Fachhochschule Köln
Cologne University of Applied Sciences

Dokumentation

Im Fach WBAII

Phase 2

Prof. Dr. Fischer

Von:

Elif Sahin

Inhaltsverzeichnis

1. Einleitung 2

2. Konzeptioneller Meilenstein / Kommunikationsabläufe und Interaktionen 3

3. Meilenstein 1 (Projektbezogenes XML Schema) 5

4. Meilenstein 2 (Ressourcen und die Semantik der HTTP- Operationen) 9

5. Meilenstein 3 (RESTful Webservice) 11

    5.1 Vorgang 11

6. Meilenstein 4 (Konzeption asynchrone Kommunikation + XMPP - Client) 12

7. Meilenstein 5 + 6 (Client - Entwicklung) 13

8. Fazit 14

9. Quellenverzeichnis 15

1. Einleitung

Im Workshop zu der Veranstaltung Web Basierte Anwendungen II wurde am 15.04.2013 am ersten Tag der 2. Phase eine kurze Einführung in das bevorstehende Projekt gegeben.
Im späteren Verlauf wurden die Studierenden jeweils in ihren Montagsgruppen zusammengeführt. In den Gruppen sollten 2-er Teams gebildet, Ideen ausgetauscht und letztendlich ein Konzept erstellt werden.

Die Phase 2 beinhaltet die Konzeption der synchronen sowie asynchronen Interaktion von Systemkomponenten unter dem Einsatz des Architekturstils REST und des Standards XMPP. Dies erfolgt unter einem Projekt, das in Meilensteine gegliedert wurde und nach der Reihenfolge aus, abgearbeitet wird.

Projektbeschreibung

Das Projekt behandelt ein verteiltes Quiz. Über einen Client werden die Fragen bei den Teilnehmern gleichzeitig angegeben, dies wird durch Publish Subscribe verteilt. Die Anmeldung zum Quiz sowie die Beantwortung der Fragen erfolgen durch REST. Das Quiz ist statisch und besteht aus festgelegten Fragen.
Entweder sind die Fragen vorgefertigt auf einem Server oder die Nutzer können selbst ein Quiz erstellen und in den Fragepool hochladen, nachdem die Überprüfung erfolgreich ist.

Den Spielern werden Fragen in allen Themengebieten, die man sich vorher aussucht, gestellt. Die Fragenkataloge sind in Schwierigkeitsgrade einfach, mittel und schwer eingeteilt ebenso ändert sich auch die Anzahl der Fragen bei „einfach 10, mittel 20, schwer 30“. Das Spiel beginnt erst dann, wenn eine Mindestanzahl erreicht ist(5). Infolgedessen haben die Spieler, die Möglichkeit miteinander zu chatten, während Sie im Warteraum auf weitere Mitspieler warten. Nutzer können auch andere Nutzer zu einem Quiz einladen.
Zu jeder Frage werden 4 Antwortmöglichkeiten angeboten. Die Punkteverteilung ist nach Antwortgeschwindigkeit gestaffelt. Die Schnelligkeit der Antworten ist gefragt und nicht nur die Richtigkeit. Wenn niemand eine Antwort auf die Frage hat, wird vom Veto Recht Gebrauch gemacht und die Frage wird aus dem aktuellen Quiz genommen und ggf. wird der Schwierigkeitsgrad bearbeitet.
Dem User werden nicht von Anfang an die Joker (50:50, Hinweis, Frage beantworten) zur Verfügung gestellt, die werden mit gesammelten Punkten erreicht. Die lokalen Quizbezogenen Punkte werden in globale Punkte umgerechnet.

Einige Informationen zu Daten, die übertragen werden müssen.
Der Nutzer besitzt einen Benutzernamen. Die Mitgliedschaft, die beantworteten Fragen und das letztes gespieltes Quiz werden angezeigt.
Die Informationsänderung bei Fragen z.B. die Zeit, Antworten der Spieler, gewonnene Punkte und die benutzten Joker müssen dargestellt werden.

