HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

COMPASS SECURITY

Hacking-Lab Mobile Event App

Bachelorarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2015

Autoren:	Oussama Zgheb, Tobias Zahner
Betreuer:	Dipl. El. Ing. FH Ivan Bütler
Projektpartner:	Compass Security Schweiz AG
Experte:	MSc ETH Inf.-Ing Philipp Sieber
Gegenleser:	Dipl. El. Ing. HTL Thomas Letsch

Abstract

Abteilung: Informatik
Namen der Studierenden: Oussama Zgheb, Tobias Zahner
Studienjahr: FS 2015
Titel der Studienarbeit: Hacking-Lab Mobile Event App
Experte: MSc ETH Inf.-Ing Phillip Sieber
Gegenleser: Dipl. El. Ing. HTL Thomas Letsch
Betreuer: Dipl. El. Ing. FH Ivan Bütler
Themengebiet: Internet-Technologien und -Anwendungen
Projektpartner: Compass Security Schweiz AG

Ausgangslage
Bei Konferenzen oder ähnlichen Veranstaltungen, besteht das Bedürfnis die einzelnen Vorträge oder Darbietungen zu bewerten. Angestrebt wird eine App, welche es ermöglicht, die einzelnen Vorträge nach verschiedenen Kriterien zu bewerten. Neben dieser Kernfunktion soll die App auch jederzeit aktuelle Infos zur Veranstaltung liefern und den Teilnehmern die Möglichkeit geben über soziale Kanäle miteinander zu interagieren.

Vorgehen/Technologien
Als Erstes wurde entschieden, für welche Plattform die App entwickelt wird. Die Entscheidung fiel auf Android, da in diesem Bereich schon Erfahrungen vorhanden waren und es eine weitverbreitete Plattform ist. Aus den Anforderungen der Aufgabenstellung wurden die folgenden Komponenten abgeleitet:
• Ein Web-Frontend um die App Inhalte zu verwalten
• Einen REST-Service als Schnittstelle zwischen App und Datenbank
• Eine Android App, welche mit dem REST-Service interagiert
Bei der Android App war schnell klar, dass Java und die Android API verwendet wird. Bei den weiteren Komponenten war der Entscheid etwas schwieriger. Aufgrund der bestehenden Infrastruktur unseres Industriepartners ist die Wahl beim REST-Service auf JAX-WS (Java) mit einer MySQL Datenbank und beim Web-Frontend auf AngularJS gefallen. Um alle Android Apps möglichst simultan auf neue Ereignisse aufmerksam zu machen, wurden Push Technologien evaluiert. Dabei stellte sich Google Cloud Messaging als beste Lösung heraus. Um möglichst früh das gelingen der Arbeit zu festigen, wurden Prototypen für die einzelnen Kernfunktionen erstellt und ausgiebig getestet.

Ergebnis
Entstanden ist eine voll funktionsfähige Event-App mit der dazugehörigen Verwaltungssoftware. Das gesamte System wird am 21.10.15 bei der Swiss Cyber Storm das erste mal produktiv eingesetzt. Obwohl die App auf die Hacking-Lab-Events ausgerichtet wurde, kann sie zukünftig mit nur wenigen Änderungen für beliebige Veranstaltungen verwendet werden. So wäre zum Beispiel die Bewertung von Konzerten oder Vorlesungen denkbar. In Zukunft könnte man aber weitere Implementationen für andere Plattformen in Betracht ziehen.

Aufgabenstellung

Das Hacking-Lab (www.hacking-lab.com) ist ein weltweit genutztes CTF und Hacking Challenge System zu Ausbildungszwecken. Im Jahr 2015 findet in der Schweiz der Final der European Cyber Security Challenge (ECSC) mit 6 Nationen aus Europa statt (DE, AT, UK, Spain, Rumänien, CH). Für diesen ECSC soll das Hacking-Lab um eine kostenlose Mobile Anwendung mit Server Applikation erweitert werden.

Die neuen Kernfunktionen umfassen:
a) Voting, Bewertung von Talks & Export der Bewertungen
b) Contribution, Gateway für Twitter/Facebook u.ä für Text und Bilder
c) Push, Notifikation der Mobile App User
d) Dashboard, Anzeige CTF Scoring & Teams
e) Agenda, zeigt das Event Schedule an
f) Rollen, Authentifizierung & Autorisierung für bestimmte Rollen

Die Arbeit soll dabei aus den folgenden Komponenten bestehen:
a) Mobile App, soll für Android und in englischer Sprache entwickelt werden
b) Frontend, soll mit AngularJS umgesetzt werden
c) Backend, soll eine REST API bieten und MySQL als Datenbank verwenden

Rapperswil, den 10.6.2015

Betreuer, Ivan Bütler

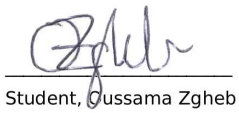
Eigenständigkeitserklärung

Leistung	Person
Backend, Frontend, Datenbank (inkl. Dokumentation)	Oussama Zgheb
Mobile App (inkl. Dokumentation)	Tobias Zahner

Ich erkläre hiermit,

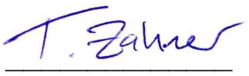
- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe
- dass ich keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt habe.

Rapperswil, den 5.6.15



Student, Oussama Zgheb

Rapperswil, den 5.6.15



Student, Tobias Zahner

Vereinbarung

1. **Gegenstand der Vereinbarung**
   Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Bachelorarbeit „Hacking-Lab Mobile Event App" von Oussama Zgheb und Tobias Zahner unter der Betreuung von Herrn Ivan Bütler der Compass Security AG geregelt.

2. **Urheberrecht**
   Die Urheberrechte stehen der Studentin / dem Student zu.

3. **Verwendung**
   Die Ergebnisse der Arbeit dürfen sowohl von allen an der Arbeit beteiligten Parteien, d.h. von den Studenten, welche die Arbeit verfasst haben, vom verantwortlichen Betreuer sowie vom Industriepartner verwendet und weiter entwickelt werden. Die Namensnennung der beteiligten Parteien ist bei der Weiterverwendung erwünscht, aber nicht Pflicht.

Rapperswil, den 5.6.15 _____
Student, Oussama Zgheb

Rapperswil, den 5.6.15 _____
Student, Tobias Zahner

Rapperswil, den 5.6.2015 _____
Betreuer, Ivan Bütler

Danksagung

An dieser Stelle möchten wir uns herzlichst bei allen Personen bedanken, die uns bei der Durchführung unserer Bachelorarbeit unterstützt haben. Besonderer Dank gilt dabei folgenden zwei Personen,

Unserem Betreuer, Herrn Ivan Bütler, welcher mit seinem grossen Know-How und hilfreichem Input eine sehr wertvolle Ansprechperson gewesen ist. Wir sind ihm sehr verbunden, dass er trotz seiner Beschäftigung als CEO der Compass Security auch noch zur späten Stunde für uns da war.

Unserem Experten, Herrn Philipp Sieber, welcher bei Problemen mit der Android App zur Stelle war und uns mit seinen Tips weitergeholfen hat.

Letztendlich möchten wir speziell unseren Familien und Freunden für ihre Unterstützung und Geduld bedanken.

Konventionen

In allen Dokumenten der Bachelorarbeit bestehen die folgenden Konventionen:

- Abbildungen, auch »Abb« genannt, ohne Quellenangabe sind durch die Autoren erstellt worden.
- Screenshots sind ausnahmslos durch die Autoren erstellt worden.
- Das Gesamtdokument ist in sechs verschiedene Teile aufgeteilt. Jeder dieser Teile beinhaltet ein oder mehre Dokumente. Diese Dokumente können ein eigenes Inhaltsverzeichnis haben. Der Aufbau dieser Arbeit richtet sich dabei nach dem Mail »M_OZ_011« von Ivan Bütler, welches im Anhang zu finden ist.

Hacking-Lab Mobile Event App

Oussama Zgheb & Tobias Zahner

11. Juni 2015

Inhaltsverzeichnis

Hacking-Lab Mobile Event App | Management Summary

Oussama Zgheb & Tobias Zahner

11. Juni 2015

Datum	Version	Änderung	Autor
06.05.15	1.0	Intial	Oussama Zgheb
07.06.15	1.1	Vorgehen, Ergebnisse	Oussama Zgheb
07.06.15	1.2	Überarbeitung	Tobias Zahner
08.06.15	1.3	Konkurrenz, Eigenleistung	Oussama Zgheb
09.06.15	1.4	Überarbeitung	Oussama Zgheb

Inhaltsverzeichnis

1 Ausgangslage

Bis anhin wurde an den Konferenzen der Compass Security Bewertungen zu Vorträgen auf Papier durchgeführt. Dies ist umständlich, zeitaufwändig und bietet keine effektive Möglichkeit die Konferenzteilnehmer an den Abstimmungen teilhaben zu lassen.

Deshalb soll eine Mobile App entwickelt werden, welches eine Voting Funktion bietet. Zudem soll die App den Besuchern helfen sich besser zurecht zu finden. *»Was findet wo statt? Welche Referenten gibt es? Wann sprechen diese? Was sind aktuelle News?«* – dies sind die üblichen Fragen der Besucher, welche durch die Mobile App beantwortet werden sollen.

1.1 Motivation

Die Möglichkeit mit neusten Technologien und in Zusammenarbeit mit einem Industriepartner die Bachelorarbeit durchzuführen ist eine grosse Motivation. Zusätzlich haben die Studierenden selbst mit dem Hacking-Lab Kontakt gehabt und wollen gerne auch etwas zu dem Projekt beitragen.

1.2 Konkurrenz

Es gibt bereits verschiedenste Mobile Voting Lösungen. Das zu entwickelnde Mobile App vereint die spezifischen Bedürfnisse an das Voting und die Möglichkeit sich zu einer Konferenz zu informieren. Darüber hinaus ist die Auswahl eines Events sowie Voting ohne zusätzliche Information (wie z.B. einen QR Code, PIN etc.) möglich, was bei den meisten bestehenden Lösungen nicht der Fall ist.

1.3 Ziele

Die folgenden Ziele sollen und können in der Bachelorarbeit erfüllt werden.

1.3.1 Soll

- Voting
  Durchführung, Auswertung und Export
- Konferenzbegleiter
  Agenda, Informationen zum Event, Informationen der Referenten
- Dashboard
  Punktestand der Hacking-Challenges, Informationen der Teammitglieder
- Soziale Kanäle
  Teilen von Fotos und Text auf sozialen Kanälen mit vorheriger Moderation

1.3.2 Kann

- Upload von Video und Ton sowie deren Veröffentlichung auf sozialen Kanälen

2 Vorgehen

Zu Beginn wurden die Anforderungen anhand der ursprünglichen Aufgabenstellung ermittelt. Anhand dieser Anforderungen und den Inputs des Betreuers wurden Vorschläge für die Benutzeroberfläche erstellt, diskutiert und der Funktionsumfang festgelegt. Danach wurden für die wichtigen Kernfunktionen Prototypen entwickelt, welche die Machbarkeit bestätigten sowie das Risiko bei der Entwicklung vermindern sollten.
Herr Ivan Bütler, der Betreuer der Bachelorarbeit wurde dabei mit regelmässigen Emails über den neusten Stand der Entwicklung informiert und bei Bedarf wurden Treffen organisiert.

Am 17.4.2015 fand eine Zwischenpräsentation an der Hochschule für Technik in Rapperswil statt. An dieser Zwischenpräsentation wurde dem Betreuer, dem Experten Philipp Sieber sowie dem Gegenleser Thomas Letsch der aktuelle Stand und das weitere Vorgehen präsentiert.

3 Ergebnisse

Es wurde eine vollständige Softwarelösung zur Bewertung von Vorträgen nach konfigurierbaren Kriterien entwickelt. Zudem wurden alle Anforderungen umgesetzt und die Arbeit dokumentiert.

3.1 Nutzen

Der Nutzen für **Veranstalter** ist sehr hoch, da nach einer Einführung für die Operatoren viel Zeit gespart werden kann:

- Die Bewertung muss nicht mehr auf Papier durchgeführt werden
- Die Auswertung geschieht automatisch, korrekt und in Echtzeit
- Die Agenda kann über eine Benutzeroberfläche angepasst werden
- Wichtige Mitteilungen können schnell und per Mausklick an fast alle Konferenzteilnehmer vermittelt werden
- Die Archivierung der Konferenzdaten ist sehr einfach, da diese alle bereits digital in einer Datenbank erfasst sind

Der Nutzen für die **Konferenzbesucher** ist, dass diese sich besser informieren können und durch das Voting sowie mit dem Teilen von Medien nun miteinbezogen werden.
Auch Personen die nicht an der Konferenz teilhaben können, haben nun die Möglichkeit direkte Einblicke zu erhalten. Dadurch wird das Interesse sowie die Bekanntheit der Konferenzen gesteigert.

3.2 Eigenleistung

Der gelb eingefärbte Teil der folgenden Abbildung Abb1 entspricht den in der Bachelorarbeit bearbeiteten Software Komponenten. Der Hacking-Lab REST Service ist nicht Teil der Bachelorarbeit. Dieser Service bestand bereits zu Beginn der Arbeit und wird durch die Compass Security betrieben.

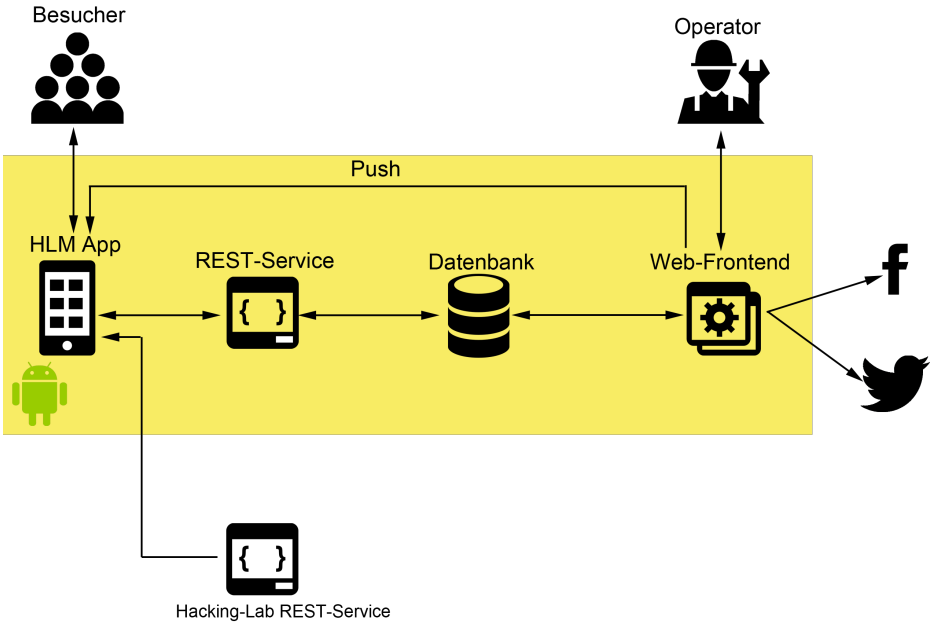


Abb1 - Komponenten Übersicht[1]

[1] Abb1 - Komponenten Übersicht

3.3 Kosten

Entwicklungs- oder Lizenzkosten gibt es keine. Es ist jedoch mit Wartungskosten für die Software, Datenbank und den Server zu rechnen.

4 Ausblick

4.1 Risiken

Alle absehbaren Risiken wurden sorgfältig behandelt. Potentielle Probleme welche durch grosse Benutzerzahlen auftreten könnten, konnten nur in der Theorie behandelt werde. Dies wäre sonst sehr Ressourcen- und Zeitaufwändig geworden. Deshalb wird empfohlen, einen Testdurchlauf durchzuführen und dabei die momentane Papierlösung für das Voting als »Backup« zu bereitzuhalten.

4.2 Verbleibende Probleme

Es sind zum Zeitpunkt der Abgabe keine Probleme oder Bugs bekannt.

4.3 Weiterführende Arbeiten

- Frontend
  Im Bereich des Frontend besteht in den Bereichen »Design« sowie »Formularüberprüfung« Verbesserungspotential
- App
  Bei der App besteht in folgenden Bereichen Verbesserungspotential:
  - Möglichkeit Videos und Audioaufzeichnungen zu teilen
  - Verbesserung der Agenda-Implementation
  - Tests auf weiteren, verschiedenen Gerätetypen

Im Allgemeinen kann das ganze System ausgebaut werden, so wären Funktionen wie eine Zahlungslösung für Event Tickets und eine iOS App denkbar.

Hacking-Lab Mobile Event App | Technical Summary Front&Backend

Oussama Zgheb & Tobias Zahner

11. Juni 2015

Inhaltsverzeichnis

1 Einführung

Bis anhin wurde an den Konferenzen der Compass Security diverse Bewertungen von Vorträgen auf Papier durchgeführt. Dies ist umständlich, zeitaufwändig und bietet keine effektive Möglichkeit die Konferenzteilnehmer an den Abstimmungen teilhaben zu lassen. Deshalb soll eine Mobile App entwickelt werden, welche eine Voting Funktion bietet. Zudem soll die App den Besuchern helfen sich besser zurecht zu finden. »Was findet wo statt? Welche Referenten gibt es? Wann sprechen diese? Was sind aktuelle News?« – dies sind die üblichen Fragen der Besucher welche durch die Mobile App beantwortet werden sollen. Darüber hinaus kann der Nutzer der App selbst Fotos und Kommentare über die Konferenz teilen.

2 Analyse

Zu Beginn der Arbeit wurden einige Sitzungen mit allen Beteiligten (Betreuer und Studenten) durchgeführt, siehe vollständige Sitzungsprotokolle im Anhang. Dabei wurden folgende Entscheidungen gefällt

- Die App soll die Möglichkeit bieten Inhalte zu teilen, mit einer Schnittstelle zu Twitter und Facebook (Sitzung vom 12.02.2015)
- Die zu erstellenden Komponenten wurden wie folgt festgelegt (Sitzung vom 19.02.2015):
  - Eine Mobile App für die Teilnehmer der Konferenz
  - Einen REST-Service als Schnittstelle zwischen Datenbank und Mobile App
  - Ein Frontend zur Verwaltung der Datenbasis
- Die Mobile App wird vorerst nur für Android entwickelt. Auf Crossplatform wurde verzichtet aufgrund der schlechteren Usability und da die Erfahrung mit entsprechenden Entwicklungs-Frameworks fehlte. Auch auf einen iOS Client wird verzichtet, da in diesem Bereich ebenfalls keinerlei Erfahrung vorhanden war. (Sitzung vom 25.02.2015)
- Ein Teil der Daten kommt vom schon bestehenden Hacking-Lab REST-Service (Sitzung vom 25.02.2015)

Daraus ergaben sich die folgenden Komponenten. Die gelb hinterlegten Komponenten in der Abbildung Abb1 waren dabei in der Bachelorarbeit zu erarbeiten.

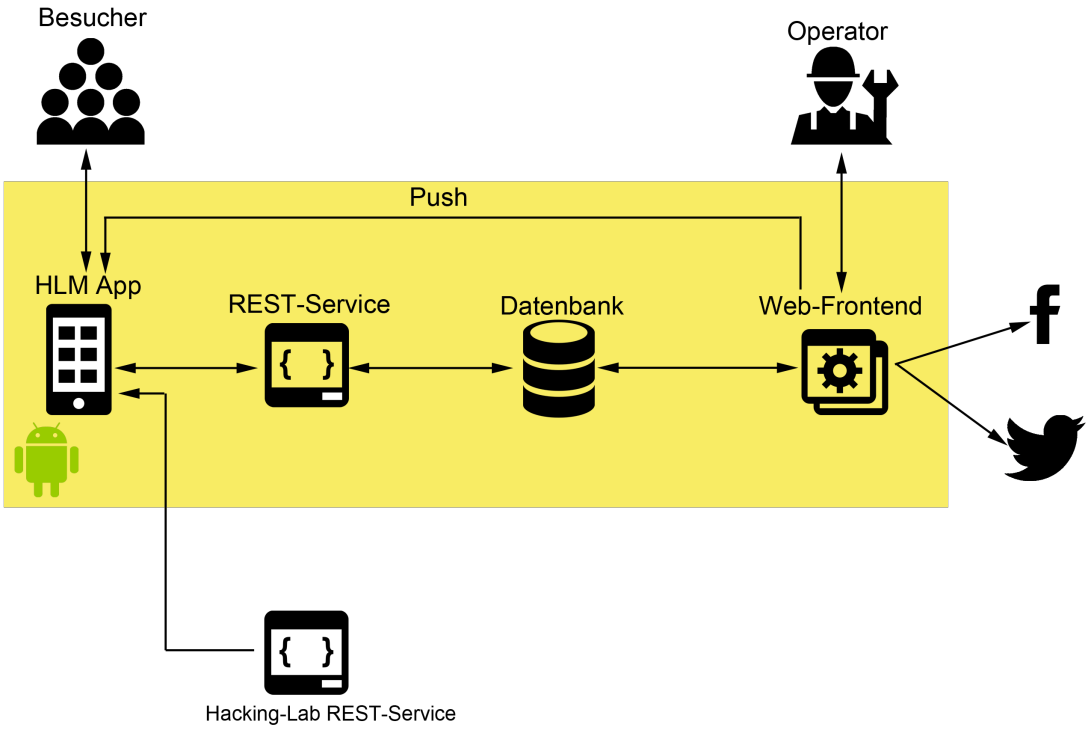


Abb 1 - System Komponenten [1]

[1] Abb 1 - System Komponenten

3 Anforderungen

Aus der Analyse und der Aufgabenstellung wurden folgende Anforderungen abgeleitet.

3.1 Nichtfunktionale Anforderungen

3.1.1 Backend

- NAnfB-1: Das Backend (bei angemessener Hardware) muss für potentiell hunderte Mobile Apps gleichzeitig ohne grossen Zeitverzögerungen (bis zu 2 Sekunden) mit Informationen versorgen.
- NAnfB-2: Das Backend muss die Möglichkeit bieten, alle Mobile Apps in »beinahe Echtzeit« mit Informationen zu versorgen (Push).
- NAnfB-3: Das Backend soll die Veranstalter eines Events unterstützen und eine einfach zu konsumierende API bieten.
- NAnfB-4: Das Backend soll eine Availability von mindestens 99% bieten.

3.1.2 Frontend

- NAnfF-1: Der Benutzer des Frontend soll nicht an eine gewisse Software gebunden sein - es soll in den üblichen Browsern funktionieren.
- NAnfF-2: Das Frontend muss ohne spezielle Browser / System Plugins lauffähig sein.

3.1.3 Mobile App

- NAnfM-1: Die Mobile App soll den Titel »Confoxy« tragen [2]
- NAnfM-2: Die App soll für das Betriebssystem Android entwickelt werden und Material Design verwenden [3]
- NAnfM-3: Die App soll in englischer Sprache umgesetzt werden.

3.2 Funktionale Anforderungen

3.2.1 Backend

- AnfB-1: Das System muss die Möglichkeit bieten, Bilder und Texte auf sozialen Kanälen veröffentlichen. Dabei sind im Rahmen der Bachelorarbeit die Dienste »Facebook« sowie »Twitter« als Muss Kriterien definiert.
- AnfB-2: Das System muss Fehlertolerant sein. Das heisst alle Elemente können nachträglich editiert, falls möglich gelöscht oder neugestartet werden können.
- AnfB-3: Es kann pro User pro Voting nur eine Stimme abgeben werden.
- AnfB-4: Die Granularität eines Sliders soll konfigurierbar sein.
- AnfB-5: Ein Voting soll anzeigen können wie viele der Jury ihre Stimme abgeben haben.
- AnfB-6: Ein Voting soll mehrere Runden unterstützen. Dabei werden alle vorhandenen Votes gelöscht und das Voting in den Zustand vor dem Start zurückgesetzt werden.
- AnfB-7: Alle Voting Daten sollen als CSV (Comma Seperated Values) exportiert werden können.
- AnfB-9: Die API soll es dem Mobile App ermöglichen sich zu registrieren und Daten anzubieten.
- AnfB-10: Nicht öffentliche API Calls / Seiten sollen Passwort geschützt sein und von den öffentlichen getrennt.
- AnfB-11: Das Backend soll eine Moderationsfunktion für geteilte Inhalte der Mobile App User bieten.
- AnfB-12: Die Steuerung des Votings soll wie in den Mockups beschrieben umgesetzt werde.

[2] Mail von Ivan Bütler, 05.06.2015 15:34, »M_TZ_001_Name App.msg«
[3] Google: Material Design, URL: http://www.google.com/design/spec/material-design/introduction.html (Stand: 06.06.15)

- AnfB-13: Das Backend ermöglicht den Upload von Daten mittels Multipart Form Data.

3.2.2 Frontend

- AnfF-1: Aktionen im Frontend müssen unabhängig vom Operator ablaufen können.
  z.B. wenn ein Voting gestartet wird und der Operator sich abmeldet / den Browser schliesst, soll das System weiterlaufen.
- AnfF-2: Das Frontend muss mehrere aktive Votings gleichzeitig verwalten können.
- AnfF-3: Das Frontend muss bei der Voting Ansicht die neusten Vote-Daten automatisch aktualisieren.

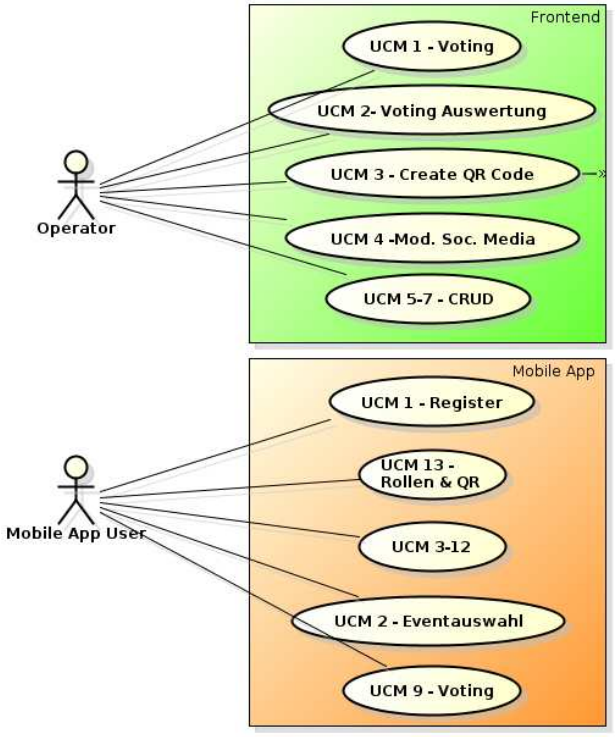
3.2.3 Mobile App

- AnfM-1: Der Benutzer registriert sich lediglich per Nickname (A-Z,a-z,0-9,_).
- AnfM-2: Bei mehreren parallelen Events soll der Benutzer den gewünschten wählen können.
- AnfM-3: Folgende Ansichten sollen in der App umgesetzt werden:
  - News: Zeigt vom Veranstalter veröffentlichte News an
  - Share-Wall: Zeigt die geteilten Inhalte der Teilnehmer an
  - Share: Unter Share kann der User ähnlich wie bei WhatsApp & Facebook einen Beitrag mit oder ohne Bild erfassen. Nach dem Erfassen des Beitrages wird dem User eine Erfolgsmeldung angezeigt. Der Operator kann die Social Items dabei revidieren und auf Facebook & Twitter posten.
  - Conference: Unter Conference gibt es nichts zu tun, sondern Informationen über den aktuellen Event. Die Informationen stammen vom HLM Server (Event Manager).
  - Speaker: Listet die Speaker auf und gibt Infos zu Ihnen aus (Name,Vorname,Titel, Abstract über seine Karriere, Bild, Link zu seinen Talks)
  - Agenda: Unter Agenda kann sich der User über die Agenda des Tages informieren. Er soll schnell herausfinden können, was als nächstes ansteht. Die Daten stammen vom HLM Server. Es soll möglich sein einen Zeitslot auszuwählen, daraufhin wird dem User alle Vorträge in diesem Slot angezeigt und deren Durchführungsort.
  - Voting: Gibt den Teilnehmern die Möglichkeit, einen Vortrag zu bewerten
  - Scoring: Unter Scoring findet der User heraus, wer hinsichtlich dem Hacking Challenge die Nase vorn hat. Die Daten kommen vom Restful Service des Dashboard. Wenn der User auf ein Team (Land) klickt, sieht man mit welchen gelösten Challenges sich die Punkte zusammensetzen.
  - Challenges: Unter Challenges kann sich der User ein Bild über die verschiedenen Challenges, unabhängig davon ob diese bereits gelöst wurden oder nicht, machen. Bei jeder Challenge kann der User die Details einer Challenge anklicken. Diese Details werden durch ein Video geliefert, das eingebunden wird. [4]
  - Teams: Unter Teams kann der User sich genauer über die Team Mitglieder der einzelnen Teams informieren. Er findet dort alle Team Members und ihr Profilbild. Diese Daten stammen vom Restful Service des Dashboard.
- AnfM-4: Die Bewertung eines Vortrags (Voting) soll durch Slider geschehen.
- AnfM-5: Durch eine spezielle Geste sollen Jury Mitglieder / Autoren die Möglichkeit haben einen QR einzuscannen und dadurch spezielle Rechte zu erhalten.

[4] Mail von Ivan Bütler, 11.05.2015, 15:46

3.3 Use Cases

Anhand der funktionalen Anforderungen und der Aufgabenstellung wurden folgende Use Cases erarbeitet. Da das Backend nicht direkt durch einen Benutzer bedient wird, wurden keine Use Cases dafür geschrieben.

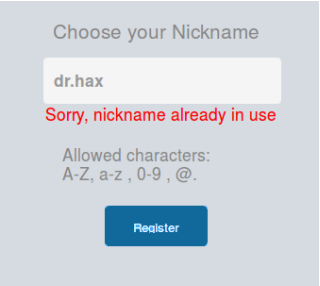


3.3.1 Mobile App

Nr.	Titel
UCM 1	Register
UCM 2	Eventauswahl
UCM 3	News
UCM 4	Social Wall
UCM 5	Share
UCM 6	Conference
UCM 7	Agenda
UCM 8	Speaker
UCM 9	Voting
UCM 10	Scoring
UCM 11	Challenges
UCM 12	Teams
UCM 13	Rollen und QR-Codes

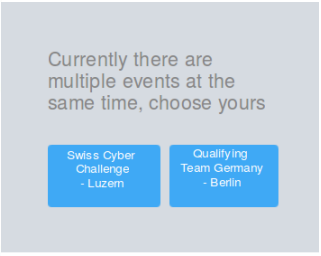
UCM 1 - Register

Precondition	-	Kein Benutzername vorhanden (im Normalfall heisst das, die App wird zum ersten Mal gestartet).
Postcondition	-	Benutzer wurde auf dem Server registriert - Benutzername und DeviceId wurden auf dem Gerät abgespeichert
Steps	1.	Der Benutzer startet die App
	2.	Der Benutzer tippt seinen gewünschten Nicknamen ins dafür vorgesehene Textfeld
	3.	Der Benutzer klickt auf den Button „register“
Variations	-	Der gewünschte Benutzername enthält unerlaubte Zeichen oder ist schon vergeben. In diesem Fall wird dem Benutzer eine entsprechende Fehlermeldung angezeigt



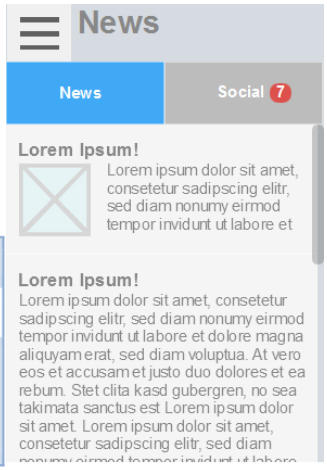
UCM 2 - Eventauswahl

Precondition	-	Benutzername und DeviceId ist vorhanden
Postcondition	-	Gewählter Event wurde zwischengespeichert auf dem Device
Steps	1.	Der Benutzer startet die App
	2.	Dem Benutzer werden die aktuell laufenden Events aufgelistet
	3.	Der Benutzer wählt den Event aus, an dem er gerne teilnehmen möchte
Variations	-	Es ist kein Event vorhanden ? Der Benutzer kann die Eventliste aktualisieren oder die App beenden. Ohne Eventauswahl ist ein Benutzen der App nicht möglich.
	-	Es ist nur ein Event vorhanden Der Event wird automatisch gewählt und der Benutzer wird auf die Startseite der App weitergeleitet.



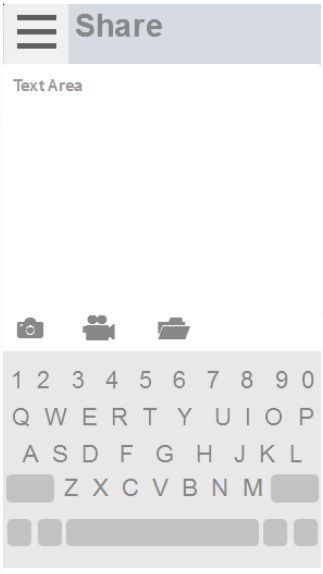
UCM 3 - News

Precondition	-	EventId ist vorhanden
Postcondition	-	News wurden aktualisiert
Steps	1.	Der Benutzer startet die App
	2.	Auf der Startseite werden ihm die aktuellen News zu seinem Event angezeigt. Diese können entweder nur aus Text oder aus Text und Bild bestehen.



UCM 4 - Social Wall

Precondition	-	EventId ist vorhanden
Postcondition	-	Social News wurden aktualisiert
Steps	1.	Der Benutzer startet die App
	2.	Der Benutzer wechselt in den Tab Social
	3.	Es werden alle veröffentlichten social Posts angezeigt. Das sind entweder Posts, die von einem autorisierten Autor stammen oder Posts, welche von der Redaktion geprüft wurden



UCM 5 - Share

Precondition	-	Benutzername, DeviceId und EventId ist vorhanden
Postcondition	-	Share Post wurde abgesetzt
Steps	1.	Der Benutzer startet die App, öffnet die Navigation und wählt „Share“
	2.	Er gibt eine Nachricht von maximal 140 Zeichen (Twitterbegrenzung) ein.
	3.	Bei Bedarf hat er die Möglichkeit dem Post ein Foto anzuhängen, welches er entweder direkt mit der Kamera macht oder aus der Gallerie auswählt.
	4.	Er klickt auf den „Share“-Button und die Nachricht wird abgeschickt.
	5.	Die Redaktion erhält den Post und kann ihn entweder akzeptieren oder ablehnen. Zusätzlich kann sie ihn auch direkt auf ihren Twitter oder Facebook Account posten.
	6.	Wenn der Post akzeptiert wird, wird er anschliessend in der Social Wall angezeigt.
Variations	-	Wenn der Benutzer die Rolle „Autor“ besitzt, wird der Post nicht von der Redaktion kontrolliert, sondern direkt veröffentlich.

