# Dokumentation
# Web-basierte Anwendungen
# Verteilte Systeme

Patrick Reimringer
Michael Freund

Inhaltsverzeichnis

# 1. Einleitung

Im Workshop Webbasierte Anwendungen 2 geht es um die Konzeptionierung und Entwicklung eines verteilten Systems. Die Interaktion zwischen den Systemkomponenten soll synchron unter Anwendung des Architekturstils REST und asynchrone unter Anwendungen von XMPP erfolgen. Des weiteren, soll ein Client erstellt werden, welcher die Funktionalität des Systems verdeutlicht.

Diese Dokumentation soll die Vorgehensweise zur Entwicklung des verteilten Systems yeigenn. Hierzu gehört die Grundidee sowie Kommunikationsabläufe und Prüfung der für das System benötigten XML Schemata, die Entwicklung der HTTP-Operationen und Ressourcen im Kontext der RESTful Webservices, die Erstellung des RESTful Webservice und XMPP Client in Java und die Entwicklung des grafischen User Interfaces.

# 2. Grundidee

Ein Programm mit dem ein User einem Gamingstream zusehen kann, erstellen kann und abonnieren kann. Dazu benötigt das Programm eine Synchrone Kommunikation mit einer serverseitigen Systemkomponenten, welche alle benötigten Ressourcen bereitstellt. So lassen sich über einen POST HTTP-Operation z. B. Streamchannel erstellen oder bei einen PUT einen User bearbeiten. Die asynchrone Interaktion findet statt, wenn ein User einen Streamchannel abonniert hat der gerade online gekommen ist, der Server sendet hier asynchron eine Nachricht an den User, die ihn drüber Informieren das gerade ein abonnierte Streamchannel online gegangen ist.

## 2.1 Kommunikationsabläufe

Die genannten Funktionen in der Grundidee werden zwischen Server und Anwendung Informationen austauschen, die in Synchrone und Asynchrone Datenübertragung unterteilt wurden.

### 2.1.1 Synchrone Datenübertragung

- Registrieren
- Profil erstellen und bearbeiten / löschen
- Channel erstellen und bearbeiten / löschen
- Channel abonnieren und kommentieren
- Themen und Kategorien suchen
- Stream starten und beenden

### 2.1.2 Asynchrone Datenübertragung

- abonnierter Channel geht online und benachrichtigt den User

# 3. Entwicklung des Projektes

## 3.1 XML Schema

Der erste Meilenstein der Phase 2 befasste sich mit der Entwicklung eines XML Schemata, auf dessen Basis alles andere aufbaut. Dazu wurden zwei Schemata entwickelt, eins für die User, welches allgemeine Informationen zum User enthält und ein zweites für die Topics.

### 3.1.1 XML Schema User

Im User Schema sind die Benutzer angelegt, dass unendlich viele Benutzer anlegen kann. Der User enthält ein Loginname  und ein Loginpassword, bei denen nur sinnvolle eingaben erlaubt sind.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="Users">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="User" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Login">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="LoginName">
                                            <xs:simpleType>
                                                <xs:restriction base="xs:string">
                                                    <xs:pattern value="[a-zA-Z0-9]*"/>
                                                </xs:restriction>
                                            </xs:simpleType>
                                        </xs:element>
                                        <xs:element name="LoginPassword">
                                            <xs:simpleType>
                                                <xs:restriction base="xs:string">
                                                    <xs:minLength value="3"/>
                                                    <xs:maxLength value="15"/>
                                                    <xs:pattern value="[a-zA-Z0-9]*"/>
                                                </xs:restriction>
                                            </xs:simpleType>
                                        </xs:element>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
```

Des weiteren werden Im User Schema eine UserID, sowie ein Channelname und allgemeine Informationen wie Vorname, Nachname angelegt.

```
<xs:element name="UserID" type="xs:int" />
<xs:element name="UserStreamChannelName" type="xs:string" />
<xs:element name="UserInformation">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="FirstName">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[a-zA-Z]*"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="LastName">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[a-zA-Z]*"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="Birthday" type="xs:string" />
            <xs:element name="City">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[a-zA-Z]*"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="Country">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[a-zA-Z]*"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

