



Web-basierte Anwendungen 2: Verteilte Systeme

Sommersemester 2013

Projektdokumentation

Ticketsystem für IT-Problemstellungen

von

Oliver Steinle und David Ostolski

Dozent: Prof. Dr. Fischer

Betreuer: David Bellingroth und Renée Schulz, Christopher Messner

Inhaltsverzeichnis

• Einleitung	3
• Idee	4
• Konzept	5
○ Analyse der Kommunikationsabläufe und Interaktionen	
○ Beschreibung Struktur der Ressourcen	
○ Beschreibung der Funktionen der Module:	
• Projektspezifisches XML Schema	9
○ Kundenliste	
○ Supporterliste	
○ Ticketliste	
○ Ticket	
• Ressourcen und die Semantik der HTTP-Operationen	15
• RESTful Webservice	16
• XMPP Server	17
• Kommunikation zwischen REST und XMPP	19
• Kommunikation zwischen Client und XMPP	20
• Client Entwicklung	21
• Fazit	31
• Quellen	32
• Erklärung	33

Einleitung

Im Zusammenhang mit dem Modul Web-basierte Anwendungen 2: Verteilte Systeme
In der zweiten Projektphase geht es um die Aufgabe ein verteiltes System zu Entwickeln, welches über 2 grundlegende Kommunikationskonzeptionen Daten miteinander austauscht.

Es besteht der Anspruch eine auf REST aufgebaute synchrone Server-Client Kommunikation, mit einer asynchronen auf XMPP basierenden Kommunikation zu verbinden und diese als konkrete Problemstellung zu definieren.

In dem Zusammenhang eine Aufgabe zu Formulieren und diese, mit gegebenen Sprachen, und Standards zu Realisieren.

Die Software, also Schnittstellen zu den Servern sowie die Clients werden in Java geschrieben.

Dieses Dokument dient der dokumentarischen Begleitung des Projektes, es werden die Kommunikationsabläufe beschrieben, Benutzerfunktionen erläutert, Schemata der Datenstrukturen beschrieben und verdeutlicht.

Darüberhinaus werden in diesem Dokument getroffene Entscheidungen zu Lösungswegen begründet.

Aufbau und die Durchführung der des Projekts wurden anhand von Themen in Meilensteine Aufgeteilt. Danach richtet sich sowohl die Projektentwicklung als auch die Struktur der Dokumentation.

Kleine Abweichungen im Bezug zu Themen und Struktur der Meilensteine können aus Gründen der Realisierbarkeit vorkommen.

Idee

Ein Ticketingsystem in einer abgegrenzten Produktivumgebung. Dieses soll einem bestimmten Benutzerkreis die Möglichkeit geben IT-technische Supportanfragen an den jeweiligen Betreuer zu übermitteln.

Die Supporter bekommen einen „Alert“ also eine Benachrichtigung in ihrem Client über neu eingegangene Anfragen, und können direkt mit dem Benutzer in Kontakt treten(synchron).

Darüberhinaus kann der Supporter eine Lösung verfassen, diese dann an den Benutzer übermittelt wird(synchron).

Sind die Lösungen für eine Archivierung relevant werden diese in die Knowledge Base übergeben und können nach Stichworten abgerufen werden(optional(asynchron))

Supportanfragen die nicht vom jeweiligen Support-Benutzer bewältigt werden können, werden über Schwerpunkt-Tags je nach Wissensschwerpunkt an andere Supporter(Fachsupporter) weitergegeben. Die Kriterien der Fachgebiete werden vom System erfasst und verwaltet.

Darüberhinaus "Eskaliert" eine Supportanfrage wenn sie eine bestimmte Zeit überschritten hat und so als Erinnerungsfunktion fungiert. die Anfrage wird nach Ablauf der angegeben Zeit in Abhängigkeit der gewählten Priorität an alle anderen Supporter weitergeleitet.

Konzept

Analyse der Kommunikationsabläufe und Interaktionen:

Akteure:

Kunden:

- Sendet Anfrage an Support
- Empfängt Lösung

Supporter:

- Empfängt Anfragen vom Kunden
- Sendet Lösung an Kunden
- Sendet Lösung an Supporter
- Sendet Lösung an KB

Module:

- Clientmodul Kunden
- Clientmodul Supporter
- Clientmodul Admin
- Knowledgebase
- Server
- Statistik

Beschreibung der Funktionen der Module:

Clientmodul Kunden:

Das Clientmodul zeigt nach Öffnen des Programms die Liste aller vom jeweiligen Benutzer (Anmeldung erforderlich) offenen Supportanfragen und deren Status (Wartend, Wird bearbeitet, Gelöst) an. Per Mausklick auf die jeweilige Supportanfrage zeigt detaillierte Informationen zur Anfrage (Datum der Anfrage, Dringlichkeit, Problembeschreibung) diese lässt sich vom Benutzer ändern und löschen.

