Fachhochschule Köln
Cologne University of Applied Sciences

Web-basierte Anwendungen 2: Verteilte Systeme

Logbuch zur Projektphase 2 – „SerienTracker“

Sommersemester 2013

Dozent: Prof. Dr. Fischer

Betreuer: Renée Schulz
David Bellingroth
Christopher Messner

ausgearbeitet von
Dennis Meyer 11084479

Gummersbach, 23.06.2013

Vorliegendes Logbuch ist Teil des WBA2. Projektes aus Phase 2.
Es werden hiermit die Themen, Überlegungen, Recherchen, Probleme und Gedanken aufgeführt, mit denen sich während der Projektentwicklung beschäftigt wurde.

17. April
Nachdem die Grundlegende Idee als Konzept entwickelt wurde, habe ich mir einen Einblick verschafft, wie Seiten im Internet zu diesem Thema gestaltet sind. Genau wurde dabei auf http://trakt.tv/, http://serienjunkies.de und http://imdb.com geschaut, welche Informationen von Interesse sein könnten. Die Anzahl der Informationen wird dabei unterschiedlich gehandhabt, wobei jede Seite auch einen anderen Ansatz verfolgt.
Auf jeden fall sollten wir uns auf das wesentliche Beschränkungen. Gerade die Community Verbindung, die eventuell als Freundefeature eingebaut werden soll, ist in der Regel sehr umfangreich.

22. April
Grundlegende Überlegung zu Kommunikationsabläufen. Zwischen wem findet ein Austausch im System statt. Dabei entstand in Zusammenarbeit die Übersicht zu synchronen Kommunikationsabläufen und asynchronen Kommunikationsabläufen.
Auch erste Gedanken zu den REST Methoden, da noch kein Vorwissen bestand, wurde vorallem mit http://www.restapitutorial.com/lessons/httpmethods.html ein kurzer Überblick verschafft.

Nachdem das Konzept soweit steht, folgt nun das angehen des ersten Meilensteins.
Dabei fand auf Grundlage vorherige Recherche und den Ideen zum Konzept die Überlegung statt, welche Objekte der realen Welt im Serientracker verwendung finden könnten.
Ergebnis: Serie, Episode, Season, Liste und User.
Es folgte die Entwicklung des XML Schemas der Serie mit Überlegung sinnvoller Elementen und Restriktionen. Dabei Beachtung gängiger Informationen wie Sender, Genre, Episodendauer etc. angelehnt an Informationen von IMDB und Trakt.tv.
Erfahrungen aus der ersten Phase sorgten dabei schnell für erste Ergebnisse und ein valides Schema.

24. April
Entwicklung des XML Schemas einer Episode, User, List und Season. Anpassung der Serie in Hinsicht auf weitere Restriktionen und Einbindung einer ID. Zur Aufteilung der Komplexität, werden die Schemen von Serie, Season und Episode jeweils aufgeteilt. Zur Identifikation werden Elemente des ID Typs angelegt und sollten untereinander referenziert werden.
Bisher Nutzung des Grundlagenwissen für XML Schemas aus Phase 1, keine weiteren Probleme.
Damit Season.xsd und Episode.xsd Bestandteil der Serie.xsd sind, aber nicht doppelt definiert werden müssen, sollten diese Schemas eingebunden wurden.

Dazu habe ich mich mit der Schemaeinbindung beschäftigt und die Varianten Import und Include ausprobiert. Für unseren Zweck wäre Inlude eine geeignete Variante, da Import eine Art Kopie erstellt und einbindet. Da Include jedoch auf die XSD zurückgreift, in der die Informationen definiert sind, muss der selbe Namespace referenziert werden.

Im Test kam es dabei noch zu Komplikationen, trotz Definition eines einheitlichen Targetnamespaces. Deshalb wurde die Anpassung vorerst wieder verworfen, sollte aber in Betrachtung gezogen werden.

26. April

Bisherige Ergebnisse in der Dokumentation erfassen. Dazu kam die Entwicklung der XML Schemas für Users und Series als Containerklassen. Da momentan jedes Element sein eigenes XML File erhält, sollte eine Möglichkeit gefunden werden, eine Übersicht aller vorhandenen Objekte eines Typs zu erhalten. Diese Schemas sind reine Behälterklassen, die alle Serien und User aufnehmen, um später eine einheitliche Liste für den Zugriff zu haben.

