Projektdokumentation
Web-basierte Anwendungen
Verteilte Systeme

Paul Will
Daniela Kucharczyk

23.06.2013

Inhaltsverzeichnis

1 Einleitung

In der zweiten Phase des Moduls Web-basierte Anwendungen: Verteilte Systeme geht es um die Entwicklung einer Applikation, die den Datenaustausch in verteilten Systemen umsetzt. Dazu soll zum einen ein RESTful Webservice erstellt werden, der die synchrone Kommunikation realisiert und zum anderen ein XMPP Server, der die asynchrone Kommunikation ermöglicht. Abschließend dient ein selbstentwickelter Client zur Repräsentation des Projektes.

Im Verlauf dieses Dokuments werden die Entwicklungsschritte zu den vorgegebenen Meilensteinen dargestellt und einzelne Ergebnisse bzw. Gedankengänge zur Ideenfindung und ihren Lösungen festgehalten.

2 Konzept

Unser System soll eine Möglichkeit bieten Interessengebiete, die das aktuelle Semester betreffen, zu organisieren. Durch das abonnieren eines Interessengebiets kann man dann alle Informationen, die diesbezüglich eingestellt sind, abrufen. Dozenten werden sowohl Inhalte über das System erstellen, speichern und editieren, als auch aktuelle Neuigkeiten bekannt geben können. Diese Meldungen, aber auch weitere wissenswerte Informationen, wie zum Beispiel anstehende Pflichttermine, werden per Newsticker an die Abonnenten gesendet.

2.1 Synchrone Kommunikation

Über das System können nach Abschluss eines Abos diesbezüglich alle vorhandenen Informationen abgerufen werden:

- Modul: Veranstaltungszeitpunkt, Veranstaltungsraum, Dozent, Credits, Beschreibung, Übungen, ggf. Praktikum
- Dozent: Fachrichtungen, laufende/vergangene Veranstaltungen, Kontakt
- Semester: Modulübersicht, Praktika, Übungen, Prüfungsfristen/-zeitraum, Stundenplan
- Studiengang: Studienverlaufsplan, Prüfungsordnung, Beschreibung, Regelstudienzeit, Studienumfang, Events

- Zeitraum: Übersicht Modulangebot, Prüfungszeitraum, Rückmeldefrist, Blockveranstaltungen

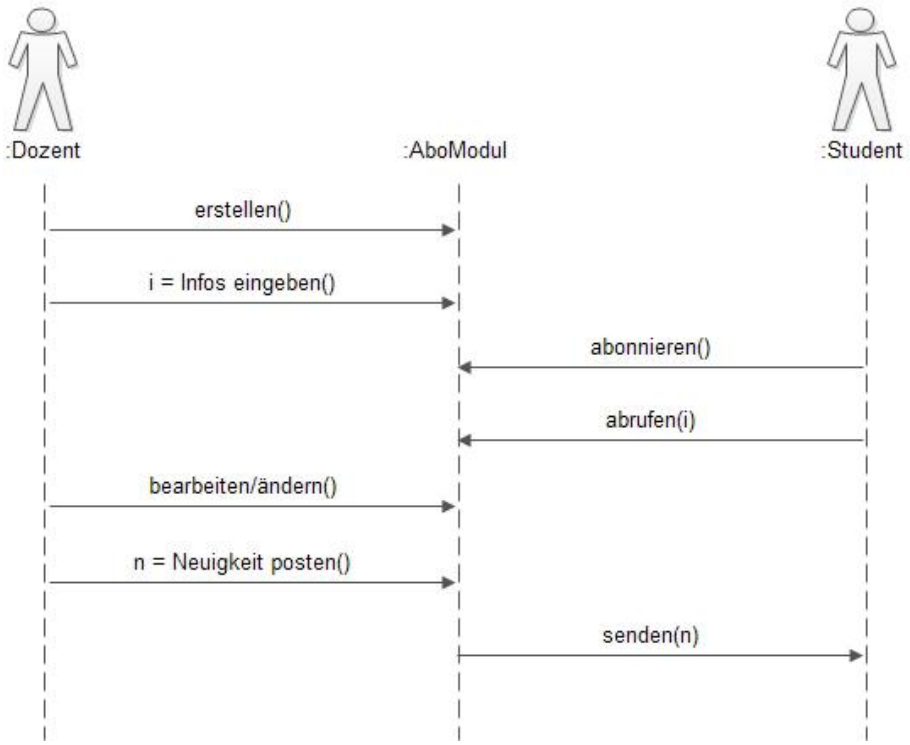
Außerdem können Dozenten über das System Beiträge erstellen, speichern und editieren, z.B.: Modulverlaufsplan, Beschreibung, Vorkenntnisse, Erwartungen, ähnliche Module, Teilnehmerzahl, Credits/Leistungsumfang, Pflichttermine.