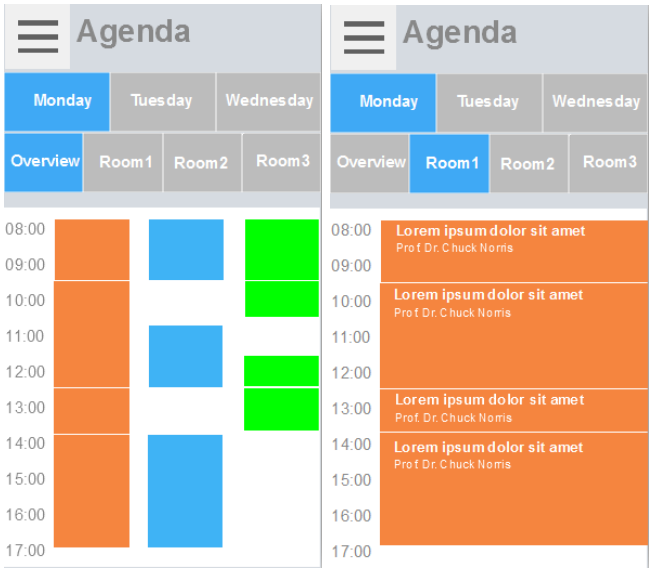
UCM 6 - Conference

Precondition	-	EventId ist vorhanden
Postcondition	-	EventInfo wurden aktualisiert
Steps	1.	Der Benutzer startet die App, öffnet die Navigation und wählt „Conference“
	2.	Ihm wird eine Beschreibung des Events angezeigt.

UCM 7 - Agenda

Precondition	-	EventId ist vorhanden
Postcondition	-	Agenda wurde aktualisiert
Steps	1.	Der Benutzer startet die App, öffnet die Navigation und wählt „Agenda“
	2.	Ihm wird ein Kalender angezeigt, der die Vorträge und sonstige Veranstaltungen an seinem Event übersichtlich darstellt.
	3.	Falls er nur die Veranstaltungen in einem bestimmten Raum sehen möchte, kann er diesen Raum auswählen.
	4.	Wenn er über einen Vortrag mehr erfahren will, klickt er auf den entsprechenden Eintrag und es öffnet sich die Detailseite zu diesem Vortrag.

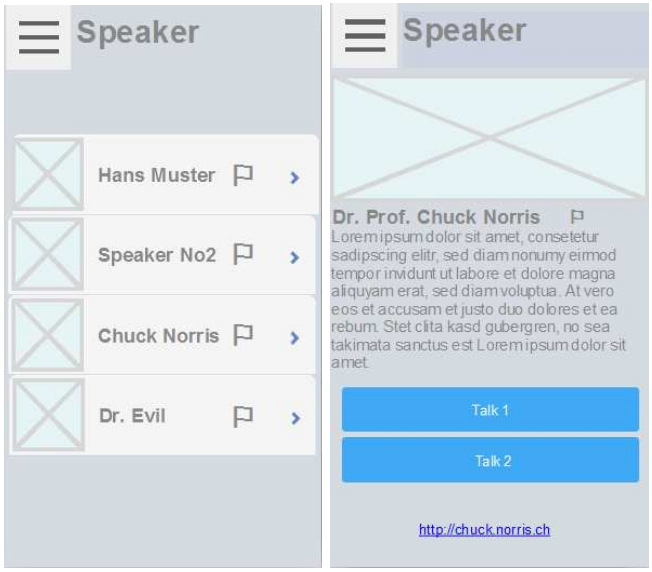
Mockups zu UCM 7 - Agenda



UCM 8 - Speaker

Precondition	-	EventId ist vorhanden
Postcondition	-	Speaker des aktuellen Events wurden aktualisiert
Steps	1.	Der Benutzer startet die App, öffnet die Navigation und wählt „Speaker“
	2.	Ihm wird eine Liste aller Speaker mit Portraitfoto, Name und Länderflagge angezeigt, die am Event einen Vortrag halten.
	3.	Wenn er über einen Speaker mehr erfahren will, klickt er auf den entsprechenden Eintrag und es öffnet sich die Detailseite zu diesem Speaker.
	4.	Auf der Detailseite wird neben einer kurzen Biographie oder Beschreibung des Speakers auch eine Liste seiner Vorträge angezeigt. Bei Bedarf kann der Benutzer direkt auf die Beschreibungen dieser Events gehen.

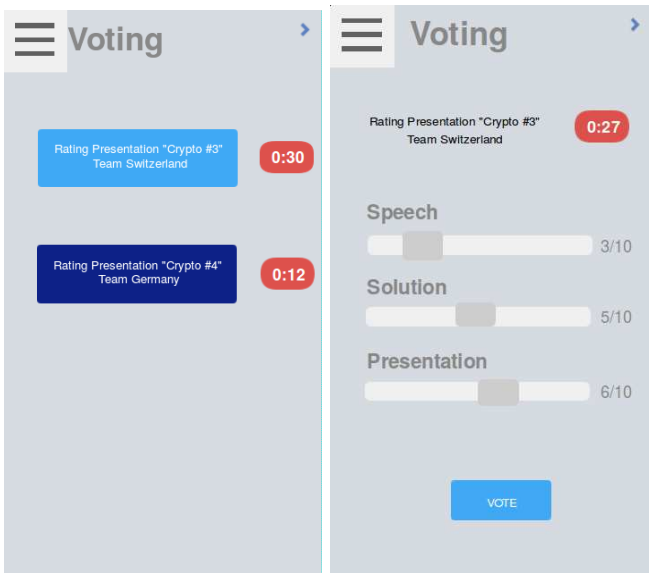
Mockups zu UCM 8 - Speaker



UCM 9 - Voting

Precondition	-	EventId, DeviceId und Benutzername ist vorhanden
Postcondition	-	Das Voting wurde abgegeben
Steps	1.	Der Benutzer erhält eine Push-Notification, dass ein Voting gestartet wurde.
	2.	Nach einem Klick auf den Push wird er direkt auf das laufende Voting umgeleitet.
	3.	Er schiebt die Slider an die gewünschten Positionen und klickt anschliessend auf den „Vote“-Button.
	4.	Das Voting wird abgeschickt und der Benutzer kann anschliessend kein zweites Mal an diesem Voting teilnehmen.
Variations	-	Statt über die Pushnotification kann der Benutzer auch selbständig über die Navigation auf „Voting“ gehen und dort das gewünschte laufende Voting auswählen.
	-	Gehört der Benutzer zur Jury, bzw. ist er im Besitz der Jury-Rolle für diesen Event, wird sein Secret-Key dem Post angehängt um ihn authentifizieren zu können.

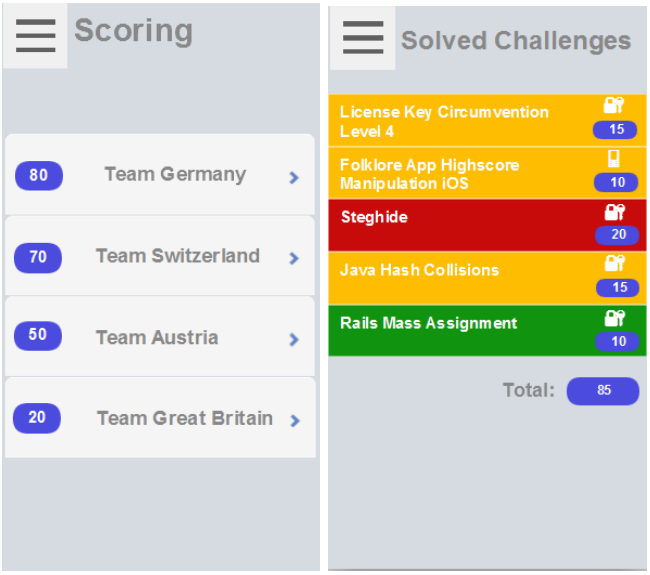
Mockups zu UCM 9 - Voting



UCM 10 - Scoring

Precondition	-	EventId ist vorhanden
Postcondition	-	Scoringinfos wurden aktualisiert
Steps	1.	Der Benutzer startet die App, öffnet die Navigation und wählt „Scoring“.
	2.	Ihm werden die Teams geordnet nach Punktzahl angezeigt
	3.	Der Benutzer klickt auf ein Team und ihm wird angezeigt, wie sich die Punktzahl zusammensetzt bzw. bei welchen Challenges das Team wie viele Punkte geholt hat.

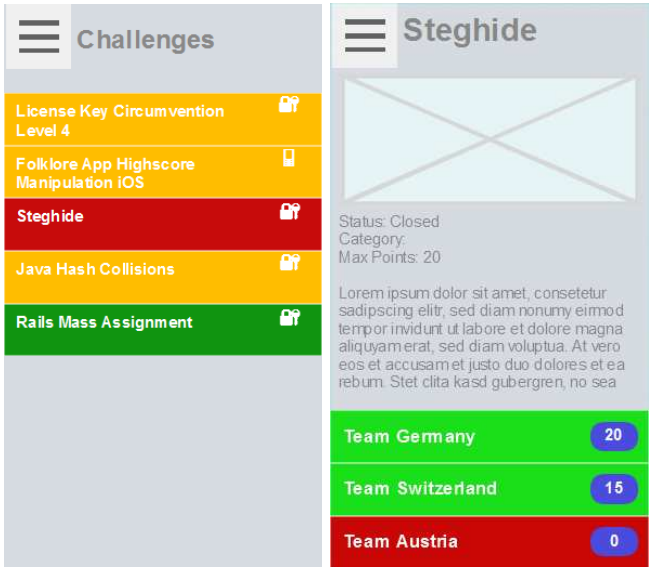
Mockups zu UCM 10 - Scoring



UCM 11 - Challenges

Precondition	-	EventId ist vorhanden
Postcondition	-	Challenges wurden aktualisiert
Steps	1.	Der Benutzer startet die App, öffnet die Navigation und wählt „Challenges“.
	2.	Die bisher veröffentlichten Challenges werden aufgelistet. Die Schwierigkeit der Challenge wird durch eine Farbe angezeigt.
	3.	Der Benutzer klickt auf eine Challenge und ihm werden die Challenge-Infos angezeigt.

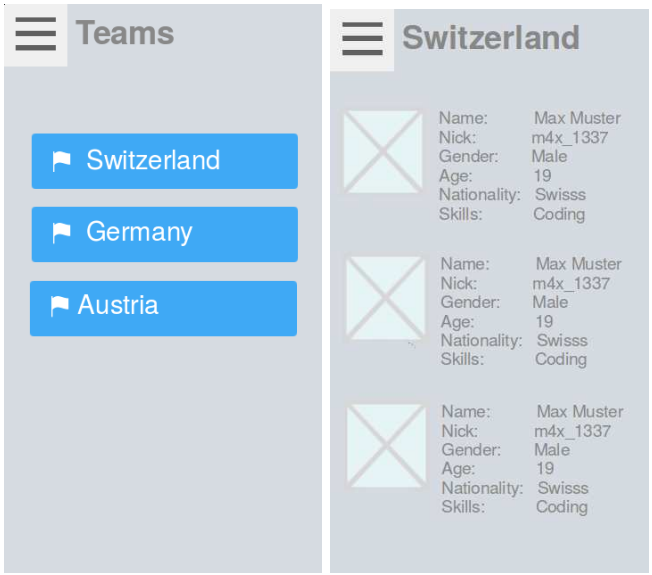
Mockups zu UCM 11 - Challenges



UCM 12 - Teams

Precondition	-	EventId ist vorhanden
Postcondition	-	Teams wurden aktualisiert
Steps	1.	Der Benutzer startet die App, öffnet die Navigation und wählt „Teams“.
	2.	Alle Teilnehmenden Teams werden mit Name, Bild und Länderflagge aufgelistet.
	3.	Der Benutzer klickt auf ein Team und ihm werden alle Teammitglieder mit Foto, Name, Vorname, Nickname, Nation und Skills aufgelistet.

Mockups zu UCM 12 - Teams



UCM 13 - Rollen und QR-Codes

Precondition	-	EventId, DeviceId und Benutzername sind vorhanden
Postcondition	-	Benutzer ist im Besitz der neuen Rolle
Steps	1.	Der Benutzer führt eine „geheime“ Geste durch.
	2.	Ein QR-Code Reader öffnet sich.
	3.	Der Benutzer liest den QR-Code den er von den Organisatoren des Events erhalten hat ein.
	4.	Der Inhalt des QR-Codes (=Secret) wird auf dem Gerät abgespeichert. Er ist nun im Besitz der neuen Rolle (entweder Autor oder Jury) für den aktuellen Event.

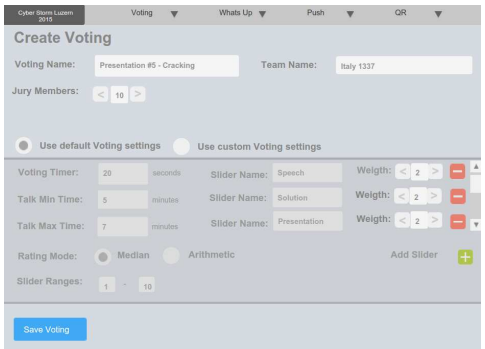
3.3.2 Frontend

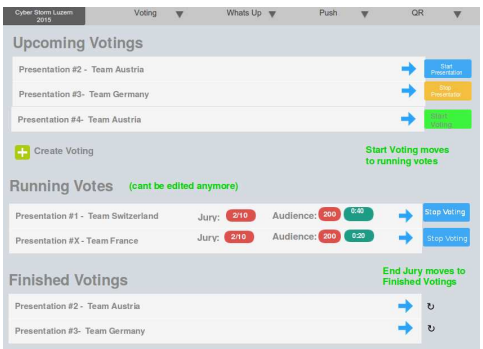
Nr.	Titel
UCM 1	Voting
UCM 2	Voting Auswertung
UCM 3	Create QR Code
UCM 4	Moderation Social Media
UCM 5	Edit
UCM 6	Create
UCM 7	Delete

UCF 1 - Voting

Precondition	-	Benutzer ist als Operator eingeloggt
Postcondition	-	Votingdaten wurden gesammelt
Steps	1.	Der Operator loggt sich auf den Event ein, der gerade stattfindet.
	2.	Der Operator klickt auf „New Voting“ (bevor der Vortrag beginnt)
	3.	Der Operator füllt die erforderlichen Felder wie Name, Anzahl Jurymitglieder, Voting-Duration und Presentation-Duration fest. Bei Bedarf kann er auch weitere Einstellungen, wie z.B. die Konfiguration der einzelnen Slider vornehmen.
	4.	Ist alles ausgefüllt, klickt er auf Submit.
	5.	Das Voting ist nun in der Votingliste drin. Sobald der Vortrag beginnt, klickt der Operator auf „Start Presentation“. Damit beginnt die Zeitmessung der Präsentation.
	6.	Sobald die Präsentation beendet ist, klickt er auf „End Presentation“.
	7.	Das Publikum hat nun noch die Möglichkeit Fragen zu stellen. Diese Zeit zählt nicht weder zur Präsentation noch zum Voting.
	8.	Sobald alle bereit sind, klickt der Operator auf „Start Voting“.
	9.	Alle Teilnehmer des Events bekommen nun eine Pushmeldung, dass das Voting läuft.
	10.	Sobald die Zeit abgelaufen ist und alle Jurymitglieder gevotet haben, klickt der Operator auf „Stop Voting“
Variations	-	Während dem Vortrag ist auch ein pausieren möglich
	-	Ist das Voting beendet, kann es reaktiviert bzw. neu gestartet werden, falls irgendetwas schief gegangen ist.

Mockups zu UCF 1 - Voting

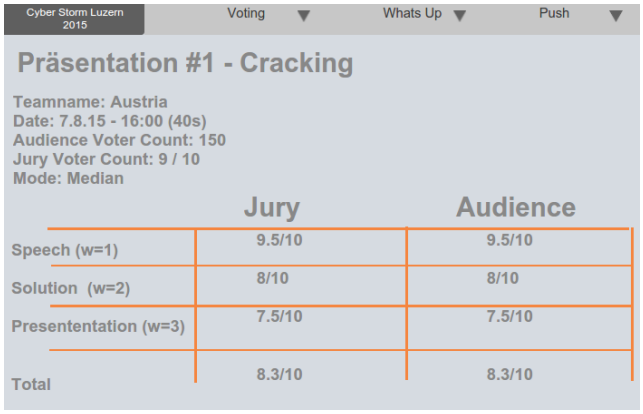




UCF 2 - Voting Auswertung

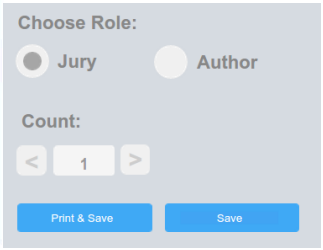
Precondition	- Benutzer ist als Operator eingeloggt - Voting ist beendet
Postcondition	
Steps	1. Der Operator loggt sich auf den Event ein und navigiert zur Votingliste. 2. Er sucht in der Liste nach einem beendeten Voting und klickt dort auf „Voting Statistics“ 3. Er sieht nun die Auswertung des Votings vor sich und kann diese bei Bedarf in ein CSV-File exportieren.

Mockups zu UCF 2 - Voting Auswertung



UCF 3 - Create QR Code

Precondition	- Benutzer ist als Operator eingeloggt - Operator hat einen Event ausgewählt
Postcondition	- QR-Code wurde generiert
Steps	1. Der Operator wählt in der Navigation „New QR-Code“ 2. Er legt die Anzahl fest und wählt aus, für welche Rolle QR-Codes erstellt werden sollen. 3. Die erstellten QR-Codes werden dem Operator angezeigt und er kann sie dem Empfänger überbringen.



UCF 4 - Moderation Social Media

Precondition	- Benutzer ist als Operator eingeloggt - Operator hat einen Event ausgewählt
Postcondition	
Steps	1. Der Operator wählt in der Navigation „Social Feed“ 2. Dort sind die erstellten Posts in vier Status gegliedert: pending, accepted, published und rejected. 3. Bei den pending-Einträgen kann der Moderator entweder auf „accept“ oder „reject“ drücken und der Post wird entsprechend behandelt. 4. Die akzeptierten Beiträge können zusätzlich optional auf Twitter und Facebook veröffentlicht werden. 5. Die abgelehnten Beiträge können bei Bedarf reaktiviert werden, sie erhalten dann wieder den Status „pending“.

UCF 5 - Edit

Precondition	- Benutzer ist als Operator eingeloggt
Postcondition	- Objekt wurde editiert
Steps	1. Der Operator wählt das Objekt aus, das er bearbeiten will und klickt auf „Edit“ 2. Der Operator editiert die gewünschten Felder und drückt auf Submit.

UCF 6 - Create

Precondition	- Benutzer ist als Operator eingeloggt
Postcondition	- Objekt wurde erstellt
Steps	1. Der Operator wählt in der Navigation „New [Object]“ 2. Der Operator füllt die 3. Der Operator editiert die gewünschten Felder und drückt auf Submit.

UCF 7 - Delete

Precondition	- Benutzer ist als Operator eingeloggt - Objekt kann gelöscht werden
Postcondition	- Objekt wurde gelöscht
Steps	1. Der Operator wählt das Objekt aus, das er bearbeiten will und klickt auf „Edit“ 2. Der Operator drückt auf den „Delete“ Button.

4 Systemarchitektur

Aus den Anforderungen wurden folgende miteinander interagierende Komponenten festgelegt. Die folgende Abbildung Abb2 zeigt die Komponenten im Zusammenspiel.

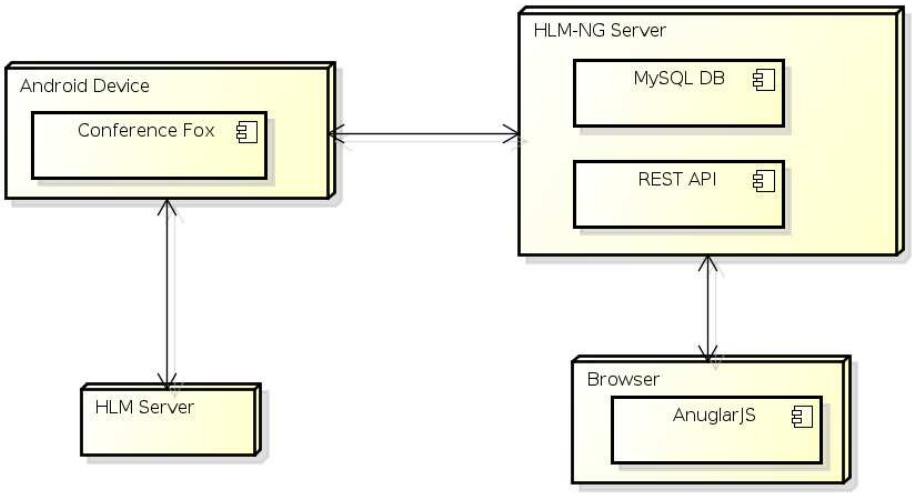


Abb2 - Komponenten Übersicht [5]

Android Device Auf dem Android Device läuft die »Conference Fox« App welche mit der REST API des HLM-NG Servers sowie mit dem bereits bestehenden HLM Server kommuniziert.

Browser Im Browser läuft die AngularJS App, welche mit der REST API des HLM-NG Servers kommuniziert. Die AngularJS App ist lediglich für die Darstellung von Daten sowie absetzen von PUT und POST Requests an den HLM-NG Server zuständig.

HLM-NG Server Auf dem HLM-NG Server läuft das Java App welches das Frontend für den Browser (AngularJS Applikation) hostet sowie die REST API welche seine Daten in der MySQL Datenbank persistiert. Dabei übernimmt das App die gesamte Business Logik.

HLM Server Der bestehende HLM Server. Dieser liefert zusätzliche Informationen an das Mobile App und ist nicht im Rahmen der Bachelorarbeit konfiguriert oder programmiert worden. Deshalb wird nicht näher darauf eingegangen.

4.1 Design

Dieses Kapitel soll die Design Entscheide anhand der Nichtfunktionalen Anforderungen und der Aufgabenstellung erläutern.

4.1.1 Frontend

Um die Anforderung AnfF-1 zu erfüllen, durfte keine Sprache wie PHP verwendet werden, da bei PHP kein Prozess selbstständig (ohne Client Browser) ausgeführt werden kann. Deshalb wurde die Kombination von AngularJS und REST Backend festgelegt, dies ermöglicht eine saubere Trennung der Businesslogik und der Darstellung. Mit der Wahl von AngularJS wurde auch die Nichtfunktionelle Anforderung NAnfF-2 erfüllt, da keine Plugins wie ActiveX oder Flash benötigt werden.

Im Allgemeinen wurden nur neue und bereits bewährte Technologien eingesetzt, um so die Zukunft der Applikation möglichst zu sichern.

[5] Abb2 - Komponenten Übersicht

4.1.2 Backend

Um die Anforderung NAnfB-1 zu erfüllen musste auf eine möglichst effiziente Implementation geachtet werden, deshalb wurde auf grosse und komplexe Frameworks wie z.B. Hibernate verzichtet. Für den Push Mechanismus, Anforderung NAnfB-2, zu realisieren wurden diverse Dienste getestet. Google Cloud Messaging stellte sich klar als Sieger heraus, da dies spezifisch für Android entwickelt wurde, der grösste Teil der Lösungen auf Google Cloud Messaging basieren und die nötige Geschwindigkeit leistet.

Sicherheit Beim Design des Backend wurde von Anfang an viel Wert auf die Sicherheit gelegt, obwohl diese keine Anforderung ist oder in der Aufgabenstellung erwähnt wurde. Folgende Sicherheitsmechanismen wurden implementiert:

- Prepared Statements
  Alle Datenbankzugriffe werden durch Prepared Statements durchgeführt.
- Logging
  Alle Zugriffe auf die Datenbank sowie Anfragen werden protokolliert.
  Ereignisse welche aussergewöhnlich sind, werden mit den entsprechenden Log Levels versehen um die Auswertung zu vereinfachen.
- Separation of Concerns
  Die API Calls für Benutzer und Operatoren wurden strikt getrennt.
- Unterdrücken von Exceptions für Clients
  Es wurden die notwendigen Einstellung vorgenommen, sodass keine ungefangenen Exceptions den Client erreichen und ihm dadurch einen potentiellen Angriffsvektor bieten.
- Escaping
  Um potentielle XSS Attacken zu verhindern wird der Input bevor er in die Datenbank gelangt mit der OWASP ESAPI und JSOUP unschädlich gemacht. Ausnahmen gibt es bei Feldern welche Sonderzeichen benutzen aber genau definiert sind, welche »Safe Values« benannt wurden. Diese sind vor allem Datum- sowie Zeitfelder und werden auf Länge sowie eine Regular Expression überprüft. Die »Safe Values« wurden eingeführt um bei den wichtigen Datum- und Zeitfeldern unabhängig der Implementation einer Library zu sein.

Das Frontend wird durch das Apache Modul »mod_but« und weitere mit einem auf Cookies basierendem Login-Mechanismus geschützt. Diese Konfiguration wurde direkt von der Compass Security übernommen und gewisse Einstellungen angepasst. Auf diese Änderungen wird im Backendmanual eingegangen und deshalb hier nicht erneut erwähnt.

5 Teil Mobile App

Datum	Version	Änderung	Autor
06.06.15	1.0	Intial	Tobias Zahner
07.06.15	1.1	Weiterarbeit	Tobias Zahner
08.06.15	1.2	Softwarearchitektur fertiggestellt	Tobias Zahner
09.06.15	1.3	Pre-version	Tobias Zahner
10.06.15	1.4	Final	Tobias Zahner

5.1 Software Architektur

Die Android wurde komplett in Android Studio[6] entwickelt. Für die Weiterentwicklung empfiehlt es sich, ebenfalls diese IDE zu verwenden. Viele Ansichten der App sind jeweils ähnlich aufgebaut, darum wird in den folgenden Kapiteln nicht auf jedes Fragment und jede Activity eingegangen. Vielmehr wird versucht, die technischen Herausforderungen zu erklären und wie diese gelöst wurden. Die resultierenden App-Seiten werden dann im folgenden Kapitel 5.2 diskutiert.

5.1.1 Projektstruktur

Die Android App wurde in folgende Komponenten unterteilt:

- activity: Enthält alle Activity-Klassen ausser MainActivity, diese befindet sich im root
- adapter: Enthält alle Adapterklassen für die diversen ListViews
- businessclasses: Enthält alle Models. Objekte dieser Klassen werden meistens durch Lesen von JSON-Files erzeugt
- database: Enthält die Klassen, die mit der lokalen Datenbank interagieren
- fragment: Enthält alle Fragment-Klassen, ausser das NavigationDrawerFragment welches für die Navigation benötigt wird und im root liegt.
- helper: Enthält diverse Helper-Klassen, die vor allem die HTTP-Requests und Posts behandeln.
- push: Enthält die Klassen, die für das Push-Handling benötigt werden.
- res: Enthält Android typisch alle Resource Files wie layout-Files, Grafiken, Strings etc.

5.1.2 Activities und Fragments

Seit Android 3.0 wird das Konzept der Activities durch Fragments ergänzt, damit sollen insbesondere Tablets besser unterstützt werden. [7] Leider ist nicht immer ganz klar, wann man Fragments und wann man Activities benutzt. Bei dieser Android-App wurde folgendes Prinzip verfolgt: Alle Seiten die direkt über die Navigation erreichar sind, werden als Fragments implementiert und der Mainactivity angehängt. Die Mainactivity besitzt dabei einen Container, in welchen das jeweilige Fragment geladen wird, wenn der Benutzer einen Menüpunkt auswählt. Die jeweiligen Detail-Seiten werden als Activities implementiert. Schematisch sieht das folgendermassen aus:

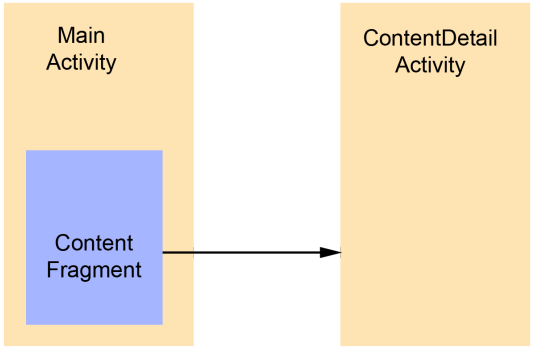


Abb1 - App-Struktur mit Activities und Fragments[8]

Diese Designentscheidung wurde aus mehreren Gründen getroffen. Hätte man z.B. bei den Navigationspunnkten jeweils immer eine neue Activity gestartet statt das Fragment ausgetauscht, hätte die Animation der Navigation nicht funktioniert, da ein neues »Fenster« aufgeht. Auch dass die Detail-Seiten als separate Activity implementiert wurden hat seine Berechtigung: Im AndroidManifest kann man nämlich sogenannte Parent-Activities von Activities festlegen. Damit muss man sich nicht weiter mit dem Navigationsfluss befassen, sondern Android weiss dann, wohin man zurück will bei einem Klick auf den BackButton. Im AndroidManifest sieht das folgendermassen aus für eine Detailactivity:

```
<activity
        android:name=".activity.ScoringDetailActivity"
        android:label="@string/solved_challenges"
```

[6] https://developer.android.com/sdk/index.html (Stand 07.06.15)
[7] http://developer.android.com/guide/components/fragments.html (Stand 06.06.15)
[8] Abb1 - App-Struktur mit Activities und Fragments

```
        android:parentActivityName=".MainActivity">
        <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.oztz.hackinglabmobile.MainActivity" />
</activity>
```

Die Navigation und das Austauschen des Contentfragments wird dabei komplett von der MainActivity aus gesteuert.

5.1.3 HTTP(S)-Requests und -Posts

HTTP-Requests dürfen nicht im UI-Thread ausgeführt werden, weshalb für einen Request ein neuer Task benötigt wird. Damit das Resultat des Requests dann in der Activity oder im Fragment verarbeitet werden kann, wurde ein Callback-Interface eingeführt. Der Ablauf bei einem HTTP-Request sieht am Beispiel SpeakerFragment folgendermassen aus:

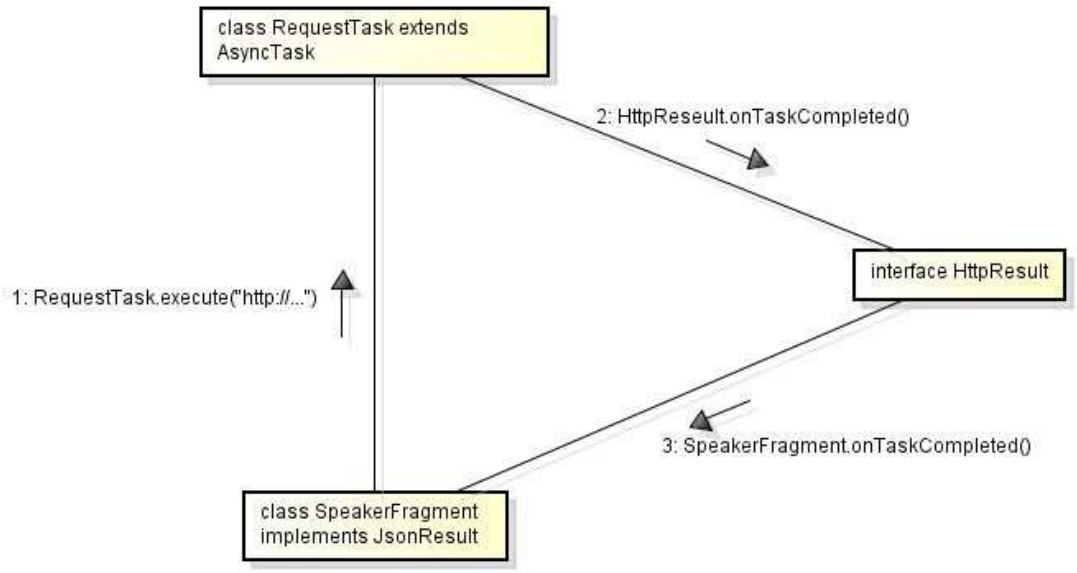


Abb2 - Ablauf von HTTP-Requests[9]

Jede Klasse, die HTTP-Requests oder -Posts senden will, muss das HttpResult-Interface implementieren. Die Klasse (in diesem Fall die SpeakerFragment-Klasse) startet dann einen neuen RequestTask welcher das HttpResult-Interface benachrichtigt, sobald der Task abgeschlossen ist. In der SpeakerFragment-Klasse kann dann das Resultat, welches normalerweise im JSON-Format vorliegt, weiterverarbeitet werden. Für HTTP-Posts wurde eine PostTask-Klasse analog zur RequestTask-Klasse geschrieben.

Auf einigen Seiten müssen mehrere Requests gleichzeitig ausgeführt werden. Damit diese beim Requests unterschieden werden können, wird bei jedem HTTP-Request ein Requestcode mitgegeben.

5.1.4 JSON-Handling

Für das verarbeiten von JSON-Inhalten wurde die Gson-Library[10] benutzt. Diese mappt sehr einfach JSON-Strings und entsprechende Modelklassen in beide Richtungen. Das sieht dann zum Beispiel folgendermassen aus:

```
News[] news = new Gson().fromJson(json, News[].class);
```

oder in umgekehrter Richtung (wird z.B. für Posts benötigt):

```
User user = (...);
String jsonString = new Gson().toJson(user);
```

[9] Abb2 - Ablauf von HTTP-Requests
[10] https://code.google.com/p/google-gson/ (Stand 07.06.15)

Wichtig für das korrekte Mapping ist der exakt gleiche Name der einzelnen Klassenvariablen beim JSON und der Modelklasse (Case Sensitive).

5.1.5 Layouts

Sämtliche XML-Layoutfiles befinden sich im Ordner res/layout. Da dieser Ordner leider keine Subfolders unterstützt, ist nur eine flache Hierarchie möglich. Die Layoutfiles wurden darum nach dem Schema »typ_name« benannt. Wobei typ entweder »activity«, »fragment« oder »item« ist. Ein Layoutfile vom Typ »item« ist dabei immer ein Item einer ListView.

5.1.6 Push-Notifications

Für die Push-Notifications wurde der Google Cloud Messaging (GCM) Service[11] verwendet. Damit der User Pushmeldungen der Confoxy-App überhaupt erhält, muss er sein Gerät bzw. seinen User beim Confoxy Server registrieren. Dies geschieht direkt beim ersten Appstart bei der Registrierung seines Benutzernamens.

Ist der Benutzer für den Push-Service registriert, bekommt er zukünftig alle Pushnachrichten von Confoxy. Das Verarbeiten der Pushnachrichten geschieht dann in der GcmMessageHandler-Klasse, die sich im push-Package befindet.