2. Konzeptioneller Meilenstein / Kommunikationsabläufe und Interaktionen

Am 22.04.2013 sollte ein grobes Konzept entworfen wurden sein. Die Idee sollte man an Hand eines Diagramms graphisch darstellen.
Die Kommunikation zwischen dem Server und Client wurde präzise und konkret, aufbauend auf das Projekt beschrieben. Die synchronen und asynchronen Verbindungen wurden ebenfalls erkundet und angepasst.

> Server
< Client

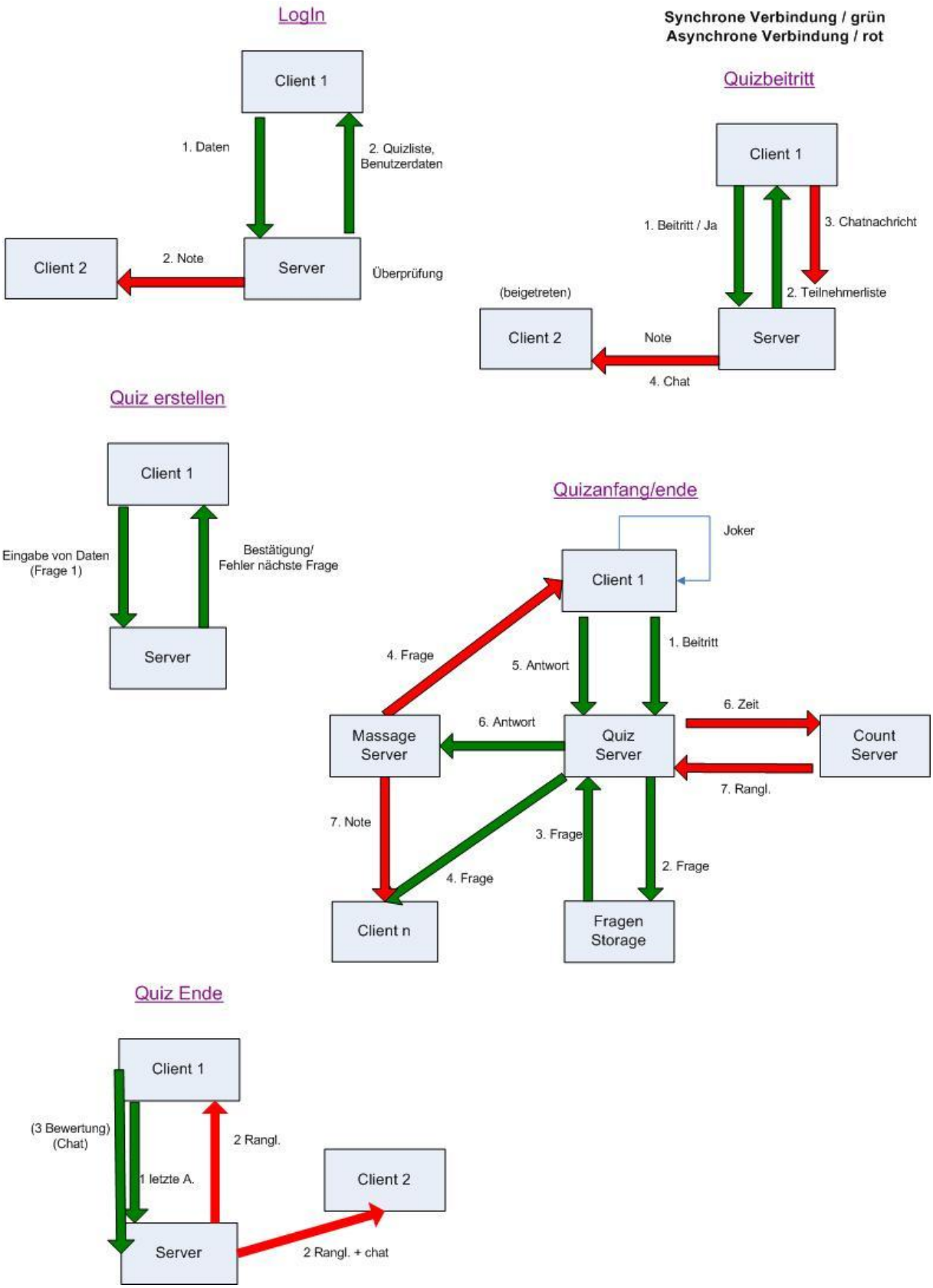
LogIn:
>Benutzername + Passwort
<Liste des Quiz(samt Informationen zum jeweiligen Quiz)
Benutzerinformationen(Benutzernamen, Profilbild, Punkte, Mitglied seit, zuletzt gespieltes Quiz)

