

Projektdokumentation
Web-basierte Anwendungen
Verteilte Systeme

Dennis Meyer
Dominik Schilling

5. Mai 2013

Inhaltsverzeichnis

1	Einleitung	3
2	Konzept	3
2.1	Zusatz - Freunde	3
2.2	Umsetzung	3
2.2.1	Synchrone Datenübertragung	3
2.2.2	Asynchrone Datenübertragung	4
3	Entwicklung des Projektes	6
3.1	Projektbezogenes XML Schema / Schemata	6
3.2	Datentypen	6
3.3	Ressourcen und die Semantik der HTTP-Operationen	7
3.3.1	Ressourcen	7
3.3.2	Parameter	8
3.3.3	Semantik der HTTP-Operationen	9
3.4	RESTful Webservice	11
3.5	Konzeption + XMPP Server einrichten	12
3.6	XMPP - Client	13
3.7	Client - Entwicklung	14
4	Projektrefektion	15
	Abbildungsverzeichnis	16
	Tabellenverzeichnis	17

1 Einleitung

2 Konzept

Die **Idee** ist, dass Serien-Interessierte über ihre zuvor favorisierten Serien und abonnierten Interessen benachrichtigt werden, sobald eine Episode dieser Serie im TV ausgestrahlt wird.

Der **Serien-Interessierte** soll Zugriff auf einen Pool von Serien bekommen, die auf einem Server gespeichert und verwaltet werden. Die einzelnen Serien soll der Nutzer dann favorisieren/abonnieren können um dann über die TV-Ausstrahlung informiert zu werden. Außerdem soll die Möglichkeit bestehen eine Episode zu bewerten und als gesehen/ungesehen zu markieren. Es besteht zudem die Möglichkeiten Serien in Listen einzuordnen und zu verwalten. (definitiv automatische Watchlist/Seenlist)

Die **Server-Anwendung** soll die Nutzer über die TV-Ausstrahlung einer Episode einer zuvor favorisierten Serie in einem vom Benutzer definierten Zeitraum informiert werden. Jeder Eintrag einer Serie enthält zudem Informationen wie Genre, anhand derer Abonnenten bezüglich eines bestimmten Themas, zusätzliche Benachrichtigungen empfangen.

Ein **Content-Admin** soll erweiterte Rechte bekommen, um die Content-Verwaltung zu übernehmen. Die Anwendung soll das Anlegen, Bearbeiten und Löschen von Serien bzw Episoden ermöglichen. Zudem ist somit das Korrigieren von Fehlern möglich, die von Usern eingeschickt werden.

2.1 Zusatz - Freunde

Serien-Interessierte sollen sich gegenseitig hinzufügen/abonnieren können um sich gegenseitig zu benachrichtigen, zum Beispiel in Form von Freund X schaut gerade Y, Freund Z hat Serie/Episode mit 8,0 bewertet oder Freund Y empfiehlt Dir Serie W.

2.2 Umsetzung

Die Anwendung ermöglicht den Austausch von Informationen zwischen Server und Anwender entsprechend den jeweiligen Funktionen.

2.2.1 Synchrone Datenübertragung

Zum einen hat der Anwender direkt die Möglichkeit auf Informationen in Form von Daten zuzugreifen und diese zu Manipulieren.

- Serien-Interessierte
 - Markieren von Episoden

- * Gesehen/Nicht gesehen
- Bewertung einer Episode
 - * Kommentar
 - * Bewertung in Zahlen
- Fehlermeldung
 - * geänderte Sendezeit, fehlerhaftes Datum
- Listen
 - * Ausgabe (Un)Watched
 - * Ausgabe vorhandene Serien
 - * Ausgabe Follower/Following (?)
- Favorisierung
 - * Anlegen
 - * Löschen
 - * Bearbeiten
 - Zeitpunkt der Benachrichtigung
- Content-Admin
 - Verwaltung der Episoden
 - * Anlegen
 - * Löschen
 - * Bearbeiten

2.2.2 Asynchrone Datenübertragung

Ein weiterer Aspekt ist das Anfordern von Informationen, wobei die entsprechenden Informationen von Seiten des Servers von Bedingungen abhängig gesendet werden, was auch mehrfach geschehen kann.

