HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

RESTful Mobile Peer to Peer Social Network

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2010

Autor(en): Martin Boos
Betreuer: Prof. Hansjörg Huser

Inhalt

1 Aufgabenstellung

1.1.1 Ausgangslage, Problembeschreibung

Peer to Peer Systeme sind eine gängige Alternative zu teureren Cluster oder Cloud Lösungen. Viele Beispiele aus der Praxis, zum Beispiel Bionic ("Open-Source Software für Volunteer Computing und Grid Computing")**, verwenden komplexe Protokolle, mit deren Hilfe Aufgaben verteilt, Ergebnisse gesammelt und konsolidiert werden können, neben vielen zusätzlichen Funktionen wie Discovery, Rekonfiguration und weiteren Systemfunktionen.**

Alle diese Systeme basieren auf low level Implementierungen dieser Protokolle. Dies bedingt, dass in der Regel ein spezieller Client heruntergeladen und installiert werden muss bevor ein Rechner in diese (P2P) Clouds integriert werden und Aufgaben gelöst werden können.

Auf der andern Seite lassen sich verteilte Systeme elegant auf REST (und RESTful Webservices) aufbauen. REST vereinfacht die Kommunikation und Serviceintegration auf einer höheren Ebene, der Applikationsebene (http Protokoll). Alle Daten werden extern mithilfe von XML dargestellt, sind also plattformneutral.

1.1.2 Technologische Rahmenbedingungen

Im Gegensatz zu den gängigen P2P Netzwerken soll das System auf dem REST Prinzip aufgebaut werden. Dieser neuartige Ansatz ermöglicht die leichte Einbindung neuer Knoten, da die Kommunikation in REST auf http basiert.

Als externe Datendarstellung soll XML verwendet werden. Damit lässt sich die Datendarstellung plattformneutral realisieren. Sicherheitsaspekte sollen fürs erste nicht berücksichtigt werden (keine Verschlüsselung der übermittelten Daten). Dies kann zu einemspäteren Zeitpunkt leicht nachgeholt werden.

Die Lösung soll gängige mobile Plattformen unterstützen, konkret Android und Mobile 7 von Microsoft. iPhone ist nicht zentral, da diese Welt sehr geschlossen und proprietär ist.

1.2 Ziele der Arbeit

1.2.1 Hauptziele:

- Erstellen einer mobilen Anwendung, welche die zwei Kommunikationsmuster Point to Point ("private Verbindung", klassische Telefonverbindung) und Publish Subscribe ("Gruppenverbindung") unterstützt.
  Im Publish Subscribe Teil muss es möglich sein, sich für verschiedene Topics einzutragen und bei allfälligen Updates automatisch informiert zu werden ("anklopfen", oder "SMS an Subsctiber"). Beispielanwendung: Events, lokal gefiltert (nur Events die in einem bestimmten Umkreis von der lokalen Position stattfinden) bzw. Hinweis auf weitere verwandte Events (auch in einem grösseren Umkreis).

- Verwalten von Themenbereichen: damit sollen sogenannte Semantic Overlay Networks (SON) den Peers zur Verfügung gestellt. Ein Beispiel wäre eine verteilte Abfrage zu einem Topic, zu dem nicht an allen Knoten relevante Informationen vorhanden sind. In diesem Fall wäre eine Abfrage aller Peers viel zu zeitaufwendig. Ein Beispiel wäre die Suche nach Musik-Events eines bestimmten Genres. Nicht alle Teilnehmer interessieren sich für alle Genres. Die

Suche nach dem interessanten Events kann sich somit auf jene Themen beschränken, die zum ausgewählten Genre gehören.

- Neben applikatorischen Funktionen (suchen, beauftragen, …) müssen weitere systemnahe Funktionen beispielsweise für die Administration des Netzwerkes bzw. Abfrage des Status der Peers und ähnliches definiert und mindestens exemplarisch implementiert werden.

- Mithilfe von REST soll ein eigenes Protokoll definiert werden. Dieses soll erweiterbar sein, so dass spätere Zusatzanforderungen in das Protokoll integriert werden können.

1.3 Zur Durchführung

Die erfolgreiche Bearbeitung dieser Aufgabe bedingt Grundkenntnisse in REST (praktische Beispiele, nicht bloss Literaturkenntnisse) sowie die Bereitschaft sich in neue und neuste Ergebnisse einzuarbeiten (MobileAnwendungen, speziell ganz neue Möglichkeiten mit Mobile 7 und Android).

Mit dem Betreuer finden in der Regel wöchentliche Besprechungen statt. Zusätzliche Besprechungen sind nach Bedarf durch die Studierenden zu veranlassen.

Alle Besprechungen sind von den Studenten mit einer Traktandenliste vorzubereiten und die Ergebnisse sind in einem Protokoll zu dokumentieren, das dem Betreuer per E-Mail zugestellt wird.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. An Meilensteinen gemäss Projektplan sind einzelne Arbeitsresultate in vorläufigen Versionen abzugeben. Über die abgegebenen Arbeitsresultate erhalten die Studierenden ein vorläufiges Feedback. Eine definitive Beurteilung erfolgt auf Grund der am Abgabetermin abgelieferten Implementierung und Dokumentation.

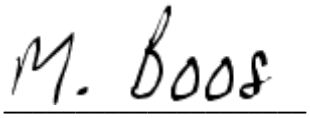
2 Erklärung über die Eigenständige Arbeit

Erklärung

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.

Rapperswil, 22. Dezember 2010



Martin Boos

3 Kurzfassung der Studienarbeit

Abteilung	**Informatik**
Name[n] der Studierenden	**Martin Boos**
Studienjahr	**FS 2010/HS 2011**
Titel der Studienarbeit	**RESTful Mobile Peer to Peer Social Network**
Examinatorin / Examinator	**Prof. Hansjörg Huser**
Themengebiet	**Software**
Projektpartner	**-**
Institut	**-**

Aufgabenstellung	Das Ziel der Studienarbeit war, eine Machbarkeitsstudie einer RESTful Peer to Peer Applikation für Mobilgeräte durchzuführen. Als Zielplattform wurde Android verwendet, wodurch es möglich war, sowohl die Serverapplikation als auch die Mobileclientapplikation in Java zu entwickeln.
Mobiles Peer to Peer	Da nicht klar war, ob Mobilgeräte auf deren Kommunikationsinfrastruktur überhaupt Peer to Peer fähig sind, musste dafür ein Prototyp erstellt werden.
Infrastruktur	Damit ein Mobileclient andere Clients finden kann, ist ein zentraler Treffpunkt nötig. Dies ist hier ein Apache Webserver mit Tomcat als Servletcontainer. Darin läuft eine Applikation basierend auf dem Jersey Framework, womit RESTful Webservices entwickelt werden können.
RESTful	Die ganze Client/Server- und Client/Client-Kommunikation erfolgt RESTful. Alle Daten werden extern mithilfe von XML dargestellt, sind also plattformneutral.
Verwendungszweck	Peer to Peer Systeme sind eine gängige Alternative zu teureren Cluster oder Cloud Lösungen.
Was wurde erreicht?	In der Arbeit konnte gezeigt werden, dass sich Mobileclients auch in einer Peer to Peer Umgebung verwenden lassen.

4 Poster

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
INFORMATIK

RESTful Mobile Peer to Peer Social Network

Martin Boos

Betreuer: Prof. Hansjörg Huser

Studienarbeit Herbstsemester 2010/2011

Themengebiet: Mobile Peer to Peer

Mobile P2P

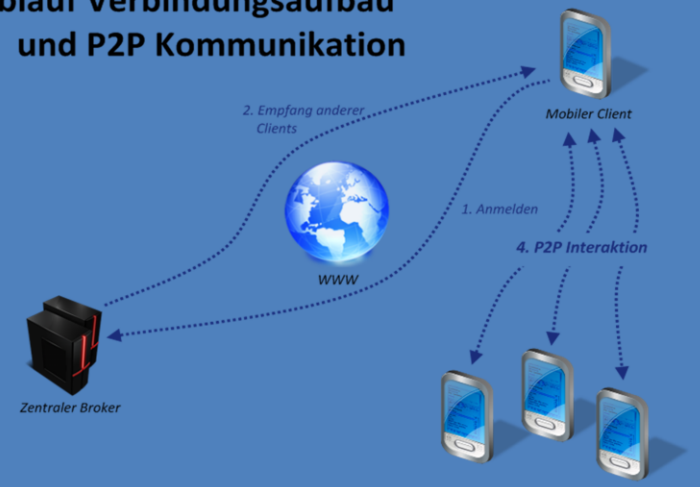
Architektur Clientseite

Variante 1	*Variante 2*
Jersey Clientapplikation	
i-jetty Webserver	Apache HTTP Client
Dalvik Virtual Machine (DVM)	
Hardware (Mobilgerät mit Android 2.2)	

Architektur Serverseite

Jersey Serverapplikation
Apache Tomcat Server
Java Virtual Machine
Server

Ablauf Verbindungsaufbau und P2P Kommunikation



Kommunikation

- RESTful
  - HTTP
  - XML

Entwicklung

- Server: Java WebService
- Android: Java

Projekt: RESTful Mobile Peer-to-Peer Social Network

Projektplan

Dieses Dokument beinhaltet Informationen betreffend der Projektplanung, der gewählten Meilensteine sowie eine Beschreibung der Arbeitspakete.

Version: 1.0 Datum: 22-12-2010

Dokumentinformationen

Änderungsgeschichte

Datum	*Version*	*Änderung*	*Autor*
03.10.2010	0.1	Erste Fassung	mb
07.10.2010	0.2	Risikomanagement	mb
11.11.2010	0.3	Diverses ergänzt	mb
22.12.2010	0.9	Dokument abgeschlossen	mb

5 Einführung (Introduction)

5.1 Zweck (Purpose)

Dieses Dokument gibt einen Überblick über die Planung und Entwicklung meiner Studienarbeit. Es soll die Grundlage für weitere Projektschritte sein.

5.2 Gültigkeitsbereich (Scope)

Dieses Dokument bezieht sich auf die Studienarbeit, welche ich im Herbstsemester 2010 an der Hochschule für Technik in Rapperswil (HSR) durchführe. Dieses Dokument wird fortlaufend erweitert und erneuert, da dieses Projekt anhand Iterativer Entwicklungsmethoden erstellt wird.

Die Angaben in diesem Dokument sind als gültig zu betrachten, da es fortlaufend dem Projektstand angepasst wird. Änderungen bleiben jedoch vorbehalten und können jeder Zeit angepasst werden.

5.3 Referenzen (References)

- /Risikomanagement/Uebersicht.xls

5.4 Übersicht (Overview)

- **Projekt Übersicht:** Behandelt die grundsätzlichen Ideen, Annahmen und Ziele des Projekts. Das beinhaltet Zweck und Ziel des fertigen Programms sowie weitere Rahmenbedingungen
- **Projektorganisation:** Informationen über die die beteiligen Personen und externe Schnittstellen.
- **Management Abläufe:** Zeigt Zeitplanung mit Iterationen und Meilensteinen in tabellarischer Form
- **Risiko Management:** Listet in tabellarischer Form die Risiken sowie deren Auswirkungen auf zu finden unter: /Risikomanagment/Risikomanagement.docx
- **Arbeitspakete:** Die zu erledigenden Arbeiten werden in Arbeitspakete aufgeteilt und priorisiert. Die Termine sind verbindlich und sollen nicht überzogen werden.
- **Infrastruktur:** Beschreibt die eingesetzten Technologien, Software und Geräte.
- **Qualitätsmassnahmen:** Massnahmen um im Verlauf des Projekts eine durchgehend hohe Qualität sicherzustellen.