Quizbeitritt:
<wartende Teilnehmer(Basisinfo, Benutzername, Profilbild, Punkte), Chat,
>Chatnachrichten

Quizanfang:
<Frage + Antwortmöglichkeiten(richtige Antwort),Joker, Medien zu Frage (Bild, Ton)
>gewählte Antwort(ob richtig oder Falsch), Zeit, evtl. benutzte Joker
<Boolean Info, ob Mitspieler geantwortet haben
<nach Antwort des letzten Mitspielers: Punktebewertung gestaffelt nach Antwortzeit

Quizende:
<Rangliste nach gewonnenen Punkten

Informationen zum Quiz:
Name, Beschreibung, Anzahl evtl. wartende Teilnehmer, Schwierigkeitsgrad, Bewertung, Themengebiet, ggf. Autor, Anzahl der Fragen


LogIn
Client 1
1. Daten
2. Quizliste, Benutzerdaten
Client 2
2. Note
Server
Überprüfung
Synchrone Verbindung / grün
Asynchrone Verbindung / rot
Quizbeitritt
Client 1
1. Beitritt / Ja
3. Chatnachricht
2. Teilnehmerliste
(beigetreten)
Client 2
Note
Server
4. Chat
Quiz erstellen
Client 1
Eingabe von Daten (Frage 1)
Bestätigung/ Fehler nächste Frage
Server
Quizanfang/ende
Joker
Client 1
4. Frage
5. Antwort
1. Beitritt
6. Zeit
Massage Server
6. Antwort
Quiz Server
Count Server
7. Rangl.
7. Note
3. Frage
2. Frage
4. Frage
Client n
Fragen Storage
Quiz Ende
Client 1
(3 Bewertung) (Chat)
2 Rangl.
1 letzte A.
Client 2
Server
2 Rangl. + chat

3. Meilenstein 1

In dieser Aufgabenstellung soll die Planung und Konzeption für das Projekt bestimmter, valider XML-Schemata durchgeführt werden. Später wird es in JAXB generiert.

Die XML-Schemata werden im Einzelnen mit Code und Erläuterung aufgeführt.

- LogIn Client

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">

<element name="userinformation">
  <complexType>
    <sequence>
      <element name="username" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="password" type="string" minOccurs="1" maxOccurs="1"/>
    </sequence>
  </complexType>
</element>
</schema>
```

Erläuterung: In der Anmeldung werden der Benutzername sowie das Passwort abgefragt.

- LogIn Server

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">

<element name="homeview">
  <complexType>
    <sequence>
      <element name="quizlist" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="userinfo" type="string" minOccurs="1" maxOccurs="1"/>
    </sequence>
  </complexType>
</element>
</schema>
```

Erläuterung: Dies ist die Sicht von der Startseite im Server. Die Benutzerinformation werden angezeigt und die Quizliste.