2.2 Asynchrone Kommunikation

Ein Newsticker soll dazu dienen die abonnierten Topics zu verwalten. So können Dozenten Neuigkeiten bekannt geben und jeder Abonnent erhält diese, sobald er sich wieder mit dem System verbindet.

2.3 Kommunikationsabläufe

Anhand des nachfolgenden Sequenzdiagramms ist sowohl die synchrone, als auch die asynchrone Kommunikation ersichtlich. Der Dozent erstellt ein Topic und kann dieses durch bearbeiten mit weiteren Informationen anreichern. Die bestehenden Topic können dann wiederum von Studenten angeschaut und abonniert werden. Wodurch sie dann über Änderungen und aktuelle Neuigkeiten per Nachricht informiert werden.



3 Entwicklung des Projektes

3.1 Projektbezogenes XML Schema / Schemata

3.1.1 Vorüberlegungen

Der erste Meilenstein umfasst die Darstellung von Daten in XML, wodurch diese für unsere Verwendung in einer dafür abgestimmten Form gespeichert werden. Um dies zu realisieren und Probleme bei der weiteren Verwendung von JAXB zu umgehen ist es notwendig den Inhalt und die Struktur der XML-Daten durch XML Schemata zu definieren.

Das System der StudyNEWS soll eine Verwaltung der Topics und Benutzer in Listen ermöglichen, welche ebenfalls als Grundlage für die asynchrone Kommunikation dienen. Außerdem soll man die Möglichkeit haben sich als Student oder Dozent anzumelden, wodurch dem Benutzer unterschiedliche Funktionen zur Verfügung stehen. Daher ergaben sich bei der Vorüberlegung folgende Objekttypen, die innerhalb der StudyNEWS von Bedeutung sein könnten und wiederkehrende Elemente beinhalten:

<u>Userliste</u>
Beinhaltet eine Auflistung aller Benutzer, wobei jeder Eintrag aus dem Nachnamen, Vornamen, Status, also als was man sich angemeldet hat, und einer ID zur Identifikation besteht

<u>Dozent</u>
Da der Dozent auch ein Topic ist, welches man abonnieren kann, ist ein detailliertes Profil notwendig. Neben dem Namen und der Adresse, die sich aus mehreren Komponenten zusammensetzt, sollen ebenfalls Informationen bezüglich der Lehre gespeichert werden. Durch eine Liste der vom Dozenten erstellten Module soll zusätzlich auf Topics aufmerksam gemacht werden.

Student

Im Gegensatz zum Dozenten benötigt der Student kein ausführliches Profil, da kein anderer Benutzer darauf zugreifen kann. Allerdings hielten wir die Kennung, Studiengang, Semester und Emailadresse trotzdem für wichtig, da diese Informationen eventuell für die asynchrone Kommunikation von Bedeutung sein könnte.