6 Projekt Übersicht (Project Overview)

Ziel der Studienarbeit ist das Erstellen eines Mobile Peer to Peer Social Networks im REpresentational State Transfer (REST)[1] Architekturstil.

6.1 Zweck (Purpose)

Peer to Peer Systeme sind eine gängige Alternative zu teureren Cluster oder Cloud Lösungen. Viele Beispiele aus der Praxis, zum Beispiel Bionic ("Open-Source Software für Volunteer Computing und Grid Computing"), verwenden komplexe Protokolle, mit deren Hilfe Aufgaben verteilt, Ergebnisse gesammelt und konsolidiert werden können, neben vielen zusätzlichen Funktionen wie Discovery, Rekonfiguration und weiteren Systemfunktionen.

Im Gegensatz zu den gängigen P2P Netzwerken soll das System auf dem REST Prinzip aufgebaut werden. Dieser neuartige Ansatz ermöglicht die leichte Einbindung neuer Knoten, da die Kommunikation in REST auf HTTP basiert.

6.2 Ziele (Objectives)

- Erstellen einer mobilen Anwendung, welche die zwei Kommunikationsmuster Point to Point ("private Verbindung", klassische Telefonverbindung) und Publish Subscribe ("Gruppenverbindung") unterstützt. Im Publish Subscribe Teil muss es möglich sein, sich für verschiedene Topics einzutragen und bei allfälligen Updates automatisch informiert zu werden ("anklopfen", oder "SMS an Subscriber"). Beispielanwendung: Events, lokal gefiltert (nur Events die in einem bestimmten Umkreis von der lokalen Position stattfinden) bzw. Hinweis auf weitere verwandte Events (auch in einem grösseren Umkreis).

- Verwalten von Themenbereichen: damit sollen sogenannte Semantic Overlay Networks (SON) den Peers zur Verfügung gestellt. Ein Beispiel wäre eine verteilte Abfrage zu einem Topic, zu dem nicht an allen Knoten relevante Informationen vorhanden sind. In diesem Fall wäre eine Abfrage aller Peers viel zu zeitaufwendig. Ein Beispiel wäre die Suche nach Musik-Events eines bestimmten Genres. Nicht alle Teilnehmer interessieren sich für alle Genres. Die Suche nach dem interessanten Events kann sich somit auf jene Themen beschränken, die zum ausgewählten Genre gehören.

- Neben applikatorischen Funktionen (suchen, beauftragen, …) müssen weitere systemnahe Funktionen beispielsweise für die Administration des Netzwerkes bzw. Abfrage des Status der Peers und ähnliches definiert und mindestens exemplarisch implementiert werden.

- Mithilfe von REST soll ein eigenes Protokoll definiert werden. Dieses soll erweiterbar sein, so dass spätere Zusatzanforderungen in das Protokoll integriert werden können.

6.3 Annahmen und Einschränkungen (Assumptions and Constraints)

Die Lösung soll gängige mobile Plattformen unterstützen, konkret Android und Mobile 7 von Microsoft. iPhone ist nicht zentral, da diese Welt sehr geschlossen und proprietär ist.

[1] http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm, 04.10.2010

7 Projektorganisation (Project Organization)

Diese Studienarbeit wird als Einzelarbeit von Martin Boos durchgeführt.

7.1 Externe Schnittstellen (external Interfaces)

Ansprechpartner und Betreuer dieser Arbeit ist Professor Hansjörg Huser, Studiengangleiter Informatik an der HSR.

8 Management Abläufe (Management Process)

8.1 Projekt Kostenvoranschlag (Project Estimates)

Für das Projekt steht das Herbstsemester 2010 zur Verfügung. Dies entspricht 14 Wochen. Aufgrund ausserordentlicher Umstände fällt bei meiner Arbeit die erste Woche aus. Die Arbeit wird also im Zeitraum von 13 Wochen vom Montag 27.09.2010 bis Donnerstag 23.12.2010 durchgeführt.

8.2 Projektplan (Project plan)

8.2.1 Iterationsplanung / Meilensteine (Iteration Objectives / Milestones)

Inception	Von	Bis
Gesamte Phase	Monatg, 27. September 2010	Freitag, 8. Oktober 2010

Elaboration	Von	Bis
Gesamte Phase	Montag, 11. Oktober 2010	Freitag, 29. Oktober 2010

Construction	Von	Bis
Iteration 1	Montag, 1. November 2010	Freitag, 12. November 2010
Iteration 2	Montag, 15. November 2010	Freitag, 26. November 2010
Iteration 3	Montag, 29. November 2010	Freitag, 10. Dezember 2010
Iteration 4	Montag, 13. Dezember 2010	Mittwoch, 15. Dezember

Transition	Von	Bis
Gesamte Phase	Mittwoch, 15. Dezember 2010	Donnerstag, 23. Dezember 2010

Meilenstein	Datum
Ende Inception Phase	Freitag, 8. Oktober 2010
Abgabe eines primitiven Technologieprototypen	Dienstag, 12. Oktober 2010
Ende Elaboration: Funtkionierender Technologieprototyp	Freitag, 22. Oktober 2010
Ende Iteration 1	Freitag, 12. November 2010
Ende Iteration 2 Cunstruction Phase	Freitag, 26. November 2010
Ende Iteration 3 Construction Phase **Ende Construction Phase**	Freitag, 10. Dezmber 2010 Mittwoch, 15. Dezember 2010

8.2.2 Besprechungen (Meetings)

Wiederholende Termine:

Betreff	Ort	Zeit
Besprechung mit Betreuer	**Nach Vereinbarung**	Wöchentlich, wenn nicht anders vereinbart

8.2.3 Abgabe (Releases)

Release	Version	Funktionalität
Final Beta	0.9	Mobile Peer to Peer ist RESTful implementiert in Form eines Chats.

9 Risiko Management (Rist Management)

9.1 Risiken

Eine tabellarische Zusammenfassung der Risiken befindet sich im Dokument /Risikomanagement/Uebersicht.xls

01 – Technologieprobleme

Beschreibung	Das gewählte Framework behindert die Entwicklung, weil es nicht wie geplant eingesetzt werden kann oder nicht die benötigte Funktionalität bietet.
Eintrittswahrscheinlichkeit	25%
Kosten der Massnahmen	28h
Schaden / Aufwand	150h
Schaden gewichtet	37.5h
Massnahmen zur Vermeidung	Zu Beginn des Projekts einen Prototypen erstellen, der alle zu verwendenden Technologien und Frameworks benutzt und Experten mit Praxiserfahrung zu Rate ziehen.
Massnahmen bei Eintritt	Alternatives Framework verwenden.
Priorität	Hoch

02 – Geringe Praxiserfahrung

Beschreibung	Aufgrund geringer Praxiserfahrung fällt die Aufwandschätzung der Arbeitspakete falsch aus.
Eintrittswahrscheinlichkeit	25%
Kosten der Massnahmen	4h
Schaden / Aufwand	50h
Schaden gewichtet	12.5h
Massnahmen zur Vermeidung	Experten zur Aufwandsschätzung beiziehen und inkrementelle Vorgehensweise mit Meilensteine verwendenn
Massnahmen bei Eintritt	Projektplan erneut überarbeiten.
Priorität	Hoch

03 – Probleme beim Umsetzen des Konzepts einer RESTful Punkt-zu-Punkt Verbindung

Beschreibung	Das erarbeitete Konzept zum Erstellen einer Punkt-zu-Punkt Verbindung mit REST zwischen zwei Nodes kann nicht oder nur sehr erschwert umgesetzt werden, weil es nicht mit einer RESTful Architektur vereinbar ist.
Eintrittswahrscheinlichkeit	15%
Kosten der Massnahmen	5h
Schaden / Aufwand	90h
Schaden gewichtet	13.5h
Massnahmen zur Vermeidung	Konzept bei der Erarbeitung detailliert genug ausarbeiten und mit entsprechender Literatur prüfen. Zusätzlich eine Prototypen entwickeln.
Massnahmen bei Eintritt	Vorhandenes Konzept anpassen oder neues erarbeiten.
Priorität	Hoch

04 – Projektdaten gehen zu einem fortgeschrittenen Zeitpunkt verloren

Beschreibung	Datenverlust in Form eines Festplattenausfalls oder Ausfall des Subversionservers.
Eintrittswahrscheinlichkeit	5%
Kosten der Massnahmen	2h
Schaden / Aufwand	200h

Schaden gewichtet	10h
Massnahmen zur Vermeidung	Regelmässig Sicherheitskopien des Projekts auf einem anderen Medium erstellen.
Massnahmen bei Eintritt	Projekt beim Stand des letzten vorhandenen Materials fortführen.
Priorität	Hoch

05 – Probleme bei Portierung auf Mobilgeräte

Beschreibung	Die Portierung der Applikation auf mobile Gerätet bereitet Probleme, weil beispielsweise benötigte Bibliotheken nicht verfügbar sind.
Eintrittswahrscheinlichkeit	15%
Kosten der Massnahmen	2h
Schaden / Aufwand	90h
Schaden gewichtet	13.5h
Massnahmen zur Vermeidung	Bewährte Frameworks einsetzen und das Projekt in Java entwickeln. Mit einem Emulator laufend prüfen und von Zeit zu Zeit auf dem Zielsystem ausführen.
Massnahmen bei Eintritt	Einschränkungen bei der Funktionalität der mobilen Clients machen.
Priorität	Mittel

06 – Krankheit

Beschreibung	Krankheit bei Projektbeteiligtem führt zu Verzögerung des Projektes
Eintrittswahrscheinlichkeit	2%
Kosten der Massnahmen	0h
Schaden / Aufwand	50h
Schaden gewichtet	1h
Massnahmen zur Vermeidung	Gesund Ernährung
Massnahmen bei Eintritt	Arzt aufsuchen
Priorität	Niedrig

10 Arbeitspakete (Work Package)

Inception

Arbeitspaket	Inhalt / Artefakte	Prio.	Aufw.	Deadline
Use Cases brief + Szenarien	Use Cases + Szenarien für 1. Hauptziel: Point-to-Point Verbindung mit REST	1	3	05.10.2010
Erste Lösungskonzepte	Lösungskonzept für Point-to-Point Verbindung mit REST	1	3	05.10.2010
Ende Inception			**6**	**08.10.2010**