- Question_Client

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">

<element name="answer">
  <complexType>
    <sequence>
      <element name="rightanswer" type="boolean" minOccurs="1" maxOccurs="1"/>
      <element name="usedjokers" minOccurs="0" maxOccurs="3">
       <simpleType>
             <restriction base="string">
                   <enumeration value="50/50"></enumeration>
                   <enumeration value="Hinweis"></enumeration>
                   <enumeration value="Frage beantworten"></enumeration>
             </restriction>
       </simpleType>
      </element>
    </sequence>
  </complexType>
</element>
</schema>
```

Erläuterung: Hier gibt es 2 Elemente rightanswer und usedjokers. Ich stelle 3 Joker zur Verfügung. Einmal den 50/50, Hinweis geben und die Frage beantworten. Der Joker wird nach gesammelten Punkten den Benutzern angeboten und nicht seit Beginn.

- Question_Server

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">

<element name="questions">
  <complexType>
    <sequence>
      <element name="question" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="rightanswer" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="falseanswer" type="string" minOccurs="3" maxOccurs="3"/>
      <element name="picturemedia" type="base64Binary" minOccurs="0"
maxOccurs="1"/>
      <element name="audiomedia" type="string" minOccurs="0" maxOccurs="1"/>
    </sequence>
  </complexType>
</element>
</schema>
```

Erläuterung: Zu den Fragen werden auch Bilder und Audio genutzt wenn es notwendig ist.

- Quiz

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">

<element name="quiz">
  <complexType>
      <sequence>
            <element name="question" type="string" minOccurs="1"/>
      </sequence>
  </complexType>
</element>
</schema>
```

- Quizend_server

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">

<element name="toplist">
  <complexType>
      <sequence>
        <element name="user" minOccurs="5" maxOccurs="5">
            <complexType>
          <simpleContent>
            <extension base="string">
              <attribute name="place" type="int" />
              <attribute name="points" type="int" />
            </extension>
          </simpleContent>
        </complexType>
      </element>
    </sequence>
  </complexType>
</element>
</schema>
```

Erläuterung: In der Topliste wird gezeigt in welchem Rang die Benutzer mit ihren Punkten gelandet sind.

- Quizlist

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">

<element name="quizlist">
  <complexType>
      <sequence>
            <element name="quiz" type="string" minOccurs="1"/>
      </sequence>
  </complexType>
</element>
</schema>
```

- Quizstart_server

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">

<element name="listofparticipants">
  <complexType>
    <sequence>
      <element name="username" type="string" minOccurs="5" maxOccurs="10"/>
    </sequence>
  </complexType>
</element>
</schema>
```

Erläuterung: In einem Quiz müssen mindestens 5 Spieler teilnehmen und bis zu 10 Spielern aufnehmen.