5.1.7 Application-Wrapper-Klasse

Viele Funktionen wie z.B. der Zugriff auf sharedPreferences[12] oder das Anzeigen eines Toasts benötigen einen ApplicationContext welcher normalerweise durch die Activity gegeben ist. Benötigt man den Context jedoch ausserhalb einer Activity, wird es schwierig. Man kann zwar von der Activity aus den Context an die Klasse übergeben, jedoch kann dies mühsam werden, wenn mehrere Klassen involviert sind und der Context über mehrere Stationen übergeben werden müsste.

Da gerade die sharedPreferences, die z.B. den Benutzernamen des Users enthalten, häufig ausserhalb der Activity gebraucht werden, brauchte es eine andere Lösung. Nach ein bisschen recherchieren wurde auf StackOverflow[13] eine elegante Lösung gefunden: Es wird eine Klasse erstellt, welche die Applicationklasse erweitert. Neben dem Context können dort auch sonstige statische Variabeln gespeichert werden. So hat man von überall her Zugriff auf diese Variabeln. In Confoxy heisst diese Wrapper-Klasse »App« und befindet sich im helper-Package.

5.1.8 Agenda

Für die Agenda wurde eine Ansicht benötigt, die einen Tag mit verschiedenen Kalendereinträgen anzeigen kann. Android bietet lediglich die Möglichkeit, mit dem Android-Kalender zu interagieren, jedoch keine View die vom Userkalender losgelöst ist. So musste eine proprietäre Library her, welche mit der »Android Week View«[14] von alamkanak auch gefunden wurde. Alternativen zu dieser Library wurden leider keine entdeckt.

Obwohl die Library Week View heisst, ist es auch möglich nur einen Tag anzuzeigen. Es tauchte allerdings das Problem auf, dass es nicht möglich ist zu bestimmen, wie die Events angeordnet werden. Das Ziel wäre gewesen, dass diejenigen Events, die im selben Raum stattfinden und damit diesselbe Farbe haben, auch untereinander platziert werden. Leider ist dies aber mit den gegebenen Library-Funktionen nicht möglich und auch die Reihenfolge in welcher die Events hinzugefügt werden hat keinen Einfluss auf die Zeichnungsreihenfolge. So kann die Ansicht, welche die Events aller Räume anzeigt im ungünstigsten Fall auch folgendermassen aussehen:

[11] https://developers.google.com/cloud-messaging/gcm (Stand 07.06.15)
[12] http://developer.android.com/guide/topics/data/data-storage.html#pref (Stand 08.06.15)
[13] http://stackoverflow.com/questions/2002288/static-way-to-get-context-on-android
[14] https://github.com/alamkanak/Android-Week-View (Stand 07.06.15)

Abb3 - Screenshot Agenda mit falscher Anordnung[15]

Für das Problem wurde auch ein Issue[16] erstellt, jedoch gibt es (Stand 07.06.2015) keine andere Lösung als die Library umzuschreiben. Dazu fehlte aber im Rahmen dieser Bachelorarbeit die Zeit.

5.1.9 Caching und lokale Datenbank

Es wurde entschieden, die Daten von confoxy möglichst zu cachen, so dass die meisten Inhalte auch offline verfügbar bleiben (sofern sie schon einmal abgerufen wurden). Das hat mehrere Vorteile, so kann z.B. schon Inhalt angezeigt werden, wenn der Request noch läuft. Bei jedem HTTP-Request wird der Content in der lokalen Datenbank abgespeichert. Wird in der App das Fragment gewechselt, so wird immer zuerst der gecachte Inhalt geladen und gleichzeitig ein HTTP-Request abgesetzt. Sobald der HTTP-Request den vollständigen Inhalt empfangen hat, wird der Inhalt aktualisiert. Gecached werden dabei alle Seiten ausser das Voting, welches nur aktuelle Inhalte anzeigen sollte. Alle anderen Seiten sind auch offline verfügbar.

5.1.10 Listview und Lazy Loading der Bilder

Eine Herausforderung stellte das Laden von Bildern aus dem Internet in eine ListView dar. Dazu eine kurze Erklärung, wie die Daten in eine ListView kommen: Es wird eine Adapterklasse geschrieben, die einen Array von Objekten entgegennimmt und in der definiert ist, wie die Objekt-Attribute auf die List-Item-View verteilt werden. Das Problem war nun, dass ein Objekt lediglich die URL des Bildes und nicht das Bild selbst enthielt. Nun kann man zwar nach dem Herunterladen des Bildes sagen, auf welchem ImageView das Bild angezeigt werden soll, jedoch enthält jedes List-Item ein ImageView und alle diese ImageViews besitzden die selbe ID. So würde das schlussendlich in jedem List-Item das gleiche Bild angezeigt.

Abhilfe schuf sowohl die UniversalImageLoader-Library[17], mit welcher das Implementieren von Lazy Loading von Bildern sehr einfach ist, als auch das View Holder Pattern[18] mit welchem man nachträglich jeden Bestandteil eines List-Items eindeutig identifizieren kann. Dies geschieht durch das Erstellen einer statischen ViewHolder-Klasse, welche alle Views eines Items hält:

```
private static class ViewHolder {
        TextView name;
        ImageView flag;
        ImageView speakerImage;
}
```

[15] Abb3 - Screenshot Agenda mit falscher Anordnung
[16] https://github.com/alamkanak/Android-Week-View/issues/131 (Stand 07.06.15)
[17] https://github.com/nostra13/Android-Universal-Image-Loader (Stand 08.06.15)
[18] http://developer.android.com/training/improving-layouts/smooth-scrolling.html#ViewHolder (Stand 08.06.15)

Die Views eines List-Items werden nun diesen ViewHolder-Attributen zugewiesen. Bevor nun ein List-Item an einer bestimmten Position zurückgegeben wird, wird dem Item das ganze ViewHolder Objekt angehängt z.B. als Tag. So ist das Item auch nachträglich noch eindeutig identifizierbar und das Bild wird an der richtigen Stelle angezeigt. Das sieht dann ungefähr folgendermassen aus (vereinfacht):

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    LayoutInflater inflater = LayoutInflater.from(context);
        View v = inflater.inflate(R.layout.item_speaker, null);
        ViewHolder holder = new ViewHolder();
    holder.name = (TextView) v.findViewById(R.id.speaker_name);
        holder.name.setText(item.name);
        holder.speakerImage = (ImageView) v.findViewById(R.id.speaker_portrait);
        if(item.media != null && item.media.length() > 0){
                imageLoader.displayImage(item.media, holder.speakerImage);
        }

        v.setTag(holder);
    return v;
}
```

5.1.11 QR-Code und Rollen

Um einem User bestimmte Rollen wie z.B. »Jurymitglied« zuteilen zu können, müssen sich diese irgendwie authentisieren können. Aus Usability-Gründen fiel die Entscheidung darauf, dem entsprechenden User ein Secret in Form eines einmalig verwendbaren QR-Codes zu übermitteln. Dieses Secret wird nach dem Einlesen auf seinem Gerät abgespeichert und bei jedem relevanten HTTP-POST mitgeschickt.

Damit die QR-Codes eingelesen werden können, wurde die ZXing Android Embedded[19] Library verwendet, welche wiederum das Core-ZXing von Google verwendet. Der Embedded QR-Code Reader hat den Vorteil, dass man QR-Codes einlesen kann, ohne dass ein Barcodescanner schon auf dem Gerät installiert sein muss.

Da der QR-Code-Reader nur für wenige bestimmte Benutzer relevant ist, sollte dieser für alle anderen Benutzer versteckt werden. Das Ziel war es, dass der Benutzer eine »geheime« Geste durchführt, welche den QR-Code-Reader öffnet. Die Geste an sich soll nicht sicherheitsrelevant sein, das heisst, falls jemand die Geste herausfindet ergeben sich dadurch keine Sicherheitslücken. Das Verstecken der Geste hat lediglich den Zweck, den normalen Benutzer nicht zu verwirren, da er nicht weiss wofür er einen QR-Code einlesen muss. Als geheime Geste wurde ein Triple-Tap rechts oben in der MainActivity auf der ActionBar festgelegt. Da Android diese Geste nicht kennt, musste diese selber implementiert werden. Bei jedem Tap in der gewünschten Zone wird folgende Methode aufgerufen:

```
@Override
public boolean onOptionsItemSelected(MenuItem item){
        if(item.getItemId() == R.id.enable_qrCode){
                int minIndex = 0;
                for(int i=1; i<tripleTap.length; i++){
                        if(tripleTap[i] < tripleTap[minIndex]){
                                minIndex = i;
                        }
                }
                tripleTap[minIndex] = Calendar.getInstance().getTimeInMillis();

        //Java Modulo can produce negative numbers
        int index = (((minIndex - 2) % tripleTap.length) + tripleTap.length)
                % tripleTap.length;