Elaboration

Arbeitspaket	Inhalt / Artefakte	Prio.	Aufw.	Deadline
Projektplan	Projektplan gemäss Vorlage weiterführen (Zeitplanung, Q-Massn. Risikomgt, Arbeitspakete, etc)	1	18	12.10.2010
Anpassung & Korrektur: Projektplan	→ Revidierter Projektplan	2	3	12.10.2010
Konfigurationsverwalt.	SVN Respository konfigurieren → Infrastruktur ist eingerichtet vor Beginn der Construction-Phase	2	1	15.10.2010
Projekt -Automation	Erstellen eines Build Scripts → Infrastruktur ist eingerichtet vor Beginn der Construction-Phase	2	4	15.10.2010
Analyse				
Use Cases fully dressed	Use Cases vollständig (fully dressed) beschreiben	1	6	18.10.2010
Nicht funktionale Anforderungen	Erfassen und beschreiben der nicht funktionalen Anforderungen	1	3	18.10.2010
Design				
Prototyp	Funktionierenden Technologieprototyp erstellen	1	16	29.10.2010
Systemsequenzdiagramme erstellen	SSDs anhand der Use Cases erstellen	1	12	21.10.2010
Komponentendiagramm	Komponentendiagramm der Architektur erstellen	1	12	24.10.2010
Logische Architektur	Design der Softwarearchitektur	1	25	24.10.2010
Ende Elaboration			**99**	**29.10.2010**

Construction

Arbeitspaket	Inhalt / Artefakte	Prio.	Aufw.	Deadline
Broker				
Basisbroker erstellen	Mit Jersey grundlegendste Funktionalitäten des Brokers implementieren	1	10	06.11.2010
Registrationsfunktion	Funktion damit sich Clients anmelden können	1	4	11.11.2010
Mobile Client				
Android GUI Prototyp	Hello World Prototyp für Android GUI entwickeln	1	6	02.11.2010
Webserver auf Android	i-jetty auf Android installieren	1	2	04.11.2010
Jerseyclient auf Android	Jersey auf Android verwenden um auf den Broker zuzugreifen	1	20	15.11.2010
P2P				
HTTP Client auf Android	Die Alternative zum Jerseyclient ist ein HTTP Client, der auf den Broker zugreift.	2	22	25.11.2010
Logisches P2P	Logisches P2P mit Broker als Controller implementieren	2	20	25.11.2010
Reelles P2P	Richtiges P2P für Mobileclients implementieren	1	35	08.12.2010
Resourcen				
JAXB beim Broker	Entity -> XML, XML -> Entity	1	8	05.11.2010
JAXB beim Client	Entity -> XML, XML -> Entity	1	16	10.11.2010
Ende Construction			**133**	**15.12.2010**

Transition

Arbeitspaket	Inhalt / Artefakte	Prio.	Aufw.	Deadline
Benutzerdokumentation	Vervollständigen	1	30	22.12.2010
Abstract / Kurzfassung	Abstract / Kurzfassung schreiben	1	2	19.12.2010
Poster	Poster erstellen	1	2	19.12.2010
Ende Transition			**34**	**23.12.2010**

11 Infrastruktur (Infrastructure)

11.1 Software

Software

- Als Entwicklungsumgebung wird Eclipse 3.5[2] verwendet
- Die verwendete Programmiersprache ist Java in der Enterprise Edition
- Zur Unterstützung von REST wird das Jersey[3] Framework verwendet
- Die Software wird auf einem Apache Tomcat[4] Server in der Version 6 deployed

Hardware

- Das Projekt wird zuerst auf einem Laptop mit Windows XP Professional SP3 im Android Emulator entwickelt
- Später soll es auf Smartphones mit Android Version 2.2 und höher portiert werden

Versionskontrolle

- Zur Versionskontrolle des gesamten Projektes (Entwicklungsdaten und Dokumentation) wird der Subversionserver der HSR verwendet.

[2] http://www.eclipse.org/, 12.10.2010
[3] https://jersey.dev.java.net/, 12.10.2010
[4] http://tomcat.apache.org/, 12.10.2010

12 Qualitätsmassnahmen (Quality Management)

12.1 Dokumentation

Dokumentationen werden nach den vorgebenen Vorlagen erstellt und korrigiert und an den aktuellen Arbeitsstand angepasst. Die komplette Dokumentation und alle anhängenden Dokumente werden auch auf dem Subversionserver gespeichert, um Sicherheitskopien und alte Versionen zu haben.

12.2 Benutzerfreundlichkeit

Da es sich beim Projekt um ein Proof of Concept (Mobile P2P) handelt und deshalb nicht für Endandwender gedacht ist, steht ein benutzerfreundliches GUI nicht an erster Stelle. Dennoch wird Wert darauf gelegt, dass es selbsterklärend und einfach zu bedienen ist.

12.3 Codequalität

Um eine gewisse Codequalität zu erreichen werden von Zeit zu Zeit Refactorings durchgeführt um beispielsweise allfällige *Long Methods* und ähnliches zu beseitigen.

Wo es nötig oder sinnvoll ist, werden Methoden und Klassen entsprechend mit Javadoc kommentiert.

Projekt: RESTful Mobile Peer-to-Peer Social Network

Software Architektur Spezifikation

(Software Architecture Document)

[Dokumentstruktur basiert auf RUP „Software Architecture Document“]

Version: 1.0 Datum: 22-12-2010

Dokumentinformationen

Änderungsgeschichte

Datum	*Version*	*Änderung*	*Autor*
21.10.2010	0.1	Erste Version erstellt	mb
27.10.2010	0.2	Dokument ergänzt	mb
03.10.2010	0.3	Architektur detaillierter beschrieben	mb
10.10.2010	0.4	Änderungen nachgetragen, Dokument aktualisiert	mb
10.11.2010	0.5	Dokument an den aktuellen Stand angepasst	mb
03.12.2010	0.6	Dokument an den aktuellen Stand angepasst (SSD, EventBus)	mb

13 Einführung (Introduction)

13.1 Zweck (Purpose)

In diesem Dokument ist die Architektur der Software dokumentiert.

13.2 Gültigkeitsbereich (Scope)

Dieses Dokument wird während der ganzen Projektdauer gültig sein und ist nur für dies Projekt bestimmt. Im weiteren Projektverlauf werden noch Änderungen und Ergänzungen am Dokument vorgenommen.

13.3 Referenzen (References)

- Systemsequenzdiagramm P2P Chat: *„P2P Chat SSD.docx“*

13.4 Übersicht (Overview)

Im Folgenden wird eine detaillierte Beschreibung der Architektur und Designentscheide des Projekts dokumentiert.

14 Architektonische Darstellung (Architectural Representation)

Es handelt sich um eine Single Client-Application. Das ausführende System benötigt ein Windows XP (oder aktueller) Die aktuelle JVM (Java Version 6) und einen PostgreSQL Server (Version 8.4).

Mobiler Client

<table>
<tr><th>Variante 1</th><th>Variante 2</th></tr>
<tr><td colspan="2">Jersey Clientapplikation</td></tr>
<tr><td>i-jetty Webserver</td><td>Apache HTTP Client

Durch den HTTP Client ist das Mobilgerät in der Lage, auf den eigentlichen Webservice zuzugreifen, welcher auf dem Broker läuft.</td></tr>
<tr><td colspan="2">Dalvik Virtual Machine (DVM)

Auf der DVM wird Bytecode ausgeführt, der durch entsprechendes Kompilieren einer Java Applikation erzeugt werden kann.</td></tr>
<tr><td colspan="2">Hardware (Mobilgerät mit Android 2.2)

Basis der Clientsoftware bzw. –umgebung stellt die Hardware selber dar. In diesem Falle sind alle Mobilgeräte auf denen Android 2.2 oder höher läuft kompatibel.</td></tr>
</table>

Zentrale Serverapplikation (Broker)

<table>
<tr><td>Jersey Serverapplikation</td></tr>
<tr><td>Apache Tomcat Server</td></tr>
<tr><td>Java Virtual Machine</td></tr>
<tr><td>Server

Es kann beliebige Hardware verwendet werden. Die einzige Voraussetzung ist, dass Java darauf lauffähig ist.</td></tr>
</table>

Umgesetzt wird Variante 2, bei der nicht das Jersey Framework auf den mobilen Clients zum Einsatz kommt. Grund dafür ist die schlechte Eignung von Jersey auf Mobilgeräten sowie die teilweise Inkompatibilität die dazu führt, dass das Framework nur teilweise funktioniert und deswegen nicht verwendet werden kann.

Probleme die dadurch entstehen sind, dass Clients nicht über eine Serverfunktion verfügen, was eine Konzeptänderung im Bereich Publish/Subscribe sowieso Peer-to-Peer mit sich bringt. Anstelle von Jerseyfunktionalitäten verfügen die Clients nur über eine Apache HTTP Client, sowie einen minimalen HTTP Listening Server, der nur auf strikt festgelegte Events (zB. die sogenannten Push-Messages von Resourceupdates) reagieren können und werden.

14.1 Architekturpattern

14.1.1 Clientsoftware

14.1.1.1 Model View Controller (MVC)

Bei der Entwicklung des Android Clients wird das Model View Controller (MVC) Pattern angewendet, um Businesslogik und Graphical User Interface (GUI) trennen zu können.

In der Android Entwicklung werden sogenannte Acitivties verwendet, welche die nötige GUI Logik enthalten. Im Gegensatz zu Swing bei einer herkömmlichen Java Applikation, wird bei Android das GUI in einem XML erzeugt. Dies bietet den Vorteil einer genaueren Trennung von GUI-Entwicklung und GUI-Logik.

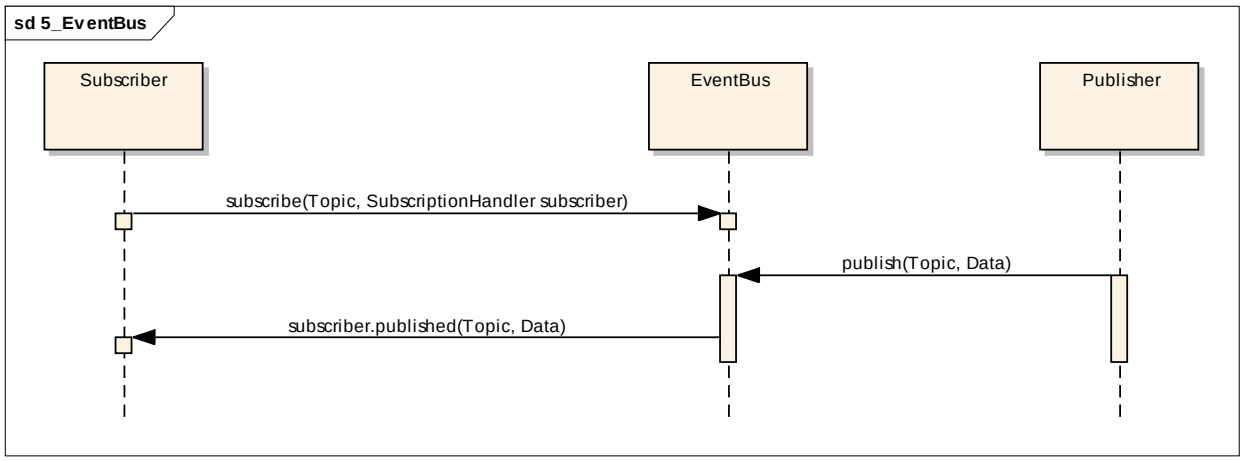
14.1.1.2 EventBus

Zur Aktualisierung des Chatfenster beim Empfang einer neuen Nachricht wurde zuerst das Observer Pattern verwendet. Die Controller-Klassen wurden von den Handler-Klassen benachrichtigt, welche wiederum das GUI informierten.