- Serien-Interessierte
 - Benachrichtung bei TV-Austrahlung
 - Freunde mit gleicher Favorisierung bei Serienstart mit Check-in benachrichtigen (?)
 - * Freund X schaut auch W
 - Empfehlung einer Serie von Freund(e) anzeigen (?)
- Content-Admin
 - Benachrichtung bei Fehlermeldung durch User

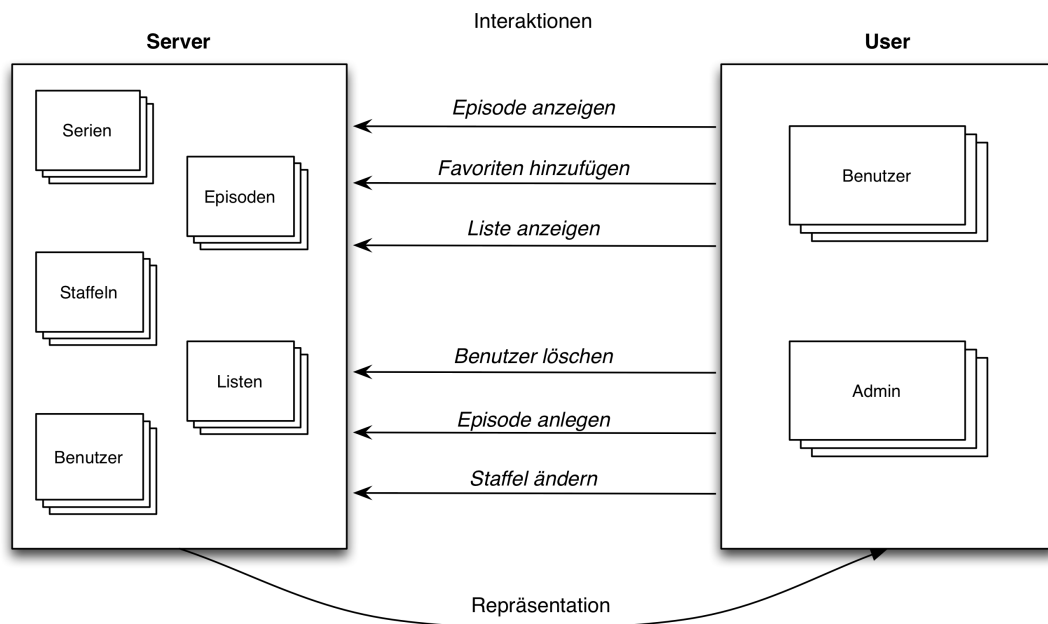


Abbildung 1: Synchrone Kommunikationsabläufe

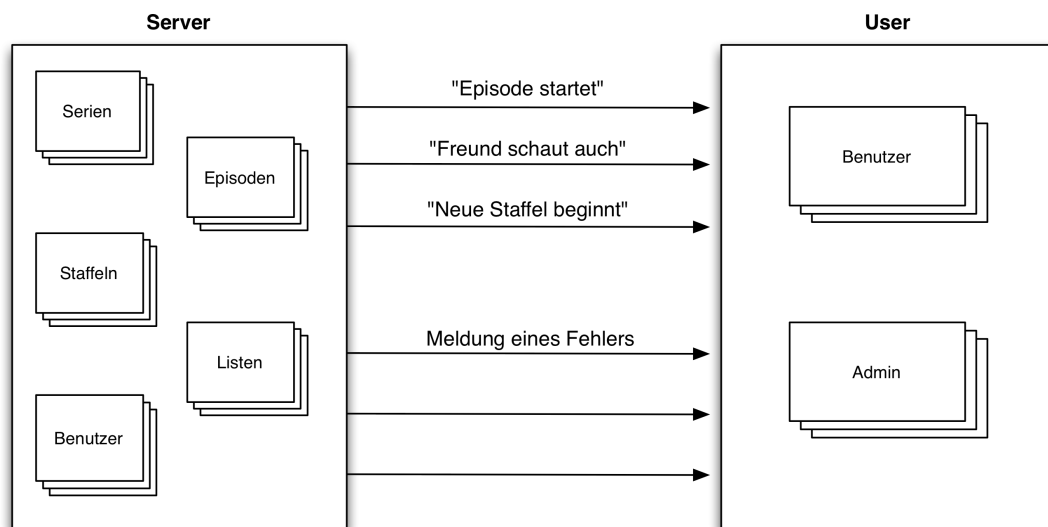


Abbildung 2: Asynchrone Kommunikationsabläufe

3 Entwicklung des Projektes

3.1 Projektbezogenes XML Schema / Schemata

Der erste Meilenstein befasst sich mit der Repräsentation von Daten in XML.