Clientmodul Supporter

Das Modul zeigt nach dem Öffnen des Programms die Liste. Er kann per Klick auf die Anfrage detaillierte Informationen (Kundendaten/Datum der Anfrage, Dringlichkeit, Problembeschreibung) anschauen, diese, falls Lösung vorhanden, beschreiben und die Anfrage samt Lösung an den Kunden zurückschicken. Darüberhinaus kann er dieser Anfrage die Beschreibung ergänzen und an einen bestimmten Fachsupporter/Abteilung weiterleiten.

Clientmodul Admin

Das Modul weißt alle Funktionalitäten von Kunde und Supporter auf und kann darüberhinaus noch Benutzer verwalten.

Knowledgebase

Sammelt Supportanfragen samt Lösung und Stichworten.
(Wurde leider nicht wie erhofft realisiert)

Struktur:

Beschreibung Struktur der Ressourcen:

Die Supportanfrage besteht aus:

- ID
- Vorname
- Nachname
- Standort
- Datum
- Beschreibung
- Themen-Tags
- Zustand

...und wird ergänzt durch den jeweiligen Supporter durch

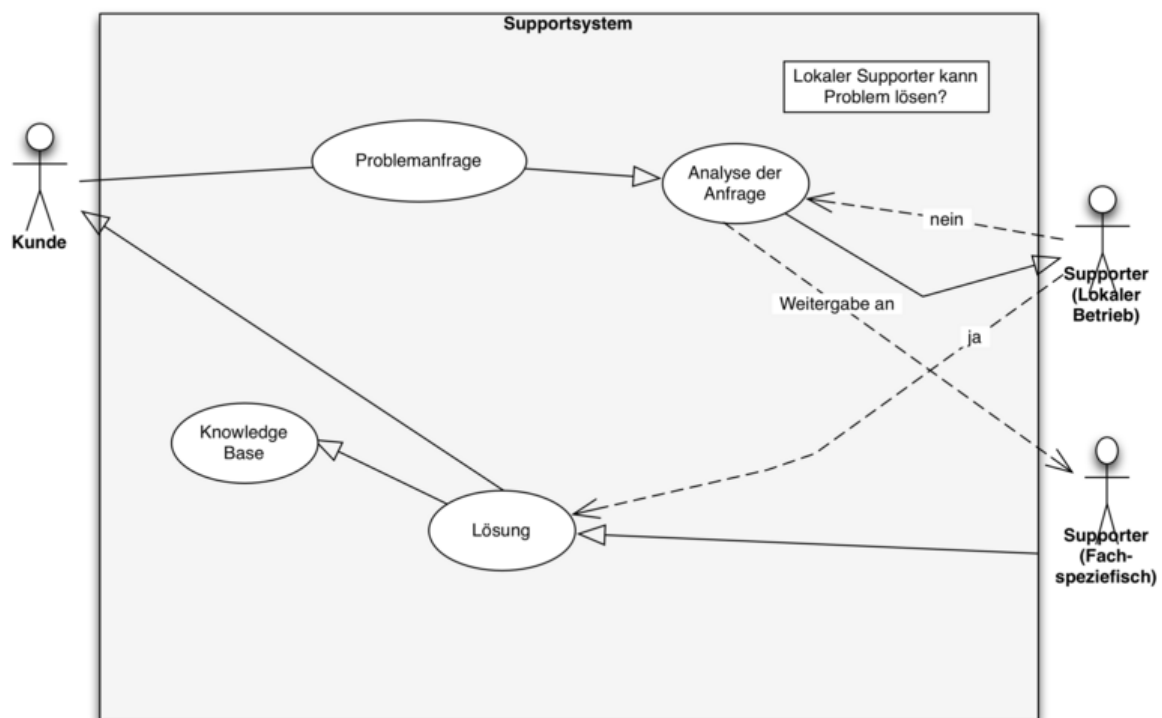
- Lösung
- Beschreibung des Lösungsansatzes (für die Weitergabe an Fachsupporter)

...und kann bei bedarf an die Knowledgebase übergeben werden mit folgenden Daten:

- Titel
- Beschreibung
- Lösung
- Themen-Tags
- Datum

Use-Case zur Datenübertragung

Ein Kunde sendet eine Supportanfrage(Synchron) an das Supportsystem die Daten aus dem Interface werden Gespeichert und synchron an den jeweiligen, den Kunden zuständigen, Supportmitarbeiter weitergeleitet. Der Supportmitarbeiter bearbeitet den Fall. Erkennt dieser, dass er nicht das notwendige Know-how besitzt um das Problem zu lösen, sendet er den Fall zurück an das Supportsystem. Dieses vergleicht die Tags mit der Liste der Fachmitarbeiter um den fall dann für diese Supporter wieder freizugeben, (Asynchron). Wenn ein Mitarbeiter diesen Fall beantworten kann, kann er diesen übernehmen wird er das Problem lösen und zurück an das System schicken. Darüberhinaus kann er den Fall für die Knowledgebase freigeben(Synchron). Nach Auswahl der Option schickt das System den relevanten Teil des Datensatzes an die Knowledgebase. Der Benutzer wird im Anschluss informiert das der Fall gelöst wurde.



