragment ist die Serialisierung eines Kommentarobjekts.

```
<comment>
  <id>DATE.getTime()</id>
  <via>$SERVICENAME</via>
  <lecture>#LectureKey</lecture>
  <fromID>$USERID</fromID>
  <fromName>USERNAME</fromName>
  <text>PLAINTEXT</text></comment>
  <attribute key="$KEYCODE" value="0-9" /> ::
</comment>
```

10.7.2 Erstellung einer Veranstaltung

Die Anfrage an den Server zur Erstellung einer neuen Veranstaltung.

```
<create type="lecture">
  <title>$Name der Veranstaltung</title>
  <from>$JabberID eines berechtigten Users</from>
</create>
```

10.7.3 Neue Veranstaltung

Eine Veranstaltung ohne Kommentare als XML serialisiert.

```
<lecture>
  <key>#LectureKey</key>
  <name>Lecture Name</name>
  <owner>Authorized User</owner>
  <date>Date.getTime()</date>
  <attribute id="1">
    <key>KEYCODE</key>
    <value>ATTRIBUTE VALUE</value>
    <count>COUNT OF VOTINGS</count>
    <named>NAME OF ATTRIBUTE</named>
```

```
    <high>NAME OF HIGH END</high>
    <low>NAME OF LOW END</low>
  </attribute>
  <attribute id="2">
    <key>KEYCODE</key>
    <value>ATTRIBUTE VALUE</value>
    <count>COUNT OF VOTINGS</count>
    <named>NAME OF ATTRIBUTE</named>
    <high>NAME OF HIGH END</high>
    <low>NAME OF LOW END</low>
  </attribute>
  <attribute id="3">
    <key>KEYCODE</key>
    <value>ATTRIBUTE VALUE</value>
    <count>COUNT OF VOTINGS</count>
    <named>NAME OF ATTRIBUTE</named>
    <high>NAME OF HIGH END</high>
    <low>NAME OF LOW END</low>
  </attribute>
</lecture>
```

Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Drabenderhöhe, den 1. Juli 2010

Manuel Krischer