Damit dabei aber sichergestellt werden konnte, dass die GUI-Klassen nicht direkt mit den Domain-Klassen kommunizieren, musste das Observer Pattern über alle Layer verteilt werden und auftretende Events wurden einfach bis oben durchgereicht. Das führte dazu, dass das Ganze sehr unübersichtlich und kompliziert wurde.

Als Alternative gab es den offiziellen EventBus[5], eine Publish/Subscribe Architektur für Java. Diese war allerdings auf dem Android nicht lauffähig. Deshalb habe ich einen solchen EventBus selber implementiert. Nun gibt es die Möglichkeit, dass sich eine bestimmte Klasse beim EventBus unter einem beliebigen Topic anmeldet. Zusätzlich muss sie einen sogennanten SubscriptionHandler übergeben, eine abstrakte Klasse, deren zu implementierende Methode die Logik enthält, die bei einem empfangenen Ereignis aufgerufen wird.

[5] http://www.eventbus.org/, 02.12.2010



14.1.1.2.1 Beispielcode

```
EventBus.subscribe("Chat", new SubscriptionHandler() {

    @Override
    public void published(String topic, Object data) {
        updateChatWindow(data);
    }
});
```

14.2 Design- und Technologieentscheide

14.2.1 REST Framework

Für Java gibt es verschiedene REST Frameworks, welche dem Entwickler die Arbeit massiv erleichtern. Ich habe mich schlussendlich für Jersey[6] entschieden auf die Empfehlung eines Angestellten bei Doodle[7], wo auch mit REST gearbeitet wird.

14.2.2 Mobileclient

Die Idee auf dem Android einen Webserver und darauf das Jersey Framework zu verwenden, wurde wieder verworfen. Mit i-jetty[8] gibt es zwar einen Webserver worauf Java Servlets funktionieren, doch funktioniert das Jersey Framework nicht auf dem Android. Das Problem dabei ist nun, wie RESTful Peer to Peer (P2P) umgesetzt werden soll, wenn Clients tatsächlich nurnoch über Client- und nicht über Serverfunktionen verfügen.

14.2.2.1 Logisches P2P

Eine Variante zur Umsetzung des P2P Konzepts mit REST wäre, mit logischen P2P Sessions zu arbeiten, die aber tatsächlich über den zentralen Server laufen und von diesem koordiniert und verwaltet werden.

[6] https://jersey.dev.java.net/, 03.11.2010

[7] http://www.doodle.com/, 03.11.2010

[8] http://code.google.com/p/i-jetty/, 03.11.2010

14.2.2.2 Android P2P Framework

Mit PeerDroid[9] gibt es eine JXTA[10] Implementation für Android, welches eventuell spatter für weitere Funktionalitäten der Applikation eingesetzt warden könnte.

14.2.2.3 Reelles P2P

In der finalen Umsetzung wird weder ein logisches P2P, noch ein Android P2P Framework eingesetzt. Stattdessen habe ich einen kleinen HTTP Listener programmiert, der auf Port 8080 hört und eingehende POSTs verarbeiten kann.

Dadurch können HTTP Requests von einem Client direkt an einen anderen gesendet warden, ohne auf einen zentralen Server angewiesen zu sein (obwohl dieser natürlich weiterhin benötigt wird).

[9] http://code.google.com/p/peerdroid/, 03.11.2010

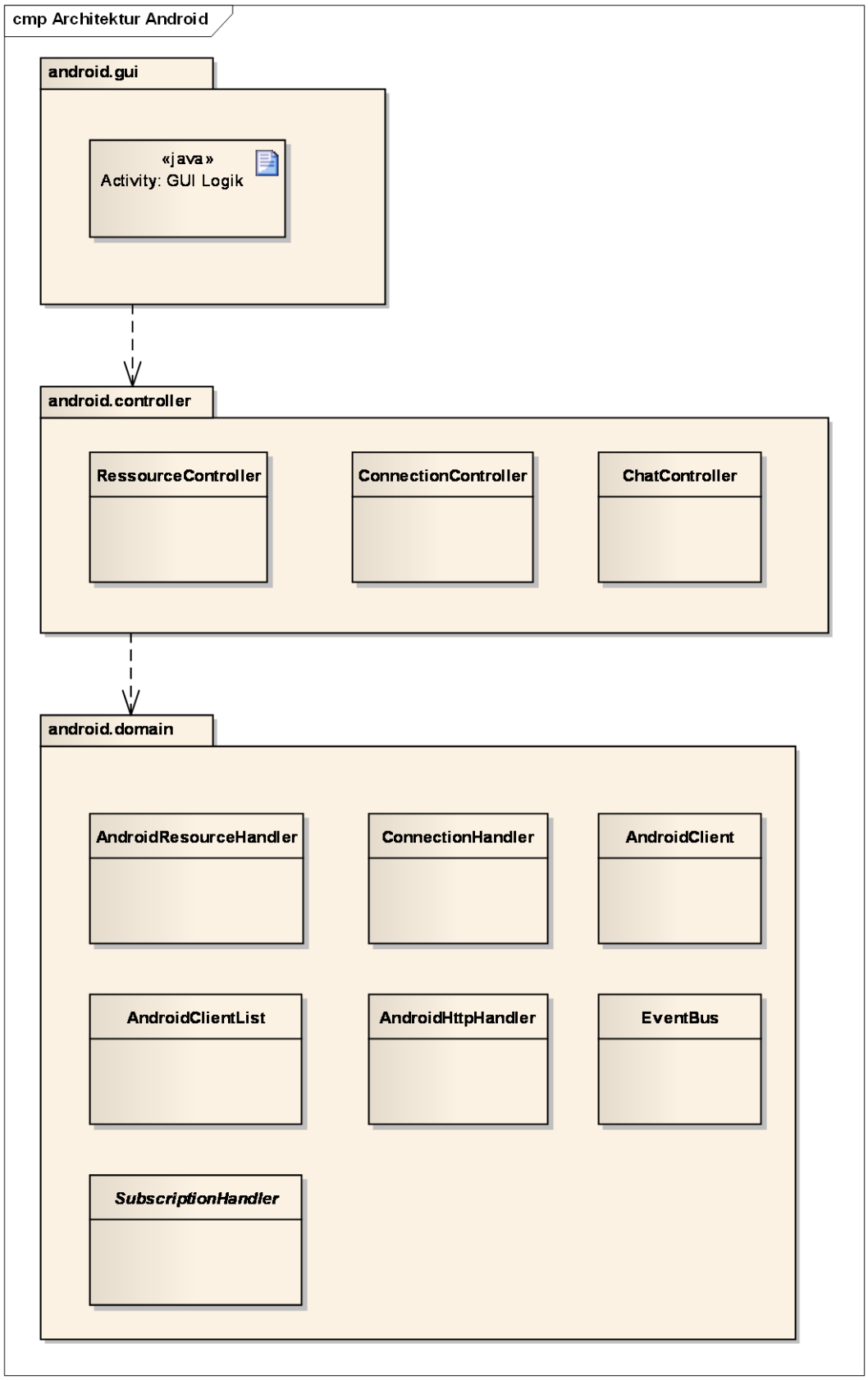
[10] https://jxta.dev.java.net/, 03.11.2010

15 Logische Architektur (Logical View)

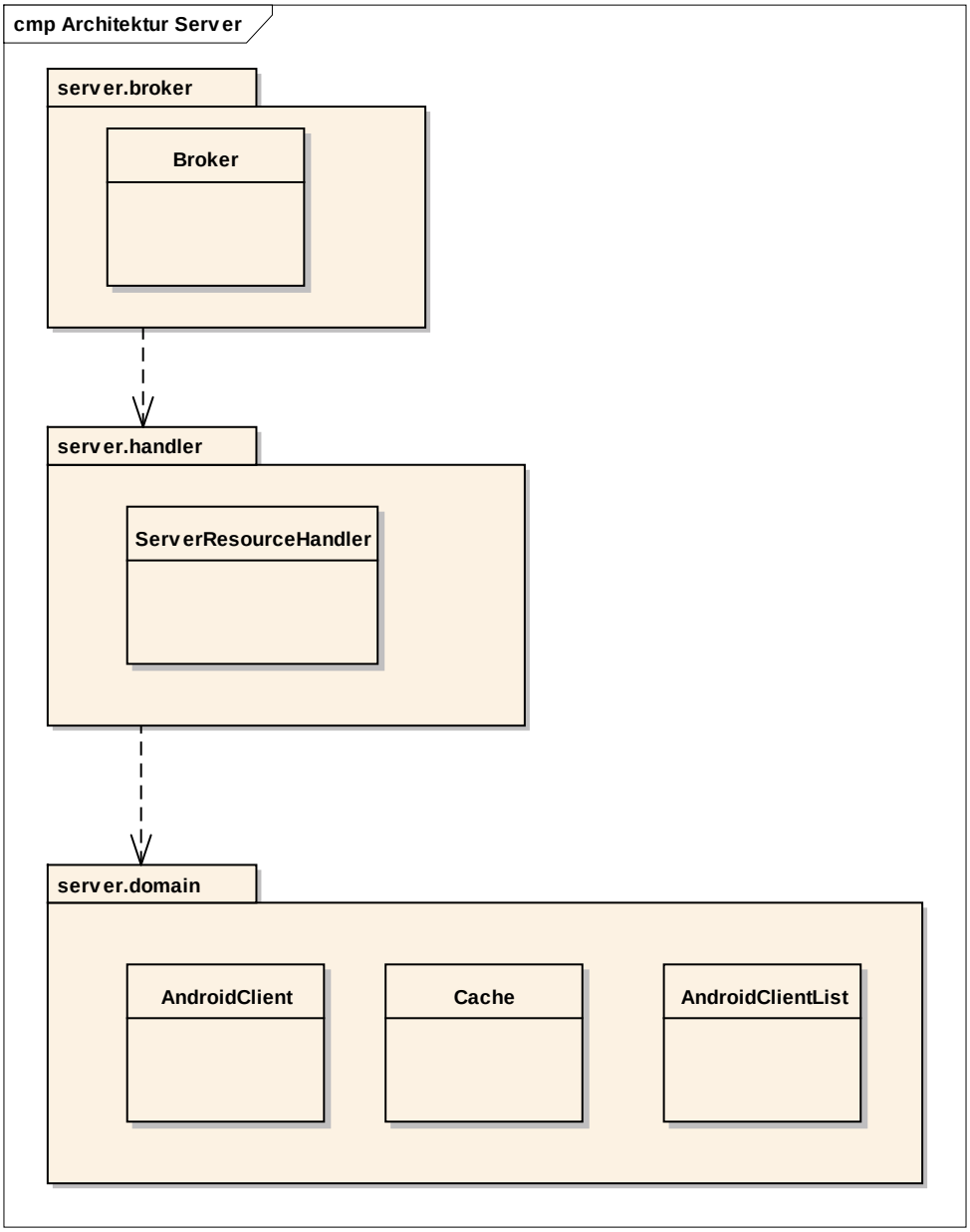
Das Projekt besteht aus zwei Teilen: Der Software für den mobilen Client und der Serveranwendung.