        long diff = tripleTap[minIndex] - tripleTap[index];
        if(diff < 600){
                startScan();
        }
}
```

Wie man sieht, werden immer die Timestamps der letzten drei Klicks gespeichert. Liegt der älteste weniger als 600ms vom neusten auseinander, das heisst wurden drei Taps innerhalb von 600ms ausgeführt, so wird der QR-Code-Scanner

[19] https://github.com/journeyapps/zxing-android-embedded (Stand 08.06.15)

gestartet. Die 600ms sind dabei ein Erfahrungswert von mehreren Messungen. Im Durchschnitt benötigt ein TripleTap zwischen 450 und 500 ms. Was durchaus erstaunlich ist, ist dass die Modulo-Funktion von Java im Gegensatz zu den meisten anderen Programmiersprachen negative Zahlen zurückgibt. Darum wird bei der Indexberechnung ein kleiner Mathematischer Kniff angewendet um den positiven Modulo zu bekommen.

5.1.12 Verwendete Libraries

Die verwendeten Libraries sind teilweise als JAR im Ordner app/libs zu finden und teilweise direkt als dependency im build.gradle File.

- Gson
  Meistverwendete Java-Library zum Handling von JSON
  https://code.google.com/p/google-gson/
  Apache License 2.0

- UniversalImageLoader
  Häufig verwendete Library um Bilder aus dem Internet in der App anzuzeigen
  https://github.com/nostra13/Android-Universal-Image-Loader
  Apache License 2.0

- Android Week View
  Einzige gefundene Library, mit welcher eine Kalender-Tagesansicht angezeigt werden kann
  https://github.com/alamkanak/Android-Week-View
  Apache License 2.0

- ZXing
  Meistverwendete Bar-Code-Library
  https://github.com/zxing/zxing
  Apache License 2.0

- ZXing Android Embedded
  Erlaubt das lesen von Bar-Codes ohne dass ZXing auf dem Gerät installiert sein muss.
  https://github.com/journeyapps/zxing-android-embedded
  Apache Licenzse 2.0

5.2 Testing und Resultate

Sämtliche Tests wurden anhand der zu Beginn festgelegten Anforderungen und Use Cases durchgeführt. Da wenig Businesslogik und viel Interaktion mit dem User getestet werden musste, wurde vor allem auf Systemtests gesetzt.

5.2.1 Test UCM 1 - Register

Die Registrierung konnte ohne Probleme umgesetzt werden. Die erlaubten Zeichen werden lokal mit einem Regex überprüft und auch ob der Name schon vergeben ist, wird erkannt, sobald der User versucht seinen Namen zu registrieren. Der Server gibt dann nämlich eine entsprechende Fehlermeldung zurück, welche von der App interpretiert wird.

Bestehende bekannte Probleme: keine

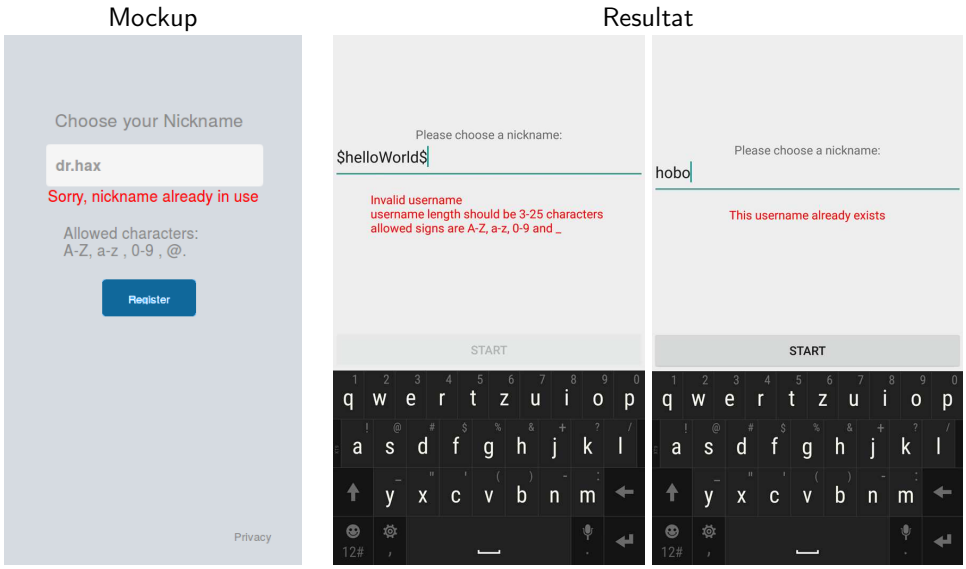


Abb4 - Mockup-Resultat-Vergleich Registrierung[20]

5.2.2 Test UCM 2 - Eventauswahl

Die Eventauswahl wird nur angezeigt, wenn kein Event oder mehr als ein Event aktiv ist. Ansonsten wird automatisch der einzige Event genommen und die Startseite angezeigt. Auch diese Seite konnte ohne Probleme implementiert werden. Nachdem ein Event ausgewählt wurde wird dieser als Umgebungsvariable gespeichert (Siehe Kapitel 5.1.7).

Bestehende bekannte Probleme: keine

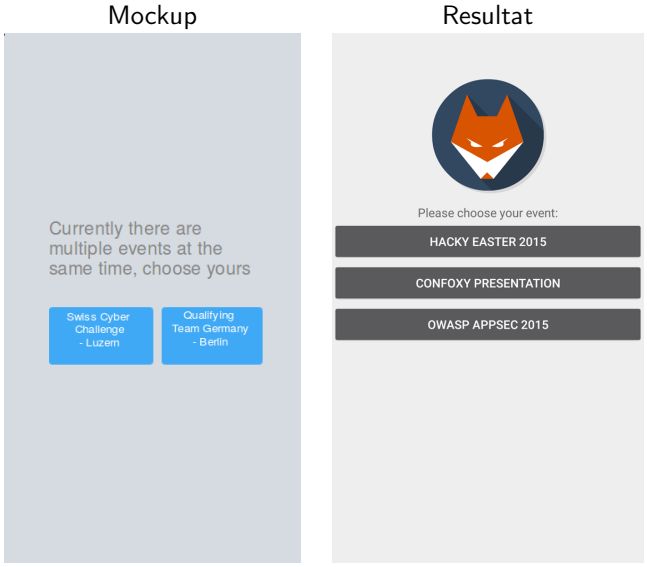


Abb5 - Mockup-Resultat-Vergleich Eventauswahl[21]

5.2.3 Test UCM 3 - News

Als Startseite nach der Eventauswahl werden die News angezeigt. Diese bestehen entweder nur aus Text oder aus Bild und Text. Hier stellte sich insbesondere das Lazy Loading von Bildern als Herausforderung heraus (Siehe Kapitel 5.1.10). Gegenüber den Mockups wurde das Layout eines Artikels mit Bild ein wenig angepasst, aus dem einfachen Grund, dass das Mockup-Layout nur mit quadratischen oder fast quadratischen Bildern gut aussieht. In der Implementation wird das Bild ganz einfach unterhalb des Textes angezeigt.

Bestehende bekannte Probleme: keine

[20] Abb4 - Mockup-Resultat-Vergleich Registrierung
[21] Abb5 - Mockup-Resultat-Vergleich Eventauswahl

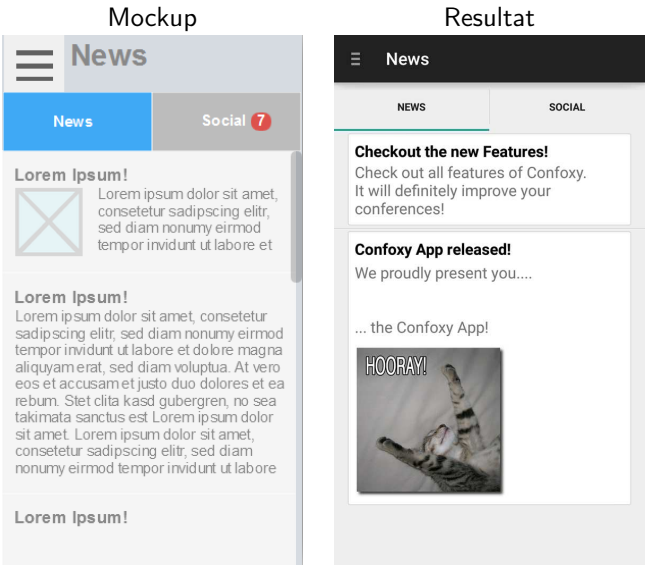


Abb6 - Mockup-Resultat-Vergleich News[22]

5.2.4 Test UCM 4 - Social Wall

Die Social Wall ist von der Implementation her genau gleich umgesetzt wie die News, es gilt darum das Gleiche wie im vorigen Kapitel.

Bestehende bekannte Probleme: keine

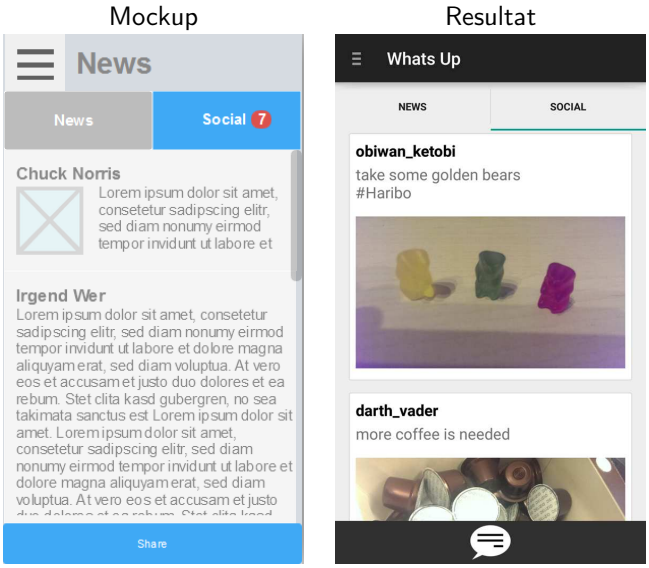


Abb7 - Mockup-Resultat-Vergleich Social Wall[23]

5.2.5 Test UCM 5 - Share

Das Sharing war eine der grössten Herausforderungen im Rahmen dieser Bachelorarbeit. Vor allem bis der Media-Upload mit einem Multipart Form Data Post funktionierte, dauerte es eine ganze Weile. Damit der Upload nicht zu lange dauert, werden die Bilder vor dem Upload auf eine Maximal-Breite bzw. -Höhe von 1000Px komprimiert. Damit sind die Bilder im Schnitt rund 50kB gross und sind auf Mobilen Screens immer noch genügend scharf. Zum Schluss gab es Probleme mit vereinzelten Geräten sowohl beim Foto schiessen, als auch beim Bild aus einem Album auswählen, welche jedoch alle gelöst werden konnten. Getestet wurde das Sharing mittlerweile auf rund 10 verschiedenen Geräten (Tablets und Smartphones).

[22] Abb6 - Mockup-Resultat-Vergleich News
[23] Abb7 - Mockup-Resultat-Vergleich Social Wall

Aus Zeitgründen musste das optionale Feature Video-Upload im Rahmen dieser Arbeit gestrichen werden. Der Upload an sich wäre zwar relativ einfach zu implementieren gewesen, jedoch wäre die Komprimierung sowie die korrekte Anzeige und Einbindung ziemlich aufwendig geworden. Dieses Feature kann aber natürlich in Zukunft noch nachgerüstet werden.

Bestehende bekannte Probleme: kein Video Sharing

Mockup Resultat

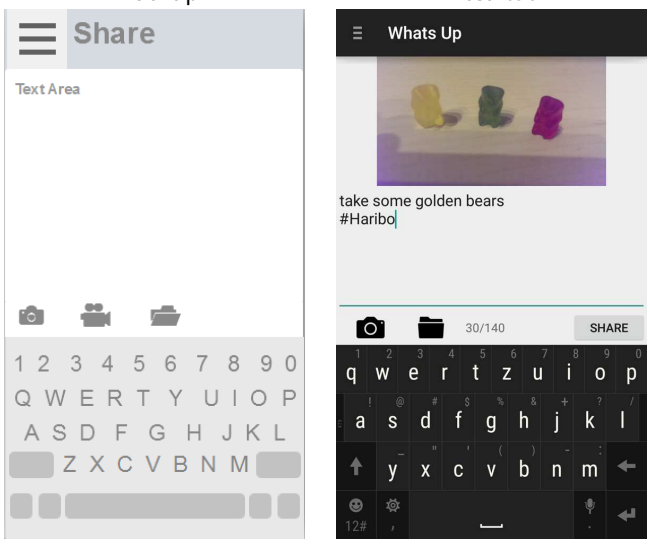


Abb8 - Mockup-Resultat-Vergleich Share[24]

5.2.6 Test UCM 6 - Conference

Die Conference-Seite ist die einfachste Seite innerhalb der App, da sie lediglich eine Beschreibung des Events enthält. Gegenüber dem Mockup ist aber kein Bild vorhanden. Dieses ging im Verlauf der Entwicklung leider vergessen und ist in der Datenstruktur serverseitig noch nicht implementiert.

Bestehende bekannte Probleme: kein Event-Bild

Mockup Resultat

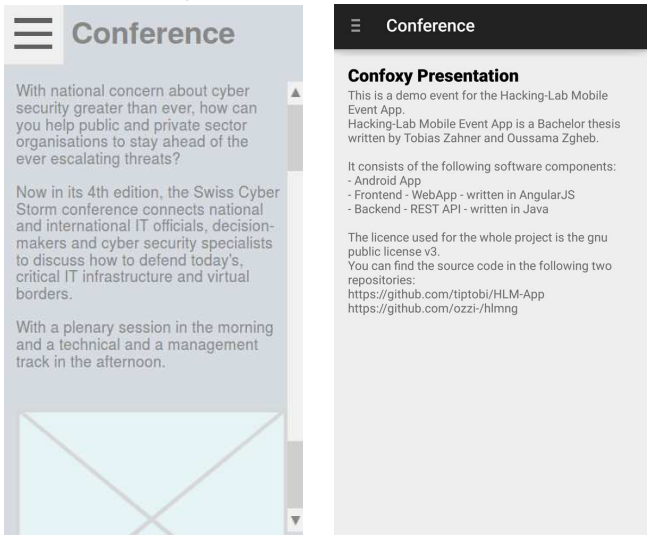


Abb9 - Mockup-Resultat-Vergleich Conference[25]

[24] Abb8 - Mockup-Resultat-Vergleich Share
[25] Abb9 - Mockup-Resultat-Vergleich Conference

5.2.7 Test UCM 7 - Agenda

Die Agenda gehörte ebenfalls zu den kompliziertesten Sachen und konnte nicht vollständig den Vorstellungen entsprechenden gelöst werden. Die verwendete Library ist zwar im Grossen und Ganzen sehr gut, hat jedoch einige Tücken, was vor allem viel Zeit während der Arbeit verbrauchte. Schlussendlich konnten alle Probleme gelöst werden bis auf jenes, dass die Anordnung der Eventitems in der Übersicht nicht bestimmt werden kann. Dadurch befinden sich Events im gleichen Raum in der Übersicht nicht zwingend immer untereinander, wie das im Mockup vorgesehen war. Siehe dazu auch Kapitel 5.1.8. Bei einem Tap auf ein Event-Item öffnet sich wie in den Anforderungen spezifiziert die Detail-Ansicht des Even-Items.

Gegenüber dem Mockup wurde das »doppelte« TabLayout durch ein einfaches ersetzt, da sonst zu wenig Platz für den Kalender blieb. Stattdessen wechselt man den Tag, indem man im Kalender nach links oder rechts wischt.

Bestehende Bekannte Probleme: Zeichnungsreihenfolge kann in der Raumübersicht nicht beeinflusst werden.

Mockup Resultat

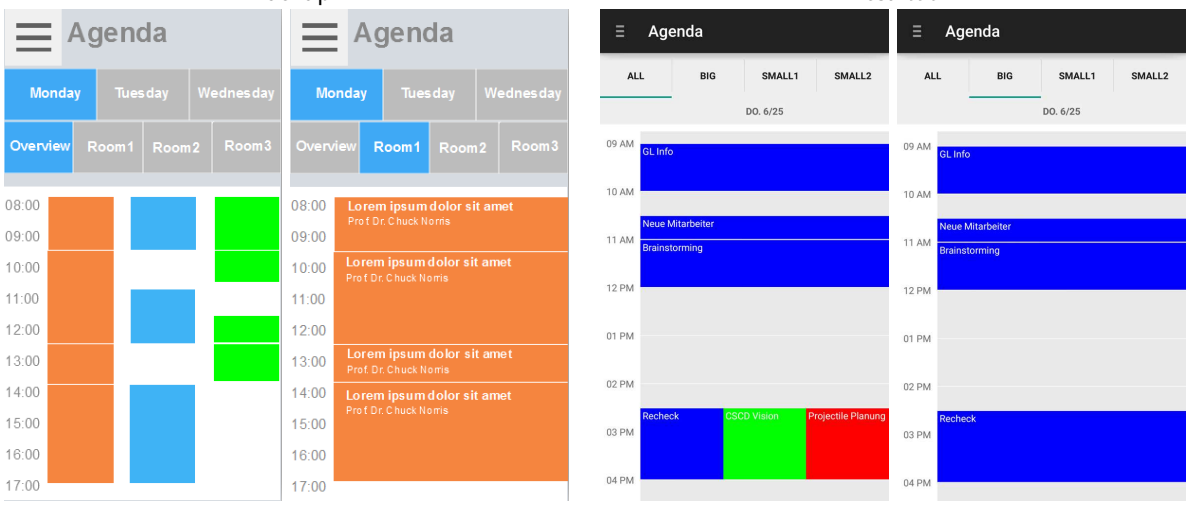


Abb10 - Mockup-Resultat-Vergleich Share[26]

5.2.8 Test UCM 8 - Speaker

Beim Speaker wurde ebenfalls ein Lazy Loading der Bilder implementiert. Ansonsten bot diese Seite wenig Komplikationen und das Mockup konnte gut umgesetzt werden.

Bestehende bekannte Probleme: keine

Mockup Resultat

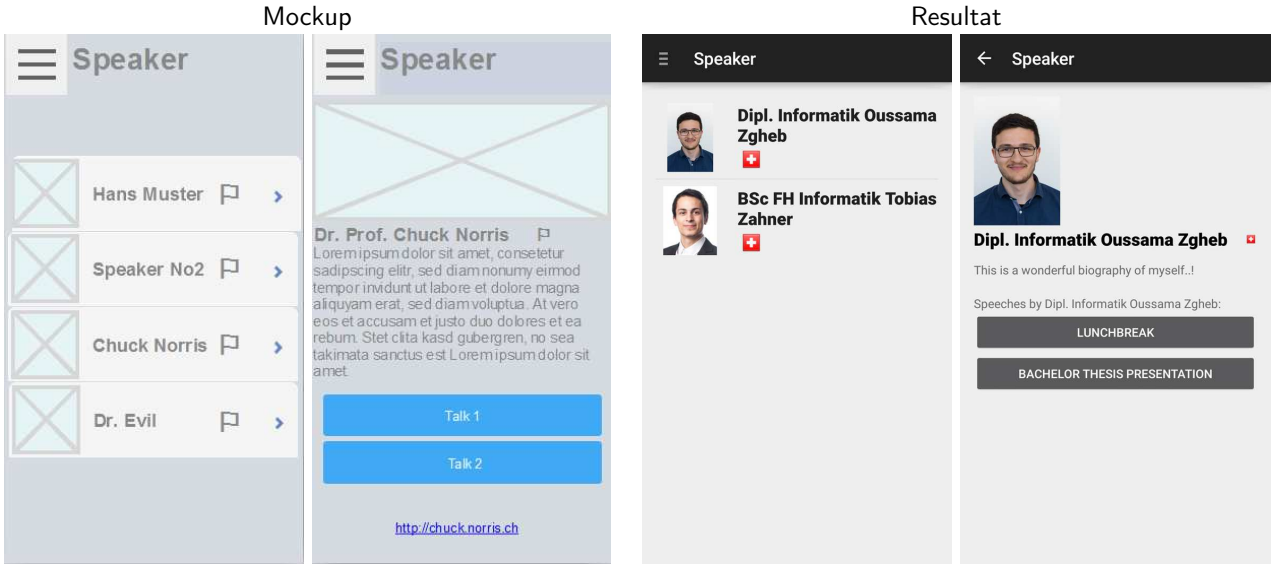


Abb11 - Mockup-Resultat-Vergleich Speaker[27]

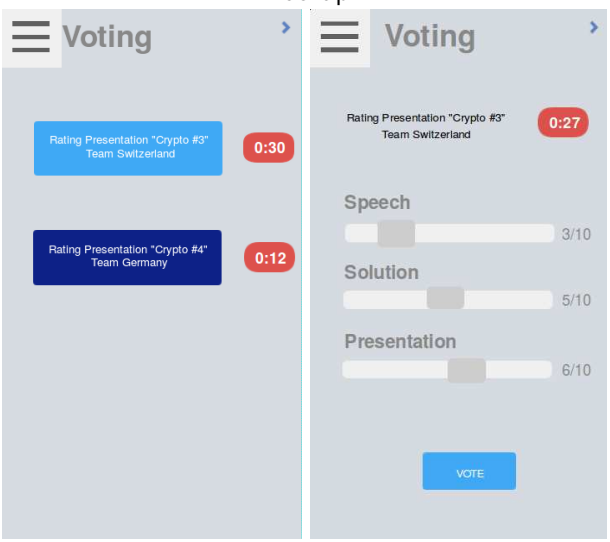
[26] Abb10 - Mockup-Resultat-Vergleich Share
[27] Abb11 - Mockup-Resultat-Vergleich Speaker

5.2.9 Test UCM 9 - Voting

Beim Voting war die ursprüngliche Idee, dass die App via Push benachrichtigt wird, wenn ein Voting beginnt, das heisst mit der Pushmeldung würde das Voting starten. Dieser Gedanke musste im Verlauf der Entwicklung aber verworfen werden, da nicht garantiert werden konnte, dass alle User den Push gleichzeitig erhalten. Stattdessen benachrichtigt die Pushnachricht den Benutzer nur noch, dass ein neues Voting startet und verlinkt den User mit dem Voting. Befindet sich der User dann auf der Voting-Seite wird die verbleibende Zeit vom Server abgerufen. Ansonsten konnte das Voting ebenfalls fast genau so vom Mockup und den Anforderungen übernommen werden. Als einzige Änderung wird der Countdown lediglich auf der Voting-Detailseite angezeigt. Auf der Votingübersichtsseite wäre dieser sehr umständlich zu implementieren gewesen und hätte in keinem Verhältnis zum Nutzen gestanden, da der Benutzer von der Push-Meldung sowieso direkt auf die Voting-Detail Seite verlinkt wird.

Bestehende bekannte Probleme: Kein Countdown in der Votingübersicht

Mockup



Resultat

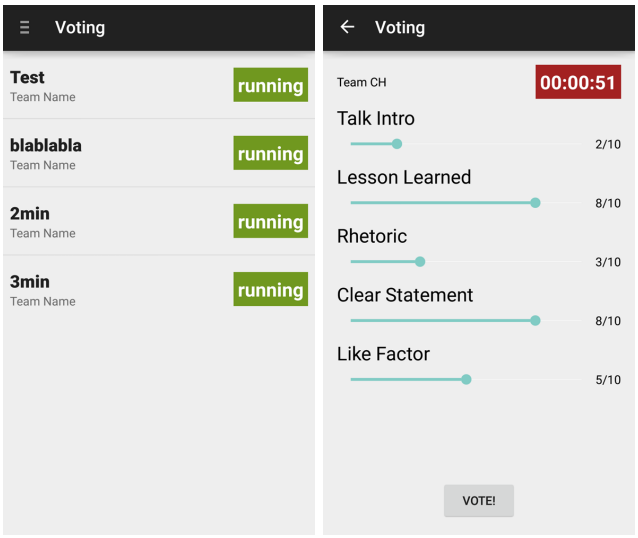


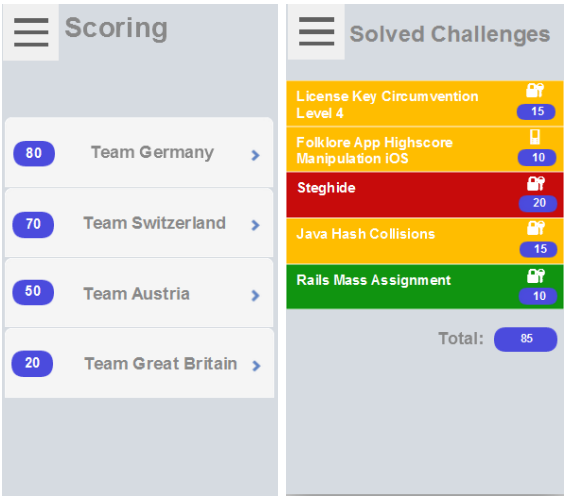
Abb12 - Mockup-Resultat-Vergleich Voting[28]

5.2.10 Test UCM 10 - Scoring

Die Scoring-Seite konnte ziemlich genau vom Mockup übernommen werden, lediglich das Design wurde etwas angepasst, da die komplett eingefärbten Einträge nicht so toll aussahen.

Bestehende bekannte Probleme: keine

Mockup



Resultat

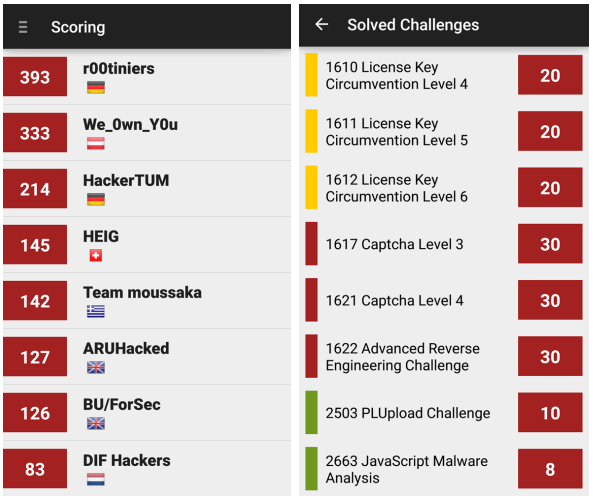


Abb13 - Mockup-Resultat-Vergleich Scoring[29]

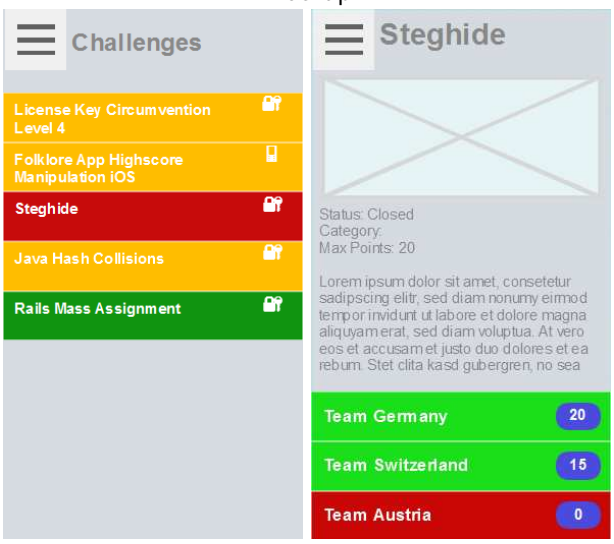
[28] Abb12 - Mockup-Resultat-Vergleich Voting
[29] Abb13 - Mockup-Resultat-Vergleich Scoring

5.2.11 Test UCM 11 - Challenges

Die Challenges-Seite konnte nicht ganz wie ursprünglich geplant umgesetzt werden. Wie beim Scoring auch, wurde das Design der Übersichtsseite angepasst. Die Challenge-Detail-Seite war leider nicht so wie im Mockup realisierbar, dies aus dem Grund, dass die dafür benötigten Daten auf Seiten des Hacking-Labs nicht vorhanden waren. Die Beschreibung der Challenges liegt im Moment nämlich nur in Form eines Videos vor[30]. Statt einer Beschreibung wird also beim Klick auf eine Challenge einfach das dazugehörige Video abgespielt.

Bestehende bekannte Probleme: Nicht alle Videos können abgespielt werden, da einige Codecs von Android nicht unterstützt werden

Mockup



Resultat

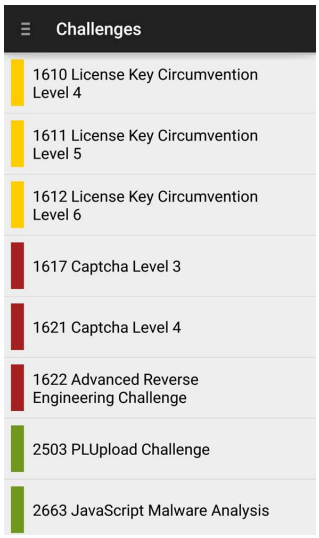


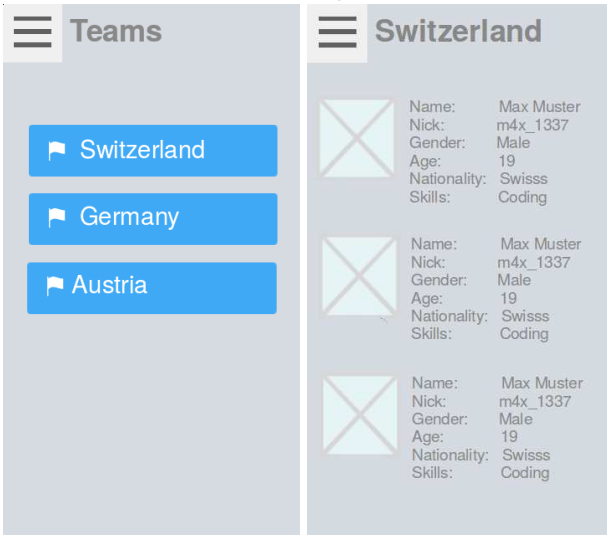
Abb14 - Mockup-Resultat-Vergleich Challenges[31]

5.2.12 Test UCM 12 - Teams

Die Teams-Seite konnte ohne grosse Probleme implementiert werden. Auch auf der Team-Detail-Seite war ein Lazy Loading der Bilder nötig, da das jedoch zuvor schon implementiert wurde, war auch das kein Problem.

Bestehende bekannte Probleme: keine

Mockup



Resultat

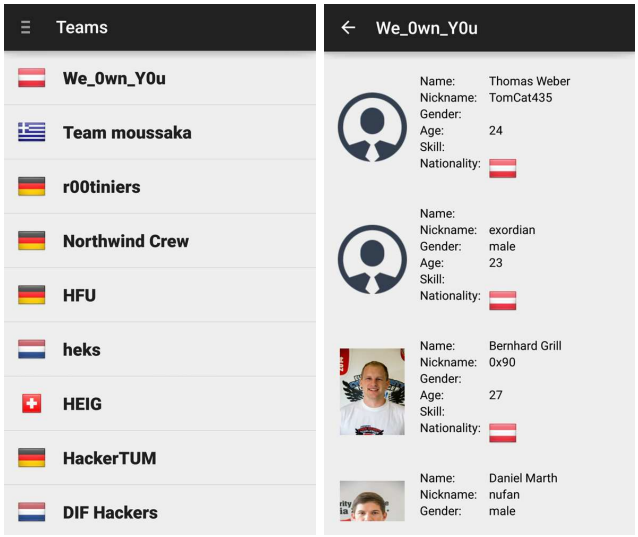


Abb15 - Mockup-Resultat-Vergleich Teams[32]

[30] Siehe dazu Mail »M_TZ_002_RE Fragen« im Anhang
[31] Abb14 - Mockup-Resultat-Vergleich Challenges
[32] Abb15- Mockup-Resultat-Vergleich Teams

5.2.13 Test UCM 13 - Rollen und QR-Codes

Der TripleTap rechts oben in der MainActivity wird sauber erkannt und der QR-Code-Reader wird ausgeführt. Wenn ein QR-Code erfolgreich eingelesen wurde, bekommt der Benutzer eine Informationsmeldung über seinen neuen Status.

Bestehende bekannte Probleme: keine

5.2.14 Test der nichtfunktionalen Anforderungen

Einige Anforderungen and die App werden nicht durch die Usecases abgedeckt. Diese werden hier nochmals überprüft:

- Die Mobile App soll den Titel »Confoxy« tragen: korrekt
- Die App soll für das Betriebssystem Android entwickelt werden und Material Design verwenden: Wurde so implementiert
- Die App soll in englischer Sprache umgesetzt werden: korrekt
- Es kann pro User pro Voting nur eine Stimme abgegeben werden: Wird serverseitig geprüft, der Benutzer bekommt aber ein Feedback falls er es versuchen sollte.

5.2.15 Verwendete Geräte

Während der Entwicklung wurd die App laufend sowohl auf einem HTC One Smartphone mit Android 5.0.2 als auch auf einem Nexus 7 Tablet mit Android 4.4.2 getestet. Für vereinzelte Tests konnten auch zusätzliche Geräte aufgetrieben werden, jedoch war auf diesen kein ausführliches Testing möglich.

5.3 Reporting

Alle präsentierten Metriken und Statistiken in diesem Kapitel beziehen sich nur auf den src-Folder. Damit wird nur der effektiv selber programmierte Code betrachtet und es werden z.B. die importierten Libraries oder die generierten Files ignoriert.

Tools Zur Erstellung der Metriken und Statistiken wurden neben den in Android Studio integrierten Tools folgende Plugins verwendet:

- Metrics Reloaded
  https://plugins.jetbrains.com/plugin/93?pr=idea
- Statistic
  https://plugins.jetbrains.com/plugin/?idea&id=4509

Da diese Tools nur sehr unübersichtlich und vereinzelt Daten darstellen, wurden hier die wichtigsten Kennzahlen herausgeschrieben:

	Java	XML	Total
Files	63	46	109
Size [kB]	185	63	248
Lines	5014	1611	6625
Minimum Lines	8	3	11
Maximum Lines	318	154	472
Average Lines	79	35	114
Classes	99		99
Average Cyclomatic Complexity	2,1		2,1
Aveage Essential Cyclomatic Complexity	1,27		1,27
Average Number of Dependencies per class	5,2		5,2
Cyclic Dependencies	0		0

Abb16 - Metriken und Statistiken[33]

[33] Abb16 - Metriken und Statistiken

6 Teil Front & Backend

Datum	Version	Änderung	Autor
29.05.15	1.0	Intial	Oussama Zgheb
03.06.15	1.1	Qualitätsmassnahmen	Oussama Zgheb
06.06.15	1.2	Frontend	Oussama Zgheb
07.06.15	1.3	Systemarchitektur	Oussama Zgheb
08.06.15	1.4	Architekturschichten, Routing	Oussama Zgheb
09.06.15	1.5	Use Cases eingefügt, UC Diagramm	Oussama Zgheb
09.06.15	1.6	Routing Hierarchie	Oussama Zgheb

6.1 Software Architektur Frontend

Das Frontend ist eine in AngularJS geschriebene WebApp, welche das Verwalten des Datenbestand sowie das Operating der Votings ermöglicht.

6.1.1 Logische Architektur

Das Frontend setzt sich aus folgenden Komponenten zusammen:

- css
  Enthält das farblich angepasste[34] CSS von Bootstrap sowie eigene Anweisungen
- fonts
  Enthält die glyphicons von Bootstrap
- img
  Enthält alle verwendeten Grafiken des Frontend (Ausgenommen Uploads)
- js
  - controller
    Enthält alle Controller für die Models
  - factory
    Enthält allgemein verwendete Funktionen
  - lib
    Enthält die verwendeten AngularJS Modue sowie andere Libraries
- template
- index.html

Controller Jedes Model hat sein eigenes Controller Modul. In diesem Modul kann es je nach Bedarf folgende Typen von Controller geben

- NewController
  Dieser Controller soll ermöglichen, dass ein neues Objekt zu erstellen
- ListController
  Dieser Controller soll ermöglichen, dass eine Liste der vorhandenen Objekte geladen wird
- IdController
  Dieser Controller soll ermöglichen, dass ein bestehendes Objekt geladen wird und bei Bedarf aktualisiert

Ziel ist es, dass die Controller lediglich die Daten über den RestService abholen, aufbereiten und weitergeben. Diese sollen keine Businesslogik beinhalten, ausser der Sortierung von Objekten.

Factory - RestService Die Factory »RestService« soll die einheitliche und einfache Verwendung der REST API für die Controller ermöglichen. Dabei werden alle HTTP Callls mit dem »$q deferred« Modul durchgeführt, sodass nicht aktiv auf die Antwort des Servers gewartet werden muss.

Factory - ToolService Die Factory »ToolService« bietet Methoden an, welche von vielen Controllern benutzt wird. So zum Beispiel die Umleitung in einen neuen State.

6.1.2 Namenskonventionen

Die in geschweiften Klammern stehenden Ausdrücke sind dabei als Variablen zu verstehen. Die Schreibweise zeigt dabei auch gleich an, ob lowerCamelCase oder CamelCase zu verwenden ist.

- {modelType} ist der Typ des Business Model.
  Also z.B. Event, QRCode, Media, Speaker etc.

[34] Lavish Bootstrap bietet einfach anpassbare CSS Dateienhttp://lavishbootstrap.com (Stand: 10.05.15)

- {templateOperationType}ist der Typ welches das Template darstellt und ist analog dem {OperationType}. Folgende Typen gibt es dabei:
  - detail
    Die Detail Ansicht eines Objekts, bei Bedarf inkludiert dies die Edit Ansicht
  - info
    Die Info Ansicht eines Objekts
  - info edit
    Die Editor Ansicht eines Objekts als Formular
  - list
    Die Listenansicht aller Objekte, inkludiert die Info Ansicht
  - new
    Die Ersteller Ansicht eines Objekts als Formular

6.1.3 Inhalte

Um eine neue Seite dem Frontend hinzuzufügen sind folgende Dateien und Anpassungen notwendig.

- Template
  Ort: /WebContent/frontend/template/{modelType}/{modelType}-{templateOperationType}.html
  Zweck: Darstellung der Daten in HTML
- Directive
  Ort: /WebContent/frontend/js/directive.js
  Zweck: Bindeglied Template, Scope
  Inhalt: Directive Namen »{modelType}{templateOperationType}«, z.B. »socialInfoEdit«
- Controller
  Ort: /WebContent/frontend/js/controller/{name}.js
  Inhalt:
  Modul mit den Namen »{modelType}«, z.B. »event«
  Controller mit dem Namen »{ModelType}{OperationType}Controller«, z.B. »EventNewController«
- Stateprovider Eintrag
  Ort: /WebContent/frontend/js/stateprovider.js
  Zweck: Routing Pfade festlegen
- App.js Eintrag
  Ort: /WebContent/frontend/js/app.js
  Inhalt: Modul Namen in das Module »hlmngApp« hinzufügen

6.1.4 Routing

Das Routing wird durch das Modul »ui-Router« durchgeführt, dies da »ngRoute« Probleme mit nested und abstrakten Views hat und im Allgemeiner weniger flexibel ist. Jede Ansicht ist dabei einem State zugeordnet. Es wird zwischen folgenden zwei Kontexten unterschieden,

- Allgemeiner Kontext
  Hier werden Events erfasst, Speakers und Push Nachrichten verwaltet.
- Event spezifischer Kontext
  Hier werden Event Inhalte verwaltet. Diese States sind an Ihrem Namen zu erkennen, alle beginnen mit »event.active.«.

Folgende Abbildung Abb17 zeigt, was beim laden einer URL abläuft.

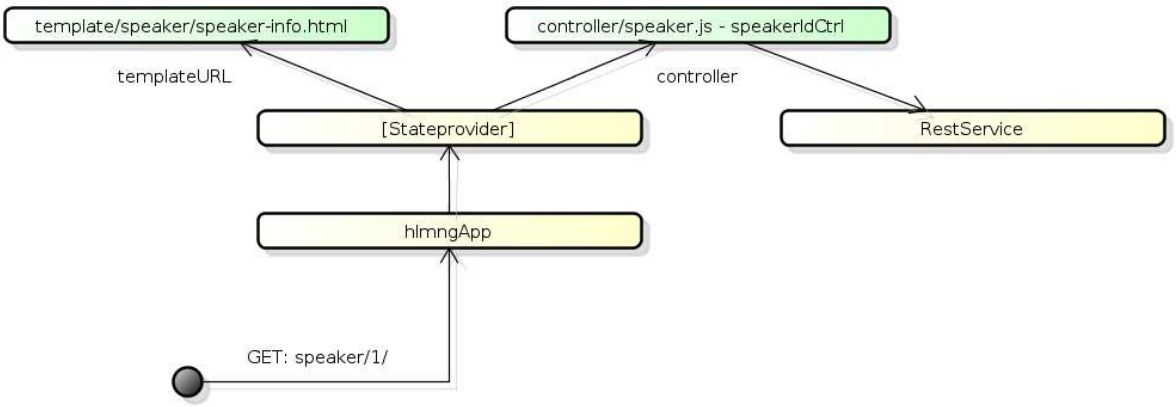


Abb17 - State Routing [35]

Folgende Abbildung Abb18 zeigt die Routing Hierarchie. Dabei ist der abstrakte Parent State immer dunkler als seine gleichfarbigen Children States eingefärbt. Links zu sehen sind die Speaker States welche sich im allgemeinen Kontext befinden. Rechts zu sehen ist der Eventspezifische Kontext.

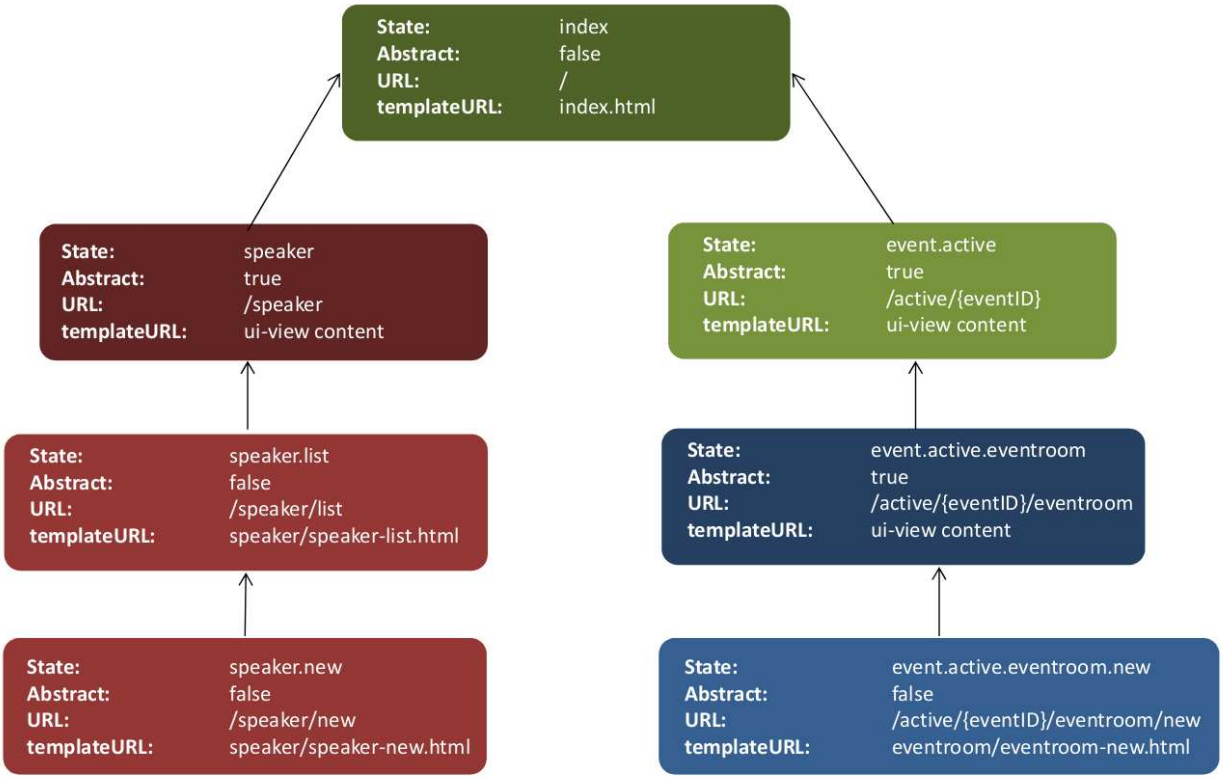


Abb18 - Routing Hierarchie [36]

6.2 Software Architektur Backend

Das Backend ist eine Java Anwendung und realisiert primär die REST API. Diese API ist grundsätzlich in zwei Bereiche aufgeteilt:

- pub - Enthält alle öffentlichen Calls
- adm - Enthält alle »privaten« Calls

[35] Abb17 - State Routing
[36] Abb18 - Routing Hierarchie

Dabei kommt es vor, dass die gleichen Calls in pub sowie adm vorkommen. Dies war zwangsläufig Nötig, da ansonsten das Frontend gewisse Calls auf pub andere wiederum auf adm ausführen hätte müssen. Eine solche Teilung der API hätte zu zusätzlicher Komplexität für alle Aufrufer geführt und wurde deshalb so vermieden.

6.2.1 Logische Architektur

Das Backend setzt sich grob aus folgenden Komponenten zusammen:

- hlmng
- db
- gcm
- testing
- log
- settings

Anschliessen werden alle Komponenten im Detail betrachtet. Folgendes Abbildung Abb19 zeigt die Systemübersicht als Pakete.

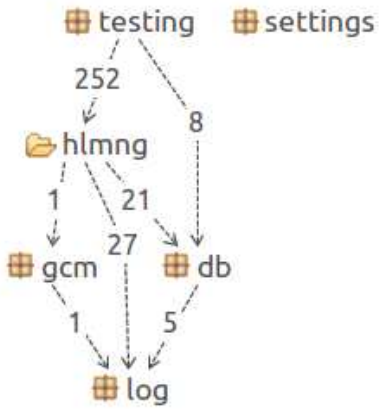


Abb19 - Package Abhängigkeiten[37]

6.2.2 Datenbank Schema

Die nachfolgende Abbildung Abb20 zeigt das gesamte Datenbank Schema in UML und biete eine Übersicht über alle Tabellen. Nachfolgend werden diese im Detail erklärt.

[37] Abb19 - Package Abhängigkeiten

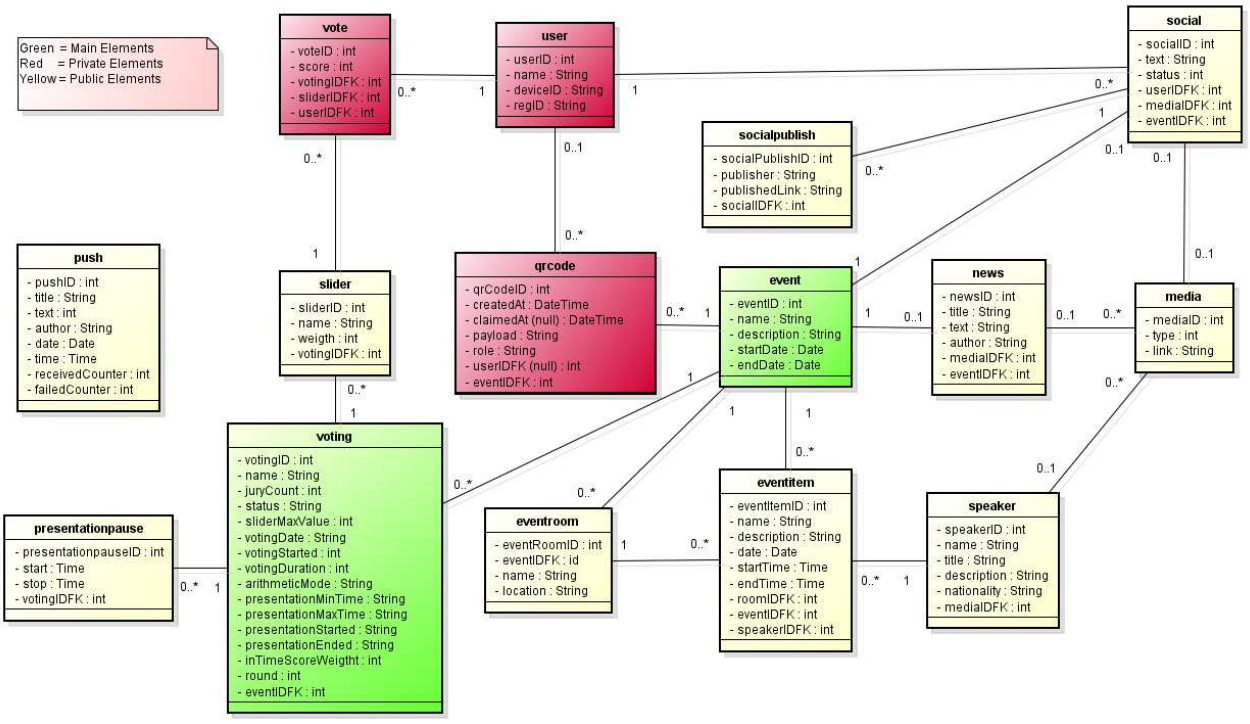


Abb20 - Datenbank Schema [38]

Die zentralen Elemente sind dabei grün hinterlegt. Die Elemente welche auf keinen Fall für den normalen Mobile App Benutzer erreichbar sein dürfen sind rot hinterlegt.

event speichert die einzelnen Events / Konferenzen und wird als Fremdschlüssel in vielen anderen Tabellen verwendet.

eventitem speichert die einzelnen Agenda Punkte, wie z.B. Vorträge und Workshops

eventroom speichert die Räume für den Event in welchem die Event Items stattfinden

media speichert den relativen Dateipfad zu den hochgeladenen Bilder und weitere Metadaten

news speichert die News Einträge

presentationpause speichert die Präsentationspausen. Dabei wird die Gesamtzeit gemessen, abzüglich der einzelnen Pausen.

Die folgende Abbildung Abb21 soll diesen Ablauf anhand einer Präsentation mit zwei Pausen verdeutlichen.

[38] Abb20 - Datenbank Schema

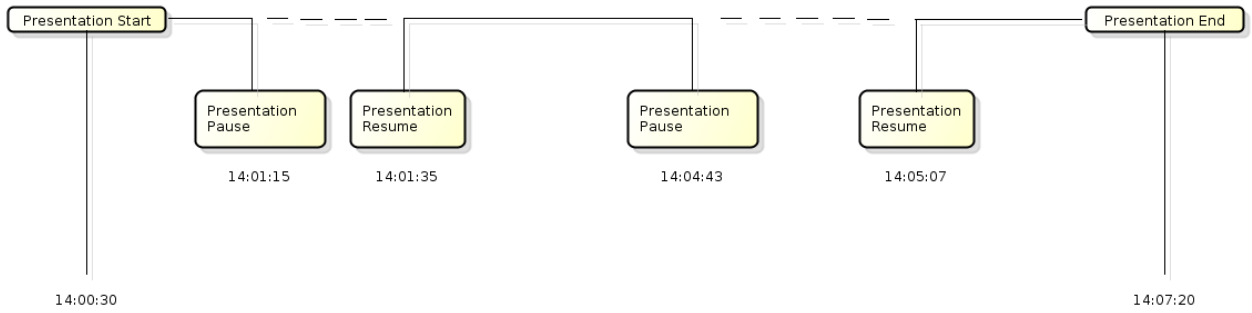


Abb21 - Präsentation Pausen [39]

Gesamte Zeit - 14:07:20 - 14:00:30 = 00:06:50

Pause #1 - 14:01:35 - 14:01:15 = 00:00:20

Pause #2 - 14:05:07 - 14:04:43 = 00:00:24

Die effektive Präsentationszeit ist demzufolge wie folgt zu berechnen:

Gesamte Zeitspanne: 6m 50s

Erste Unterbrechung 20s

Zweite Unterbrechung 24s

Berechnung: 6m 50s - 20 s - 24 = 6m 6s Gesamtzeit

push speichert die Push Einträge

qrcode speichert die QR Codes welche für die Authentisierung verwendet werden

slider speichert die einzelnen Slider, also die Bewertungskriterien pro Voting

social speichert die Social Einträge

socialpublish speichert die Links zu den auf den Sozialen Netzwerken veröffentlichen Social Einträgen

speaker speichert die einzelnen Referenten

user speichert die registrierten Mobile App Benutzer

vote speichert die einzelnen Bewertungen pro Slider

voting speichert die Votings

6.2.3 Datenbank Constraints

Die folgende Tabelle listet die Aktionen von »On Update« und »On Delete« für jede Foreign Key Reference auf.

[39] Abb21 - Präsentation Pause

Reference	On Update	On Delete
event	cascase	cascade
user	cascade	no action
media	cascade	no action
social	cascade	no action
voting	cascade	cascade
speaker	cascade	no action
event room	cascade	no action

Begründung On Update Es sollen alle Mutationen der ID's weitergegeben werden, da ansonsten inkonsistente Daten auftauchen.

Begründung On Delete Falls »no action« gewählt wurde ist dies deshalb so, weil die Objekte auch ohne FK noch von Nutzen sind oder zur Archivierung notwendig sind. Lediglich »event« und »voting« sollen beim Löschen dies weitergeben, dies da es in diesem Falle Sinn macht, einen ganzen Event / fehlgeschlagenes Test Voting zu löschen.

6.2.4 Model

Unter dem Ordner »Model« befinden sich alle Business Models. Diese enthalten bloss Konstruktoren sowie Setter und Getter. Um das Mapping zu automatisieren ist es wichtig, dass die Klasse und die darin enthaltenen Felder die gleichen Namen tragen wie im Datenbank Schema. Dabei sind die Namen in Java in UpperCamelCase anzugeben, die Tabellennamen in lowercase.

6.2.5 AuthChecker

Für gewisse sensitive Aktionen der Mobile App, wird eine Authentisierung und Authorisierung durchgeführt. Der AuthChecker ermöglicht die Überprüfung der HTTP Authorization Header sowie der selbst eingeführten »X-QRCode« Header.
Der Return Value der Methoden ist dabei ein »AuthResult« welches Informationen darüber erhält ob die Login Daten gültig waren oder falls nicht was das Problem war.
Das Ziel bei dem HTTP Authorization Header ist, zu überprüfen ob der Benutzer des Mobile App tatsächlich der ist, für den er sich ausgibt. Dies ist z.B. relevant beim Erstellen von Social Items. Das Secret / Password jedes Benutzers ist dabei die Device ID.
Das Ziel bei dem HTTP X-QRCode Header ist, zu überprüfen ob der Benutzer tatsächlich spezielle Berechtigungen erhalten soll. Das Secret ist dabei die Payload des QR Codes welcher der Benutzer einscannt.

6.2.6 Voting

Um das Frontend mit Daten zu den aktuell abgebenden Stimmen zu versorgen (Anforderung AnfF-3) wurden die dafür benötigten Calls im Backend zur Verfügung gestellt. Dabei werden die Daten nicht an das Frontend gepusht sondern ein Polling Mechanismus eingesetzt. Dies wurde deshalb so gelöst, da eine serverseitige Push Lösung zu komplex und Ressourcenintensiv wäre.

6.2.7 GCM

Diese Klassen implementieren die Push Notifikationen für die Mobile Clients.

Empfängerliste Die Empfängerliste wird zusammengestellt aus allen Einträgen der »User« Tabelle. Dort sind alle registrierten Mobile Clients inklusive RegID vorhanden. Die RegID ist die Registrierungsnummer bei dem GCM Dienst. Pushes die im Kontext eines Events stehen, werden an alle Mobile Apps gesendet, unabhängig davon, welcher Event dort aktiv ist. Dies mit der Überlegung, dass möglichst wenig Zeit vergeht für die Erstellung der Empfängerliste (minimaler Delay zum Senden des Push) und da meist sowieso nur ein Event aktiv ist (unnötige Komplexität falls pro Mobile App der aktive Event gemeldet und gespeichert werden muss).

6.2.8 Persistenz

Bei der Entwicklung wurde bewusst auf ein Framework wie z.B. Hibernate [40] verzichtet, dies da keine Erfahrung mit solchen Frameworks vorhanden ist und die Flexibilität so hoch sie möglich gehalten werden wollte. Bei falscher Verwendung (z.B. durch Unwissen) oder Bugs in der Implementierung solcher Frameworks könnten die Folgen von schlechter Performance bis hin zu gravierenden Sicherheitslücken reichen.

Die Prepared Statements werden per Reflection automatisch generiert. Dies geschieht bei der ersten Verwendung und wird danach für die restliche Laufzeitdauer der Applikation gespeichert. Jede »Model« Klasse verfügt über ein eigenes Data Access Object (DAO) welches durch den GenDaoLoader instanziiert wird. Das DAO erstellt nun mit Hilfe des QueryBuilder die Prepared Statements und speichert diese.

Das Zusammenspiel der Komponenten beim Erstellen der Statements wird im folgenden Diagramm Abb22 gezeigt.

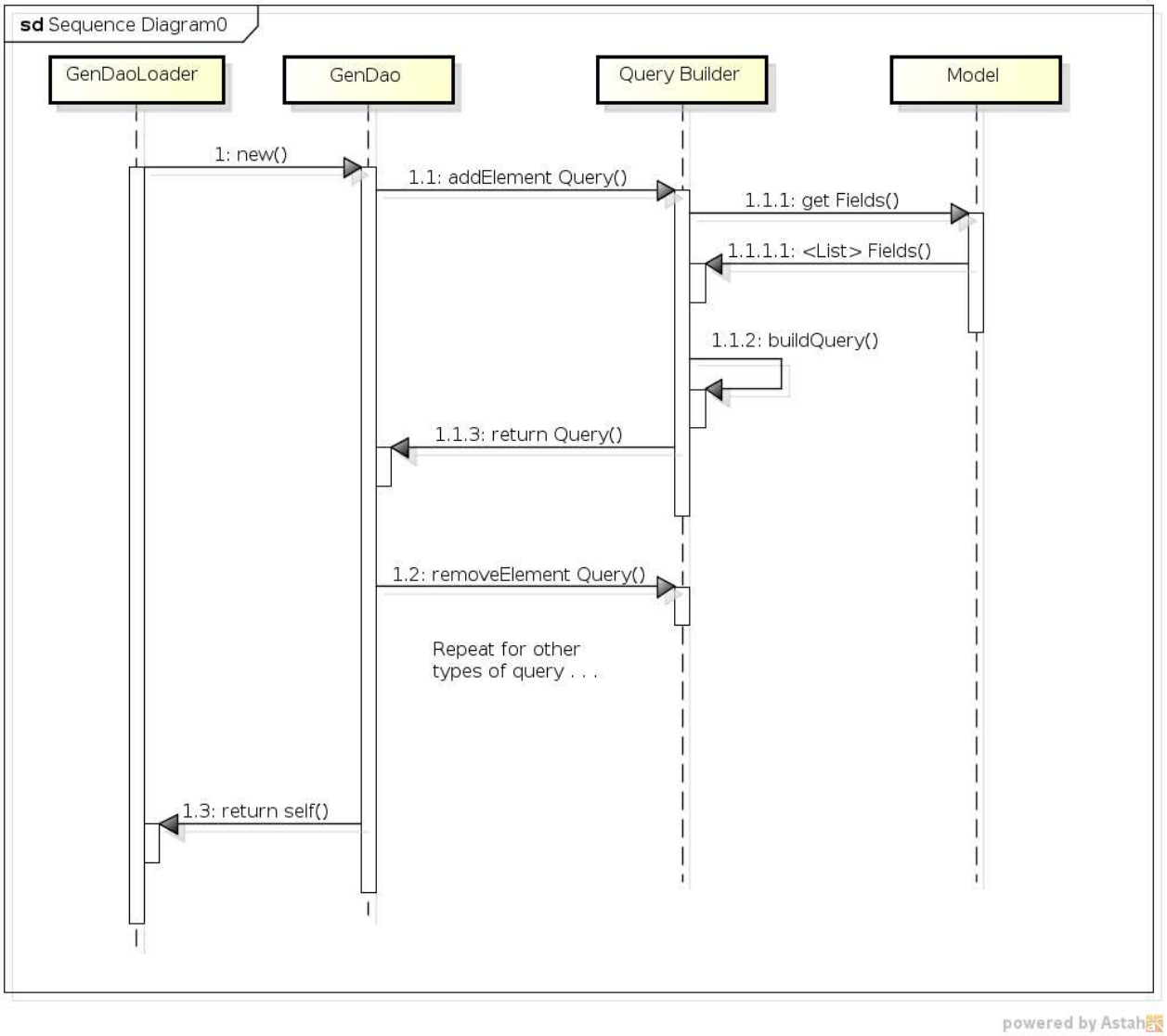


Abb22 - Ablauf der automatischen Generierung der Prepared Statements [41]

Gewisse Models benötigen zusätzliche Methoden, diese haben also einen eigenen Dao welcher den GenDao erben. Dieser Sachverhalt wird in der folgenden Abbildung Abb23 verdeutlicht.

[40] `http://hibernate.org/` (Stand: 10.05.15)
[41] Abb22 - Ablauf der automatischen Generierung der Prepared Statements

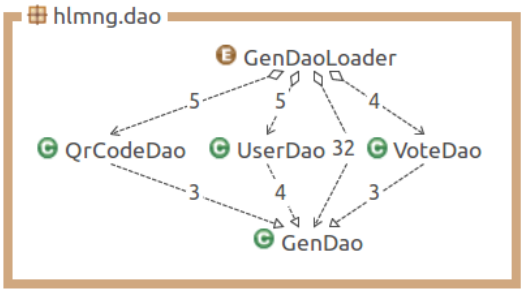


Abb23 - Vererbung der DAO Klassen [42]

6.2.9 Ressource

Die Ressourcen sind die Klassen welche die eigentliche REST Funktionalität bereit stellen. Dabei wird zwischen den »Pub« und den »Adm« Ressourcen unterschieden. Diese Trennung existiert um gewisse API Calls öffentlich, andere wiederum privat zu realisieren (Stichwort: Separation of concerns).

Die »Adm« Ressourcen werden durch das Frontend verwendet und enthalten zum Teil andere / mehr Funktionalität als die »Pub« welche z.B. von den Mobile Apps verwendet werden.

Jede Ressource hat einen Base Path welche dem Model entspricht und eine oder mehrere Methoden welche einem Call auf der REST API entsprechen. Alle diese Aktionen verwenden die dazugehörigen DAOs des Model. Der Ablauf eines REST Call der ein neues Element erstellt ist in der Abbildung Abb24 zu sehen.

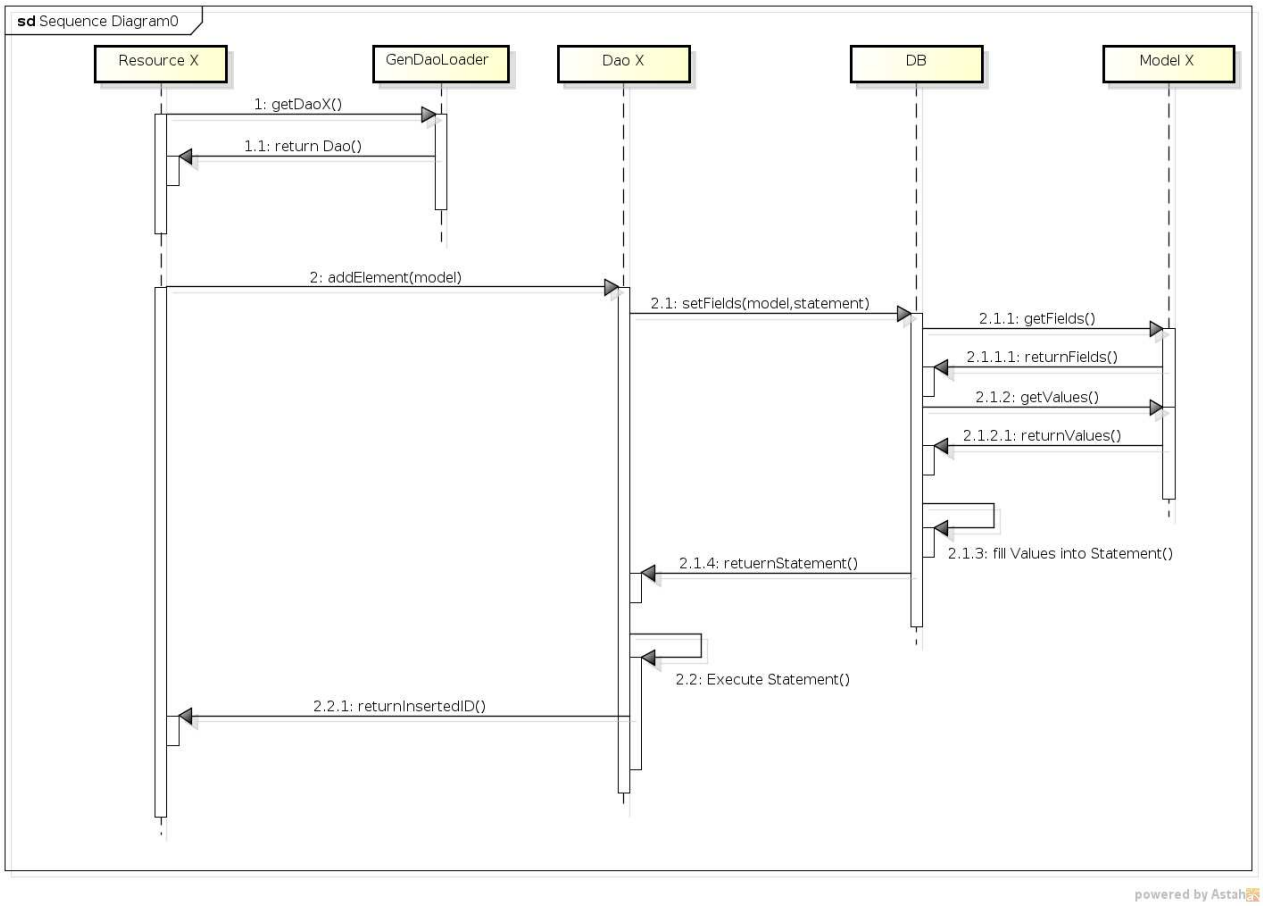


Abb24 - Ablauf einer Resourcen GET Methode[43]

[42] Abb23 - Vererbung der DAO Klassen
[43] Abb24 - Ablauf einer Resourcen GET Methode

6.2.10 Performance

Bei der ganzen Entwicklung wurde bewusst auf die Performance geachtet. Folgend zwei besonders erwähnenswerte Features:

Injizierung Um der potentiellen grossen Zahl von Benutzern standzuhalten, wurde versucht die Anzahl von Calls möglichst klein zu halten. So wurde bei gewissen Calls mit einem Fremdschlüssel das gesuchte Attribut des Fremdschlüssel Objektes injiziert. So zum Beispiel bei einem GET auf einen Speaker, dieser hat einen Foreign Key welcher auf ein Media Element zeigt. Nun müsste das Mobile App also zwei Calls durchführen um einerseits den Speaker sowie das Bild zu erhalten. Nun wurde einfach ein weiteres Feld Namens »media« injeziert welches die URL zu dem Bild enthält, ein Call fällt nun effektiv weg.

»Newest« Calls Bei gewissen Calls wie z.B. News oder Social wollen die meisten Benutzer lediglich die neusten Einträge sehen. Deshalb wurde der »Newest« Call implementiert, welcher die neusten 15 (konfigurierbarer Wert) Elemente in sortierter Reihenfolge, also neustes zuoberst, ausgibt. Dadurch müssen nicht alle, potentiell mehrere dutzend Elemente, gesendet werden. Bei Bedarf können selbstverständlich auch alle Elemente abgerufen werden.

6.3 Fremdcode

Dieses Kapitel dient zur klaren Deklarierung von Fremdcode welcher in der Bachelorarbeit verwendet wurde.

6.3.1 Libraries

Alle JAR Dateien unter »src/Libraries« sowie »src/WebContent/WEB-INF/lib« sind Fremdcode in kompilierter Form. Es folgt eine Liste aller Libraries mit einer Begründung, einem Link zu deren Internetpräsenzen und Lizenz.

- angular-ezfb
  EZFB ist das einzige AngularJS Modul zur Benutzung der Facebook SDK. Es bietet grosse Vorteile bei der Benutzung gegenüber der »puren« Javascript Facebook SDK.
  URL: https://github.com/pc035860/angular-easyfb
  Lizenz: MIT
- angularJS
  URL: https://angularjs.org/
  Lizenz: MIT
- commons-codec
  Die empfohlene Codecs Library für Apache Tomcat.
  URL: https://commons.apache.org/proper/commons-codec/
  Lizenz: Apache License 2.0
- esapi
  ESAPI wird stets weiterentwickelt und an die neusten Bedrohungen angepasst. Zudem ist die Benutzung sehr einfach gestaltet und es werden mächtige Konfigurationsmöglichkeiten geboten.
  URL: https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API
  Lizenz: BSD
- facebook4j
  Facebook4J bietet alle gewünschten Features und funktioniert »out of the box« bestens. Von den gleichen Entwicklern wie Twitter4J.
  URL: http://facebook4j.org/
  Lizenz: Apache Licence 2.0
- genson
  Genson bietet automatischen JSON Support für JAX-RS implementationen.
  URL: https://github.com/owlike/genson
  Lizenz: Apache Licence 2.0
- hk2
  Das de facto Framework für Dependency Injection für Glassfish.

URL: https://hk2.java.net/2.4.0-b24/
Lizenz: »CDDL + GPLv2 with classpath exception«

- imgscalr
  Imgscalr benötigt keine externen Libraries und profitert von Hardware Beschleunigung.
  URL: https://github.com/thebuzzmedia/imgscalr
  Lizenz: Apache Licence 2.0

- jackson
  Wird benötigt für Jersey.
  URL: https://github.com/FasterXML/jackson
  Lizenz: Apache Licence 2.0

- jquery
  Die grösste, aktive entwickelte JS Bibliothek.
  URL: https://jquery.com/
  Lizenz: MIT

- jaxb
  Wird benötigt für Jersey / JAX-WS
  URL: https://jaxb.java.net/
  Lizenz: »CDDL v1.1 and GPL v2«

- jersey
  Unkomplizierte Konfiguration im Vergleich zu Spring MVC und mehr Kontrolle über Details.
  URL: https://jersey.java.net/
  Lizenz: »CDDL 1.1 and GPL v2 with the Classpath Exception«

- json-simple
  Schnelles manuelles en- und decoding.
  URL: https://code.google.com/p/json-simple/
  Lizenz: Apache Licence 2.0

- jsoup
  Ergänzt sich bestens mit der ESAPI, welche auch Code von JSOUP verwendet.
  URL: http://jsoup.org/
  Lizenz: MIT

- mimepull
  De facto Standard API für MIME Anhänge in Java.
  URL: https://mimepull.java.net/
  Lizenz: »CDDL v1.1 and GPL v2«

- mysql-connector
  Der offizielle MySQL Connector des Herstellers Oracle
  URL: https://dev.mysql.com/downloads/connector/
  Lizenz: GPL v2

- persistence-api
  De facto Standard API.
  URL: http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html
  Lizenz: »CDDL v1.1 and GPL v2«

- spin
  Einfache, skalierbare Spinner ohne externe Abhängigkeiten.
  URL: https://github.com/fgnass/spin.js
  Lizenz: MIT

- twitter4j
  Twitter4J bietet alle gewünschten Features und funktioniert »out of the box« bestens. Von den gleichen Entwicklern wie Facebook4J.
  URL: http://twitter4j.org/en/index.html
  Lizenz: Apache Licence 2.0

- ui-router
  Bieter viel mehr (benötigte) Möglichkeiten als ngRouter.

URL: https://github.com/angular-ui/ui-router
Lizenz: MIT

- ui-bootstrap
  Das AngularJS Modul für Bootstrap.
  URL: https://github.com/angular-ui/bootstrap
  Lizenz: MIT

- validation-api
  Wird benötigt für JAX-WS.
  URL: http://beanvalidation.org/
  Lizenz: Apache Licence 2.0

- zxing
  Zxing wird von Google aktiv entwickelt und wird auch bei Android eingesetzt.
  URL: https://github.com/zxing/zxing
  Lizenz: Apache Licence 2.0

6.3.2 Code Snippets

Gewisse Codezeilen wurden im Internet als Lösung auf ein bestimmtes Problem gefunden und übernommen. Alle diese Stellen sind im Sourcecode als Kommentar deklariert, inklusive Link. Es handelt sich dabei um je 3-5 Zeilen welche bei Bedarf angepasst wurden.

6.4 Reporting

Die folgenden Reports wurden erstellt, folgende Ziele zu erreichen:

- STAN4J - Aussagen zur Software Qualität geben zu können
- JUnit - Die Korrektheit der Software zu prüfen
- JavadDoc - Die Wartung und Weiterentwicklung zu vereinfachen
- GUI Mockup Vergleich - Die Überprüfung ob alle Use Cases und Anforderungen des Front & Backend erfüllt wurden

6.4.1 STAN4J

Um Aussagen zur Code Qualität zu machen wurde STAN4J 2.1 [44] benutzt. Dabei zu beachten ist, dass dies sich nur auf den Backend Teil der Software beziehen. Folgende Screenshots zeigen die wichtigsten Ausschnitte des ganzen Reports welcher im Anhang zu finden ist.

[44] STAN4J - Structure Analaysis for Java, URL: `http://stan4j.com`(Stand 08.06.15)

Metrics Summary

Metric	Value
Number of Libraries	1
Number of Packages	11
Number of Top Level Classes	79
Average Number of Top Level Classes per Package	7.18
Average Number of Member Classes per Class	0.05
Average Number of Methods per Class	7.96
Average Number of Fields per Class	3.09
Estimated Lines of Code	5713
Estimated Lines of Code per Top Level Class	72.32
Average Cyclomatic Complexity	1.63
Fat for Library Dependencies	0
Fat for Flat Package Dependencies	26
Fat for Top Level Class Dependencies	267
Tangled for Library Dependencies	0%
Average Component Dependency between Libraries	0%
Average Component Dependency between Packages	28.18%
Average Component Dependency between Units	18.39%
Average Distance	-0.46
Average Absolute Distance	0.46
Average Weighted Methods per Class	12.98
Average Depth of Inheritance Tree	1.35
Average Number of Children	0.35
Average Coupling between Objects	4.14
Average Response for a Class	13.75
Average Lack of Cohesion in Methods	29.76

Metric Ratings

Count Metrics

Metric	Rating	Linear
Number of Top Level Classes	20 40 60 80	✓
Number of Methods	25 50 100 200	✓
Number of Fields	10 20 40 80	✓
Estimated Lines of Code	200 300 400 500	✓
Estimated Lines of Code	30 60 120 240	✓

Complexity Metrics

Metric	Rating	Linear
Cyclomatic Complexity	10 15 20 30	✓
Fat	30 60 120 240	✓
Fat	30 60 120 240	✓
Fat	30 60 120 240	✓
Tangled	0 1	✓
Tangled for Library Dependencies	0 1	✓
Average Component Dependency between Libraries	0 .5 1	✓
Average Component Dependency between Packages	0 .5 1	✓

Robert C. Martin Metrics

Metric	Rating	Linear
Distance	-1 -.5 0 .5 1	✓
Average Absolute Distance	0 .4 .5 1	✓

Chidamber & Kemerer Metrics

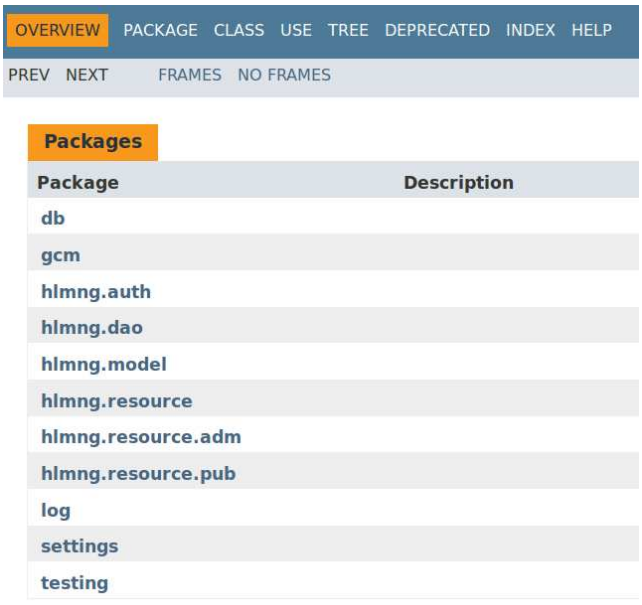
Metric	Rating	Linear
Weighted Methods per Class	0 100 200 300	✓
Depth of Inheritance Tree	4 6 8 10	✓
Average Depth of Inheritance Tree	2 1 0	✓
Coupling between Objects	0 25 250	✓
Response for a Class	0 100 1000	✓

6.4.2 JUnit

Folgender Screenshot zeigt, dass alle 29 JUnit [45]Tests erfolgreich durchgeführt worden konnten.

6.4.3 Javadoc