In Hinblick auf die Entwicklung eines inhaltlich logischen Schemas, kam die Überlegung auf, ob man Abhängigkeiten zu Restriktionen schaffen kann.
Eine Serie könnte beispielsweise in den USA entstanden sein und dort auf dem Sender „ABC“ laufen. Kommt die Serie in Deutschland, so kann sie logischerweise nicht auf diesem Sender eingetragen sein, sondern muss auf einen deutschen Sender verweisen.
Recherche nach solch einer Definition innerhalb des XSD führte vorerst zu keinem Ergebnis.

Nachtrag:
Da dieser Anwendungsfall innerhalb des Projektes nicht beachtet werden wird und das definierte Datum vorerst auf den Ursprungsrelease ausgelegt ist, kann ein solcher Widerspruch nicht auftreten. Dennoch könnte ein Objekt erstellt werden, dass inhaltlich Länder und unpassende Sender verbindet. Mit diesem Problem könnte man sich in der Clientausarbeitung beschäftigen und eventuelle Bedingungen durch Abfragen realisieren.

29. April

Bisherige Bezeichnungen für Elemente und Attribute wurden kontrolliert und durch passendere ersetzt. Dabei viel speziell auf, das viele wiederkehrende Elemente wie Titel, Überblick, Bilder... innerhalb der einzelnen Schemas vorkommen.
Zum testen der Schemen wurden XML Dateien der einzelnen Typen erstellt und auf Validierung geprüft werden. Dabei ergeben sich noch weitere Informationen, die im Schema definiert werden sollten.
Erstellung einer Testuser.xml, Users.xml, list.xml sowie Anpassung der zugehörigen Schemas. Hin und wieder traten Komplikationen innerhalb der XSD auf, die jedoch hauptsächlich auf Syntaxfehler beruhten und behoben werden konnten.

30. April

Erstellen weiterer XML Dateien und Test auf Validierung. Season.xml, Seasons.xml, Serie.xml, Series.xml mit erneuter Anpassung der zugehörigen XSD Dateien.
Zur Vorbereitung auf den 3. Meilenstein, wurde im REST und HTTP Buch von Stefan Tilkov Kapitel 1-4 gelesen. Dabei wurde unter anderem ein erster Überblick zu Ressourcen gewonnen. Bereits die XML Schemas befassten sich dabei mit den grundlegenden Objekten, die später in einer Ressource auftreten sollten.
Diese Erkenntnis bestätigt die Notwendigkeit vorhandener Typen, die zuvor für den Serientracker erkannt wurden.

1.Mai

Für die Dokumentation wurde der Einsatz von Latex beschlossen. Daher folgte Einbindung der vorhandenen Texte in die entsprechende Datei, sowie Einarbeitung ins Programm mit Tests einiger Codebeispiele. Speziell die Seite http://en.wikibooks.org/wiki/LaTeX/ lieferte dabei einen guten Einblick in die einzelnen Funktionen.
Dazu Erstellung einer Lists.xml und Anpassung der vorhandenen lists.xsd.
Gemeinsame Identifikation der Ressourcen sowie Verwendungsmöglichkeiten der REST Methoden. Dazu kommt die Entwicklung entsprechender Abbildungen zur Veranschaulichung in der Dokumentation. Innerhalb der Gruppe fand dabei die Einigung auf Ressourcen recht schnell statt und die entsprechenden Operationen ebenfalls.

Auch dabei wurde auf das Rest und HTTP Buch von Stefan Tilkov sowie die Inhalte der Vorlesung gesetzt.
Größere Probleme oder Verständnisprobleme gab es bisher keine oder konnten selbstständig gelöst werden.

3. Mai
In Hinblick auf die asynchrone Kommunikation, muss noch eine Datenstruktur für die entsprechenden Nachrichten definiert werden. Daher wurde eine Message.xsd mit Testdatei angelegt und getestet.
Die Idee einer Message XML ist, dass für jede Art von Nachricht eine eigene XML erstellt und per ID eindeutig gespeichert wird. Genauer heißt das, dass z.B. Benachrichtigung zum Start einer bestimmten Folge oder Benachrichtung zum Start der Episode eines bestimmten Genres wieder verwendet werden können.