15.1.1 Mobiler Client

Beim Androidclient wird das Model-View-Controller Pattern (MVC-Pattern) angewendet. Dadurch können GUI-Entwicklung, GUI-Handling und Businesslogik sauber getrennt werden.


cmp Architektur Android
android.gui
«java»
Activity: GUI Logik
android.controller
RessourceController
ConnectionController
ChatController
android.domain
AndroidResourceHandler
ConnectionHandler
AndroidClient
AndroidClientList
AndroidHttpHandler
EventBus
SubscriptionHandler

15.1.2 Serveranwendung



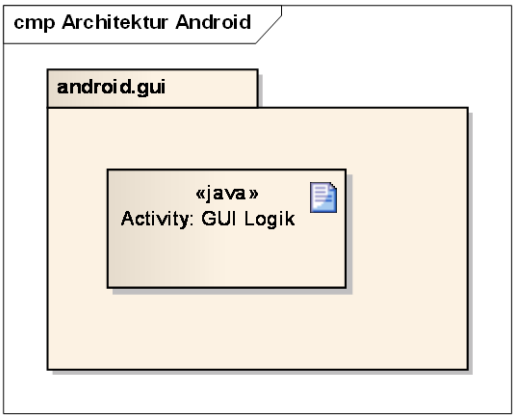
15.2 Design Pakete (Architecturally Significant Design Packages)

15.2.1 Package android.gui

15.2.1.1 Beschreibung des Package

In diesem Package befindet sich die GUI-Logik. Es ist die Schnittstelle zwischen den Controllern und der Benutzeroberfläche an sich. Anders als bei Swing ist die Definition des GUIs nicht auch in dieser Datei gespeichert, sondern in einem eigenen XML-Dokument („AndroidClient/res/layout/main.xml“).

15.2.1.2 Diagramme



Diese Klasse ist eine sogenannte Activity, welche beim Starten der App auf dem Android-Gerät ausgeführt wird. GuiController muss deshalb von der Klasse „android.app.Activity“ erben.

15.2.1.3 Schnittstellen

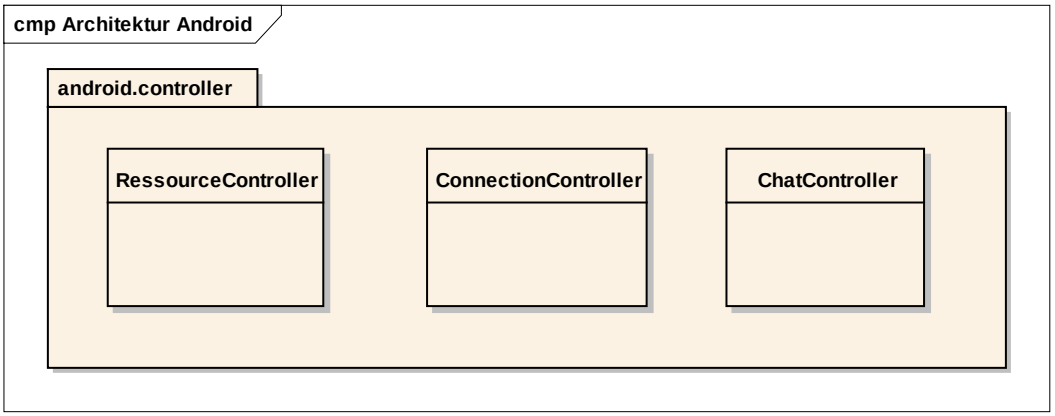
Diese Klasse dient als Schnittstelle zwischen der GUI-Definition im XML-Dokument und den Controllern aus dem Package android.controller, welches später beschrieben wird.

15.2.2 Package android.controller

15.2.2.1 Beschreibung des Package

Dieses Package dient als Schnittstelle zwischen der GUI-Logik (der Activity) und den Domainklassen des Mobileclients. Dadurch kann vermieden werden, in der Activity direkt Instanzen von Domainklassen zu erzeugen. Den Controllern werden die Werte vom GUI übergeben und diese erzeugen dann die benötigten Objekte.

15.2.2.2 Diagramme

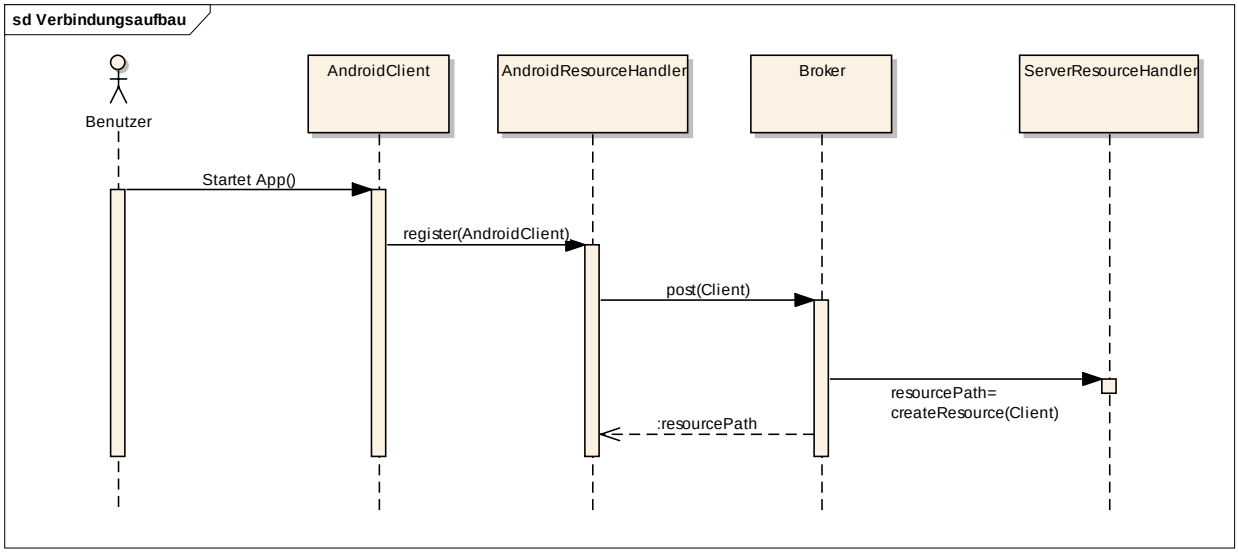


15.2.2.3 Schnittstellen

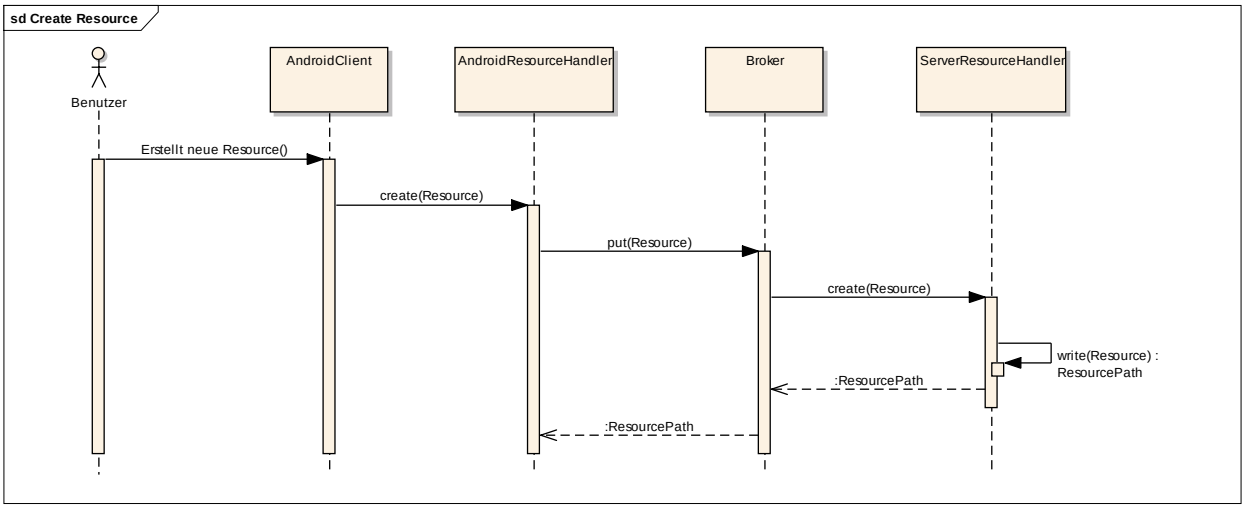
Dadurch dass in der Acitivity Klasse Instanzen dieser beiden Klassen angelegt werden, muss nicht aus dem GUI direkt auf die Domainklassen (zB. zur Erzeugung neuer Resourcen) zugegriffen werden.

15.2.2.4 Operationen

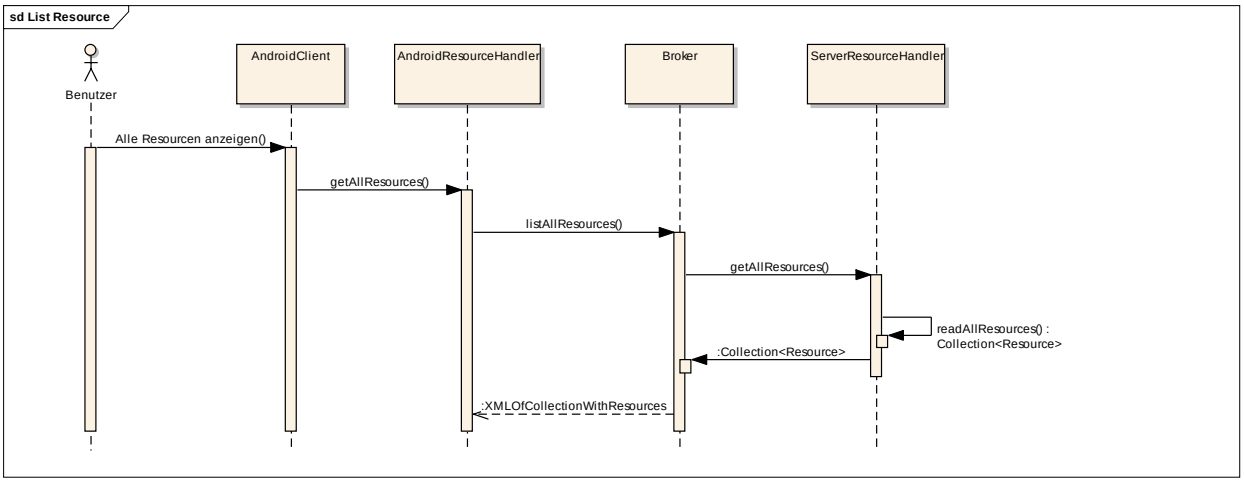
15.2.2.4.1 Verbindungsaufbau und Registration des Clients