Modul

Das Modul ist ein weiteres Topic, welches abonniert werden kann und soll dem Abonnenten einen Überblick über die wichtigsten Inhalte geben. Außerdem soll der Ersteller die Möglichkeit haben das Modul online oder offline zu setzen, je nachdem, ob es eine laufende Veranstaltung ist.

Topicliste

Aufzählung aller Topics und wichtiges Element für die asynchrone Kommunikation, da durch das Abonnieren eines Topics die Benachrichtigungen ausgelöst werden. Bestandteil eines Listeneintrags ist die Bezeichnung, Kürzel und der Status.

3.1.1 Umsetzung

Nachdem die Vorüberlegungen abgeschlossen waren, definierten wir für jeden identifizierten Objekttypen ein eigenes Schema und belegten die einzelnen Elemente/Attribute mit Datentypen. Der Gedanke dabei war die Datenstruktur relativ einfach zu halten und die spezifischen Informationen bezüglich der Objekte Student, Dozent und Modul erfassen zu können. Die Schemata Userliste und Topicliste wurden umgesetzt, um die spätere Listendarstellung und die Vergabe von IDs für die Instanzen von Student, Dozent und Modul zu vereinfachen. Auf Grundlage jedes Schemas erstellten wir dann zusätzlich mindesten ein XML Dokumente mit Beispieldaten . Das hatte den Vorteil, dass wir durch Validierung die Schemata auf Korrektheit und Sinnhaftigkeit bezüglich unseres Projekts prüfen konnten.

3.2 Ressourcen und die Semantik der HTTP-Operationen

Für die Umsetzung der synchronen Kommunikation sollte vorerst eine theoretische Auseinandersetzung mit HTTP-Operationen und Ressourcen im Kontext RESTful Webservice erfolgen. Wodurch sich für uns folgende grundlegende Schritte herausstellten, um unseren Dienst zu entwickeln:

1. Welche Ressourcen werden benötigt?
2. Mit welchen URIs werden diese Ressourcen repräsentiert?
3. Welche HTTP-Operationen werden von den jeweiligen Ressourcen unterstützt?

3.2.1 Ressourcen

Eine Ressource ist eine abstrakte Schnittstelle zu einem Objekt oder Dienst und stellt einen derzeitigen oder beabsichtigten Zustand dar. Folglich ergeben sich im Rahmen unseres Projektes zu den bereits definierten Objekttypen die Primärressourcen Student, Dozent und Modul. Desweiteren werden die Listenressourcen Userliste und Topicliste zur Verwaltung der einzelnen Elemente festgelegt. Für die Repräsentation unserer Ressourcen verwenden wir `application/xml` als zugeordneter Medientyp, was sich daraus ergibt, dass wir unsere Daten in XML speichern und die Ressourcen somit XML verarbeiten und zurückgeben.

3.2.2 Festlegen der URIs

URIs dienen zur Adressierung der Ressourcen und ermöglichen somit die Interaktion zwischen Server und Client. Daher ist für das effiziente arbeiten wichtig, dass für jede Ressource eine eindeutige URIs festgelegt wird. Bei dem Entwurf dieser URIs stellten wir allerdings fest, dass sich einige Funktionen die gleichen URIs verwenden. Obwohl dies kein Problem für die Lokalisation der einzelnen Funktionen darstellt, da die Unterscheidung über die Art des HTTP-Request stattfindet, haben wir für den eigenen Überblick aussagekräftiger URIs verwendet. Die Abschnitte, die bei den für unsere Ressourcen bestimmten URIs in geschweiften Klammern stehen, werden später durch die konkrete ID einer Instance bestimmt.