Damit bei der Verwendung der XML Dateien bei der späteren Verarbeitung mit JAXB keine Probleme auftreten, ist es notwendig eine Validierung der Dateien durch Definition zugehöriger XML Schemas durchzuführen. Vorteil bei der Verwendung eines Schemas ist, neben der Kontrolle auf Wohlgeformtheit und der Verwendung definierter Datentypen und Strukturen, auch das festlegen von Restriktionen.

Hinsichtlich des zugrunde liegenden Konzeptes und den benötigten Informationen, gibt es viele Elemente innerhalb der Dateien, die nur mit Strings realisiert werden können. Das Problem bei freier Definition besteht darin, dass die Datensätze sehr fehleranfällig sind, wenn es um die Benutzung durch Menschen geht. Rechtschreibfehler beim Namen des Landes, des Fernsehsenders oder des Genres, würden für den Leser bzw. Interessenten der Anwendung noch kein Problem darstellen, da er vermutlich deuten könnte was gemeint ist. Informationstechnisch ist es aber von Vorteil, die Datensätze möglichst konsistent und reichhaltig anzulegen.

Das System des Serientrackers beruht darauf, die Verwaltung von Serien anhand von Listen und Tags wie Gesehen, Ungesehen zu ermöglichen. Wie bereits in der Besprechung der Umsetzung erwähnt, wird anhand dieser Informationen auch die Asynchrone Datenübertragung realisiert. Das Abonnieren von Informationen zu laufenden Serien des Genre Krimi, greift bei Benachrichtigung auf Datensätze zu, die diesem Elementwert unter dieser eindeutigen Zeichenfolge zugeordnet sind. Formulierungsfehler wie Krimie oder Crime, führen dementsprechend zu Komplikationen, weil sie die Reinheit der Informationssätze beschädigen.

3.2 Datentypen

Zu Beginn des ersten Meilensteins, steht die Analyse der vorhandenen Informationstypen. Bei der Definition eines XML Schemas, sollte neben der entsprechenden Datensatzstruktur und Reihenfolge, auch festgelegt werden, durch welche Datentyp die einzelnen Informationen repräsentiert werden sollen.

3.3 Ressourcen und die Semantik der HTTP-Operationen

3.3.1 Ressourcen

Als Vorbereitung für die Umsetzung der synchronen Kommunikationsvorgänge, steht die theoretische Auseinandersetzung mit REST im Mittelpunkt. Der erste Schwerpunkt dabei ist die Identifizierung vorhandener Ressourcen des Serientrackers. Bereits beim Konzipieren der XML Schemas mit beispielhaften XML Datensätzen, musste überlegt werden, für welche Elemente es möglich ist Entitäten der realen Welt zu ermitteln.

Bei einer Ressource geht es, ähnlich wie bei XML Dateien, nicht darum wie die darin enthaltenen Informationen im letztendlichen Kontext repräsentiert werden, sondern welche Informationen diese enthalten. Entsprechende Objekte der Außenwelt werden beschrieben und wie die Wurzelemente bei XML, stellen sie einen bestimmten Objekttyp dar. Eine identifizierte Ressource, ist eine Schnittstelle zur Außenwelt und sollte daher dem Kontext entsprechend gut durchdacht werden. Die Auseinandersetzung im Rahmen mit den XML Schemas lieferte dabei einen Überblick über vorhandene Primärressourcen. Dabei handelt es sich um die Oberklassen der vorhandenen Objekttypen Serie und User.

Desweiteren ist es möglich vorhandene Subressourcen zu identifizieren, die sich dadurch auszeichnen, dass sie selbst Bestandteil einer Ressource sind. Aufgrund der komplexen Struktur einer Serie, die neben den allgemeinen Informationen noch die Informationen zu mehreren Staffeln und entsprechenden Episoden enthalten, wurde früh die systematische Aufteilung festgelegt. Da es auch innerhalb der Anwendung von Interesse sein kann, eine einfache Repräsentation der Staffelübersicht oder der Episodenübersicht einer Staffel zu ermöglichen, bietet es sich gerade bei diesen Typen an, diese Objekte als eigene Ressource zu designen. Dazu kommt, dass eine Episode zum Beispiel im Kontext einer Serie am meisten Sinn macht, durchaus aber auch für sich existieren kann.