Zuvor verfolgter Ansatz mit Inkludierung der XML Schemas konnte erfolgreich umgesetzt werden.
Nach Recherche im Internet, lieferte mir vor allem die Seite http://www.xfront.com/ZeroOneOrManyNamespaces.html ein gutes Verständnis darüber, wie entsprechende Funktionen umgesetzt werden sollten.

5. Mai
Aufgrund von Komplikationen beim Entwickeln der JAXB, bei denen die ObjectFactory nur die Informationen der letzten XSD enthielten, wurde eine Möglichkeit gefunden dieses Problem zu beseitigen mithilfe der korrekten Inkludierung der Schemen untereinander.

Mit dieser Vorarbeit wurde die XML Dateien zur Episode validiert und die Message.xsd in die Masterinklude eingebunden. Zusätzlich wurde an der Projektdokumentation weiter gearbeitet und Tilkov Kapitel 5 und 6 zur Vertiefung gelesen.

6. Mai
Entwicklung des EpisodesServices/SeriesServices in Bezug auf die GET Methoden. Anwendung von POST und PUT Methoden noch etwas unklar, wenn es um das Schreiben der Informationen geht. Dazu weitere Ausarbeitung der Dokumentation.

7. Mai
Entwicklung des ListsService, SeasonService und UserService mit GET Methoden auf einzelnen und mehrere Objekte. Dazu Beschäftigung mit http://www.torsten-horn.de/techdocs/jee-rest.htm#JaxRsHelloWorld-Grizzly zum Einrichten des Servers (bereits vorhanden) sowie Grundlagen in Entwicklung der entsprechenden Javamethoden bei REST. Speziell die Codebeispiele auf angegebener Seite lieferten dabei einen grundlegendes Verständnis darüber, wie entspechende HTTP Methoden in Java umgesetzt werden können. Tests mit vorhandenen Server lieferten erste Ergebnisse.

12. Mai
Weitere Beschäftigung mit der Einbindung der POST und DELETE Methodes des EpisodesService, SeasonsServices und UsersServices. Anpassung der GET Methoden an vorhandene Änderung mit Einbindung der For-each Schleife.
Dazu Aktualisierung der vorhandenen Dokumentation.

Weitere Fragen die sich ergaben:
Abfrage per Query Elementen.
Wo wird ein Abonnement eines Genres geregelt? -> Ausgabe der Liste eines Genres, über List wie bei normalen Abo geplant ->Shows->Series->Serie->Genre=...?
Würde alle Listen durchlaufen und alle zurückgeben wo Genre vertreten. Hierfür eventuell Genrelisten entwickeln, die nur Serien dieses Typs aufnehmen?

Eine Möglichkeit wäre eine zusätzliche GET Methode in Series, die als Queryparamter eine Genre abfrage ermöglicht -> Abfrage aller Serien eines Genres und anschließend speichern in einer Liste möglich.

Wie werden Abonnements geregelt? User erstellt Liste, kann aber auch Watchlist etc sein, da Definition sehr frei. Unklar welche User bei welcher Serie benachrichtigt werden wollen.

Zusätzliche Ressourcen der User? subscribedSeries und subscribedGenre, Liste aller Serien über die man Nachrichten haben will, Zugriff per GET und POST würde einfaches verwalten dieser Liste ermöglichen, userList, alle listen eines Users, für Profil interessant, Abfrage wäre aber auch über Queryparameter per listID?userID=... denkbar.

Ressource List allgemein, da publicList interessant für alle sein kann. Eventuell sogar subscriptions -> series, genres, network, airday, airtime.

Zu diesen Ansätze sollte nochmal eine Absprache innerhalb der Gruppe erfolgen.

13. Mai
Weitere Arbeit an der Dokumentation.
Dazu Anpassung der XML Dateien und Schemas unter Verwendung des neuen ID Typs mit anschließender Validierung. Identifizierung mit Kürzel für entsprechenden Typ gut, jedoch wirkt der Hash etwas zu beliebig. Für die interne Verarbeitung wird das aber kein Problem darstellen und dieses

In der aktuellen Servicesstruktur des REST Services, sollte die Strukturierung und Auslagerung bestimmter Codeanteile überlegt werden. Der Marshaller/Unmarshaller Service sollte als einzelne Struktur ausgelagert werden und in den entsprechenden Dateien aufgerufen werden. Momentan wird dieser Schritt in jeder Javadatei getätigt und könnte durch eine ausgelagerte Methode vereinfacht werden.
Grundsätzliche Sturkturierung.