Ressource Dozent:

- http://www.example.com/dozent
- http://www.example.com/dozent/{id}
- http://www.example.com/dozent/add
- http://www.example.com/dozent/{id}/edit
- http://www.example.com/dozent/{id}/delete

Ressource Student:

- http://www.example.com/student
- http://www.example.com/student/{id}
- http://www.example.com/student/add
- http://www.example.com/student/{id}/edit
- http://www.example.com/student/{id}/delete

Ressource Modul:

- http://www.example.com/modul
- http://www.example.com/modul/{id}
- http://www.example.com/modul/add
- http://www.example.com/modul/{id}/edit
- http://www.example.com/modul/{id}/delete

Ressource Userliste:

- http://www.example.com/userliste

Ressource Topicliste:

- http://www.example.com/topicliste

3.2.3 HTTP-Operationen

Die Anwendungsfälle bezüglich der einzelnen Ressourcen werden mit den vier Anfragetypen GET, POST, PUT und DELETE abgedeckt. Wobei den HTTP-Methoden gemäß Konvention folgende Bedeutung zukommt:

- **GET** liefert eine oder mehrere Ressourcen in einer bestimmten Repräsentation zurück
- **POST** erzeugt eine Ressource
- **PUT** aktualisiert eine bereits vorhandene Ressource
- **DELETE** löscht eine Ressource

Ressource: Dozent

Beschreibung	HTTP Methode	URI
Dozent laden	GET	/dozenten/{id}/
Dozent erstellen	POST	/dozenten/add
Dozent aktualisieren	PUT	/dozenten/{id}/edit
Dozent löschen	DELETE	/dozenten/{id}/delete
Neuigkeit posten	PUT	/dozenten/{id}/news/{id}

Ressource: Student

Beschreibung	HTTP Methoden	URI
Student laden	GET	/student/{id}
Student erstellen	POST	/student/add
Student aktualisieren	PUT	/student/{id}/edit
Student löschen	DELETE	/student/{id}/delete
Abo abschließen	PUT	/student/{id}/abo
Abo abbestellen	DELETE	/student/{id}/abo/{id}/delete
Neuigkeit laden	GET	/news

Ressource: Modul

Beschreibung	HTTP Methoden	URI
Modul laden	GET	/modul
Modul erstellen	POST	/ modul/add
Modul aktualisieren	PUT	/modul/{id}/edit
Modul löschen	DELETE	/ modul /{id}/ delete

Ressource: Userliste

Beschreibung	HTTP Methoden	URI
Liste aller Benutzer	GET	/user

Ressource: Topicliste

Beschreibung	HTTP Methoden	URI
Liste aller Topics	GET	/topics

Wie zu erkennen ist, stellt nicht jede Ressource alle Methoden zur Verfügung. Die Ressourcen Userliste und Topicliste beschränken sich lediglich auf eine Anfrage mit GET, da sie nur als Verzeichnis dienen.

3.3 RESTful Webservice

Ein RESTful Webservice soll in Java erstellt werden. Dazu gelten folgenden Bedingungen:

- Mindestens zwei Ressourcen müssen implementiert sein
- JAXB soll für das marshalling / unmarshalling verwendet werden
- Die Operationen GET, DELETE und POST müssen implementiert sein
- Es sollen mindestens einmal PathParams und einmal QueryParams verwendet werden

Die umgesetzten Ergebnisse sind unter folgendem Link zu finden: https://github.com/Butterfly3108/wba2_2_studynews/tree/master/wba22_studynews/src/restfulWebservice

3.4 Konzeption + XMPP Server einrichten

Hierbei handelt sich um einen konzeptionellen Meilenstein für die genauere Planung der asynchronen Kommunikation, wobei folgende Punkte bearbeitet werden sollen:

- Leafs (Topics) sollen für das Projekte definiert werden
- Wer ist dabei Publisher und wer Subscriber?
- Welche Daten sollen übertragen werden?