- Userinformation

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="Userinformation">
       <complexType>
         <sequence>
             <element name="username" type="string" minOccurs="1" maxOccurs="1"/>
             <element name="password" type="string" minOccurs="1" maxOccurs="1"/>
             <element name="profilepic" type="base64Binary" minOccurs="0"
maxOccurs="1"/>
             <element name="realforename" type="string" minOccurs="0"
maxOccurs="1"/>
             <element name="reallastname" type="string" minOccurs="0"
maxOccurs="1"/>
             <element name="website" type="string" minOccurs="0" maxOccurs="1"/>
             <element name="detailinfo" type="string" minOccurs="0"
maxOccurs="1"/>
         </sequence>
       </complexType>
  </element>
</schema>
```

Erläuterung: In diesem XML-Schema werden genauere Informationen über den Benutzer mitgeteilt.

4. Meilenstein 2

In diesem Meilenstein sollen die Ressourcen und die jeweiligen HTTP-Operationen, die im Projekt anwendbar sind aufgeschrieben werden, sowie ein XML-Schema erstellt werden.
Im Vorfeld werden die Begriffe Ressourcen und HTTP-Operationen näher erklärt.

Ressourcen:
Ressourcen bilden das zentrale Konzept von REST. Die Ressourcen werden als Objekte ausgezeichnet, die eindeutig Identifizierbar sind. Über den Austausch von einen oder mehreren Repräsentationen erfolgt eine Interaktion. Im Web werden für die Identifikation von Ressourcen URI´s verwendet.

Beispiel: http://url.de/ressource/

HTTP-Operationen:
REST-konforme Dienste müssen einige Operationen vorsehen, die auf alle Dienste angewendet werden können.
In diesem Projekt werden bei REST vier Standard Operationen von HTTP verwendet. Diese lauten:

GET	Zeigt eine Repräsentation einer Ressource an
PUT	Ändert die Inhalte einer Ressource (neue Ressource kann angelegt werden oder bestehende Ressource kann mit PUT ersetzt werden)
POST	Fügt der Ressource Inhalte hinzu (die Ressource existiert schon)
DELETE	Löscht Inhalte der Ressource

Die vorläufigen Ressourcen im Projekt lauten:

Spiel: bestehend aus Teilnehmer, Quiz, Fragen, Antworten, Rangliste, Joker, Antwortzeit, richtige/falsche Antwort

Quizliste: bestehend aus Quiz

Quiz: bestehend aus Bewertung und Fragen

Fragen: bestehend aus Antworten

Benutzerliste: bestehend aus Benutzern

Antwort: bestehend aus richtigen/falschen Antworten

Hierbei wurden die Ressourcen Spiel und Antwort im späteren Verlauf hinzugefügt. Zumal sich das „Spiel“ nach mehreren Reflexionen und Beispielen als unumgänglich herausgestellt hat. Die Ressource „Antwort“ sollte vorerst in die Ressource „Frage“ untergebracht werden. Jedoch wurde es nach einigen Überlegungen als eine eigene Ressource in Betracht gezogen, da es übersichtlicher ist.

Die HTTP-Operationen wurden auf die Ressourcen „Quizze“ und „User“ aufbauend dargestellt und in Unterbereiche aufgeteilt.
Die unten erfassten Operationen waren die ersten Grundgedanken für das Projekt.

HTTP- Operationen

Quizze

GET	Quizze auflisten
PUT	/
POST	Quizze hinzufügen
DELETE	/

Quizze/ Quiz

GET	/
PUT	neues Quiz hinzufügen
POST	/
DELETE	Quiz löschen

Quizze/ Quiz/ Frage

GET	Fragen auflisten
PUT	neue Frage
POST	Fragen ändern
DELETE	Frage löschen

Quizze/ Quiz/ Antwort

GET	Antwort ausgeben
PUT	neue Antwort
POST	Antwort ändern
DELETE	Antwort löschen

Users

GET	Userliste
PUT	/
POST	User hinzufügen
DELETE	/

Users/ UserID

GET	Daten angeben
PUT	neuer User
POST	Angaben bearbeiten
DELETE	User löschen

5. Meilenstein 3

In diesem Meilenstein soll in Java ein RESTful Webservice erstellt werden.
Hierzu soll die Progammierschnittstelle JAXB verwendet werden um marshalling/unmarshalling zu ermöglichen. Bei marshalling werden strukturierte Daten in ein Format umgewandelt, die eine Übermittlungen zu anderen Prozessen erlauben. Bei unmarshalling werden auf der Empfängerseite die Daten in ihre ursprüngliche Struktur wiederhergestellt.
Die Ressourcen und die Operationen müssen implementiert werden zugleich sollen PathParams und QueryParams eingesetzt werden.

5.1 Vorgang