18. Mai
Update des EpisodesServices, ListsServices, SeasonsServices, UsersServices mit entsprechend Anpassungen an aktuelles Format sowie Ausbau der POST und DELETE Methoden. Weiterhin noch etwas Unklarheit was das Anlegen der QueryParameter angeht. Bisherige Codebeispiele führen noch nicht zum erwünschten Ziel. Geplant ist die Einbindung in der Listmethode, um die Abfrage nach bestimmten Kategorien zu ermöglichen.

26. Mai
Vorbereitung des 4 und 5 Meilensteins durch theoretische Auseinandersetzung mit XMPP. Dazu wurden das Buch XMPP The Definitive Guide und diverse Quellen im Internet verwendet. Noch besteht etwas Unklarheit mit dem Begriff Leaf. In vorhandener Literatur fällt dies nur im Zusammenhang der Nodes. Konzept von Publisher/ Subscriber ist etwas eindeutiger.

In Bezug auf Meilenstein 5, wurde sich mit dem Service Discovery Prinzip und disco#items sowie disco#info vertraut gemacht.

27. Mai
Entwicklung des ListsServices zur Verwendung der neu ausgelagerten und angelegten List Dateien der REST Methode. Dazu Entwicklung des ListsDataHandler.

29. Mai
Weitere Vertiefung der XMPP Problematik.
Auseinandersetzung mit Nodes und die vorhandenen Topics innerhalb des Systems. Grundsätzlich also alles das, was für die Anwender von Interesse ist und Abonniert werden kann. In unserem Fall Serien oder bestimmte Genrelisten. Bei Ausarbeitung
Auch Informationen zu anderen Usern, sofern die Freundefunktion umgesetzt wird.

01. Juni
Ausbau der List.xsd zur Verwendung als Genreliste. Dazu Änderungen an entsprechenden Schemas und Beispielxml. Einbindung entsprechender Query Abfrage in den RESTService und Update der Dokumentation.

02. Juni
Arbeit an der Dokumentation

04.Juni
Erneute Auseinandersetzung zu XMPP, JID und Payload.
Weiterhin Recherche über http://xmpp.org/extensions/xep-0271.html und Einblick in die praktische Umsetzung auf
http://www.igniterealtime.org/builds/smack/docs/latest/documentation/extensions/disco.html.
Thema für mich momentan noch etwas schwer zu fassen.
Unterschiedliche Bedeutung von Node: Service Discovery: Bereich der XMPP Entität, der den Umgang mit Anfragen und die Antwort bezüglich bestimmte Protokolle regelt.

Publish Subscribe: Bestimmter Feed oder Channel, der in einem PubSub Service gehostet wird. Die Art des Channels wird durch den Payload charakterisiert.

05.Juni
Entwicklung von realen Beispieldatensätzen anhand diverser Internetseiten. Grundlegend hierbei http://trakt.tv und http://imdb.com . Definition innerhalb der Series.xml und keine Aufteilung auf einzelne Daten der Einfachheit halber.

06. Juni
Weitere Beispieldaten.

07.Juni
Beschäftigung mit dem PubSub Prinzip und der praktischen Umsetzung
http://www.igniterealtime.org/builds/smack/docs/latest/documentation/extensions/pubsub.html

09. Juni
Konzeptskizzen für Client entwickeln. Dabei Berücksichtigung der vorhandenen Funktionen.

10.Juni
Anlegen des Clients und Beschäftigung mit Java Swing anhand diverser Tutorialvideos auf Youtube, weitere Seiten im Netz und Beispielen wie auf http://www.zum.de/Faecher/Inf/RP/Java/Dokumente/Java%20Tutorial%20%20Einfache%20GUI.htm.