3.4.1 Topics

Die Verwendung von Topics ist ein Type-basierter Ansatz und ermöglicht das Versenden von Nachrichten, obwohl sich Sender (Publisher) und Empfänger (Subscriber) nicht kennen. Bei dieser Umsetzung bekunden die Empfänger ihr Interesse, indem sie einen Typ von Objekten abonnieren und erhalten dann alle Events dieses Typs. Durch den Einsatz von Topics braucht der Sender nur an ein Ziel zu

versenden und erreicht trotzdem alle relevanten Empfänger. Wann gesendete Nachrichten bezüglich eines Topics empfangen werden bestimmt bei diesem Modell der Abonnent. Übertragen auf unser Projekt lassen sich somit Objekte der Typen Modul und Dozent, als unsere Topics bestimmen. Dies könnten beispielsweise Topics mit der Bezeichnung WBA2, EMI, MA2, Stenzel oder Konen sein, abhängig davon welche Topics von den Sendern erstellt werden.

3.4.2 Publisher / Subscriber

Bei unserem Projekt werden die Rollen schon bei der Registrierung der Benutzer und des dabei gewählten Status festgelegt, was bedeutet, dass das der Dozent als Nachrichtensender und die Studenten als Nachrichtenempfänger fungieren. Durch diese Festlegung wird nicht nur vorgeschrieben, wer Nachrichten zu Topics senden oder empfangen darf, sondern auch, wer berechtigt ist neu Topic zu erstellen und bestehende zu löschen.

3.4.3 Payload Daten

Um die Struktur und den Inhalt der zu übertragenden Daten zu definieren und besser verarbeiten zu können, erstellten wir als Grundlage ein XML Schema, welches aus wenigen, unserer Auffassung nach wissenswerten Elementen zusammengesetzt ist und wie folgt aussieht:

```
<xs:element name="notification" >
      <xs:complexType>
            xs:sequence>
                  <xs:element name="datum" type="xs:date" />
                  <xs:element name="verfasser" type="xs:string" />
                  <xs:element name="topic" type="xs:string" />
                  <xs:element name="nachricht" type="xs:string" />
            </xs:sequence>
      </xs:complexType>
</xs:element>
```

3.5 XMPP - Client

Es soll ein XMPP Client entwickelt werden, dafür soll folgendes berücksichtigt werden:

- Mittels der Anwendung soll es möglich sein Leafs zu abonnieren, Nachrichten zu empfangen und zu veröffentlichen.
- Ermöglichen Sie die Übertragung von Nutzdaten (Beispielsweise Simplepayload)
- Leafs und ggf. mögliche Eigenschaften der Leafs sollen angezeigt werden können (Service Discovery)

3.6 Client - Entwicklung

Abschließend soll ein Client erstellt werden, welcher das Projekt repräsentiert. Dafür ist eine Nutzung des REST Webservices und des XMPP Servers erforderlich:

- Das System (RESTful Webservices sowie XMPP Server) muss über ein grafisches User Interface nutzbar sein
- Verwendung von Swing wird empfohlen
- Getrenntes Ausführen auf unterschiedlichen Systemen von Client und Server /Webservice sollte optimaler Weise möglich sein

3.6.1 Vorüberlegungen

Zu Beginn machten wir uns erst einmal Gedanken über ein mögliches Layout, welches alle gewünschten Funktionen beinhaltet. Dabei stand für uns ausschließlich die gewünschte Funktionalität im Mittelpunkt.

Eine Navigationsleiste mit Buttons soll dazu dienen die verschiedenen Bereiche der StudyNEWS zu erreichen. Neben dem Bereich, um das eigene Profil zu bearbeiten oder zu löschen, sollen ebenfalls die Bereiche Abonnements und Modul umgesetzt werden. Die Komponenten dieser Seiten soll so gestaltet werden, dass je nach Benutzerstatus unterschiedliche Funktionen zur Verfügung stehen.

3.6.2 Umsetzung

Während der Umsetzung mit Java Swing wurde die GUI ständig verändert und weiterentwickelt, um auftretende Schwierigkeiten zu beheben. Alle Funktionen auch immer noch nicht umgesetzt werden oder laufen fehlerhaft.