Um mit dem Webservice zu beginnen, musste man als erstes die Jars/Libraries in das Projekt integrieren, die dafür notwendig waren wie z.B. vom Grizzly-Servers und Jersey-Frameworks. Die ermöglicht REST im Rahmen von Webservices zu ermöglichen und zu vereinheitlichen.

Die JAXB Klassen werden hier nicht aufgeführt, sie befinden sich in Github unter „Generate“ sowie der Webservice unter „minirestwebservice“.
Im „minirestwebservice“ ist die Klasse „Server.java“ für den URL verantwortlich und die Klasse „client.java“ für die exakte Eingabe.

Die Ressourcen „Quizze“ und „User“ werden mit ihren Operationen im Webservice implementiert. Beide Ressourcen haben eine eigene Klasse und werden durch die Klasse „xml_worker“ im Package Ressource zusammengeführt. Im „xml_worker“ wird das marshalling/unmarshalling eingelesen und die Ressourcen können Informationen abrufen.
Die HTTP-Operationen, die in Meilenstein 2 festgelegt wurden, sind in die Klassen „Quizze“ und „User“ eingesetzt.

Verwendung von PathParams und QueryParams
PathParam
wurde anhand einer User-ID implementiert, um eine explizite Ressource zu erhalten.
QueryParams
wurden in beiden Klassen umgesetzt und ist für die Suche zuständig.

In einem der Beratungsterminen wurden wir darauf angesprochen, wie man die Sicherheit gegen unerwünschte Modifikationen zu gewährleisten hat.
Im Buch gibt es ein Verfahren, dass dieses Problem beheben kann.
HMAC (Keyed- Hashing for Message Authentication) versucht die Nachricht um eine Signatur anzureichern, die dem Server wie auch dem Client als einen bekannten geheimen Schlüssel erkannt wird. Jedoch konnte das Verfahren in meinem Projekt nicht realisiert werden können.

6. Meilenstein 4

Meilenstein 4

Hierbei wird die Planung der asynchronen Kommunikation in den Vordergrund gestellt. Die Topics sollen für das Projekt sowie die übertragenen Daten ermittelt werden. Zudem soll Publisher und Subscriber für die gefundenen Topics erkundet bzw. die Position dargestellt werden.

Für das Projekt wurde nur ein bedeutendes Topic definiert:

1. Leaf/Topic	Quiz
2. Publisher	Beitreter
3. Subscriber	Beigetrittener
4. Daten	Nachrichten

Im allgemeinen Spiel also im Quiz existieren die User. Entweder sind sie im Quiz registriert oder werden noch Beitreten. Die Daten, die zwischen Beigetrittener und Beitreter ausgetauscht werden sind die Nachrichten.

7. Meilenstein 5 + 6

Da diese beiden Meilensteine nicht gelöst wurden, gebe ich eine kurze Erläuterung, was für Aufgaben hinsichtlich der beiden Meilensteine gemacht werden mussten.

In diesen beiden Meilensteinen sollte zuerst ein XMPP-Server (Openfire) installiert und später ein Client entwickelt werden.
Mithilfe der Anwendung soll man die Leafs abonnieren, Nachrichten empfangen und veröffentlichen.
Um den Client darstellen zu können benötigt man den REST Webservice und die XMPP Server. Man soll den Client und Server auf verschiedenen Systemen ausführen können.

8. Fazit

In Phase 2 des WBA2 Workshops habe ich mich gründlich mit dem Thema RESTful-Webservices beschäftigt.
Da das Modul für mich Neuland war, musste ich viel Zeit, Kraft und Geduld für die Themenbereiche investieren und mich einarbeiten.
Ich denke, dass ich mein bestes gegeben habe um das Projekt so weit wie möglich zu bringen, obwohl mich mein Gruppenpartner im Stich gelassen hat.

Ich bin der Meinung, dass man für die Bearbeitung der Aufgaben in den Phasen mehr Zeit benötigt hat und vor allem mehr Einarbeitung, denn die Vorlesungen waren nicht genügend.
Es soll kein Kritik sein, jedoch fühle ich mich als Anfänger so.

Trotzdem hoffe ich auf ein ausreichendes Ergebnis.
Um meinen Zustand erkenntlich zu machen hatte ich frühzeitig mit Herrn Fischer per E-Mail und mit den Betreuern Kontakt aufgenommen.

Der Webservice funktioniert nicht und die Meilensteine 5+6 konnte ich nicht machen. Dafür habe ich versucht die Dokumentation ausführlich zu bearbeiten.

9. Quellenverzeichnis

- www.w3school.com
- www.wikipedia.de
- www.openbook.galileocomuting.de
- www.stackoverflow.com
- REST und HTTP von Stefan Tilkov
- Verteilte Systeme von Tannenbaum

Es wurde noch auf weiteren Internetseiten recherchiert, die aber nicht zum Kontext beigetragen haben.