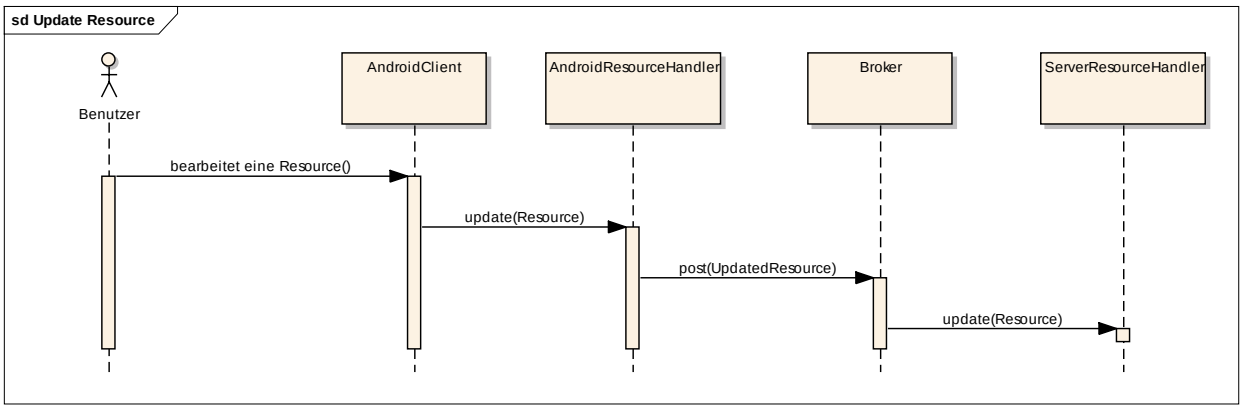
15.2.2.4.2 Erstellen einer neuen Ressource



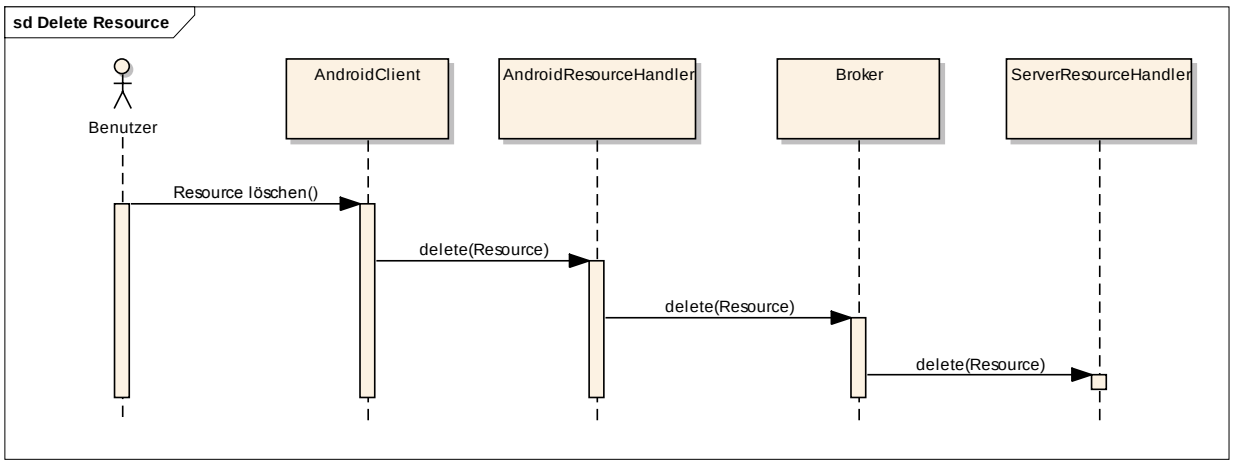
15.2.2.4.3 Anzeigen der vorhandenen Resourcen



15.2.2.4.4 Bearbeiten einer Ressource



15.2.2.4.5 Löschen einer Ressource

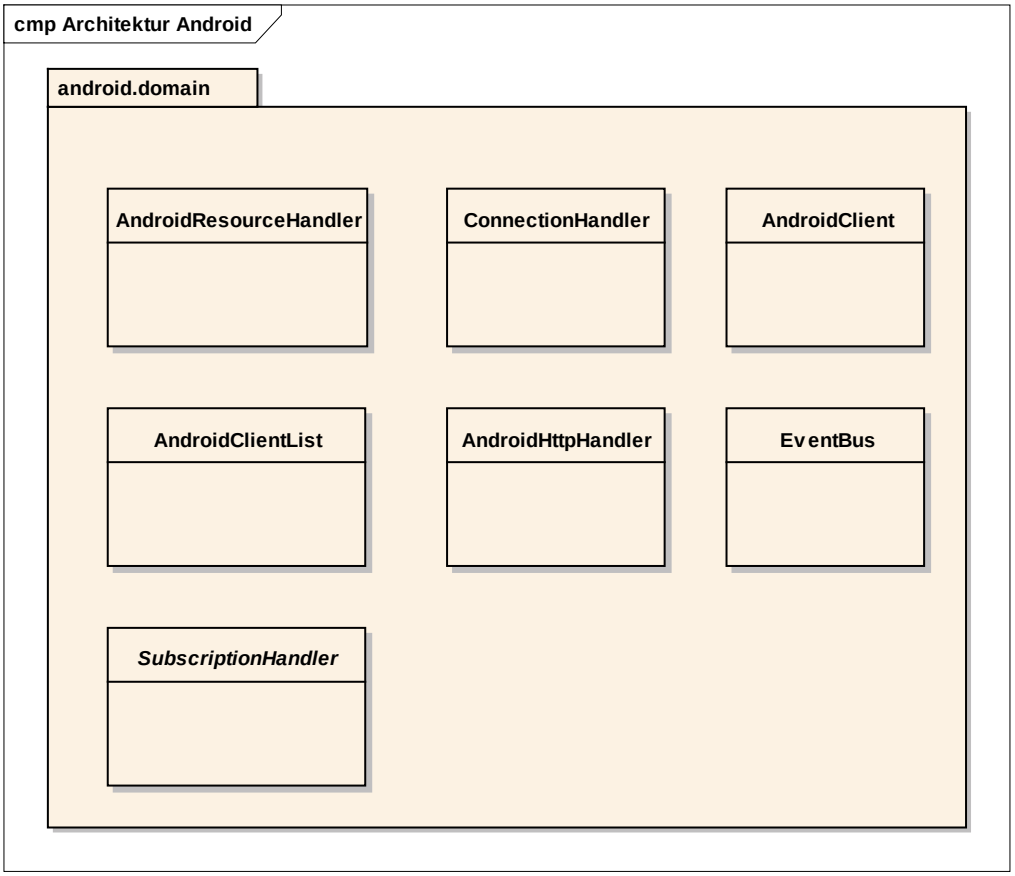


15.2.3 Package android.domain

15.2.3.1 Beschreibung des Package

Dieses Package beinhaltet alle Domainklassen des Mobileclients.

15.2.3.2 Diagramme



15.2.3.3 Schnittstellen

Die Klassen AndroidResourcehandler und ConnectionHandler sind die Schnittstelle zwischen den Domain- und den Controllerklassen.

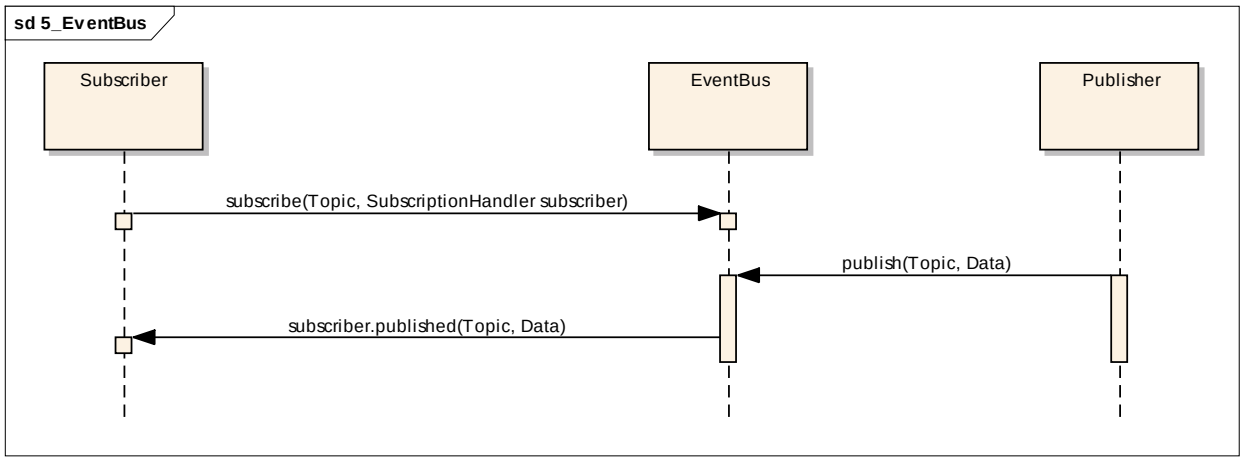
Die P2P Schnittstelle des AndroidClients ist ein HTTP Listener auf Port 8080.

Die Klassen *AndroidClient* und *AndroidClientList* sind identisch mit den gleichnamigen Klassen des Brokers. Nur die Annotationen fürs XML-marshalling und -unmarshalling unterscheiden sich, da nicht dasselbe Framework verwendet werden kann.

15.2.3.4 Operationen

15.2.3.4.1 Publish/Subscribe mit EventBus

Als Alternative zum Observer Pattern wird ein EventBus verwendet.



15.2.3.4.2 P2P Chat

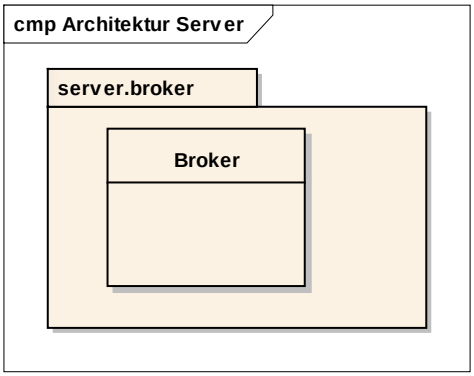
Diagramm befindet sich in seperatem Dokument „P2P Chat SSD.docx“

15.2.4 Package server.broker

15.2.4.1 Beschreibung des Package

Der Broker ist der Kern der Serveranwendung. Alle ankommenden und abgehenden Anfragen und Antworten laufen über ihn.

15.2.4.2 Diagramme



15.2.4.3 Schnittstellen

Die Klasse Broker ist die öffentliche Schnittstelle des Services. Nachfolgend sind die URLs aufgelistet, an die der Client seine Anfragen (GET, POST, PUT und DELETE) schicken kann. Jegliche Kommunikation wird per HTTP gemacht und alle Resourcen als XML übermittelt.