Unter dem Element StreamChannel werden neue Channels angelegt mit jeweiligen Channelinformationen, sowie Kommentare und einen Viewer Zähler.

```
<xs:element name="StreamChannel">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="ChannelTopic">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[a-zA-Z0-9]*"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="ChannelDescription">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="ChannelHeadline" type="xs:string" />
                        <xs:element name="ChannelInformation" type="xs:string" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="ChannelComments">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="ChannelComment" maxOccurs="unbounded">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="CommentID" type="xs:int" />
                                    <xs:element name="CommentDate" type="xs:date" />
                                    <xs:element name="CommentData">
                                        <xs:complexType>
                                            <xs:choice>
                                                <xs:element name="CommentHeadline" type="xs:string" />
                                                <xs:element name="CommentText" type="xs:string" />
                                            </xs:choice>
                                        </xs:complexType>
                                    </xs:element>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="TotalViewerCount">
                <xs:simpleType>
                    <xs:restriction base="xs:int">
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

### 3.1.2 XML Schema Topics

Im Topic Schema werden neue Topics angelegt, mit einer ID und einem Namen.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="Topics">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Topic" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="TopicID" type="xs:int" />
                            <xs:element name="TopicName">
                                <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                        <xs:pattern value="[a-zA-Z0-9]*"/>
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:element>
                            <xs:element name="TopicViewerCount" type="xs:int" />
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

### 3.2 Ressourcen und Semantik der HTTP-Operationen

Bei den Verschiedenen Kommunikationsabläufen müssen unterschiedliche Ressourcen abgefragt werden. So muss bspw. bei einer bestimmten User suche eine Get-Operation auf eine Ressource erfolgen  die einen User eindeutig identifiziert . Aus den erarbeiteten Kommunikationsabläufen ist ersichtlich welche Ressourcen benötigt werden.

**HTTP-Operationen**

### 3.2.1 GET

1. Methoden für User-Profil
   - alle registrierten User anzeigen
   - bestimmten User suchen
   - Login-Passwort des User-Profils abfragen

2. Methoden für Streamchannel
   - bestimmten Streamchannel anzeigen
   - Kommentare eines Streamchannels anzeigen

3. Methoden für Topics
    - bestimmtes Topic anzeigen
    - Topics anyeigen

### 3.2.2 POST

1. Methoden für User-Profil
    - ein neues Profil erstellen

2. Methoden für Streamchannel
    - Streamchannnel erstellen
    - Kommentare für einen Streamchannel erstellen

3. Methoden für Topics
    - neues Topic erstellen

### 3.2.3 DELETE

1. Methoden für User-Profil
    - bestimmten User löschen

2. Methoden für Streamchannel
    - Streamchannel löschen
    - Kommentare eines Streamchannels löschen

### 3.2.4 PUT

1. Methoden für User-Profil
    - bestimmten User-Profil bearbeitenn
    - Login Passwort des User-Profils ändern

2. Methoden für Streamchannel
    - Streamchannel bearbeiten

## 3.3 RESTful Webservice

Zur Umsetzung des RESTful Webservice wird auf den Grizzly-Server aufgesetzt. Java-Methoden werden somit über HTTP-Operationen und Ressourcen angesprochen. Mit den von JAXB erstellten Klassen wird das marshalling und unmarshalling der XML-Dokumente durchgeführt. Mit den Methoden der Klassen wird dann das gewünschte Objekt XML-formatiert übertragen bzw. geändert.

### 3.4 XMPP Server und Client

#### 3.4.1 XMPP API

Ein XMPP Server wird mit Hilfe von Openfire eingerichtet, dazu wird die Anwendung gestartet und ein Admin angelegt. Nach dem erstellen des Server kann man nun mehrere Nutzer anlegen und Rechte verteilen. Als Schnittstelle zum XMPP-Server wird SMACK in der Version 3.3 genutzt.

#### 3.4.2 Publish-Subscribe Prinzip

Das Publish-Subscribe Prinzip beruht auf einer verbindungslosen Kommunikation zwischen unterschiedlichen Usern, wobei es immer nur einen „Redner“ (Publisher) und viele „Hörer“ (Subscriber) gibt. Die Kommunikation läuft über einen Server, welcher in der Mitte steht.
Ein Publisher erstellt einen sogenannten Node auf dem Server für den sich Subscriber anmelden können und an dem Publisher Nachrichten senden kann. Interessiert sich ein Subscriber für ein Thema (Node) so kann er sich für dieses Thema anmelden und bekommt alle Nachrichten vom Publisher.

Das Publish-Subscribe Prinzip beruht in unserem Programm auf einen Topic über welchen der Streamchannel mit dem User kommunizieren könnnen. Die User subscriben sich bei dem Streamchannnel und ein Streamchannel schickt allen Usern eine Nachricht, damit diese wissen das der Streanchannel online ist.
Aufgrund der asynchronen Architektur muss weder der Server auf die Antwort eines Streamchannels warten noch ein Streamchannel auf die Antwort eines Users um das Programm weiter zu nutzen.

### 3.5 Client Entwicklung

Mit Hilfe von JAVA Swing wird ein Interface erstellt, mit der es möglich ist die Funktionen zu testen. Es handelt sich hierbei nur um ein einfaches Interface das nur zu Testzwecken entwickelt wurde.