Um die nachfolgende Weiterentwicklung des Backend zu vereinfachen, wurde wo sinnvoll die Javadoc Notationen verwendet. Der generierte Report zum Zeitpunkt der Abgabe kann im Anhang gefunden werden, da dieser mehre dutzend Seiten lang ist.



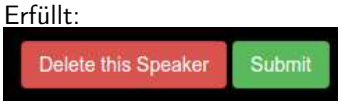
[45]http://junit.org/

6.4.4 GUI Mockup Vergleich

Die Elemente der Frontend GUI wurden für jede View aus den Mockups übernommen. Zusätzlich wurden Features wie Pagination und eine Suche hinzugefügt. Aus Zeitgründen konnte das exakte Layout der Mockups nicht genau übernommen werden. Die Navigation wurde übernommen, mobile-tauglich gemacht und optisch verbessert.

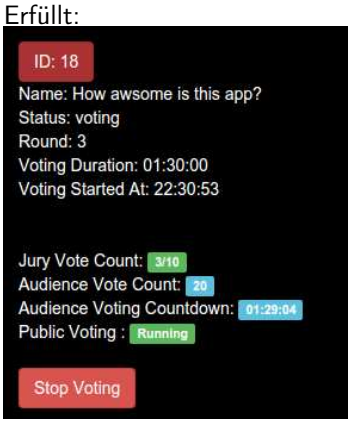
Das Bedienungskonzept der GUI ist im Frontendmanual ausführlich erklärt.

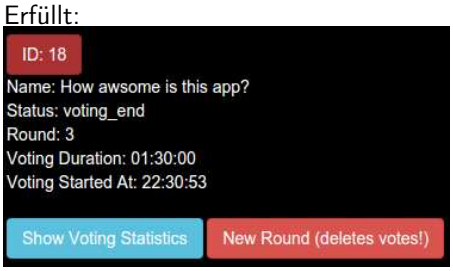
- AnfB-1: Das System muss die Möglichkeit bieten, Bilder und Texte auf sozialen Kanälen veröffentlichen.
  Erfüllt:
  

- AnfB-2: Das System muss Fehlertolerant sein. Das heisst alle Elemente können nachträglich editiert, falls möglich gelöscht oder neugestartet werden können.
  Erfüllt:
  

- AnfB-3: Es kann pro User pro Voting nur eine Stimme abgeben werden.
  Erfüllt: Im Code wir ein Check durchgeführt, wurde mittels doppelten Vote HTTP POSTs mehrfach getestet

- AnfB-4: Die Granularität eines Sliders soll konfigurierbar sein.
  Erfüllt:
  

- AnfB-5: Ein Voting soll anzeigen können wie viele der Jury ihre Stimme abgeben haben.
  Erfüllt:
  

- AnfB-6: Ein Voting soll mehrere Runden unterstützen. Dabei werden alle vorhandenen Votes gelöscht und das Voting in den Zustand vor dem Start zurückgesetzt werden.
  Erfüllt:
  

- AnfB-7: Alle Voting Daten sollen als CSV (Comma Seperated Values) exportiert werden können.
  Erfüllt:

	A	B	C	D	
1	General Settings				
2	##############				
3					
4	votingID	name	juryCount	status	slide
5	18	How awsome is this app?	10	voting_end	
6					
7					
8	Sliders & Weibghts				
9	################				
10					
11	sliderID	name	weight	votingIDFK	
12	83	Design	1	18	
13	84	Functionality	1	18	
14	85	Implementation	2	18	
15		In Time Score	1	''	
16					
17	Voting Results				
18	############				
19		Jury	Audience		

- AnfB-9: Die API soll es dem Mobile App ermöglichen sich zu registrieren und Daten anzubieten.
  Erfüllt: Im Code / siehe Mobile App

- AnfB-10: Nicht öffentliche API Calls / Seiten sollen Passwort geschützt sein und von den öffentlichen getrennt.
  Erfüllt:
  Passwort - Siehe Backend Manual - wurde nicht durch die Software selbst sondern mit Apache gelöst.
  Getrennt - Siehe Code / API Dokumentation

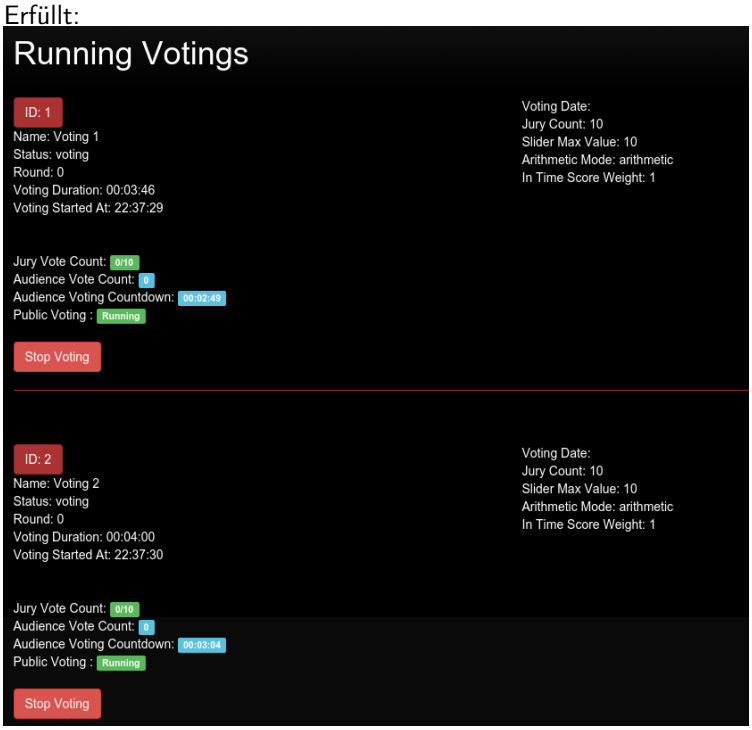
- AnfB-11: Das Backend soll eine Moderationsfunktion für geteilte Inhalte der Mobile App User bieten.

- AnfB-12: Die Steuerung des Votings soll wie in den Mockups beschrieben umgesetzt werde.
  Erfüllt: Siehe Frontend selbst und Code

- AnfB-13: Das Backend ermöglicht den Upload von Daten mittels Multipart Form Data.
  Erfüllt:

- AnfF-1: Aktionen im Frontend müssen unabhängig vom Operator ablaufen können.
  z.B. wenn ein Voting gestartet wird und der Operator sich abmeldet / den Browser schliesst, soll das System weiterlaufen.
  Erfüllt: Siehe Frontend selbst

- AnfF-2: Das Frontend muss mehrere aktive Votings gleichzeitig verwalten können.
  Erfüllt:



- AnfF-3: Das Frontend muss bei der Voting Ansicht die neusten Vote-Daten automatisch aktualisieren.
  Siehe AnfF-2 Screenshot

6.5 Testing

6.5.1 Integrationstests

Nach jedem neu implementierten Feature wurde vor dem Push in das Repository das neue Feature getestet und allfällig andere betroffene Module überprüft. Diese Tests wurden auf Seiten des Backend durch JUnit durchgeführt, auf Seiten des Frontend durch manuelle Funktionsprüfung mit einem Browser. So wurde sichergestellt, dass die Software jederzeit in einem deploybaren Zustand vorhanden ist.

JUnit Um den JUnit Tests den Zugriff auf die Datenbank zu gewähren, wurde bei dem Generic Dao ein Flag »isTest« eingeführt. Falls dieses Flag gesetzt ist, veranlasst der Dao die Datenbankverbindung direkt mit den Daten des HLMNGSettings File zu eröffnen. Dies wurde so gelöst, da in JUnit die automatische Verwendung von »META-INF/context.xml« nicht funktioniert.

Verwendete Browser Das Frontend Testing wurde mit den zwei weit verbreiteten Browsern Chrome sowie Firefox durchgeführt. Es wurden Versuche mit dem Testingframework »Jasmine« unternommen um Unit Tests für AngularJS durchzuführen. Leider traten diverse Probleme bei der Benutzung auf, welche auch nicht durch den Betreuer oder weiteren Kontaktpersonen gelöst werden konnten. Deshalb wurden auf automatisierte Tests verzichtet, dies nicht zuletzt auch deshalb, da die Programmlogik grösstenteils im Backend steckt und der AngularJS Teil lediglich für die Darstellung zuständig ist.

6.5.2 STAN4

Um sicherzustellen, dass keine zirkulären Abhängigkeiten bestehen sowie Hinweise zu den gängigen Code Metriken zu erhalten, wurde STAN4J eingesetzt. Dies mit dem Ziel den späteren Ausbau sowie den Austausch einzelner Komponenten zu erleichtern.

6.5.3 GCM Benchmark

Um eine Aussage über die Erfüllung der Nichtfunktionalen Anforderung NAnfB-1 zu machen, wurde die Gesamtzeit von GCM Pushes gemessen. Unter Gesamtzeit ist die Zeitdauer zwischen Absenden der Nachricht an Google bis zum Empfang der Nachricht auf den Clients zu verstehen. Es wurde mit einem Client getestet, da die Anzahl keine Rolle spielen sollte [46]. Natürlich ist dabei die Netzwerkinfrastruktur ausschlaggebend an der die Mobile Apps verbunden sind.

Nachfolgend sind die Messungen zu finden, welche auf einem Laptop an einer 5 MB/s Swisscom Verbindung per WLAN durchgeführt wurden. Die Messungen wurden am 05.03.15 um 14:30 durchgeführt.

Messung Nr.	Gesamtzeit
1	1.54 s
2	1.03 s
3	2.13 s
4	1.82 s
5	3.03 s
6	2.89 s
7	1.75 s

6.5.4 Availability

Um die Nichtfunktionale Anforderung NAnfB-4 zu überprüfen wurde die Uptime des Server, welcher durch die Compass Security betrieben wird, mittels dem Linux Befehl »uptime« überprüft sowie dem Log des letzten Restart überprüft. Leider konnten keine längerfristigen Tests durchgeführt werden, da bei jeder Version deployed wurde. Die folgende Tabelle zeigt die am längsten laufenden Laufzeiten.

Start	Ende	Erreichte Laufzeit
06.05.15 - 13:22	09.05.15 - 16:02	3 Tage, 2 Stunden, 40 Minuten
11.05.15 - 09:55	17.05.15 - 11:32	6 Tage, 1 Stunde, 37 Minuten
30.05.15 - 15:45	04.06.15 - 14:34	4 Tage, 22 Stunden, 49 Minuten

6.5.5 Performance Benchmark

Um zu testen, ob die Software auch mehrere Clients simultan bedienen kann wurde folgender »Benchmark« erstellt für die Bestätigung der Nichtfunktionalen Anforderung NAnfB-2:

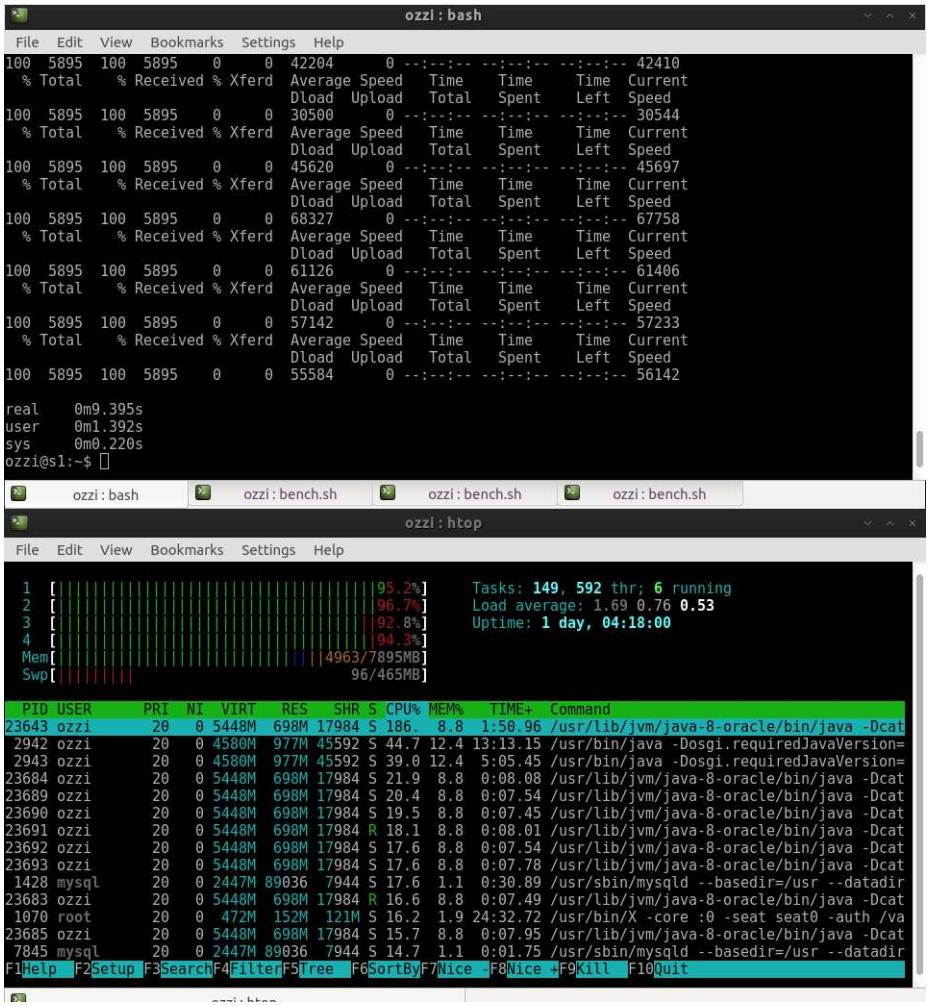
```
#!/bin/bash
for i in {1..10}
do
        curl -i -H "Accept: application/json" -H "Content-Type: application/json"
        -X GET 'https://localhost:8443/hlmng/rest/adm/social' -k > /dev/null
done
```

Dieser wurde parallel in 4 Shells mit dem Time Befehl gestartet, um die Ausführungszeit zu messen. Nach und nach wurde das Maximum im For Loop erhöht und die Ausführungszeit überwacht.

6.5.6 Fazit

Es konnten 4x 100 Calls (parallel) innerhalb von 10 Sekunden bedient werden. Dies auf einem Ultrabook mit einer i7 4500U CPU [47]. Ich bin zuversichtlich, dass mit einer aktuellen Serverhardware ohne Probleme ein Konferenz mit mehreren hunderten Leuten durchgeführt werden kann. Der folgende Screenshot zeigt den Versuchsaufbau und die jeweilige Auslastung des Computer.

[46] Making the Most of Google Cloud Messaging, http://developer.android.com/training/cloudsync/gcm.html(Stand: 08.06.2015)
[47] http://ark.intel.com/de/products/75460/Intel-Core-i7-4500U-Processor-4M-Cache-up-to-3_00-GHz (Stand: 01.06.15)


ozzi : bash
File Edit View Bookmarks Settings Help
100 5895 100 5895 0 0 42204 0 --:--:-- --:--:-- --:--:-- 42410
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 5895 100 5895 0 0 30500 0 --:--:-- --:--:-- --:--:-- 30544
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 5895 100 5895 0 0 45620 0 --:--:-- --:--:-- --:--:-- 45697
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 5895 100 5895 0 0 68327 0 --:--:-- --:--:-- --:--:-- 67758
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 5895 100 5895 0 0 61126 0 --:--:-- --:--:-- --:--:-- 61406
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 5895 100 5895 0 0 57142 0 --:--:-- --:--:-- --:--:-- 57233
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 5895 100 5895 0 0 55584 0 --:--:-- --:--:-- --:--:-- 56142
real 0m9.395s
user 0m1.392s
sys 0m0.220s
ozzi@s1:~$
ozzi : bash
ozzi : bench.sh
ozzi : bench.sh
ozzi : bench.sh
ozzi : htop
File Edit View Bookmarks Settings Help
1 [||||||||||||||||||||||||||||||||95.2%]
2 [||||||||||||||||||||||||||||||||96.7%]
3 [||||||||||||||||||||||||||||||||92.8%]
4 [||||||||||||||||||||||||||||||||94.3%]
Mem[||||||||||||||||||||||||4963/7895MB]
Swp[|||||||| 96/465MB]
Tasks: 149, 592 thr; 6 running
Load average: 1.69 0.76 0.53
Uptime: 1 day, 04:18:00
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
23643 ozzi 20 0 5448M 698M 17984 S 186. 8.8 1:50.96 /usr/lib/jvm/java-8-oracle/bin/java -Dcat
2942 ozzi 20 0 4580M 977M 45592 S 44.7 12.4 13:13.15 /usr/bin/java -Dosgi.requiredJavaVersion=
2943 ozzi 20 0 4580M 977M 45592 S 39.0 12.4 5:05.45 /usr/bin/java -Dosgi.requiredJavaVersion=
23684 ozzi 20 0 5448M 698M 17984 S 21.9 8.8 0:08.08 /usr/lib/jvm/java-8-oracle/bin/java -Dcat
23689 ozzi 20 0 5448M 698M 17984 S 20.4 8.8 0:07.54 /usr/lib/jvm/java-8-oracle/bin/java -Dcat
23690 ozzi 20 0 5448M 698M 17984 S 19.5 8.8 0:07.45 /usr/lib/jvm/java-8-oracle/bin/java -Dcat
23691 ozzi 20 0 5448M 698M 17984 R 18.1 8.8 0:08.01 /usr/lib/jvm/java-8-oracle/bin/java -Dcat
23692 ozzi 20 0 5448M 698M 17984 S 17.6 8.8 0:07.54 /usr/lib/jvm/java-8-oracle/bin/java -Dcat
23693 ozzi 20 0 5448M 698M 17984 S 17.6 8.8 0:07.78 /usr/lib/jvm/java-8-oracle/bin/java -Dcat
1428 mysql 20 0 2447M 89036 7944 S 17.6 1.1 0:30.89 /usr/sbin/mysqld --basedir=/usr --datadir
23683 ozzi 20 0 5448M 698M 17984 R 16.6 8.8 0:07.49 /usr/lib/jvm/java-8-oracle/bin/java -Dcat
1070 root 20 0 472M 152M 121M S 16.2 1.9 24:32.72 /usr/bin/X -core :0 -seat seat0 -auth /va
23685 ozzi 20 0 5448M 698M 17984 S 15.7 8.8 0:07.95 /usr/lib/jvm/java-8-oracle/bin/java -Dcat
7845 mysql 20 0 2447M 89036 7944 S 14.7 1.1 0:01.75 /usr/sbin/mysqld --basedir=/usr --datadir
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice - F8Nice + F9Kill F10Quit

7 Fussnoten Index

Hacking-Lab Mobile Event App | Projektmanagment

Oussama Zgheb & Tobias Zahner

11. Juni 2015

Datum	Version	Änderung	Autor
04.04.15	1.0	Intial	Oussama Zgheb
10.04.15	1.1	Risikomanagement eingefügt	Oussama Zgheb
06.06.15	1.2	RUP Diagramm	Oussama Zgheb
08.06.15	1.3	Ergänzungen und Risikomanagment	Tobias Zahner
10.06.15	1.4	Exportieren der Grafiken, Erklärungen	Oussama Zgheb

Inhaltsverzeichnis

1 Management Abläufe

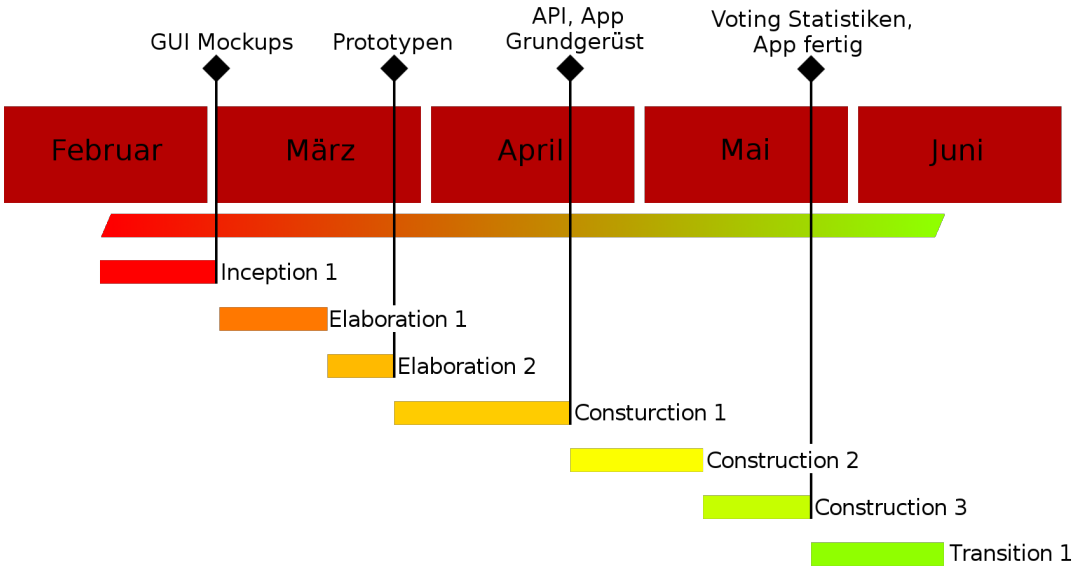
1.1 Projekt Zeitaufwand

Der Aufwand für eine Bachelorarbeit beträgt pro Student 360 Stunden [1]

Somit liegt der Gesamtaufwand mit zwei Studenten bei 720 Stunden. Die Bachelorarbeit startete am 16.02.15 und endet am 12.06.15. Dies sind 16 Wochen welche für die Arbeit zur Verfügung stehen.

1.2 Projektplan

Die folgende Abbildung AB1 zeigt den groben Projektplan mit allen RUP Phasen sowie den wichtigsten Meilenstein. Diese Abbildung soll eine schnelle Übersicht über die Entwicklungsarbeit der Bachelorarbeit geben.



AB1 [2]

1.2.1 Meilensteine

Nachfolgend sind Meilensteine, gruppiert zu deren Komponente, aufgelistet.

REST API

- RA-1 - Einarbeitung in das Thema und Ausarbeitung des Grundkonzepts
- RA-2 - Fertigstellung des DB Schema
- RA-3 - Erstellung eines Prototypen mit GET/POST/PUT

[1] Abschnitt »Zeitlicher Rahmen« - https://www.hsr.ch/Hinweise-fuer-externe-Auftragg.7263.98.html?&no_cache=1&type=98 (Stand 06.06.15)
[2] Zeitplan der Bachelorarbeit, Eigene Abbildung

- RA-4 - Implementierung von Login-Funktionalität
- RA-5 - Fertigstellung aller Models in der API
- RA-6 - Testing sowie Fertigstellung der Dokumentation

Frontend

- BE-1 - Erstellung Prototyp einer Seite mit AngularJS und REST API Call
- BE-2 - Fertigstellung der Eventauswahl, Event Erstellung
- BE-3 - Fertigstellung der Voting Funktion
- BE-4 - Fertigstellung der restlichen Seiten (QR,Speaker, etc.)
- BE-5 - Voting (Start,Pause,Export)
- BE-6 - Grafisches Styling sowie Testing

Mobile App

- AP-1 - Views (ausser Agenda und Voting) inkl. Navigation fertiggestellt
- AP-2 - Verknüpfung mit REST-Daten hergestellt
- AP-3 - Voting implementiert
- AP-4 - Agenda implementiert
- AP-5 - Sharing implementiert

1.2.2 Zeitplanung

Nachfolgend sind alle RUP Phasen chronologisch aufgelistet. Dabei sind die Ziele der Phasen sowie deren Meilensteine vermerkt.

Inception Zeitrahmen: 16.2.15 - 1.3.15 (14 Tage)
Geplanter Zeitaufwand: 50h

- Projekt Kontext ermittelt
- Funktionen sowie Anforderungen festgelegt
- Entwicklungsumgebung eingerichtet

Elaboration 1 Zeitrahmen: 2.3.15 - 15.3.15 (14 Tage)
Geplanter Zeitaufwand: 85h

- GUI Mockups fertiggestellt
- Isolierte Prototypen programmiert (RA-1,RA-3)
- Datenbank sowie Architektur ausgearbeitet (RA-2)

Elaboration 2 Zeitrahmen: 16.3.15 - 22.3.15 (7 Tage)
Geplanter Zeitaufwand: 50h

- Zusammenspiel der einzelnen Komponenten definieren
- Prototypen von App und API miteinander verknüpfen
- Basis Funktionalität ausarbeiten (RA-4)
- Push Prototyp

Construction 1 Zeitrahmen: 22.3.15 - 19.4.15 (28 Tage)
Geplanter Zeitaufwand: 180h

- Datenbank ist fertig inklusive Testdatenbestand
- App Grundgerüst steht (alle Views / Navigation) (AP-1)
- API fertiggestellt (RA-5, RA-6)
- Prototyp Frontend (BE-1)

Construction 2 Zeitrahmen: 20.4.15 - 10.5.15 (21 Tage)
Geplanter Zeitaufwand: 150h

- Frontend CRUD Operationen (inkl. QR Code) (BE-2, BE-3, BE-4)
- Mobile App verwendet ausschliesslich Daten der API (AP-2)

Construction 3 Zeitrahmen: 11.5.15 - 24.5.15 (14 Tage)
Geplanter Zeitaufwand: 105h

- Voting (BE-5)
- Design des Frontend (BE-6)
- Funktionalität der App vollständig implementiert (AP-3, AP-4, AP-5)

Transistion 1 Zeitrahmen: 25.5.15 - 12.6.15 (18 Tage)
Geplanter Zeitaufwand: 100h

- Testing der fertigen Komponenten und deren Zusammenspiel
- Manuals und Dokumentation fertiggestellt
- Pufferzeit

1.3 Meetings

Bis die GUI ausgearbeitet war, wurden bei jeder Revision ein Meeting gehalten, bei welchem die Änderungen mit dem Betreuer diskutiert wurden.
Während der Implementation wurden je nach Bedarf des Betreuers oder der Studierenden ein Meeting angefordert. Die Studierenden haben sich regelmässig per Email mit dem Betreuer in Kontakt gesetzt und darin den aktuellen Stand sowie das weitere Vorgehen beschrieben.

2 Risikomanagement

Im Risikomanagement wurden die potentielle Risiken bei der Umsetzung der Bachelorarbeit aufgelistet. Diese wurden bewertet, analysiert und anhand der gewonnen Erkenntnissen wurde versucht vorbeugende Massnahmen zu ermitteln sowie umzusetzen.

Die Formel des totalen Schadensfaktor lautet wie folgt:
Maximaler Schaden (in Stunden) x Eintrittswahrscheinlichkeit (Prozent in Dezimalschreibweise) x Gewichteter Schaden (Auswirkung auf Projekt) = totaler Schadensfaktor

2.1 Risikomanagement Front und Backend

Nr	Titel	Beschreibung	max. Schaden	Eintrittswahr-scheinlichkeit	Gewichteter Schaden	Tot.
R1	Push Notifikation	Die Push Mechanismus ist unzuverlässig (langsam, kommt nicht an)	7h	0.2	3	4.2
R2	CSV Auswertung	Die Voting Auswertung ist fehlerhaft	10h	0.15	5	7.5
R3	Soziale Kanäle	Das teilen auf den sozialen Kanälen funktioniert nicht	5h	0.25	1	1.25
R4	Synchronisation	Statusänderungen beim Voting (Pause/Start etc.)	8h	0.3	4	9.6
R5	Performance	Die Performance ist bei vielen Mobile Apps ungenügend	10h	0.1	2	2

Vorbeugung

R1 – Früh einen Prototypen erstellen und Testen, Best Practices einhalten sowie Mechanismus dahinter verstehen

R2 – Unit Tests der Rechenfunktion, von Hand überprüfen

R3 – Auf bewährte Libraries zurückgreifen, API Dokumentation lesen

R4 – Früh einen Prototypen erstellen, einfache Calls auf Seiten der REST API bieten, Testing

R5 – Performance Benchmarks durchführen, Connection Pooling verwenden, keine „bloat“ Frameworks verwenden

2.2 Risikomanagement Mobile App

Nr	Titel	Beschreibung	Max. Schaden	Eintritts-wahrscheinlichkeit	Gewichteter Schaden	Total
R1	HTTP-Verkehr	Der Austausch mit dem Server funktioniert nicht	8h	0.2	5	8
R2	Mediasharing über HTTP	Mediasharing über HTTP funktioniert nicht	10h	0.3	2	6
R3	Agenda	Keine passende Library ist vorhanden um eine Tagesagenda anzuzeigen	14h	0.5	4	28
R4	Multi-Device	App sieht auf einigen Devices schlecht aus oder funktioniert gar nicht	6h	0.3	1	1,8
R5	HTTPS-Unterstützung	Zertifikat kann nicht akzeptiert werden oder es kann sonst keine Verbindung zur HTTPS-Seite hergestellt werden.	8h	0.4	3	9,6

Vorbeugung

R1 - Als erstes den Netzwerkverkehr sicherstellen, da fast alle App-Teile darauf angewiesen sind.

R2 - Sobald im Backend der Fileupload implementiert ist, versuchen einen erfolgreichen POST mit einem Mediafile zu machen.

R3 - Früh informieren, ob eine Library vorhanden ist und falls nicht genug Zeit einplanen zur selbständigen Implementierung.

R4 - Während der ganzen Entwicklung sowohl auf einem Smartphone als auch auf einem Tablet testen.

R5 - Sobald HTTPS Serverseitig implementiert ist, das Zertifikat importieren und so schnell wie möglich auf HTTPS umstellen.

2.3 Umgang mit Risiken

Die Arbeitspakete wurden grosszügig geplant, sodass ein Puffer für allfällige Probleme bereits integriert ist. Um Risiken mit den eingesetzten Technologien vorzubeugen, wurden funktionierende Prototypen entwickelt.
Bei Problemen welche nicht durch die Studenten selber lösbar sind, soll ein Treffen mit dem Betreuer oder dem Experten veranlasst werden.

3 Infrastruktur

Die Codierung sowie die Dokumentation wird auf den persönlichen Laptops der Studenten durchgeführt.
Auf dem durch die HSR bereitgestellten VServer läuft die Zeiterfassungsapplikation sowie die REST API, bis ein Server an der Compass Security eingerichtet wird.

3.1 Entwicklungsumgebung

Folgend wir eine Übersicht über die verwendeten Tools und Programme gegeben,

3.1.1 IDE

Front und Backend Eclipse EE (Kepler) bietet die Möglichkeit einen Integrierten Tomcat Server zu verwenden, sodass mit einem Knopfdruck die neuste Version lokal *deployed* wird. Unterstützt werden zudem Syntax Überprüfung und Highlighting für JS, HTML und CSS.

Mobile App Für die Mobile App wurde Android Studio verwendet, was von Google empfohlen wird. Das früher verwendete Eclipse mit dem entsprechenden Plugin ist immer noch sehr fehleranfällig und eher nicht zu empfehlen.

3.1.2 Tools

UML Diagramme - Astah Community
Grafikbearbeitung - Photoshop CS4, Gimp 2.
Textverarbeitung - Lyx, Microsoft Office, LibreOffice
Zeiterfassung - Redmine

3.1.3 Versionierung

Der Code wird mit GIT auf GitHub versioniert. Dabei besitzen beide Teammitglieder ein eigenes Repository.

3.2 Dokumentverwaltung

Um die Projektdokumente für beide Teammitglieder verfügbar zu machen, wird Dropbox eingesetzt. Dropbox bietet automatische Synchronisierung, Sicherung sowie Versionierung der einzelne Dokumente.

4 Aufteilung

Folgendes Kapitel legt die Aufteilung der Tätigkeiten sowie Verantwortlichkeiten fest.

4.1 Beide Studenten

- GUI Mockups
- Zusammenspiel der Komponenten
- Entwicklung Frontend
  Konnte durch unvorhergesehene Probleme auf Seiten Mobile App nicht durch Tobias Zahner wahrgenommen werden
- Dokumentation der Bachelorarbeit

4.2 Oussama Zgheb

- Design und Umsetzung der Datenbankstrukturen
- Design und Entwicklung der REST API
- Evaluierung und Implementierung einer Push Methodik
- Evaluierung und Prototyp der Twitter Anbindung
- Implementierung der Social Network Anbindung zu Facebook & Twitter

4.3 Tobias Zahner

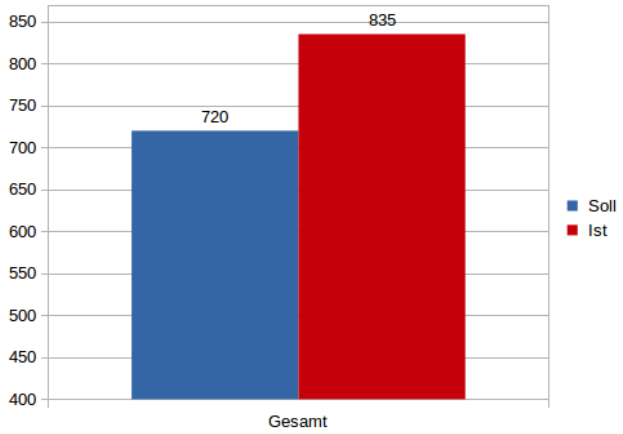
- Evaluierung und Prototyp der Facebook Anbindung
- Design und Entwicklung - Hacking-Lab Mobile App für Android

5 Zeitauswertung

Der Arbeitsaufwand der gemäss Vorgabe durch die einzelnen Studenten geleistet werden muss, beträgt 360 Stunden. Die folgenden Abbildungen wurden aus Redmine, dem Zeiterfassungstool extrahiert und aufbereitet. Alle Werte sind dabei in Stunden zu verstehen.

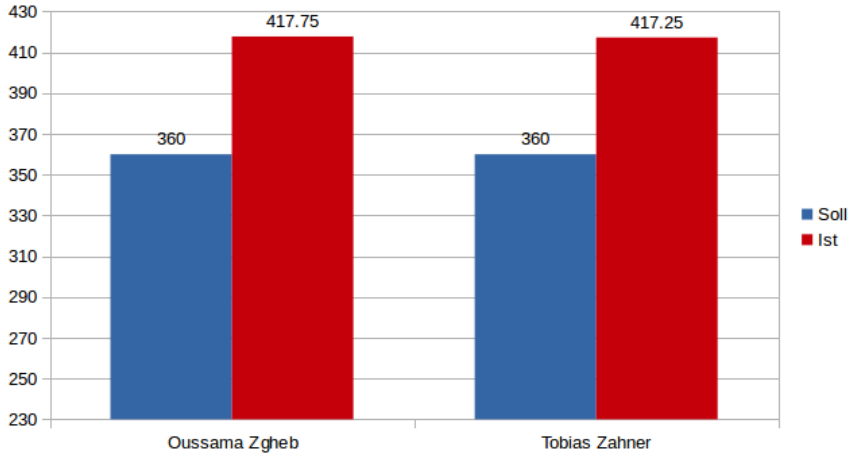
5.1 Gesamtzeit

Zu sehen ist, dass die Gesamtzeit um 115h überschritten wurde. Die Erklärung dafür ist unter »5.5 Zusammenfassung« zu finden.



5.2 Gesamtzeit pro Student

Zu sehen ist, dass beide Studenten beinahe gleich viel Stunden in die Bachelorarbeit investiert haben.



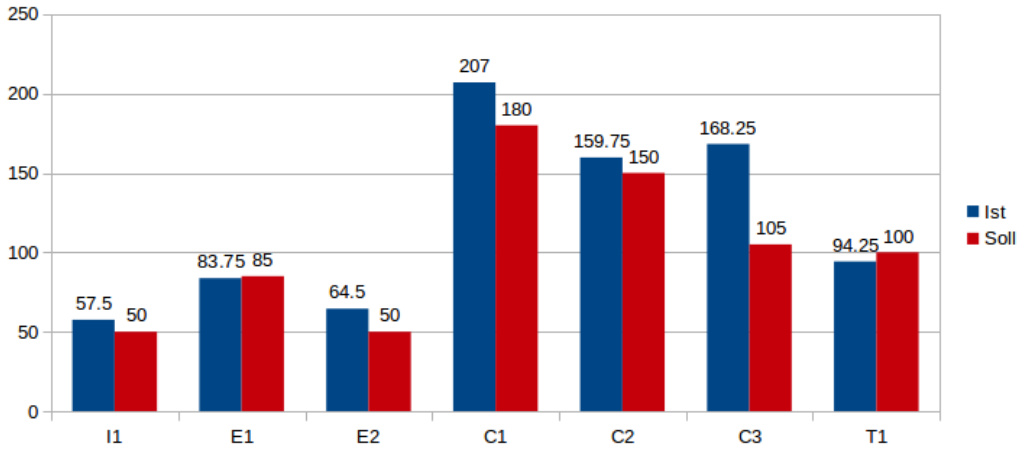
5.3 Zeitaufwand pro Aktivität

Die nachfolgende Tabelle zeigt alle Aktivitäten und deren Aufwand über die knapp 5 Monate Bachelorarbeit. Zu sehen ist, dass die meiste Zeit mit der Implementierung verbracht wurde.

Aktivität	Benutzer	2015-2	2015-3	2015-4	2015-5	2015-6	Gesamtzeit
Analyse & Design		*4.50*	*4.00*	*12.00*	*8.00*		*28.50*
	Oussama Zgheb	**4.00**	**4.00**	**12.00**	**0.75**		**20.75**
	Tobias Zahner	**0.50**			**7.25**		**7.75**
Implementation			*191.00*	*177.00*	*253.75*	*21.00*	*642.75*
	Oussama Zgheb		**87.50**	**72.25**	**136.75**	**9.50**	**306.00**
	Tobias Zahner		**103.50**	**104.75**	**117.00**	**11.50**	**336.75**
Dokumentation			*8.25*	*1.25*	*3.50*	*52.75*	*65.75*
	Oussama Zgheb		**8.25**	**1.25**	**3.50**	**23.75**	**36.75**
	Tobias Zahner					**29.00**	**29.00**
Test				*5.00*		*5.00*	*10.00*
	Oussama Zgheb			**5.00**			**5.00**
	Tobias Zahner					**5.00**	**5.00**
Requirements		*28.00*	*8.00*	*4.50*		*2.50*	*43.00*
	Oussama Zgheb	**8.50**	**2.50**	**1.75**		**2.50**	**15.25**
	Tobias Zahner	**19.50**	**5.50**	**2.75**			**27.75**
Business Modeling		*12.75*	*3.50*	*0.75*		*2.00*	*19.00*
	Oussama Zgheb	**9.75**	**2.25**			**2.00**	**14.00**
	Tobias Zahner	**3.00**	**1.25**	**0.75**			**5.00**
Deployment						*11.00*	*11.00*
	Oussama Zgheb					**11.00**	**11.00**
Allgemein		*0.75*	*8.50*	*0.75*	*3.00*	*2.00*	*15.00*
	Oussama Zgheb	**0.75**	**4.50**	**0.75**	**3.00**		**9.00**
	Tobias Zahner		**4.00**			**2.00**	**6.00**
Gesamtzeit		**46.00**	**223.25**	**201.25**	**268.25**	**96.25**	**835.00**

5.4 Zeitaufwand pro Iteration

Das untenstehende Diagramm zeigt den eingeplanten und tatsächlichen Zeitaufwand pro Iteration.
Vor allem in den *Construction* Phasen wurde mehr Zeit für die Imlpementation benötigt.



5.5 Zusammenfassung

Wie zu sehen ist, ist das Stundentotal deutlich überschritten worden. Folgende Faktoren haben dazu beigetragen:

- Unterschätzte Einarbeitungszeit bei der Mobile App Entwicklung für verschiedene Geräte (Tablet, Smartphone)
- Technische Probleme mit der Agenda beim Mobile App
- Probleme mit der Sharing Funktion im Mobile App auf anderen Geräten

Die direkten Folgen daraus waren, dass die Entwicklung des Frontend durch den Studenten Oussama Zgheb alleine durchgeführt werden mussten, welches bei ihm zu einem deutlichen Mehraufwand führte. Zusätzlich sind die Probleme mit der Sharing Funktion erst gegen Ende der Transition Phase entdeckt worden, sodass diese repariert werden mussten und zu einer Verzögerung in der Dokumentationsarbeit von Tobias Zahner führten, welche wiederum durch Oussama Zgheb aufgeholt wurden.

6 Meeting Protokolle

Sämtliche Meeting Protokolle sind aus Gründen der Übersichtlichkeit im Anhang zu finden. Die Meetings sind mit Datum, den anwesenden Personen, Ort sowie allfälligen zusätzlichen Dateien versehen.

7 Fussnoten Index

HLM-NG | Persönlicher Bericht

Oussama Zgheb & Tobias Zahner

11. Juni 2015

Datum	Version	Änderung	Autor
05.06.15	1.0	Eigener Teil	Oussama Zgheb
07.06.15	1.1	Eigener Teil	Tobias Zahner

Persönlicher Bericht

Oussama Zgheb

Als ich die Ausschreibung zu dieser Arbeit sah war es für mich bereits klar, dass ich diese Arbeit als meine erste Priorität wählen werde. Als Hacking-Lab Benutzer war ich sehr erpicht etwas dem Projekt beitragen zu können und die Möglichkeit mit neuen Technologien sowie viel Freiraum zu arbeiten besiegelten meine Wahl.

Vor dem Beginn der Bachelorarbeit hatte ich noch nie das Vergnügen mit AngularJS zu arbeiten. Um nicht bei Null anfangen zu müssen, hatte ich bei CodeSchool [1] einen Onlinekurs durchgeführt. Spätestens nach diesem Kurs hatte ich den Narren an AngularJS gefressen. Die anfänglichen Diskussionen zu den Anforderungen und die GUI Mockups waren sehr ausführlich und stellten sich bei der Entwicklung mehrmals als sehr wertvoll heraus. Die Umsetzung der REST API mit JAX-WS stellte sich als sehr »angenehm« heraus. Ich bin froh, die Entscheidung getroffen kein Framework wie Hibernate gewählt zu haben. Dies vor allem da weitere Einarbeit nötig gewesen wäre und sich somit die Prototypen herausgezögert hätten. Auch mit der Performance der eigenen Lösung bin ich äussert zufrieden und konnte dabei sehr viel mehr lernen, als wenn ich einfach ein bestehendes Framework konfiguriert hätte.

Leider kam der Entwicklungsprozess der App nicht so wie geplant voran. Deshalb lag es an mir alleine das Frontend, anstatt wie geplant mit der Mithilfe von Tobias, zu entwickeln. Dies gelang aber gerade durch die Vorbereitung und der Freude an den Technologien sehr gut, lediglich beim Design konnte ich nicht soviel Zeit wie gewünscht investieren. Die Zusammenarbeit mit Tobias war sehr angenehm und durch die zuvor absolvierte Studienarbeit vertraut.

Als schwierig erwies sich, die Voting Funktion im Frontend umzusetzen. Dies da die Operationen und Auswertungen teils sehr komplex sind. Dank hilfreichem Feedback von Ivan Bütler sowie ausgiebigem Testing sowie Optimierungen wurde diese Aufgabe zu meiner vollen Zufriedenheit gelöst.

Rückblickend auf die gesamte Bachelorarbeit muss ich sagen, dass ich zuversichtlich bin ein hochwertiges und modernes System entwickelt zu haben an welchem sich die Nachfolgenden Entwickler gut zurecht finden werden. Dabei bin mir sicher, dass der erste produktive Einsatz erfolgreich sein wird und freue mich zu darauf an diesem Event anwesend zu sein. Ich habe sehr viel gelernt, nicht nur die eingesetzten Technologien sondern die Entwicklung eines Systems von der Planung hin zu der Entwicklung bis zum Testing über verschiedene Komponenten. Auch in den Themen Projektmanagement und in der engen Zusammenarbeit mit einer Firma habe ich äusserst wertvolle Erfahrungen gewonnen, von welchen ich mir sicher bin, dass diese mir im späteren Berufsleben sehr von Nutzen werden sein.

[1] `https://www.codeschool.com/courses/shaping-up-with-angular-js`

Tobias Zahner

Das Projekt Hacking-Lab Mobile hörte sich für mich bei der Ausschreibung von allen Projekten am interessantesten an, da mich vor allem die App-Entwicklung sehr interessierte und auch der Hacking-Lab Event sehr vielversprechend klang.

Die ersten Wochen verbrachten wir vor allem mit GUI Mockups, die im Nachhinein zwar sehr wichtig waren, in meinen Augen aber ein wenig zu viel Zeit verbrauchten. So blieben für die Implementationen aller Komponenten lediglich 3 Monate. Da ich schon ein wenig Android-Erfahrung hatte übernahm ich vorerst den App-Teil und Oussama den Backend-Teil. Zu Beginn hatte ich einige Startschwierigkeiten, vor allem die korrekte Multi-Device-Unterstützung (Smartphones & Tablets) machte mir lange zu schaffen. Nach viel Einarbeitung gelang es mir schliesslich das Grundgerüst aufzustellen. Zum Glück hatte Oussama schon sehr früh den REST-Service zum Laufen gebracht, so dass ich immer mit den »richtigen« Daten arbeiten konnte. Als dann auch in der App der HTTP-Verkehr funktionierte ging es eine Weile sehr rasch voran. Zwar gab es immer wieder leichte Ausbremsungen wie z.B. die Umschaltung von HTTP zu HTTPS, dennoch war der Fortschritt wöchentlich erkennbar. Bis einen Monat vor der Abgabe war ich sehr gut im Zeitplan und ziemlich früh hatte ich dann alle Seiten mal implementiert mit dem Wissen, dass auf vielen Seiten noch Details zu tun sind. Ich war aber guter Dinge Oussama beim Frontend noch unterstützen zu können.

Leider erwiesen sich dann einige dieser Details als zeitraubende Knacknüsse, die viel Zeit in Anspruch nahmen. Insbesondere die Implementierung und vor allem das Debugging der Agenda, des Votings und des Sharings stellten sich als Zeitfresser heraus. So verbrauchte ich fast den gesamten Zeitpuffer - den wir zum Glück bei Projektplanung eingeplant hatten - für die Optimierung und Fehlerbehebung der App. Es stellte sich heraus, dass ich zwar zu Beginn des Projekts alle grossen Risiken beseitigen konnte, die grosse Menge der kleinen Details jedoch genauso viel Zeit in Anspruch nahm.

Kurz vor Schluss der Arbeit, als eigentlich nur noch Dokumentation angesagt war, stellte sich noch heraus, dass einige Funktionen wie z.B. das Sharen von Fotos auf vereinzelten Geräten nicht funktionierte. Da ich auf meinen zur Verfügung stehenden Testgeräten die Fehler nicht reproduzieren konnte, stellte sich ein Debuggen als sehr schwierig heraus. So wurden die letzten Wochen noch einmal sehr zeitaufwendig.

Insgesamt habe ich im Rahmen dieser Bachelorarbeit sehr viel über die Android-Entwicklung gelernt. Viele Dinge musste ich lange recherchieren, diese werde ich aber nun sicher nicht mehr vergessen. Müsste ich noch einmal von vorne beginnen, würde ich das Meiste gleich machen, lediglich die Agenda würde ich wohl versuchen von Grund auf selber zu implementieren ohne eine Library. Die Zusammenarbeit mit Oussama war einmal mehr sehr angenehm, leider konnte ich ihn zum Schluss aufgrund der oben genannten Ereignisse nicht wie gewünscht unterstützen, was die Entwicklung des Frontends anging. Dennoch ziehe ich insgesamt ein positives Fazit und bin vom entstandenen Projekt absolut überzeugt.

Hacking-Lab Mobile Event App | API

Oussama Zgheb

11. Juni 2015

Datum	Version	Änderung	Autor
11.03.15	1.0	Erste Version	Oussama Zgheb
13.03.15	1.1	Calls für Events & Speaker	Oussama Zgheb
17.03.15	1.2	Calls für Media	Oussama Zgheb
23.03.15	1.3	Calls für Event Item & Event Room	Oussama Zgheb
24.03.15	1.4	JSON Beispiele hinzugefügt	Oussama Zgheb
25.03.15	1.5	Delete Operationen sowie Formatierung	Oussama Zgheb
30.03.15	1.6	Code Formatierung, Cals für QR Code	Oussama Zgheb
31.03.15	1.7	QR Code Überprüfung , File Upload	Oussama Zgheb
31.03.15	1.8	Null Kennzeichnungen	Oussama Zgheb
01.04.15	1.9	Unique Kennzeichnung, Voting Duration	Oussama Zgheb
09.04.15	2.0	»Newest« Calls, Spam Schutz	Oussama Zgheb
14.04.15	2.1	Informationen & Konventionen gruppiert und erweitert	Oussama Zgheb
20.04.15	2.2	Event Active Felder, X-QRCode Header	Oussama Zgheb
27.04.15	3.0	Adm / Pub URL's (Änderungen v. Sitzung 23.04.15)	Oussama Zgheb
06.05.15	3.1	Voting Triggers, »/sliders« Call nun auch für Pub	Oussama Zgheb
08.05.15	3.2	User Listing für Pub entfernt, Social Status update	Oussama Zgheb
14.05.15	3.3	SpeakerIDFK in Event Item nachgeführt	Oussama Zgheb
17.05.15	3.4	Presentation Calls entfernt, Typo, Voting Pauses	Oussama Zgheb
19.05.15	4.0	Voting Calls	Oussama Zgheb
20.05.15	4.1	Zusätzliche Event Calls	Oussama Zgheb
22.05.15	4.2	Newest Calls verschoben, inTimeScoreWeight	Oussama Zgheb
28.05.15	4.3	Publish Calls	Oussama Zgheb
03.06.15	4.4	Voting Export Calls	Oussama Zgheb

Inhaltsverzeichnis

1 Einleitung

Dieses Dokument soll eine ausführliche Übersicht über die Rest API des Backend geben. Dabei sollen alle möglichen »Calls«, die zu erwartenden Antworten sowie Konventionen und Benutzungsinformationen aufgeführt werden. Ziel ist es den Entwickler des Mobile Apps sowie des Backends eine Hilfe bei der Erweiterung sowie Unterhalt von HLM-NG zu sein.

2 API

2.1 Informationen und Konventionen

2.1.1 Legende

- **S & Q** User Authentisierung benötigt (Siehe Kapitel »Login«)
- **[Null]** kennzeichnet Felder die »null« sein dürfen und können
- **[!]** kennzeichnet Felder die automatisch gesetzt werden. Dies bedeutet, dass bei einem POST / PUT diese nicht mitgeliefert werden müssen. Falls diese trotzdem mit gegeben werden, werden diese schlichtweg ignoriert.
- **[Unique]** kennzeichnet Felder welche Einzigartig sein müssen, falls etwas hinzugefügt / geändert wird und einen nicht einzigartigen Wert enthält wird ein Bad Request zurückgeliefert.
- **{varname}** Variablen werden in dieser Dokumentation mit geschweiften Klammern gekennzeichnet

2.1.2 Zeit und Daten

Das Format für »Date« **YYYY-MM-DD**.
Das Format für »Time« **HH:MM**
Das Format für »Date Time« **YYYY-MM-DD HH:MM**

2.1.3 Login

- Bei den als **S** gekennzeichneten Calls wird ein Basic **HTTP-Auth** erwartet[1]
  Das Passwort ist dabei gleich der DeviceID. Kann 401, 403 zurückliefern.
  - Bei erfolgreicher Authentifizierung wird die eigentliche Aktion durchgeführt
  - Bei fehlgeschlagener Authentifizierung wegen falschen Anmeldedaten wird erneut ein HTTP-Auth geschickt
  - Bei einem Fehler bei dem Anmeldedaten Parser wird ein Bad Request geschickt
- Bei den als Q gekennzeichneten Calls kann nebst der Basic HTTP-Auth ein zusätzliches Header Feld mit dem Namen »X-QRCode« mitgesendet werden. Bei erfolgreicher Auswertung des Wertes werden die Calls mit spezieller Berechtigung ausgeführt (z.B. Voting als Jury). Der Wert dieses Header ist gleich der Payload des QR Codes.

2.1.4 Diverses

- Eine fixe Reihenfolge der Properties ist bei JSON (standardmässig) nicht gewährleistet[2]
- Der Befehl »POST«
  - Liefert (falls erfolgreich) das eingefügt Element als JSON zurück
- Der Befehl »PUT »

[1] The Internet Society: HTTP Authentication (1999), https://www.ietf.org/rfc/rfc2617.txt (Stand 13.05.2015)
[2] ECMA: ECMAScript Language Specification (1999), Kapitel 4.3.3, http://www.ecma-international.org/publications/files/ECMA-ST-ARCH/ECMA-262,%203rd%20edition,%20December%201999.pdf (Stand 10.05.2015)

  - Updatet lediglich - Es wird kein neues Element erstellt falls unter die definierte ID nicht existiert
  - Ignoriert eine abweichende ID

- Spam Schutz
  Bei folgenden Aktionen wird beim überschreiten einer Threshold für eine gewisse Zeit zu jeder Anfrage ein 429 zurückgeliefert (too many requests).
  Die genauen Einstellungen sind »FileSettings« zu entnehmen.
  - Bei allen Anfragen die eine Benutzerauthentisierung fordern
  - Erstellung oder Update eines Benutzerprofils
- Last Update Time
  Bei jeder Resource gibt es einen call Namens »/lastupdate« welcher den Zeitpunkt (als Unix Timestamp) angibt, an welchem irgend ein Element geändert wurde. So kann der Client erfahren (durch vergleichen des letzten Zeitpunktes) ob sein Datenbestand noch aktuell ist.
- Alle Pfade werden relativ angegeben
- Thumbnails
  Bei allen »media« Links kann »_thumb« angehängt werden, um ein Thumbnail des besagten Bildes zu erhalten
- Caching
  Sämtliche Medien Ressourcen offerieren Client Caching durch ETag [3]
  Hier zu sehen ist die erste Anfrage für »shot0008.png«.
  Der Server sendet dem Client ein Etag mit welcher sich der Client speichern soll.