Schnittstelle	Beschreibung	URL	HTTP-Methode
Registration	Ein Client registriert sich immer am Anfang beim Broker. Der Client muss dabei ein Client-Objekt im XML Format posten. Parametername „client“, Daten als XML.	/broker/register	POST

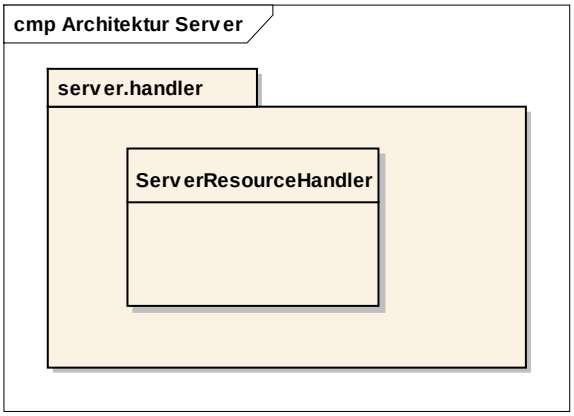
Abmeldung	Client meldet sich wieder ab. Dazu postet er wieder dasselbe Clientobjekt wie bei der Registration	/broker/unregister	POST
Client abfragen	Ein bestimmter Client kann anhand seiner eindeutigen ID abgefragt werden.	/broker/clients/*{uniqueId}*	GET
Alle Clients abfragen	Alle angemeldeten Clients können abgefragt werden.	/broker/clients/all	GET
Resource erstellen	Der Client kann eine neue Resource auf dem Broker erstellen. Der Name wird vom Broker bestimmt (Zwecks Eindeutigkeit), aber dem Client zurückgeliefert. Der Client muss ein XML einer von Resource abgeleiteten Klasse übermitteln	/broker/resources/create/*{type}*	nicht implementiert
Resource löschen	Der Client übermittelt die zu löschende Resource, von der er Ersteller / Besitzer sein muss.	/broker/resources/delete/{type}	nicht implementiert
Resource abfragen	Der Client möchte eine oder mehrere Resourcen abfragen. Er übermittelt die Anfrage im XML	/broker/resources/get/{type}	nicht implementiert
Resource bearbeiten	Der Client übermittelt erneut die bearbeitete Resource um so die bereits vorhandene zu überschreiben	/broker/resources/update/{type}	nicht implementiert

15.2.5 Package server.handler

15.2.5.1 Beschreibung des Package

In diesem Package befinden sich die Handlerklassen, beispielsweise zur Umwandlung “Objekt nach XML” oder “XML nach Objekt” oder zur Verwaltung von Publish/Subscribe Events.

15.2.5.2 Diagramme

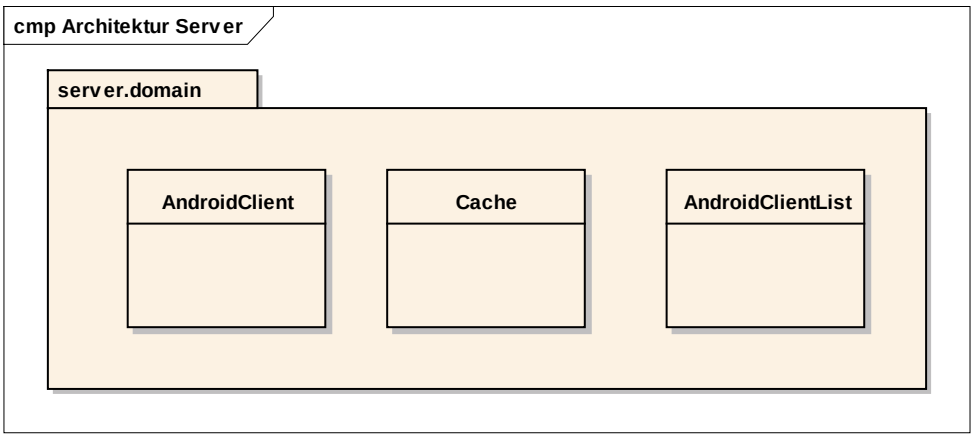


15.2.6 Package server.domain

15.2.6.1 Beschreibung des Package

Hier befinden sich die Domainklassen der Serveranwendung.

15.2.6.2 Diagramme



15.2.6.3 Schnittstellen

Die Klassen *AndroidClient* und *AndroidClientList* sind identisch mit den gleichnamigen Klassen des Mobileclients. Nur die Annotationen fürs XML-marshalling und -unmarshalling unterscheiden sich, da nicht dasselbe Framework verwendet werden kann.

16 Prozesse und Threads (Process View)

16.1 AndroidClient

Auf dem AndroidClient läuft ein HTTP Listener um POSTs zu empfangen. Damit der Client beim Lesen des Streams nicht blockiert, werden alle empfangen POSTs in einem eigenen Thread verarbeitet.

16.2 Broker

Jedesmal wenn sich ein Client an- oder abmeldet informiert der Broker alle anderen Clients darüber. Diese Funktion läuft in einem eigenen Thread um während dieser Zeit nicht den Broker zu blockieren.

17 Datenspeicherung (Data View)

Alle Daten werden zurzeit in XML-Dateien sowohl auf dem Server, als auch auf dem Androidclient direkt auf ein Speichermedium geschrieben und auch davon wieder gelesen.

18 Grössen und Leistung (Size and Performance)

Die Einschränkungen sind durch Apache Tomcat 6 gesetzt und liegen jenseits der benötigten Kapazität.

19 Persönlicher Bericht

19.1 Einleitung

Die ersten Probleme mit der Studienarbeit gab's, bevor sie überhaupt angefangen hatte. Dank der guten Unterstützung der Studienleitung Informatik war es mir schlussendlich doch noch möglich, mit einer Woche Verspätung ein spannendes Thema in einer Einzelarbeit zu bearbeiten.

19.2 Einzelarbeit

Eine Einzelarbeit hat natürlich sowohl Vor- als auch Nachteile. Super war, dass ich völlig unabhängig war und Arbeiten konnte, wo ich gerade wollte und auch wann ich wollte. Dafür wäre ich oftmals froh gewesen, mit einem Teampartner Ideen und Gedanken austauschen zu können, gerade wenn man selber nicht mehr weiter weiss.

19.3 Entwicklung

Während der Studienarbeit habe ich, wie zu erwarten war, sehr viel gelernt. Gerade im Bereich REST und Android, was ja auch meine Hauptthemen waren.

Die Arbeit war sehr spannend und lehrreich. So wie es aussieht, kann ich meine Bachelorarbeit dann auf den Ergebnissen meiner Studienarbeit aufbauen, worauf ich mich sehr freue.

19.4 Betreuung

Betreut wurde ich von Herrn Hansjörg Huser. Zusätzlich hat sich auch Herr Josef Joller grosszügig zur Verfügung gestellt.

Immer wenn nötig haben wir eine Sitzung vereinbart. Das ging problemlos und sehr flexibel per Mail.

19.5 Schattenseiten

Wie vermutlich überall hatte auch meine Studienarbeit ihre Schattenseite. Die ersten Wochen war es für mich überhaupt kein Problem, das Projekt im Androidemulator zu testen. Irgendwann kam ich aber an einem Punkt an, wo ich mit dem Emulator nicht mehr weiter kam. Natürlich habe ich sofort meine Betreuer informiert und nochmals darum gebeten, mindestens zwei Testgeräte mit Android 2.2 zu erhalten. Dies hat sich dann in die Länge gezogen und mein Projekt stand für ungefähr zwei Wochen still, weil erst danach zwei Smartphones bestellt wurden.

20 Anhang

20.1 Sitzungsprotokolle

Sitzungsprotokoll

Projekt: **RESTful Mobile P2P Social Network**
Woche: **39**
Datum: **Mittwoch, den 29. September 2010**

Teammitglieder / Kürzel:
Martin Boos / mb

Traktanden:

- Kickoff Meeting

Diskussion / Beschlüsse:

- Use Cases und Szenarien für 1. Hauptziel erarbeiten
- P2P hat höhere Priorität als Pub/Sub
- Erste Lösungskonzepte für P2P Verbindung mit REST erarbeiten
    - Paper lesen

Sitzungsprotokoll

Projekt: **RESTful Mobile P2P Social Network**
Woche: **40**
Datum: **Dienstag, den 05. Oktober 2010**

Teammitglieder / Kürzel:
Martin Boos / mb

Traktanden:

- Arbeitsstand anschauen

Diskussion / Beschlüsse:

- Risikoanalyse machen (nicht generisch) und mit Projektplan mailen
- Use Cases für Pub/Sub:
  - vervollständigen
- Prototyp mit CRUD: Ressourcen und XML
- Projektplan weitermachen (Arbeitspakete)
- Abgabe nächstes Mal:
  - primitiver Prototyp in REST
  - weitergeführten Projektplan mit Risikomanagement

Sitzungsprotokoll

Projekt: **RESTful Mobile P2P Social Network**
Woche: **41**
Datum: **Dienstag, den 12. Oktober 2010**

Teammitglieder / Kürzel:
Martin Boos / mb

Traktanden:

- Arbeitsstand anschauen

Diskussion / Beschlüsse:

- Nächster Technologieprototyp: i-jetty auf Android
- Erfahrungen dokumentieren
    - zB. Framework: was geht, was nicht, warum, usw.

Sitzungsprotokoll

Projekt: **RESTful Mobile P2P Social Network**
Woche: **42**
Datum: **Freitag, den 22. Oktober 2010**

Teammitglieder / Kürzel:
Martin Boos / mb

Traktanden:

- Arbeitsstand anschauen
- Probleme mit i-jetty auf Android

Diskussion / Beschlüsse:

- Recherche: JMS auf Android, JMS über HTTP
- Apache HTTPClient auf Android -> Prototyp bis nächste Woche
- Callback über HTTP / Callback mit REST -> sonst Polling als Workaround
- SSDs mit wichtigen Operationen (zB. listTopics(), register(), connect())
- Einarbeiten: Android GUI
- Systemkonzepte weitertreiben: SSDs, Prototypen, usw.