Zur Ordnung der einzelnen Elemente, gibt es entsprechende Listenressourcen wie Series, Seasons, Users und Episodes, welche alle Elemente des zugehörigen Typs aufnehmen und sammeln.

Ein Kernelement der Anwendung, wird das Benachrichtigen der Benutzer über bestimmte Ereignisse sein, die sich durch Abonnements in Form von Listen verwalten lassen. Gerade bei einer möglichen Kategorisierung wie Serie nach dem Genre Drama, wäre es durchaus interessant gewesen Listenressourcen mit entsprechenden Filtern zu definieren. Aufgrund der Vielzahl von Kategorisierungsmöglichkeit, wird dieser Schritt aber nicht über einzelne Ressourcen stattfinden, sondern über die Anwendung mit Verweisen innerhalb der Listen geregelt.

Die Auseinandersetzung führte zu folgenden Ressourcen, wobei die angegebenen URI nur den charakteristischen Abschnitt widerspiegelt. Da für die Umsetzung an sich, die URI eher als eine ID in Form von Zeichen steht und für die letztendliche Auswertung nicht unbedingt notwendig ist, wird auf eine Darstellung in Form mit Schema und Pfadangabe in der Übersicht verzichtet.

Tabelle 1: Ressourcen des Serientrackers

Ressource	URI	Methode
Liste aller Serien	/series/	GET, PUT
Einzelne Serie	/series/{id}	GET, POST, DELETE
Liste aller Staffeln	/seasons/	GET, PUT
Einzelne Staffel	/seasons/{id}	GET, POST, DELETE
Liste aller Episoden	/episodes/	GET, PUT
Einzelne Episode	/episodes/{id}	GET, POST, DELETE
Liste aller User	/users/	GET, PUT
Einzelne User	/users/{id}	GET, POST, DELETE
Liste aller Listen	/lists/	GET, PUT
Liste eines Users	/lists/{id}	GET, POST, DELETE

Jedes Element, dass für die synchrone Kommunikation von Interesse ist, besitzt eine Ressource auf ein einzelnes Element und die Gesamtheit einer Liste. Bei den URI wird der Zugriff über entsprechende ID's auf die Liste deutlich und es zeigt sich eine Folge der einfachen Ressourcen. Durch die Konzipierung der Untertypen Staffel und Episode als einzelne Ressource, wird eine komplexe Hierarchie verhindert, die in Form von /series/{id}/seasons/{id}/episodes/{id} dennoch realisierbar gewesen wäre.

3.3.2 Parameter

Die Repräsentation einer spezifischen Ressource wurde im vorherigen Abschnitt mittels **Path-Parameter** realisiert, das heißt, der Parameter ID war Teil der eigentlichen URI.

Als Alternative können allerdings auch sogenannte **Query-Parameter** zum Einsatz kommen. Diese eignen sich meistens um Ressourcen noch weiter zu verfeinern und sind in der Regel optional.

Exemplarisch könnte ein Query-Parameter in diesem Projekt bei der Ressource Users zum Einsatz kommen, um nur die Benutzer der Gruppe Admin zu repräsentieren: /users/?group=admins. Aber auch bei einem Zugriff auf eine einzelnen Episode einer Staffel einer Serie können Query-Parameter genutzt werden, Beispiel: /episodes/?serie_id=1&season_id=2. Jedoch wären die Parameter an dieser Stelle obligatorisch.

Als Alternative zu den beiden Varianten bietet sich noch der **HTTP-Header** des Clienten an. Da diese Art von Parameterübertragung eher untypisch ist, wird an dieser Stelle nicht weiter drauf eingegangen.

3.3.3 Semantik der HTTP-Operationen

Nachdem die Ressourcen identifiziert sind, muss nun bestimmt werden, welche Operationen (HTTP-Methoden) unterstützt werden und welche Semantik sich dahinter verbirgt.

Dafür stehen die vier Methoden GET (Informationen auslesen), PUT (Informationen anlegen), POST (Informationen ändern) und DELETE (Informationen löschen) zur Verfügung. Nicht jede Ressource muss alle Methoden bereitstellen. Für das Projekt wurden die Ressourcen erneut betrachtet und die benötigten Methoden samt Semantik herausgearbeitet.