```
GET /hlmng/rest/pub/media/image/png/shot0008.png HTTP/1.1
Host: localhost:8443
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:37.0) Firefox/37.0
Accept: text/html,application/xhtml+xml,application/xml;
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Cache-Control: private, no-transform, max-age=1800
Expires: Thu, 01 Jan 1970 01:00:00 CET
Etag: "1712108524"
Content-Type: image/png Transfer-Encoding: chunked
Date: Tue, 28 Apr 2015 11:23:34 GMT
```

Hier zu sehen ist die erneute Anfrage für »shot0008.png«.
Der Client sendet »If-None-Match« mit, anhand diesem Wert entscheidet der Server ob das Bild 304 (not modified) oder 200 (nochmals neu senden) ist.

```
GET /hlmng/rest/pub/media/image/png/shot0008.png
HTTP/1.1 Host: localhost:8443
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:37.0) Firefox/37.0
Accept: text/html,application/xhtml+xml,application/xml;
Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate
Connection: keep-alive
If-None-Match: "1712108524"
Cache-Control: max-age=0
.
HTTP/1.1 304 Not Modified
Server: Apache-Coyote/1.1
Cache-Control: private, no-transform, max-age=1800
Expires: Thu, 01 Jan 1970 01:00:00 CET
Etag: "1712108524"
Date: Tue, 28 Apr 2015 11:23:37 GMT
```

[3] W3.org: Header Field Definitions, www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.19 (Stand: 10.04.15)

2.2 Event

eventID - Die eindeutige ID des Events
name - Der Name des Events
description - Ein kurzer Info Text zum Event
startDate - Datum beginn des Events
endDate - Datum letzter Tag des Events (kann gleich wie »from« Feld sein)
active - Gibt an ob der Event aktiv sein soll. Falls ein Event nicht aktiv ist, wird er im Frontend nicht angezeigt