Sitzungsprotokoll

Projekt: **RESTful Mobile P2P Social Network**
Woche: **43**
Datum: **Freitag, den 29. Oktober 2010**

Teammitglieder / Kürzel:
Martin Boos / mb

Traktanden:

- Arbeitsstand anschauen

Diskussion / Beschlüsse:

- SVN Accounts für Huser und Joller erstellen und sofort mailen
- Komponentendiagramm fertig
- Architektonische Darstellung:
    - bei jedem Layer noch beschreiben, was darin enthalten ist usw.
    - MVC Pattern auf Android und andere verwendete Pattern
    - Client Server / P2P
- Prototyp für logisches P2P
- Szenarien implementieren
- Server -> Ressourcen cachen

Sitzungsprotokoll

Projekt: **RESTful Mobile P2P Social Network**
Woche: **45**
Datum: **Freitag, den 12. November 2010**

Teammitglieder / Kürzel:
Martin Boos / mb

Traktanden:

- Arbeitsstand anschauen
- Benötigte Handies

Diskussion / Beschlüsse:

- Handies werden bestellt
- SSD für P2P Chat machen

Sitzungsprotokoll

Projekt: **RESTful Mobile P2P Social Network**
Woche: **47**
Datum: **Freitag, den 26. November 2010**

Teammitglieder / Kürzel:
Martin Boos / mb

Traktanden:

- Arbeitsstand anschauen
- Benötigte Handies

Diskussion / Beschlüsse:

- Handies werden bestellt
- SSD für P2P Chat machen

20.2 Risikomanagement

Risiko Kosten Template

Risiko Analyse
Projektname: RESTful Mobile Peer-to-Peer Social Network
Projektmanager: Martin Boos
Datum Kalkulation: Donnstertag 7. Oktober 2010

50

Risiko Bewertungen								
Risk ID	**Risiko**	**Auswirkung**	**Massnahme**	**Kosten der Massnahmen in Stunden**	**Max. Schaden in Stunden**	**Wahrschein-lichkeit des Eintreffens**	**Gewichteter Schaden in Stunden**	**Priorität**
R01	**Framework behindert die Entwicklung, weil sie nicht wie geplant eingesetzt werden kann oder nocht die Benötigte Funktionalität bietet**	**Mehraufwand durch Framework wechsel**	**Recherche und Prototypen**	**28.00**	150	25%	**38**	**Hoch**
R02	**geringe Praxiserfahrung**	**Falsche Aufwandschätzung**	**Experten(Betreuer) zur Aufwandschätzung beziehen**		100	50%	**50**	**Hoch**
R03	**Problem beim umsetzten einer RESTfull Punkt-zu-**	**Vorhandenes Konzept anpassen oder neues**	**Konzept genug genau Ausarbeiten**	**5.00**	90	15%	**14**	**Hoch**

	Punkt Verbindung:	**erarbeiten**	**und mit Literatur prüfen**					
R04	**Projektdaten gehen zu einen vortgeschrittenen Zeitpunkt verloren**	**Projekt bei Stand des letzten vorhanden Materials fortführen**	**Backup**	**2.00**	200	5%	**10**	**Hoch**
R05	**Problem bei der Portierung auf Mobilgeräte**	**Einschränkung des Funktionsumfang bei den mobilien Clients**	**Bewährte Frameworks einsetzten. Entwicklung in Java(Android)**	**2.00**	90	15%	**14**	**Mittel**
R06	**Krankheit bei Projektbeteiligten**	**Weniger Zeitbudget**	**Gesunde Ernährung**	**0.00**	50	2%	**1**	**Niedrig**
	Total Kosten in Arbeitspaketen enthalten			**37**				
	Total Rückstellungen						**126**	

20.3 Zeiterfassung

Zeiterfassung

Projekt:
Teammitglied: **Martin Boos**

Durchschnittliche Soll-Arbeitszeit: **16.0**

Woche	Tag	Datum	Tätigkeit	Zeit	Zeit/Wo	Total	Soll
Woche 0							
					0.0	**0.0**	0.0
Woche 1							
					0.0	**0.0**	16.0
Woche 2	Mittwoch	29-09-2010	Kick-off Meeting	1.0			
	Samstag	02-10-2010	Unterlagen zu REST lesen	5.0			
	Sonntag	03-10-2010	Recherche zum Thema REST und Programmierung auf Android	5.0	**10.0**	**10.0**	32.0

Woche 3	Montag	04-10-2010	Projektplan beginnen	3.0			
	Montag	04-10-2010	Arbeitspakete erstellen	1.5			
	Montag	04-10-2010	Erste Lösungskonzepte für Point-to-Point mit REST erarbeiten	2.0			
	Dienstag	05-10-2010	Sitzung	1.0			
	Dienstag	05-10-2010	Hello World Prototyp erstellen mit Jersey Framework erstellen	4.5			
	Mittwoch	06-10-2010	Erste HelloWorld App-Prototypen auf Android	4.0			
	Donnerstag	07-10-2010	Risikomanagement	5.0			
	Freitag	08-10-2010	UseCases für Publish/Subscribe	6.0			
	Sonntag	10-10-2010	Projektplan weiterführen	5.0	**32.0**	**42.0**	48.0
Woche 4	Montag	11-10-2010	Jersey Prototyp um JAXB XML Handling erweitern	6.0			
	Dienstag	12-10-2010	Sitzung	1.0			
	Mittwoch	13-10-2010	Versuch Jersey auf Android zu bringen inkl. Prototyp	10.0			
	Freitag	15-10-2010	Architektur dokumentieren	3.0			
	Sonntag	17-10-2010	Recherche: Jersey auf Android	6.0			
					26.0	**68.0**	64.0
Woche 5	Montag	18-10-2010	Recherche und Doku: Alternativen für Android	4.5			
	Freitag	22-10-2010	Sitzung	1.0			
	Sonntag	24-10-2010	Systemsequenzdiagramme machen	6.0			
					11.5	**79.5**	80.0

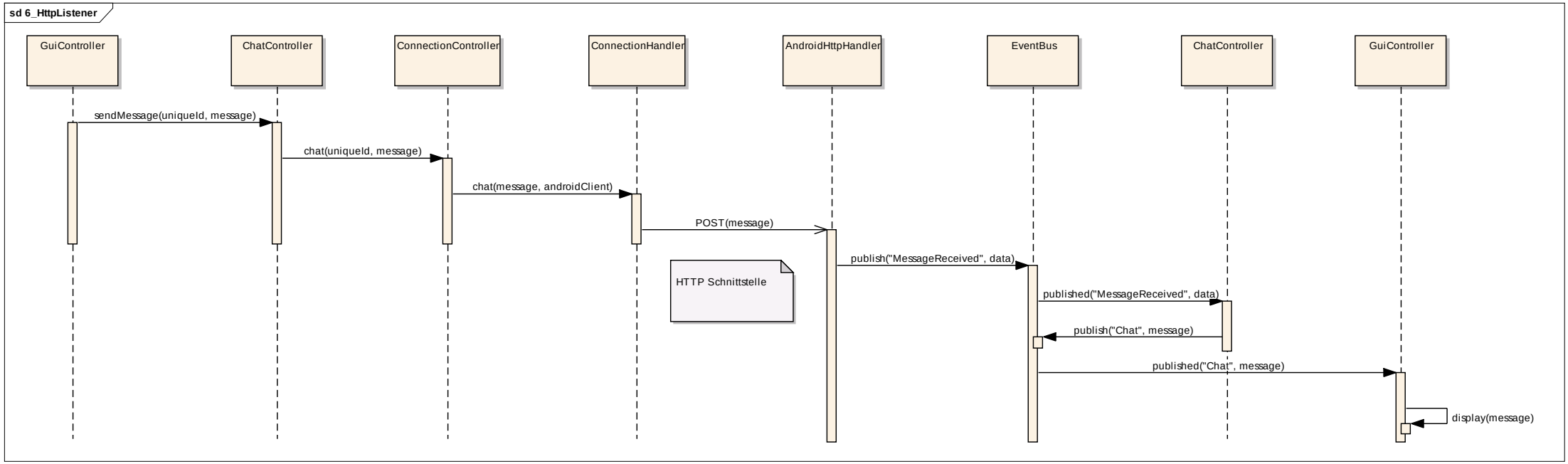
Woche 6	Montag	25-10-2010	SSDs weitermachen	4.0			
	Montag	25-10-2010	Genauer in Entwicklung mit Android einarbeiten	4.0			
	Mittwoch	27-10-2010	Tutorial für Android GUI Entwicklung lesen	2.0			
	Mittwoch	27-10-2010	Android GUI Prototyp erstellen	2.5			
	Donnerstag	28-10-2010	GUI für AndroidClient erstellen	2.0			
	Donnerstag	28-10-2010	Prototyp: Apache HTTP Client auf Android als Jersey Alternative	2.0			
	Freitag	29-10-2010	Sitzung	1.5			
	Samstag	30-10-2010	HTTP Client Prototyp weitermachen	4.0			
	Sonntag	31-10-2010	Komponentendiagramm der Architektur	4.0			
	Sonntag	31-10-2010	Architektur detaillierter spezifizieren	4.0	**30.0**	**109.5**	96.0
Woche 7	Mittwoch	03-11-2010	Architekturdoku weiterführen	2.0			
	Mittwoch	03-11-2010	P2P Prototyp auf Android für Machbarkeit von Mobile P2P	6.0			
	Donnerstag	04-11-2010	Komponentendiagramme aktualisieren	3.0			
					11.0	**120.5**	112.0
Woche 8	Freitag	12-11-2010	Sitzung: Geräte sind nötig zur Weiterentwicklung	1.0			
	Sonntag	14-11-2010	Doku weiterführen	2.0			
					3.0	**123.5**	128.0

Woche 9			Weiterentwicklung steht still aufgrund unzureichender Infrastruktur				
					0.0	**123.5**	144.0
Woche 10	Freitag	26-11-2010	Sitzung: Geräte werden bestellt	1.0			
	Freitag	26-11-2010	Architekturdoku verbessern	2.0			
					3.0	**126.5**	160.0
Woche 11	Dienstag	30-11-2010	Geräte sind angekommen				
	Mittwoch	01-12-2010	Portierung der bestehenden Applikation vom Emulator auf Geräte	10.0			
	Donnerstag	02-12-2010	Android Tutorials lesen für Multithreading	4.0			
	Sonntag	05-12-2010	HTTP Listening Server für Android entwickeln (Jersey Alternative)	8.0			
	Freitag	03-12-2010	Komponentendiagramme anpassen	3.0			
	Sonntag	04-12-2010	SSD für P2P Chat	2.0			
					27.0	**153.5**	176.0
Woche 12	Montag	06-12-2010	Probleme mit Entwicklung: Android Lifecyclemodell -> Problemlösung suchen	4.0			
	Mittwoch	08-12-2010	Ersetzung des ObserverPattern durch eigenen	6.0	**20.5**	**174.0**	192.0

			EventBus			
	Mittwoch	08-12-2010	Dokumentation nachtragen	3.0		
	Donnerstag	09-12-2010	Sitzung	0.5		
	Freitag	10-12-2010	Bugs beheben	3.0		
	Sonntag	12-12-2010	Bugs beheben und besseres Exceptionhandling	4.0		
Woche 13	Montag	13-12-2010	Dokumentation nachtragen	2.0		
	Mittwoch	15-12-2010	Änderungen am MobileClient und Fehlerkorrektur	5.0		
	Mittwoch	15-12-2010	Dokumentation aktualisieren	2.0		
	Mittwoch	15-12-2010	Plakat und Abstract erstellen	2.5		
	Freitag	17-12-2010	Zeiterfassung vervollständigen	2.5		
					14.0	**188.0**	208.0
Woche 14	Dienstag	21-12-2010	Notizen von Sitzungen als Sitzungsprotokolle digital erfasst	3.0		
	Mittwoch	22-12-2010	Abgabedokumente erstellen	10.0		
	Mittwoch	22-12-2010	Projekt abschliessen	4.0		
					17.0	**205.0**	224.0

Total Ist - Arbeitszeit:	**205.0**	
Total Soll - Arbeitszeit:		**224.0**
Differenz:	**-19.0**	

20.4 P2P Chat System Sequenz Diagramm



21 Literaturverzeichnis

Bayer, T. (2002). REST Web Services - Eine Einführung. Mannheim.

Ghelen, G., & Pham, L. (2005). Mobile Web Services for Peer-to-Peer Applications. RWTH Achen University.

Rodriguez, A. (2008). RESTful Web services: The basics.