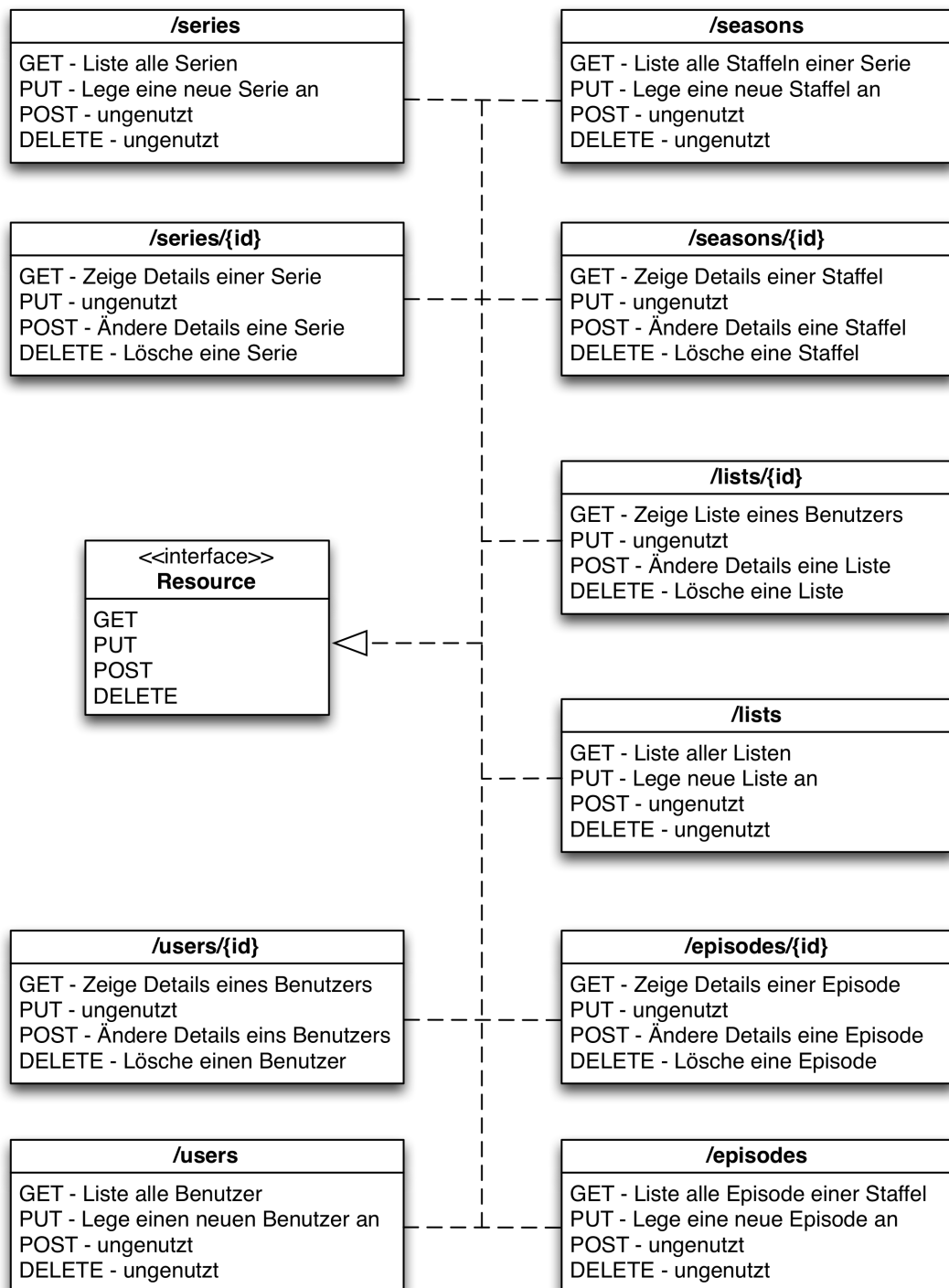


Abbildung 3: Bedeutung der http Methoden

3.4 RESTful Webservice

3.5 Konzeption + XMPP Server einrichten

3.6 XMPP - Client

3.7 Client - Entwicklung

4 Projektreflektion

Abbildungsverzeichnis

1	Synchrone Kommunikationsabläufe	5
2	Asynchrone Kommunikationsabläufe	5
3	Bedeutung der http Methoden	10

Tabellenverzeichnis

1	Ressourcen des Serientrackers	8
---	---	---