```
{
        "eventID": 1,
        "name": "test",
        "description": "desc",
        "from": "2015-06-10",
        "to": "2015-06-11",
        "active": true
}
```

pub/event
GET - Liefert alle Events

pub/event/{id}
GET - Liefert den besagten Event

pub/event/{id}/speakers
GET - Liefer alle Speaker welche ein Event Item an diesem Event leiten

pub+adm/event/{id}/eventrooms
GET - Liefert alle Event Rooms welche zu Event {id} gehören

pub+adm/event/{id}/eventitems
GET - Liefert alle Event Items welche zu Event {id} gehören

pub+adm/event/{id}/news
GET - Liefert alle News Items welche zu Event {id} gehören

pub+adm/event/{id}/news/newest
GET - Liefert nur die neusten (höchste ID zuerst) 15 (Standardwert) Einträge. Es wird empfohlen diesen Call zu werden, ausser der User wünscht explizit alle Einträge.

pub+adm/event/{id}/socials
GET - Liefert alle Social Items welche zu Event {id} gehören

pub+adm/event/{id}/social/newest
GET - Liefert nur die neusten (höchste ID zuerst) 15 (Standardwert) Einträge. Es wird empfohlen diesen Call zu werden, ausser der User wünscht explizit alle Einträge.

pub+adm/event/{id}/votings
GET - Liefert alle Votings welche zu Event {id} gehören

pub+adm/event/{id}/pushes
GET - Liefert alle Pushes welche zu Event {id} gehören

adm/event
GET - Liefert alle Events
POST - Erstellt neuen Event

adm/event/{id}
GET - Liefert den besagten Event
PUT - Updatet den besagten Event
DELETE - Löscht den besagten Event

adm/event/{id}/eventrooms
GET - Liefert alle Event Rooms welche zu Event {id} gehören

adm/event/{id}/eventitems
GET - Liefert alle Event Items welche zu Event {id} gehören

2.3 EventItem

eventItemID - Die eindeutige ID des Event Item
name - Der Name des Item
description - Die Beschreibung des Item
date - Das Datum an welchem das Item stattfindet
startTime - Der Beginn des Item
endTime - Das Ende des Item
roomIDFK - Ein FK zu dem Raum in welchem das Item stattfindet
eventIDFK - Ein FK zu dem Event in welchem das Item stattfindet
speakerIDFK - Ein FK zu dem Speaker welcher das Item moderiert

```
{
        "eventItemID": 1,
        "name": "Advanced Cryptography",
        "description": "description of item..",
        "date": "2015-06-15",
        "startTime": "13:00",
        "endTime": "14:15",
        "roomIDFK": 1,
        "eventIDFK": 1,
        "speakerIDFK": 1
}
```

pub/eventitem
GET - Liefert alle Event Items

pub/eventitem/{id}
GET - Liefert besagtes Event Item

adm/eventitem
GET - Liefert alle Event Items
POST - Erstellt neues Event Item

adm/eventitem/{id}
GET - Liefert besagtes Event Item
PUT - Updatet besagtes Event Item
DELETE - Löscht besagtes Event Item

2.4 EventRoom

eventRoomID - Die eindeutige ID des Raumes
name - Die Raumbezeichnung
location - **[Null]** Eine genauere Bezeichnung zum Raum
eventIDFK - Ein FK zu dem Event in welchem der Raum verwendet wird

```
{
        "eventRoomID": 1,
        "name": "1.206",
        "location": "1",
        "eventIDFK": 1
}
```

pub/eventroom
GET - Liefert alle Event Rooms

pub/eventroom/{id}
GET - Liefert den besagten Event Room

adm/eventroom
GET - Liefert alle Event Rooms
POST - Erstellt neuen Event Room

adm/eventroom/{id}
GET - Liefert den besagten Event Room
PUT - Updatet den besagten Event Room
DELETE - Löscht den besagten Event Room

2.5 Media

mediaID - Die eindeutige ID jedes Medien Objektes
type - Der Kennzeichner des Medien Objektes. »jpeg«, »png« (TODO weitere Typen wie Video etc.)
link - Ein absoluter Link zur Ressource

```
{
        "mediaID": 1,
        "type": "jpg",
        "link": "http://localhost:8080/hlmng/rest/media/jpeg/1.jpg",
}
```

pub/media
GET - Liefert alle Medien

pub/media/{id}
GET - Liefert das besagte Medien Objekt

pub/media/{typ}/{filename}
GET - Liefert die gewünschte Medien Ressource
z.B. http://.../rest/media/image/png/image.png"

pub/media/upload
POST - **S** - Ermöglicht einen Upload von Media Dateien mittels Multipart-Data [4].
Die möglichen Return Codes sind (nebst den üblichen Authorization needed / Bad Request).

- 500 - Server konnte das File nicht speichern
- 415 - Falscher / fehlender / unbekannter Mimetype
- 413- Datei zu gross
- 200 - Hochgeladen, Objekt wird als JSON geliefert, sodass sich der Client die ID speichern kann

adm/media
GET - Liefert alle Medien

adm/media/{typ}/{filename}
GET - Liefert die gewünschte Medien Ressource
z.B. http://.../rest/media/image/png/image.png"

adm/media/upload
Gleich wie /pub/media/upload, ohne Authentisierung

[4] The Internet Society: Returning Values from Forms (1998), https://www.ietf.org/rfc/rfc2388.txt (Stand: 05.04.15)

adm/media/{id}
GET - Liefert das besagte Medien Objekt
DELETE - Löscht die verlinkte Medien Datei, der Eintrag in der Datenbank wird **nicht gelöscht.**
Dies da auf das Medien Objekt verlinkende Objekte weiterhin den Link benutzen können und dieser dann ein 404 zurückliefert.

2.6 News

newsID - Die eindeutige ID zu dem News Eintrag
title - Der Titel des News Eintrag
text - Der Text des News Eintrag
media - Der direkte Link zur angehängten Medien Datei
author - Der Autor (frei wählbarer String)
mediaIDFK - **[Null]** Ein FK zu der angehängten Medien Datei
eventIDFK - Ein FK zu dem Event zu welchem die News gehören

```
{
        "newsID": 1,
        "title": "Breaking news!"
        "text": "Team Switzerland takes the lead!",
        "media": "http://localhost:8080/hlmng/rest/media/jpg/1.jpg",
        "author": "Max Muster",
        "mediaIDFK": 1,
        "eventIDFK": 1
}
```

pub/news
GET - Liefert alle News Einträge

pub/news/{id}
GET - Liefert den besagten News Eintrag

adm/news
GET - Liefert alle News Einträge
POST - Erstellt neuen News Eintrag

adm/news/{id}
GET - Liefert den besagten News Eintrag
PUT - Updatet den besagten News Eintrag
DELETE - Löscht den besagten News Eintrag

2.7 Presentationpause

presentationpauseID - Die eindeutige ID der Presentationpause
start [!] - Der Startzeitpunkt der Pause
stop [!] [Null] - Der Stopzeitpunkt der Pause
votingIDFK - Ein FK zu dem dazugehörigen Voting

```
{
        "presentationpauseID": 1,
        "start": "14:02:03",
        "stop": "14:03:28",
        "votingIDFK": 1
}
```

adm/presentationpause/{id}
PUT - Updatet besagten Presentationpause Eintrag

adm/presentationpause/
POST - Erstellt neuen Presentationpause Eintrag

2.8 Publish

adm/publish/twitter
POST - Erstellt einen neuen Twitter Post, erwartet ein Social Item (max. 140 Zeichen)

adm/publish/facebook
POST - Erstellt einen neuen Facebook Post, erwartet ein Social Item (aktuelles User Access Token muss bestehen)

adm/publish/facebook/updatetoken
POST - Updatet das User Access Token welches für die Facebook Posts verwendet wird. Dieses Token wir per Facebook JS API zur Verfügung gestellt.

2.9 Push

pushID - Die eindeutige ID des Push
title - Der Titel der Push Nachricht
author - Der Autor (frei wählbarer String)
date - Das Datum an dem die Nachricht gesendet wurde
time - Der Zeitpunkt an dem die Nachricht gesendet wurde
receivedCounter - Die Anzahl Mobile Apps welche die Nachricht erhalten haben
failedCounter - Die Anzahl Mobile Apps an welche die Nachricht nicht gesendet werden konnte (Details siehe Log)

```
{
        "pushID": 1,
        "title": "push title",
        "author": "mmuster",
        "date": "2015-05-20",
        "time": "14:15",
        "receivedCounter": 10,
        "failedCounter" : 2
}
```

pub/push
GET - Liefert alle Push Einträge

/push/{id}
GET - Liefert besagten Push Eintrag

adm/push
GET - Liefert alle Push Einträge
POST - Erstellt neuen Push Eintrag
Der Post sollte etwa wie folgt aussehen:

```
{
        "author": "mmuster",
        "text": "it works",
        "title": "unbelievable"
}
```

adm/push/{id}
GET - Liefert besagten Push Eintrag
PUT - Updatet besagten Push Eintrag *****
DELETE - Löscht den besagten Push Eintrag *****
***** Die Push Notifaction auf dem Mobile App kann logischerweise nicht nachträglich geändert / gelöscht werden.

2.10 QR Code

qrCodeID - Die eindeutige ID des QR Code
createdAt - **[!]** Der Erstellungszeitpunkt des QR Code
claimedAt - **[!] [Null]** Der Zeitpunkt an dem der QR Code eingelöst wurde
payload - **[!] [Unique]** Der geheime Wert eines QR Code.
role - Die Rolle welche der QR Code legitmiert [jury,author]
userIDFK - **[Null]** Ein FK zu dem User welcher den QR Code »geclaimt« hat
eventIDFK - Ein FK zu dem Event in welchem der QR Code gültig ist

```
{
        "qrCodeID": 1,
        "createdAt": "2015-06-21 22:59:59",
        "claimedAt": "2015-06-21 23:09:13",
        "role": "jury",
        "userIDFK": 1,
        "eventIDFK": 1
}
```

adm/qrcode
GET - Liefert alle QR Codes
POST - Erstellt neuen QR Code

adm/qrcode/{id}
GET - Liefert den besagten QR Code
PUT - Updatet den besagten QR Code
DELETE - Löscht den besagten QR Code

adm/qrcode/{id}/render
GET - Liefert den QR Code als Bild zurück

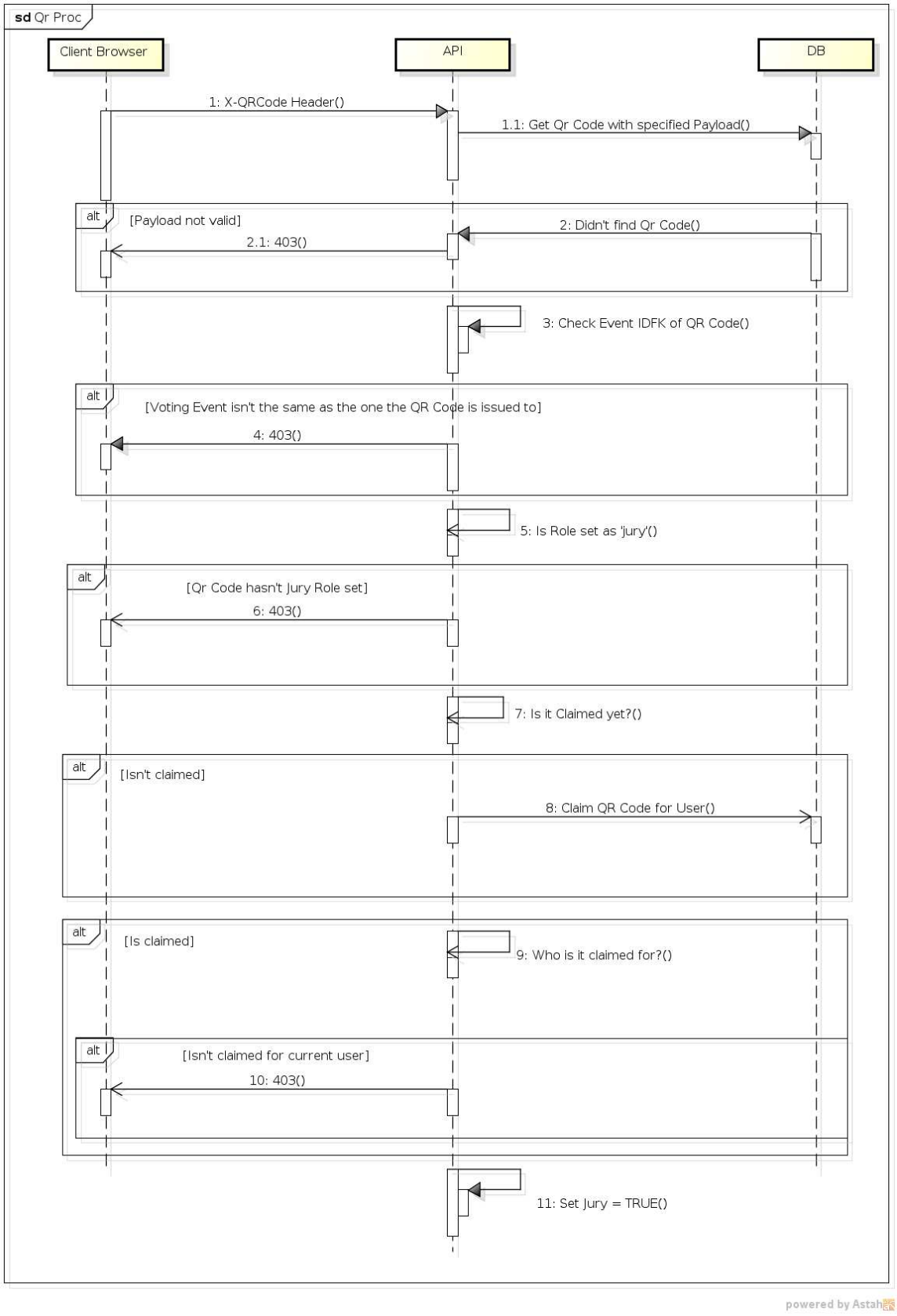
2.10.1 QR Code Payload

Der QR Code dient als geheimer Schlüssel, welche ein Mobile App User einlesen kann. Aus dem QR Code gewinnt man die Payload, welche als geheimer Schlüssel dient um beim Server gewisse Privilegien (siehe Rolle) zu erlangen. Gleichzeitig enthält die Payload Informationen für das Mobile App, welche die Rolle sowie den Gültigkeitsbereich festlegen. Dies ist so gelöst, da kein Mobile App User auf die QR Codes zugreifen darf und so auch keine Metainformationen zu dem eingescannten QR Code erhalten kann.

Payload:
[role]-[eventId]-[secret]
Role - Siehe Feld oben »role«
EventId - Die ID zu dem Event welcher der QR Code berechtigt ist
Secret - 20 Zeichen zufälliger geheimer Schlüssel

2.10.2 QrCode Überprüfung

Folgendes Diagramm soll den Ablauf und die möglichen Return Codes aufzeigen,



Sequenz Diagramm - QR Code Übeprüfung [5]

[5] Eigene Darstellung

2.11 Slider

sliderID - Die eindeutige ID des Slider
name - Die Bezeichnung des Slider
weigth - Die Gewichtung des Sliders
votingIDFK - Ein FK zu dem dazugehörigen Voting

```
{
        "sliderID": 1,
        "name": "Language",
        "weigth": 2,
        "votingIDFK": 1
}
```

pub/slider
GET - Liefert alle Slider

pub/slider/{id}
GET - Liefert dem besagten Slider

adm/slider
GET - Liefert alle Slider
POST - Erstellt neuen Slider

adm/slider/{id}
GET - Liefert dem besagten Slider
PUT - Updatet dem besagten Slider
DELETE - Löscht dem besagten Slider

adm/slider/{id}/votescorejury
GET - Liefert den gesamt Score der Jury für den besagten Slider

adm/slider/{id}/votescoreaudience
GET - Liefert den gesamt Score des Publikums für den besagten Slider

adm/slider{id}/votes
GET - Liefert alle Votes zu dem besagten Slider

2.12 Social

socialID - Die eindeutige ID des Social Eintrages
text - Der Text des Eintrages
status [!] - Der Moderationsstatus [pending,accepted,published,rejected]
authorName [!] - Der Username des Social Items, falls der User nicht gefunden wird ist das Feld »unknown«
media - **[Null]** Der direkte Link zu der angehängten Medien Datei (falls vorhanden)
userIDFK - Ein FK zu dem User der den Social Eintrag erstellt hat
mediaIDFK - **[Null]** Ein FK zu dem Medien Objekt welches angehängt sein kann
eventIDFK - Ein FK zu dem Event in welchem der Social Eintrag erstellt wurde

```
{
        "socialID": 1,
        "text": "Nice presentation!",
        "status": "pending",
        "authorName": "Harrison Ford",
        "media": "http://localhost:8080/hlmng/rest/media/jpg/1.jpg",
        "userIDFK": 1,
        "mediaIDFK": 1,
        "eventIDFK": 1
}
```

pub/social
GET - Liefert alle Social Einträge
POST - **S** - Erstellt neuen Social Eintrag, immer mit dem Moderationsstatus »pending«, ausser QR Code Auth wird mitgesendet

pub/social/{id}
GET - Liefert den besagten Social Eintrag

adm+pub/social/{id}/publications
GET - Liefert alle Publications

adm/social
GET - Liefert alle Social Einträge
POST - Erstellt neuen Social Eintrag

adm/social/{id}
GET - Liefert den besagten Social Eintrag
PUT - Updatet den besagten Social Eintrag
DELETE - Löscht den besagten Social Eintrag

2.13 Settings

pub+adm/settings
GET - Liefert gewisse Einstellungen der Applikation zurück

2.14 Speaker

speakerID - Die eindeutige ID des Speaker
name - Der Vor- und Nachname des Speaker
description - Ein kurzer Info Text zum Speaker
media - **[Null]** Der direkte Link zu der angehängten Medien Datei (falls vorhanden)
nationality - Das Kürzel zu dem Heimastland des Speaker. **Achtung:** Die Nationality wird immer in toUpperCase umgewandelt.
title - **[Null]** Der akademische Titel des Speaker, falls vorhanden
mediaIDFK - Ein FK zu dem Speaker Bild

```
{
        "speakerID": 1,
        "name": "Richard Stallman",
        "title": "PhD MIT",
        "media": "http://localhost:8080/hlmng/rest/media/jpg/1.jpg",
        "description": ".. freedom activist and computer programmer ..",
        "nationality": "USA"
        "mediaIDFK": 1
}
```

pub/speaker
GET - Liefert alle Speaker

pub/speaker/{id}
GET - Liefert den besagten Speaker

adm/speaker
GET - Liefert alle Speaker
POST - Erstellt einen neuen Speaker

adm/speaker/{id}
GET - Liefert den besagten Speaker

PUT - Updatet den besagten Speaker
DELETE - Löscht den besagten Speaker

2.15 User

userID - Die eindeutige ID welche den Benutzer identifiziert
name - Der Benutzername
deviceID - Die eindeutige ID jedes Android Gerätes
regID - Die Registrations ID bei dem Google Cloud Messaging Dienst

```
{
        "userID": 3,
        "name": "hmuster",
        "deviceID": "1234567890987654321",
        "regID": "1aa23cd45ef6789fg098hk765432ff1..."
}
```

pub/user
POST - Erstellt neuen user. Falls der Username bereits existiert wird 422 (unprocessable entity / exists) zurückgeliefert.

pub/user/{id}/changeregid
PUT - **S** - Updatet die »regID« des Users welcher sich anmeldet. Ein im JSON abweichender Username wird zurückgewiesen. Alle Variablen ausser »regID« werden ignoriert.

adm/user
GET - Liefert alle user
POST - Erstellt neuen user. Falls der Username bereits existiert wird 422 (unprocessable entity / exists) zurückgeliefert.

adm/user/{id}
GET - Liefert den besagten User
PUT - Updaten den besagten User
DELETE - Löscht den besagten

2.16 Vote

voteID - Die eindeutige ID des Vote (= eine Stimme)
score - Die abgebende Bewertung, beginnend bei 0 bis zu sliderMaxValue von Voting
isJury - Sagt aus ob die Vote von einem Jury Mitglied gemacht wurde oder nicht
sliderIDFK - Ein FK zu dem Slider über den die Bewertung abgeben wurde
userIDFK - Ein FK zu dem Benutzer welcher »gevotet« hat

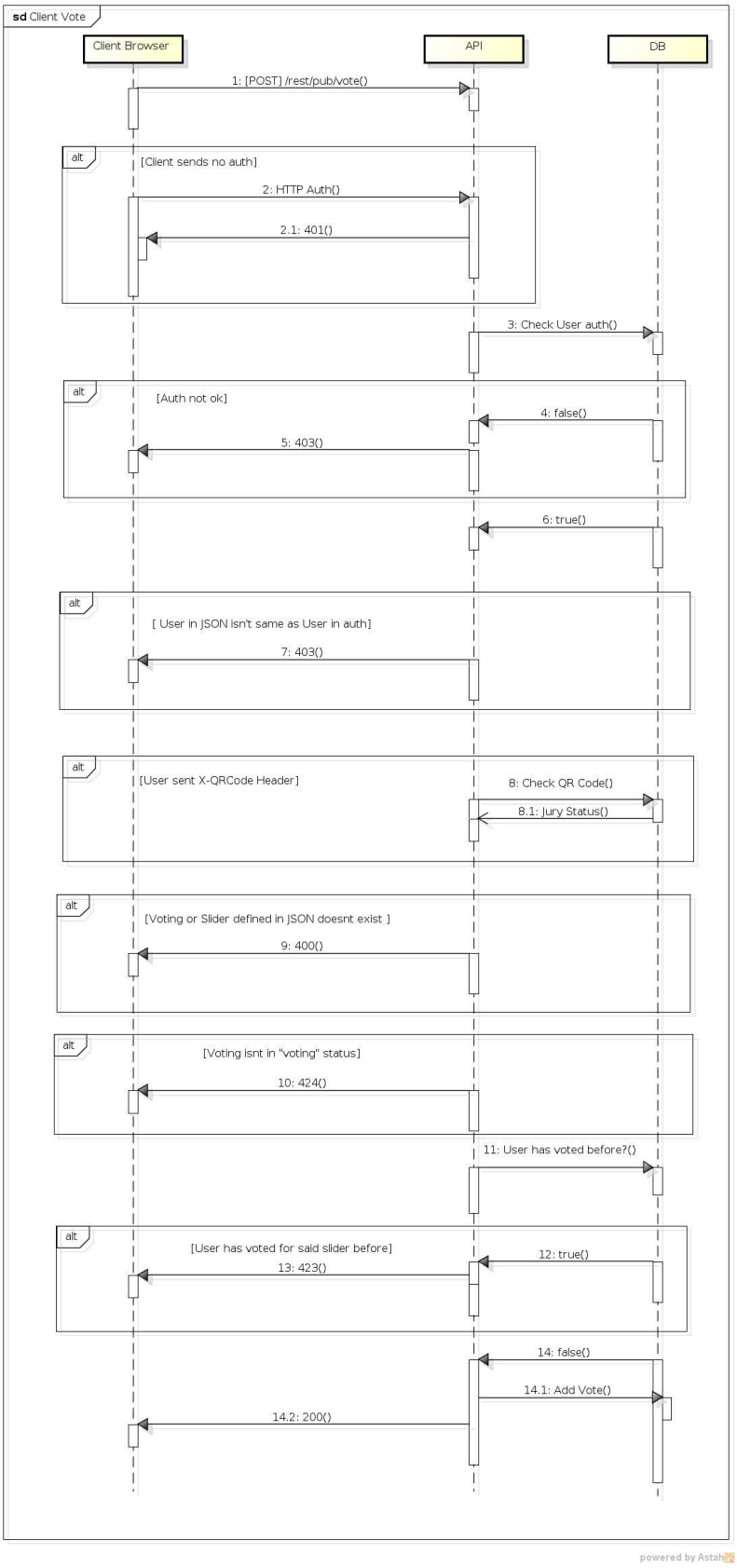
```
{
        "voteID": 1,
        "score": 7,
        "isJury": true,
        "sliderIDFK": 1,
        "userIDFK": 1
}
```

pub/vote
POST - **S** - **Q** - Erstellt neuen Vote
Es ist dabei nur ein Vote pro User für ein Slider möglich, bei erneutem Versuch wird HTTP Code 423 (locked) zurückgegeben.

adm/vote/{id}
GET - Liefert den besagten Vote
PUT - Updatet den besagten Vote

2.16.1 Vote Ablauf

Folgendes Diagramm soll den Ablauf und die möglichen Return Codes aufzeigen,



Sequenz Diagramm - Vote Ablauf [6]

[6] Eigene Darstellung

2.17 Voting

votingID - Die eindeutige ID des Votings
name - Der Name des Votings
juryCount - Die Anzahl anwesender Jury Mitglieder
status - Der Status des Voting (pre_presentation,presentation,presentation_end,voting,voting_end)
sliderMaxValue - Der maximale Wert der Slider
votingDate [!] [Null] - Das Datum an welchem das Voting auf den Status »voting« gesetzt wurde
votingStarted - **[!] [Null]** Der Zeitpunkt an welchem das Voting gestartet wurde (also status=voting)
votingDuration - Die Zeit in der, nach dem erhaltenen Push, abgestimmt werden darf
arithmeticMode - Die Auswahl der mathematischen Funktion zur Auswertung (median,todo)
presentationMinTime - Die minimal erlaubte Zeit für eine Präsentation
presentationMaxTime - Die maximal erlaubte Zeit für eine Präsentation
presentationStarted [!] [Null] - Der Zeitpunkt an dem die Präsentation gestartet wurde
presentationEnded [!] [Null] - Der Zeitpunkt an dem die Präsentation beendet wurde
inTimeScoreWeight - Legt die Gewichtung fest, mit welcher der Score für die Einhaltung der Zeit gewichtet wird
round - Die Durchführungsrunde des Voting
eventIDFK - Ein FK zu dem Event in welchem das Voting stattfindet

```
{
        "votingID": 1,
        "name": "Voting 1",
        "juryCount": 15,
        "status": "running",
        "sliderMaxValue": 10,
        "votingDate": "2015-05-05",
        "votingStarted": "14:04:27",
        "votingDuration": "00:00:50",
        "arithmeticMode": "median",
        "presentationMinTime": "00:05:00",
        "presentationMaxTime": "00:07:00",
        "presentationStarted": "14:30:30",
        "presentationEnded": "14:37:50",
        "inTimeScoreWeight": 1,
        "round": 1,
        "presentationIDFK": 1
}
```

pub/voting
GET - Liefert alle Votings

pub/voting/{id}
GET - Liefert besagtes Votings

pub/voting/{id}/sliders
GET - Liefert alle Slider des besagten Voting

adm/voting
GET - Liefert alle Votings
POST - Erstellt neues Voting

adm/voting/{id}
GET - Liefert besagtes Voting
PUT - Updatet besagtes Voting

adm/voting/{id}/sliders
GET - Liefert alle Slider des besagten Voting

adm/voting/{id}/totalscoreaudience
GET - Liefert den Score über alle Slider des Voting des Publikums

adm/voting/{id}/totalscorejury

GET - Liefert den Score über alle Slider des Voting der Jury

adm/voting/{id}/votes
GET - Liefert alle Votings aller Slider des Voting

adm/voting/{id}/votes/count
GET - Anzahl Votes (es muss dabei für jeden Slider ein Vote vorhanden sein um als gültig zu gelten)

adm/voting/{id}/duration
GET - Die gesamte Dauer der Präsentation (inkl. Abzug der Pausen), 00:00:00 falls noch nicht beendet

adm/voting/{id}/getpause
GET - Das momentan aktive (also noch keine Stopp-Zeit eingetragene) Pausenelement

adm/voting/{id}/presentationpauses
GET - Alle Pausenelemente

adm/voting/{id}/ispaused
GET - True oder False

adm/voting/{id}/presentationintime
GET - True oder False, je nach dem ob die »Duration« innerhalb des Range definiert durch minPresentationTime und maxPresentationTime liegt

adm/voting/{id}/audiencevotingover
GET - True oder False - True falls Started+Duration<Current

adm/voting/{id}/votes/audience
GET - Alle Votes welche durch das Publikum erstellt wurden

adm/voting/{id}/votes/audience/count
GET - Anzahl Votes des Publikum (es muss dabei für jeden Slider ein Vote vorhanden sein um als gültig zu gelten)

adm/voting/{id}/votes/jury
GET - Alle Votes welche durch die Jury erstellt wurden

adm/voting/{id}/votes/jury/count
GET - Anzahl Votes der Jury (es muss dabei für jeden Slider ein Vote vorhanden sein um als gültig zu gelten)

adm/voting/{id}/export
GET - CSV Datei mit Voting Infos und allen Votes

adm/voting/event/{id}/exportall
GET - CSV Datei mit allen Votings des Events (Infos & alle Votes)

adm/voting/event/{id}/exportallranked
GET - CSV Datei mit Audience + Jury Score für alle Votings des Events um eine Rangliste zu erstellen

2.17.1 Voting Triggers

Falls ein Voting durch einen Put Befehl in den Zustand »presentation_end« oder »voting« gesetzt wird, werden die Mobile App's per Push über diese Änderung informiert. Der Push der automatisch versendet wird, sieht wie folgt aus:

```
{
        "author": "vote_event",
        "date": "2015-05-06",
        "failedCounter": 2,
        "pushID": 9,
        "receivedCounter": 1,
        "text": "{ \"votingID\": 0 , \"name\": \"TEST\" }",
```

```
        "time": "16:24:55",
        "title": "presentation_end"
}
```

Das Feld »author« bezeichnet dabei, dass es sich nicht um einen regulären Push handelt und enthält »vote_event« als Wert. Das Feld »title« bezeichnet dabei den Zustand (also presentation_end oder voting). Das Feld Text enthält wiederum als JSON in einem String die notwendigen Metainformationen wie die ID und Namen des gestarteten Voting.

2.18 Time

/pub/time
Liefert die aktuelle Serverzeit im Format HH-mm-ss.SSS.

3 Fussnoten Index

Hacking-Lab Mobile Event App | Backend Installation Guide

Oussama Zgheb

11. Juni 2015

Datum	Version	Änderung	Autor
14.04.15	1.0	Initial	Oussama Zgheb
16.04.15	1.1	Weitere Settings erfasst	Oussama Zgheb
29.05.15	1.2	Facebook & Twitter	Oussama Zgheb
01.06.15	1.3	Installation, Unterhalt	Oussama Zgheb
10.06.15	1.4	Common Mistakes, SSL	Oussama Zgheb

Inhaltsverzeichnis

1 Einleitung

Dieses Dokument soll einer mit der Materie vertrauten Person eine Hilfestellung zur Konfiguration und Installation des HLM-NG Backend geben.

2 Voraussetzungen

Die Software wurde auf unter Ubuntu 12.04 LTS, Lubuntu 14.10 & 15.04 sowie Debian 7 entwickelt und getestet , weitere Betriebssysteme sind höchst wahrscheinlich auch möglich einzusetzen.

Folgende Software wird auf dem Zielgerät benötigt:

- Tomcat 7
- MySQL 5.6
- Eclipse EE (Kepler oder neuer)

2.1 Google Cloud Messaging

Google Cloud Messaging [1] wird eingesetzt, um Push Meldungen auf die Mobile Apps zu verteilen. Um den Dienst benutzen zu können, braucht man ein Google Konto [2] sowie einen ApiKey. Um den ApiKey zu erhalten muss man unter `https://console.developers.google.com/project` ein neues Project erstellen. Nun findet man unter »APIs & Auth« den Dienst »Google Cloud Messaging for Android«, dieser mussa aktiviert werden. Unter »APIs & Auth -> Credentials« kann ein »Key for server applications« (API Key) erstellt werden. Diese Daten werden später benötigt.

2.2 Facebook

2.2.1 App erstellen

Um auf eine Facebook Seite zu Posten wird eine App benötigt, diese kann unter `https://developers.facebook.com/` erstellt werden. Zuerst muss jedoch ein Facebook Konto bestehen sowie auf Facebook angemeldet sein. Wichtig: Der App Typ muss dabei »Website« sein und die Site URL richtig gesetzt sein, z.B. »https://fix.confoxy.com/hlmng/frontend/«. Nun findet man im Dashboard der App folgende Daten: »App ID« und »App Secret«.

2.2.2 Konfiguration

Nun müssen die Parameter der Facebook4J Library [3] hinterlegt werden. Dies geschieht in der Datei facebook4j.properties im Rootverzeichnis »src«. Die Datei sollte also wie folgt aussehen:

```
oauth.appId=****************
oauth.appSecret=********************************
oauth.permissions=publish_actions, manage_pages, publish_pages
```

Diese Tokens laufen nie ab, es Bedarf also keiner nachträglichen Änderungen mehr. [4]. Nach Bedarf kann auch »debug=true« hinzugefügt werden, diese liefert wertvolle Informationen auf System.out.

2.3 Twitter

2.3.1 App erstellen

Um die Twitter API zu benutzen muss man unter `https://apps.twitter.com/` ein App erstellen. Dafür wird ein Twitter Account benötigt, bei bedarf sollte dieser zuerst erstellt und angemeldet sein. Wichtig ist, dass beim erstellen

[1] `https://developer.android.com/google/gcm/gs.html`
[2] `https://accounts.google.com/signup`
[3] `http://facebook4j.org/`
[4] `https://developers.facebook.com/docs/facebook-login/access-tokens`

der App das Feld »Website« mit der richtigen URL ausgefüllt wird, also z.B. »https://fix.confoxy.com/hlmng/frontend/«. Nach dem erstellen der App sollte man nun unter »Keys and Access Tokens« die benötigten Daten finden. Folgende Parameter benötigt, »Consumer Key«, »Consumer Secret«, »Access Token«, »Access Token Secret«.

2.3.2 Konfiguration

Nun müssen die Parameter der Twitter4J Library [5] hinterlegt werden. Dies geschieht in der Datei twitter4j.properties im Rootverzeichnis »src«. Die Datei sollte also wie folgt aussehen:

```
oauth.consumerKey=**********************
oauth.consumerSecret=**************************************************
oauth.accessToken=******************************************
oauth.accessTokenSecret=******************************************
```

Diese Tokens laufen nie ab, es Bedarf also keiner nachträglichen Änderungen mehr, ausser das App wird durch den Twitter Account entfernt. [6]. Nach Bedarf kann auch »debug=true« hinzugefügt werden, diese liefert wertvolle Informationen auf System.out.

2.4 Tomcat

Um sicherzustellen, dass genügend Ressourcen vorhanden sind, wird empfohlen den Java Heap Space etwas grösser als die Default Einstellung festzulegen. Dies geschieht im File »/etc/default/tomcat7«. Dort sollte die Zeile »JAVA_OPTS« wie folgt aussehen:

```
JAVA_OPTS="-Djava.awt.headless=true -Xmx768m -Xms384m -XX:+UseConcMarkSweepGC"
```

Xmx ist dabei die maximale Heap Size, Xms die initiale Grösse. Bei grosser Benutzeranzahl muss die maximale Heap Size eventuell erhöht werden.

3 Installation Server

3.1 Ordner erstellen

Das Backend muss Daten auf dem Filesystem ablegen können. Empfohlen ist, die folgenden Ordner unter **»/var/lib/hlmng«** abzulegen. Es werden die Unterordner »logs«, »media«, und »qr« benötigt. Wichtig dabei ist, dass der User tomcat7 owner ist. Dies kann wie folgt übernommen werden:

```
chown tomcat7 /var/lib/hlmng/*
```

3.2 HLMNGSettings

Unter **»src/settings/HLMNGSettings.java«** können die Parameter für das Backend vorgenommen werden. Nachfolgend werden alle erklärt,

3.2.1 Pfade

- jdbcPath (String) = Der JDBC Pfad zur Datenbank
  z.B. »jdbc:mysql://127.0.0.1/hlmng«, wobei hlmng der Datenbank Schema Name ist
- pubURL = Der Pfad unter welchem der Public Rest API Teil sein soll
  z.B. »/pub«
- admURL = Der Pfad unter welchem der Admin Rest API Teil sein soll
  z.B. »/adm«

[5] http://twitter4j.org/
[6] https://dev.twitter.com/oauth/overview/faq

- qrFileRootDir (String) = Der Pfad unter welchem alle die gerenderten Qr Codes abgelegt werden z.B. »/var/lib/hlmng/qr/«
- mediaFileRootDir (String)= Der Pfad unter welchem alle heraufgeladenen Dateien abgelegt werden z.B. »/var/lib/hlmng/media«
- logFileRootDir (String) = Der Pfad unter welchem die Logfiles abgelegt werden (falls aktiv) z.B. »/var/lib/hlmng/logs«
- restAppPath (String) = Der Pfad unter welchem die Schnittstelle erreichbar sein soll z.B. »https://fix.confoxy.com/hlmng/rest«
- gcmURL (String) = Die URL der Google Cloud Messaging API um Nachrichten zu senden z.B: »https://android.googleapis.com/gcm/send«
- Der Subpfad zur Rest API wird im File »WebContent/web.xml« festgelegt:

```
<servlet-mapping>
        <servlet-name>
                Jersey REST Service
        </servlet-name>
        <url-pattern>
                /rest/*       <---- */
        </url-pattern>
</servlet-mapping>
```

3.2.2 Login Daten

- jdbcUser (String) = Der JDBC User für die Datenbank
- jdbcPassword (String) = Das JDBC Passwort für den User
- apiKey (String) = Der vom Google erhaltene Key für ihre API's

3.2.3 Facebook

- facebookAppId (String) = Die Facebook App ID
- facebookPageId (String) = Die Facebook Page ID auf welche gepostet werden soll

3.2.4 Weiteres

- cacheTime (int) = Cache Max-Age in Sekunden
- qrCodeWidth , qrCodeHeight (int)= Die Dimension der gerenderten Qr Codes in Pixel
- maxMediaImageSizeMB (Double) = Die maximale Dateigrösse für die Datei uploads in Megabyte
- logSysErr (boolean) = Falls Wahr wird jeglicher Log Output auf System.Error umgeleitet anstatt direkt in das Logfile geschrieben zu werden
- selectLimit (int) = Die Anzahl von Elementen die bei limitieren Queries geladen werden sollen (z.B. bei /newest calls)
- maxActionsAllowed (int) = Die Anzahl Aktionen (siehe API Dokumentation -> Spam Schutz) welche in »actionGraceTime« durchgeführt werden dürfen
- actionGraceTime (int) = Die Aktionen Zeitfenster (in ms)
- Angestrengtheit (int) = Die Länge der QR Code Payload in Bit
- mediaUploadThumbnailPixel (int) = Die maximale Höhe oder Länge eines Thumbnails

3.3 MySQL

3.3.1 Konfiguration

Auf der MySQL Server Instanz muss ein Schema mit dem Namen »hlmng« erstellt werden (andere Namen sind möglich, diese müssen dann auch so in die Konfiguration übernommen werden). In dieses Schema müssen alle sich unter **»src/doc/6_SQL_Creates«** befindende SQL Dateien importiert werden. Dies ist möglich mit dem »source« Befehl oder einer GUI wie MySQL Workbench [7]. Nun sollte ein Benutzer mit den benötigten Berechtigungen (delete,insert,select,update) angelegt werden, dies sollte etwa wie folgt aussehen:

```
CREATE USER 'user'@'localhost' IDENTIFIED BY '************';
GRANT DELETE,INSERT,SELECT,UPDATE ON hlmng.* TO 'user'@'localhost';
```

3.3.2 Verbindungsdaten

Unter **»hlmmng/WebContent/META-INF/context.xml«** werden die Informationen für die MySQL Verbindung festgelegt, siehe »url« sowie »username« und »password«.

```
<Resource name="jdbc/hlmng" auth="Container" type="javax.sql.DataSource"
driverClassName="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost:3306/hlmng?autoReconnect=true"
username="user" password="password" />
```

3.4 HTTPS

Als erstes muss ein Keystore mit einem Cert erstellt werden.

```
keytool -genkey -alias tomcat -keyalg RSA
```

Nun muss im server.xml von Tomcat der Connector konfiguriert werden, wichtig ist, dass der User Tomcat dieses File lesen kann.

```
<Connector executor="tomcatThreadPool" port="8080"
protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443" />

<Connector SSLEnabled="true" acceptCount="100" clientAuth="false"
disableUploadTimeout="true"  enableLookups="false" maxThreads="25"
keyAlias="hlmng" port="8443" keystoreFile="/home/ozzi/.keystore"
keystorePass="*****" protocol="org.apache.coyote.http11.Http11NioProtocol"
scheme="https" secure="true" sslProtocol="TLS" />
```

4 Installation Software

4.1 Eclipse Einstellungen

Tipp: Das angelegte Projekt sollte vom Typ »Dynamic Web Project« sein, damit ein Tomcat Server in Eclipse hinzugefügt werden kann und somit auf einfache Weise lokal deployed werden kann.

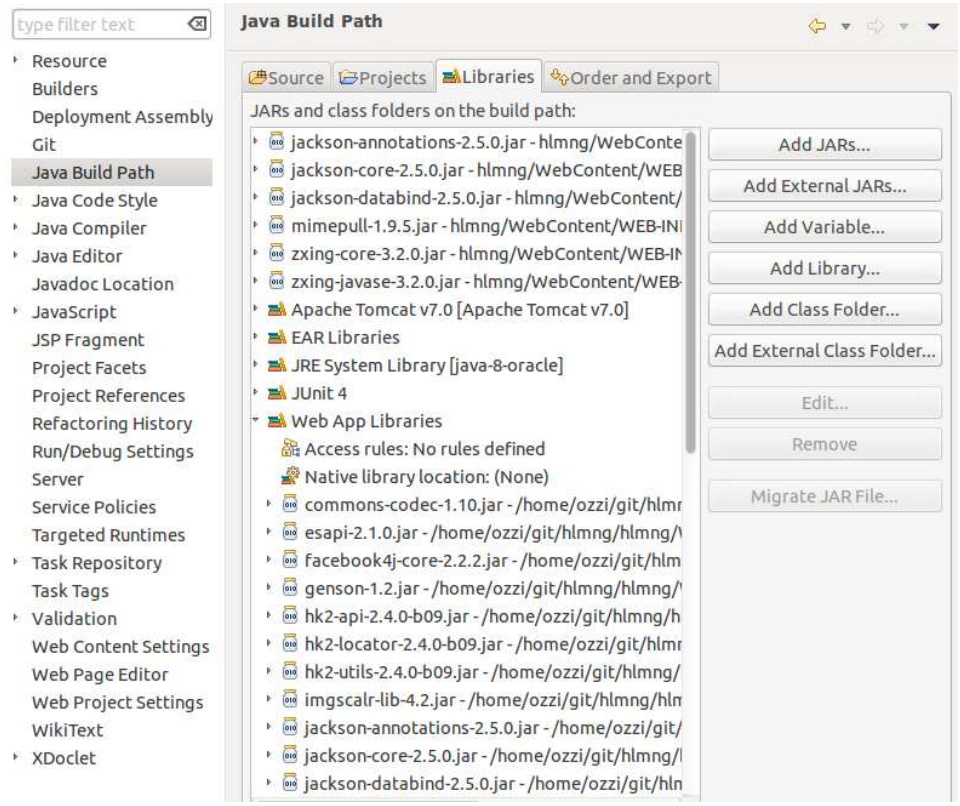
Bei Eclipse EE ist es wichtig, das »Compiler compliance level« auf 1.7 zu setzen. Da ansonsten bei der Kompilation mit OS X oder potentiell weiteren Betriebssystem die folgende Exception auftritt:

```
Unsupported major.minor version 52.0
```

Das »Compiler compliance level« kann in den Properties unter »Java Compiler« gefunden werden, siehe untenstehenden Screenshot.

[7] http://www.mysql.de/products/workbench/

Zudem müssen die Libraries im Buildpath wie folgenden Screenshot hinzugefügt werden.



4.2 Version Control

Die neuste Version kann per GitHub [8] bezogen werden.

4.3 Kompilaton

4.3.1 Eclipse

Unter Eclipse kann man das Projekt wie folgt exportieren:
Project Explorer -> Rechtsklick -> Export -> War File

4.3.2 Commandline

```
cd /path/to/project
jar cvf hlmng.war *
```

[8] https://github.com/ozzi-/hlmng

4.4 Installation

Das erstellte WAR File muss in den »WebApps« Ordner von Tomcat kopiert werden. Dieser befindet sich bei Debian unter »/var/lib/tomcat7/webapps/«. Die Anwendung sollte nun automatisch deployed werden und sogleich erreichbar sein.

4.4.1 Überprüfung

Falls alles erfolgreich installiert wurde, sollte unter `https://localhost:8443/hlmng/frontend/` das Frontend geladen werden.

Als Test sollte erfolgreich ein »Speaker« erstellt werden können. Unter `https://localhost:8443/hlmng/rest/adm/speaker` sollte der soeben erstellte »Speaker« nun als JSON angezeigt werden. Zudem sollte das Log frei von Warnings oder Errors sein.

4.4.2 Common Mistakes

- Falsch Verbindungsdaten in »context.xml«

```
com.mysql.jdbc.exceptions.jdbc4.MySQLNonTransientConnectionException:
Could not create connection to database server. Attempted reconnect 3 times.

oder

Caused by: java.sql.SQLException: Access denied for user 'hlmdng_db' . .
```

- »Seite konnte nicht gefunden werden«
  Sicherstellen, dass die Pfade richtig konfiguriert sind und die Applikation von Tomcat geladen wurde.
- Nicht das Log Konsultieren
  Viele Fehler können durch das konsultieren des Log lokalisiert werden.
- Falsche Systemzeit
  Falls eine falsche / abweichende Systemzeit auf dem Server (sowie auf dem System des Operator Browsers) kann es zu Fehlern bei Countdown und Timer anzeigen geben. Auch Twitter sowie Facebook können bei abweichender Systemzeit die Posts verweigern (respektive deren Oauth Implementiertung).
- Twitter Publishing: Error 500
  Dieser Fehler ist zu sehen in der Browser JS Konsole und macht sich bemerkbar, wenn beim Posten auf Twitter der Spinner nicht verschwindet / nichts passiert. Dies liegt höchstwahrscheinlich an falschen Tokens in der Datei »src/twitter4j.properties«.
- Facebook Post: Error 424
  Dieser Fehler erscheint in der Browser JS Konsole, wenn in »src/facebook4j.properties« die Tokens falsch sind oder in »WebContent/frontend/js/app.js« die falschen Facebook ID hinterlegt ist.

5 Unterhalt

Folgende Unterhaltsarbeiten können im längerfristigen Betrieb auftreten:

- Expiry der API Token
  Neue Token generieren und in der Applikation einfügen.
- Festplattenspeicher
  QR Codes auf dem Dateisystem können gelöscht werden, bei Bedarf werden diese neu angelegt.
  Alte Log Dateien können migriert oder gelöscht werden.

- Arbeitsspeicher
  Da kein langfristiger Testlauf mit mehren hunderten Benutzern im Rahmen der Entwicklung der Bachelorarbeit simuliert werden konnte, sollte beim ersten produktiven Einsatz die Arbeitsspeicher Auslastung überwacht werden, z.B. mit JStat [9].

  - Datenbank
    Der übliche Unterhalt der Datenbank sollte gewährleistet sein.

- Libraries
  Es werden einige Libraries verwendet, diese sollten periodisch oder bei bekannten Sicherheitslücken aktualisiert werden und das Projekt neu kompiliert, getestet und deployed werden.

- ui-Router
  Da das Modul ui-Router noch in Entwicklung ist, kann es durchaus sein, dass es noch grössere Änderungen geben wird. Diese müssten bei einem Update des Moduls eventuell nachgeführt werden.

[9] `https://docs.oracle.com/javase/7/docs/technotes/tools/share/jstat.html`

6 Fussnoten Index

Hacking-Lab Mobile Event App | Frontend User Manual

Oussama Zgheb

11. Juni 2015

Datum	Version	Änderung	Autor
29.05.15	1.0	Initial	Oussama Zgheb
30.05.15	1.1	Bedienkonzept, Moderation, Voting	Oussama Zgheb
01.06.15	1.2	Events, Workflow	Oussama Zgheb

Inhaltsverzeichnis

1 Einleitung

Dieses Dokument soll einer mit der Materie vertrauten Person eine Hilfestellung zur Konfiguration und Installation des Frontend geben.

2 Voraussetzungen

- Das Frontend wurde entwickelt und getestet mit Chromium (Chrome) 41 sowie Firefox 37/38.
  Weitere moderne Browser sollten ohne Probleme auch funktionieren.
- Das Frontend ist dank Bootstrap auch auf Mobile Geräten benutzbar.
- Javascript muss aktiviert sein sowie eventuell das Zertifikat hinzugefügt werden.
- Es wird auf gewissen Seiten das Facebook SDK eingebunden. [1]
  Gewisse Browser Plugins (Adblock Plus, NoScript) können die Funktionalität beeinträchtigen.

3 Installation

Folgendes Kapitel richtet sich an die Person welche das Frontend einrichten und unterhalten wird. Wissen zu den eingesetzten Technologien wird empfohlen (AngularJS, Linux).

3.1 app.js

Der AngularJS Applikation muss bereits beim Start gewisse Parameter bekannt sein, deshalb müssen unter **»src/WebContent/frontend/js/app.js«** zwei Parameter angepasst sein. Der Rest Path muss festgelegt sein, um eine Verbindung aufzubauen. Die Facebook App ID muss bekannt sein, bevor der zugehörige Controller startet. »app.js« sollte also folgende Zeilen enthalten:

```
var apiUrl = '../rest/adm/';
var facebookAppId=1234578901234567;
```

3.2 Zugriffsschutz

Das Frontend selbst verfügt über keine Authentisierung sowie Authentifizierung, dies wird auf dem betroffenen Webserver mittels dem Apache Plugin »mod_but« realisiert.

Dabei können gewisse URLs als Private gekennzeichnet werden, welche dann ein Cookie verlangen. Falls dies fehlt wird auf eine Loginmaske weitergeleitet (HTTP Code 302). Auf dieser Maske werden die Login Daten überprüft und je nachdem ein Cookie gesetzt. Dieser Vorgang sowie weitere Informationen sind in der Dokumentation »mod_but« [2] der Compass Security zu finden.

3.3 httpd.conf

In der Datei **»/opt/apache_but/httpd/conf/httpd.conf«** wurden folgende zusätzliche Anpassungen zu der Default Config der Compass Security vorgenommen.

3.3.1 mod_rewrite

Das Modul »mod_rewrite« ist so konfiguriert, dass automatisch HTTPS forciert wird. Dies muss jedoch für **»/rest/-pub/media/*«** deaktiviert werden, da externe APIs (Facebook & Twitter) Zugang benötigen und selbst signierte Zertifikate nicht akzeptiert werden.

[1] https://developers.facebook.com/docs/javascript
[2] Ivan Bütler, »MOD_BUT Documentation« , 10. Aug. 2009

```
RewriteEngine On
LogLevel alert rewrite:trace4
RewriteCond %{HTTPS} off
RewriteCond %{SCRIPT_FILENAME} !\/hlmng\/rest\/pub\/media\/*
RewriteRule ^(.*)$ https://%{HTTP_HOST}%{REQUEST_URI}
```

3.3.2 mod_but

Das Modul »mod_but« übernimmt die »Login Funktion«, wichtig dabei ist, dass der Pfad zu **»/rest/adm/*«** sowie **»/frontend/*«** durch diesen Mechanismus gesichert ist.
Folgende Einstellungen müssen dabei vorgenommen werden:

```
MOD_BUT_SESSION_FREE_URL '(^\*\$)|(^/hlmng/rest/pub/.*)|(^/error/)|(^/renew)|
(^/favicon\.ico\$)|(^/info)|(^/en/)|(^/de/)|(^/static/)|(^/fbws/)'
```

```
<Location /hlmng/frontend/>
        MOD_BUT_LOGON_REQUIRED      On
        ProxyPass http://localhost:8080/hlmng/frontend/ timeout=10
</Location>

<Location /hlmng/rest/adm/>
        MOD_BUT_LOGON_REQUIRED      On
        ProxyPass http://localhost:8080/hlmng/rest/adm/ timeout=10
</Location>

<Location /hlmng/rest/pub/>
        MOD_BUT_LOGON_REQUIRED      Off
        ProxyPass http://localhost:8080/hlmng/rest/pub/ timeout=10
</Location>
```

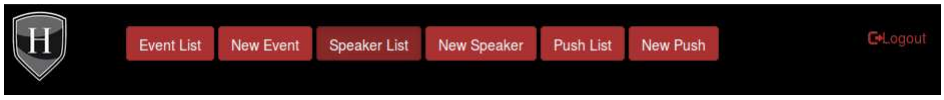
4 Benutzung

Folgendes Kapitel soll den Operatoren des Frontend das Grundlegende Konzept erläutern sowie als Nachschlagewerk dienen.

4.1 Allgemeines Bedienkonzept

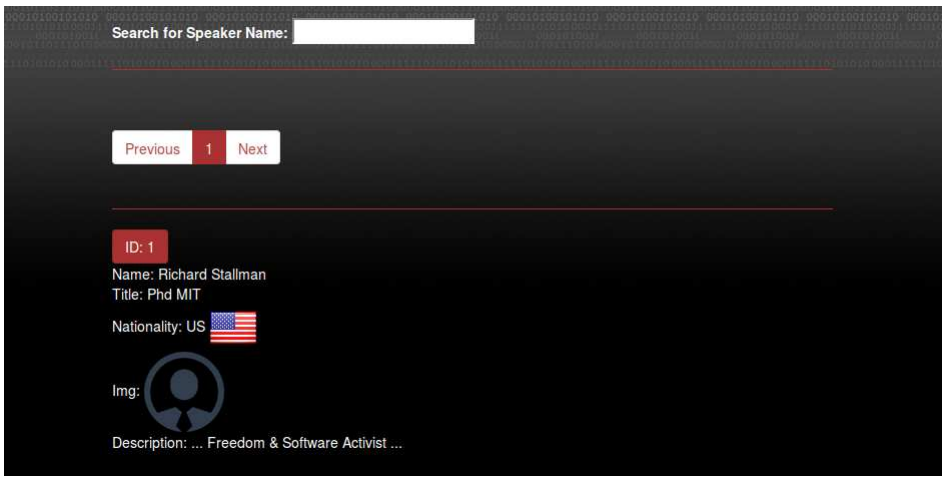
4.1.1 Menüleiste

In der Menüleiste sind die momentan verfügbaren Ansichten eingeblendet. Die aktive Ansicht ist dabei dunkler eingefärbt, in diesem Beispiel »Speaker List«. Per Klick auf »Logout« kann jederzeit die Session mit dem Server beendet werden.



4.1.2 Ansicht

Jede Ansicht hat eine andere Funktion, sei es eine Übersicht, ein Formular zur Erstellung eines Objektes und so weiter.
»List« Ansichten verfügen wo sinnvoll über ein Suchfeld, die Liste wird in Echtzeit gefiltert. Bei List Ansichten bei denen viele Objekte erwartet werden, ist zudem eine »Pagination« Funktion aktiv, welche es erlaubt die Objekte auf mehreren Seiten aufzuteilen. *Tipp*: Bei einer Suche über das Suchfeld werden alle Seiten durchsucht und auf einer Seite zusammengeführt.



4.1.3 Operationen

Objekt editieren Auf den roten »ID« Button des gewünschten Objektes klicken 

–> 

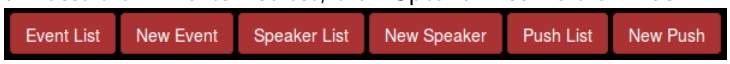
–> 

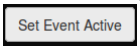
Objekt löschen Objekt editieren –> 

***** Nicht alle Objekt Typen können gelöscht werden da Abhängigkeiten untereinander existieren

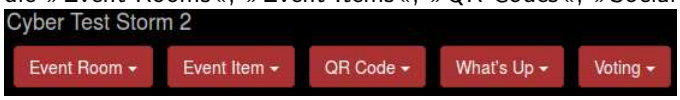
4.1.4 Bereiche

Globaler Bereich Im Globalen Bereich wird alles, was ausserhalb eines Event Kontextes passiert, verwaltet. Dies umfasst die »Events« selbst, die »Speaker« sowie die »Push« Funktion.



Event spezifischer Bereich Um Objekte eines Events zu verwalten, muss man in den Event spezifischen Bereich. Dies geschieht durch den 

Knopf welcher sich bei jedem Event Objekt befindet. Dieser Bereich umfasst die »Event Rooms«, »Event Items«, »QR Codes«, »Social Items«, »News« sowie »Votings«.



4.2 Events

Beim Erstellen eines Events können folgende Parameter gesetzt werden,

ID Hier sollte man die gleiche Event ID wie auf dem Hacking-Lab Server verwenden, sodass das Mobile App die Scores und Teams anzeigen kann

Name Der Name welcher angezeigt werden soll

Description Ein aussagekräftige Beschreibung des Events

From Das Startdatum des Events

To Das Enddatum des Events - darf gleich dem From Datum oder später sein

Active Falls die Checkbox aktiv ist, wird der Event in der Eventauswahl im Mobile App angezeigt

4.3 Voting

Beim Voting geht es darum, dass ein Team seine Lösung zu einer Hacking-Lab Challenge präsentiert. Die Zeit der Präsentation soll dabei gemessen und danach ein Voting gehalten werden. Das Voting dient einerseits zum Einbezug der Zuschauer (Audience Votes) sowie zur offiziellen Bewertung durch die Jury (Jury Votes).

4.3.1 Voting Parameter

Beim Erstellen eines Voting können folgende Parameter gesetzt werden,

Name Der Name welche angezeigt werden soll

Voting Duration Die Zeit für welche das Audience Voting laufen soll, die Jury kann Voten bis Zustand

Jury Member Count Legt fest, wie viele Jury Mitglieder anwesend sind, dies dient lediglich zur späteren Überprüfung wie viele tatsächlich gevotet haben

Presentation Min Time Legt fest, wie lange die Präsentation mindestens sein muss

Presentation Max Time Legt fest, wie lange die Präsentation maximal sein darf

Rating Mode Legt fest, was die zu verwendende Formel bei der Auswertung sein soll

Slider Max Value Der maximale Score welcher für jeden Slider abgegeben werden kann

In Time Score Weight Der Gewichtungsfaktor für den In Time Score, welcher für die Einhaltung des Präsentations Zeitfenster vergeben wird

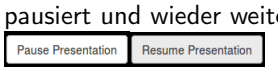
Slider Jeder Slider repräsentiert einen Aspekt des Vortrags, welcher bewertet werden soll. Jeder einzelne hat einen eigenen Gewichtungsfaktor

4.3.2 Zustände

Ein Voting befindet sich immer in einem der folgenden Zuständen,

pre_presentation Die Präsentation wurde erfasst, jedoch noch nicht gestartet.

presentation Die Präsentation hat begonnen, die Zeit wird gemessen. Die Präsentation kann dabei beliebig oft pausiert und wieder weitergeführt werden.



presentation_end Die Präsentation wurde beendet, die Mobile Apps werden benachrichtigt und zeigen dem Benutzer an, dass bald ein Voting startet.

voting Das Voting hat begonnen, die Mobile Apps werden erneut benachrichtigt und die Benutzer können nun ihre Votes abgeben.

voting_end Das Voting wurde beendet. Die Statistiken sind nun verfügbar.

4.3.3 Statistiken

Die Statistiken sind verfügbar, sobald das Voting im Zustand »voting_end« ist. Die Bewertung sowie alle Votes können als CSV (Comma-separated values) exportiert werden.

Falls eine Präsentation den zeitlich vorgegeben Rahmen eingehalten hat, wird dies mit dem maximal möglichen »Slider Max Score« und der definierten Gewichtung »In Time Weight« in der Jury Voting belohnt.
***** Bei der automatischen Berechnung ist die momentan einzige implementierte Formel »Arithmetic«.

4.4 Autorisierung mittels QR Codes

Um gewissen Mobile App Benutzer spezielle Berechtigungen zu geben, kann ein QR Code für eine Rolle generiert werden. Dieser Code kann im App einmalig eingelesen werden und ermöglicht nun dem Benutzer des App spezielle Berechtigungen.

Folgend sind die zwei Rollen genauer beschrieben:

4.4.1 Jury

Die »Jury« Rolle ermöglicht den Mobile Benutzern ihre Votes auf dem Speicher mit dem Jury Flag zu speichern.

4.4.2 Author

Die »Author« Rolle ermöglicht den Mobile Benutzern ihre Social Items ohne vorherige Überprüfung zu veröffentlichen.

4.5 Moderation von Social Items

Um Vandalismus zu verhindern gibt es ein Moderationssystem für die Social Items. Ein durch einen Mobile App Benutzer neu erstelltes Social Item wird mit dem Zustand »pending« erstellt. Dieses kann nun akzeptiert werden (»accepted« - sichtbar für alle Benutzer) oder abgelehnt werden (»rejected«).

4.5.1 Publishing

Um Leute auf der ganzen Welt am Event teilhaben zu lassen, gibt es die Möglichkeit Social Items im Zustand »accepted« auf Facebook sowie Twitter zu publizieren. Dabei kann nur der Text oder auch das hochgeladene Bild (falls vorhanden) publiziert werden. Sobald dies passiert ist, ändert der Zustand sich auf »published« und enthält nun Links zu den publizierten Beiträgen.

Facebook Um ein Social Item auf Facebook zu publizieren muss ein Facebook User im Browser angemeldet sein, falls dies nicht der Fall ist erscheint ein »Connect with Facebook« Button. Der angemeldete Benutzer muss zusätzlich das dazugehörige App den Zugriff gewähren.

Twitter Um ein Social Item auf Twitter zu publizieren gibt nebst der 140 Zeichen Begrenzung [3] keine Voraussetzungen, diese können also ohne vorherigen Login publiziert werden.

[3] Twitter: Character Counting, `https://dev.twitter.com/overview/api/counting-characters` (Stand: 10.05.15)

4.6 Allgemeiner Workflow

1. Erstellen eines Events
2. Erstellen der Speaker, falls noch nicht vorhanden
3. Erstellen der Event Rooms
4. Erstellen der Event Items
5. Freischaltung des Events vor dem Event beginn

Bei Bedarf:

1. Erstellen der QR Codes
2. Erstellen der Votings
3. Moderation der Social Items
4. Erstellen von News / Push Meldungen

5 Fussnoten Index

Hacking-Lab Mobile Event App | Terminologie

Oussama Zgheb & Tobias Zahner

11. Juni 2015

Datum	Version	Änderung	Autor
09.03.15	1.0	Intial	Oussama Zgheb
20.03.15	1.1	Erweiterung Terminologie	Oussama Zgheb
03.06.15	1.2	Komponenten	Oussama Zgheb
08.06.15	1.3	Ergänzungen	Tobias Zahner

1 Einleitung

Dieses Dokument schafft die Grundlage einer Programmweiten Definition der Bezeichnungen schaffen.

Dadurch sollen Verwirrungen und Unstimmigkeiten in dem Front- sowie Backend nicht entstehen.

2 Terminologie

2.1 Komponenten

2.1.1 Mobile App

Das »Mobile App« (auch »Mobile Client«) bezeichnet den Teil der Software die auf dem Android Gerät des Benutzers läuft. Dieses App trägt die Bezeichnung »ConFoxy«.

2.1.2 Hacking-Lab Server

Der »Hacking-Lab Server« ist der bestehende Server [1] und dessen Rest API.

2.1.3 HLM-NG Server

Der »HLM-NG Server« ist der Server auf welchem das Front sowie Backend gehostet werden.

2.1.4 Frontend

Das Frontend ist eine Webanwendung welche durch das Team bedient wird. In dieser Webanwendung werden alle notwendigen Dinge für einen Event konfiguriert und gesteuert.

2.1.5 REST API

Die Rest API (auch »Backend«) ist die Schnittstelle der Datenbank zum »HLM-NG Server« sowie zu den »Mobile App's«.

Calls Ein Call ist ein Aufruf einer Ressource der REST API.

2.2 Event

Ein »Event« bezeichnet eine Veranstaltung/Konferenz. Diese Veranstaltung kann mehrere Tage lang sein. Die Veranstaltung ist an einem Ort mit mehreren Räumen welche sich während der ganzen Veranstaltung nicht ändern.

2.2.1 Event Item

Ein »Event Item« bezeichnet ein Ereignis in einer Veranstaltung. Also z.B. eine Präsentation, ein Talk oder die Mittagspause. Ein Ereignis gehört zu einem Raum und hat einen Start- sowie Endzeitpunkt.

2.3 Speaker

Ein »Speaker« ist eine Person welche ein Ereignis durchführt. Also z.B. der Referent einer Präsentation.

[1] https://www.hacking-lab.com/

2.4 Whats Up

2.4.1 News

»News« sind Nachrichten welche im Backend erfasst wurden um die Veranstaltungsteilnehmer zu informieren. Also z.B. der Vortrag XYZ findet in Raum 123 anstatt 321 statt.

2.4.2 Social

»Social« sind Beiträge von Konferenzteilnehmern. Diese Beiträge werden zuerst durch das Moderationsteam im Backend freigeschaltet.

2.4.3 Share

Unter »Share« wird das hinzufügen von Beiträgen für »Social« verstanden.

2.5 Push

Ein »Push« ist eine kurze Meldungen an die Konferenzteilnehmer. Diese kann für dringende »News« verwendet werden.

2.6 Voting

Ein »Voting« ist eine Abstimmung über eine Präsentation einer »Challenge«. Die Abstimmung wird im Backend erfasst und per App durchgeführt.

2.6.1 Vote

Ein »Vote« ist eine Abstimmung einer Person für ein »Voting«. Also z.B. Max Muster hat bei dem »Voting« XYZ eine Bewertung von 8/6/9 abgeben.

2.6.2 Scoring

Das »Scoring« zeigt die aktuelle Rangliste der Teams an und wie die totale Punktezahl zustande gekommen ist.

2.6.3 Slider

Ein »Slider« ist eins von mehreren Kriterien die bei einem »Voting« bewertet werden.

2.7 Rollen

2.7.1 User

Ein »User« oder auch »Benutzer«, bezeichnet einen Konferenzteilnehmer der das »Mobile App« verwendet.

2.7.2 Moderator

Ein »Moderator« arbeitet im Backend und moderiert die »News« sowie den »Social« Kanal auf dem »HLM Server«

2.7.3 Operator

Ein »Operator« erfasst und steuert »Events« sowie »Votings« auf dem »HLM Server«

2.7.4 Author

Ein »Author« kann ohne vorherige Überprüfung Beiträge in den »Social« Kanal einfügen.