Panels der einzelnen Seiten wurden angelegt und mit entsprechenden Objekten gefüllt. Prinzipiell ist die Herangehensweise verständlich und führt schnell zu Ergebnissen. Genaue Details wie Formatierung fallen noch etwas schwerer. Zur Vereinfachung der Layoutentwicklung wurde mit dem Layout Manager Miglayout gearbeitet.
Dazu fanden sich auf http://www.miglayout.com/ weitere Tutorials. Die grundlegende Arbeitsweise wurde verstanden, aber die praktische Umsetzung führte dennoch zu einigen Schwierigkeiten, weshalb hier nochmal genauer getestet werden sollte.
Der entsprechende Aufbau orientierte sich dabei noch an den Konzeptskizzen und wird anhand von Beispielen wie auf http://zetcode.com/tutorials/javaswingtutorial/ umgesetzt.

12. Juni
Update Dokumentation sowie Kontrolle vorhandener Ausarbeitungen.

13. Juni und 14. Juni
Weitere Entwicklung am Client, HomeGUI, MessageSettingGUI, EditSerieGUI und weitere Unterseiten. Beim Vergleich zur Konzeptskizze fällt auf, dass die Umsetzung deutlich aufwendiger ist als geplant und einige Funktionen in der Skizze eingebaut wurden, die in einem gesamten System nett wären, für unsere Anwendung aber die Möglichkeit überschreiten. Dazu kommt, dass das Layout grundsätzlich eher Webseiten orientiert ist. Eine leicht Überarbeitete Skizze bestimmter Elemente wurde angefertigt, zudem die Trennung zwischen User und Admin überlegt.
Beschäftigung mit Operationen auf vorhandene Daten durch die JAXB Objekte.

15. Juni
Bei der Arbeit am Client wird speziell mit Cardlayout herumprobiert. Dazu kommt weitere Arbeit an der Dokumentation.

18. Juni
Beschäftigung mit bisher vorhandenem Projektstand, hierbei speziell der XMPP Entwicklung.

19. Juni
Update des vorhandenen Datenbestandes in Betracht auf Client.
Aufbauend auf erster Version des Client und der Klassen wurde eine neue Version erstellt und in der Funktionalität ausgebaut.
Dafür wurden bereits entsprechende Funktionen auf den Datenbestand eingebaut.

Darauf aufbauend folgt die Erweiterung des Datenbestandes, wie Anpassung vorhanden Beispiele an reale Daten der IDs durch den Hasher. Dazu Einbinden von Grafiken dem Hinzufügen weiterer Serien, um beim Start des Clients bereits einen Datensatz zu haben um Funktionalitäten zu testen. Entwicklung von Genrelisten und speziell die Betrachtung der Nodeelemente.

20. Juni
Entwicklung der newContenGUI Übersicht des neuen Clients, aufbauend auf vorheriger Ausarbeitung und der ersten Clientversion. Arbeit an der Dokumentation.
Da hierbei speziell auf CardLayout gesetzt wird, weitere Beschäftigung damit. Während der Entwicklung auf trat das Problem auf, dass die jeweilige Deklaration nicht ganz gepasst hat. Diverse Recherche im Internet wie auf
http://docs.oracle.com/javase/tutorial/uiswing/layout/card.html, führte noch nicht zum gewünschten Ergebnis. Entsprechende Lösung war dann simpler als gedacht, da das falsche Element an die Panels angefügt wurde. Ansonsten war die Ausarbeitung in Ordnung

22. Juni und 23. Juni
Entwicklung der RegisterGUI, auch hier aufbauend auf die alte Version. (Welche im folgenden Cleanup des Projektes entfernt wurde). Dazu die Funktionalität in Verbindung mit Openfire, sodass neue Registrierungen einen entsprechenden User dauerhaft anlegen. Weiterhin funktionale Beschäftigung mit der newSeasonContent.

Finale Ausarbeitung der Dokumentation, Betrachtung und Bewertung des letztendlichen Projektstandes und Vorbereitung der Abgabe.

Persönliches Fazit:
Permanente Auseinandersetzung mit den Themen des Projektes notwendig. Letztendlich sorgte die geringe Zeit dafür, dass nicht jedes Thema in gewünschter Tiefe betrachtet werden konnte, wie angestrebt. Verständnis ist vorhanden, im Laufe des Projektes wurde jedoch deutlich, dass man sich bereits in der Konzeptentwicklung zu hohe Maßstäbe gesetzt hat.