Projektspezifisches XML Schema

Für das Ticketsystem werden folgende XML-Schemata Entwickelt

- Userliste
- Ticketliste
- Ticket

diese werden in 3 XML-Schemata abgebildet.

Jeder Kunde, Supportmitarbeiter bekommt eine ID und einen Benutzergrad zugewiesen. Ebenfalls bekommen Tickets eine Eindeutige ID und einen Zustand. Eine detaillierte Beschreibung der Datenstruktur:

Folgende Dateien Repräsentieren die XML-Schemata

UlisteShema.xsd

TListeShema.xsd

TicketShema.xsd

Der Schemata erstellt mit folgender Struktur:

Userliste:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
            elementFormDefault="qualified"
            targetNamespace="http://example.org/ticket"
            xmlns:tn="http://example.org/ticket">

    <xs:element name="userdb">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="user" type="tn:ct_profile"
minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="ct_profile">
        <xs:sequence>
            <xs:element name="id" type="xs:nonNegativeInteger"/>
            <xs:element name="jabber" type="xs:string"/>
            <xs:element name="Vorname" type="xs:string"/>
            <xs:element name="Nachname" type="xs:string"/>
            <xs:element name="Standort" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

Web-basierte Anwendungen 2: Verteilte Systeme

```
<xs:element name="status" type="xs:string"/>
<xs:element name="KnowHows" minOccurs="0">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="KnowHow"
type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="Tickets" type="tn:cttickets"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="cttickets">
  <xs:sequence>
    <xs:element name="ticketId"
type="xs:nonNegativeInteger" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
</xs:schema>
```

Ticketliste:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  targetNamespace="http://example.org/ticket"
  xmlns:tn="http://example.org/ticket">

  <xs:element name="ticketlist">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="teintrag" minOccurs="0"
maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ticket_id"
type="xs:nonNegativeInteger"/>
              <xs:element name="betreff"
type="xs:string"/>
              <xs:element name="datum"
type="xs:dateTime"/>
              <xs:element name="zustand"
type="tn:st_zustand"/>
              <xs:element
name="bearbeitungszustand" type="xs:boolean"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Web-basierte Anwendungen 2: Verteilte Systeme

```

        <xs:element name="tags">
            <xs:complexType>
                <xs:sequence>
                    <xs:element
name="tag" type="xs:string" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>

<xs:simpleType name="st_zustand">
    <xs:restriction base="xs:string">
        <xs:enumeration value="niedrig"/>
        <xs:enumeration value="normal"/>
        <xs:enumeration value="hoch"/>
        <xs:enumeration value="kritisch"/>
    </xs:restriction>
</xs:simpleType>
</xs:schema>
```

Ticket:

```

<xs:element name="Ticket">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="tn:ct_ticket">
                <xs:attribute name="id"
type="xs:nonNegativeInteger" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>

<!-- ComplexType zum Root-Element mit Zwei weiteren Kind-
Elementen vom Typ ComplexType -->
<xs:complexType name="ct_ticket">
    <xs:sequence>
        <xs:element name="Infos" type="tn:ct_info"/>
        <xs:element name="Antworten">
            <xs:complexType>
                <xs:sequence>
```

Web-basierte Anwendungen 2: Verteilte Systeme

```

        <xs:element name="Antwort"
type="tn:ct_antwort" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>

<!-- ComplexType zum Element "Infos"-->
<xs:complexType name="ct_info">
    <xs:sequence>
        <xs:element name="Betreff" type="xs:string"/>
        <xs:element name="User">
            <xs:complexType>
                <xs:simpleContent>
                    <xs:extension base="xs:string">
                        <xs:attribute name="id"
type="xs:nonNegativeInteger" use="required"/>
                    </xs:extension>
                </xs:simpleContent>
            </xs:complexType>
        </xs:element>
        <xs:element name="Standort" type="xs:string"/>
        <xs:element name="SupporterList">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="Supporter"
minOccurs="0" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:simpleContent>
                                <xs:extension
base="xs:string">
                                    <xs:attribute
name="id" type="xs:nonNegativeInteger" use="required"/>
                                </xs:extension>
                            </xs:simpleContent>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:complexType>
        </xs:element>

        <xs:element name="Datum" type="xs:dateTime"/>
        <xs:element name="Beschreibung" type="xs:string"/>
        <xs:element name="Zustand" type="xs:string"/>
    </xs:sequence>
</xs:complexType>
</xs:element>

```

Web-basierte Anwendungen 2: Verteilte Systeme

```
<xs:element name="Tags">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Tag"
type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>

<!-- ComplexType zum Element "Antwort" -->
<xs:complexType name="ct_antwort">
  <xs:sequence>
    <xs:element name="Supporter">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:string">
            <xs:attribute name="id"
type="xs:nonNegativeInteger" use="required"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="Antwort" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
</xs:schema>
```

Datentypen

Für die semantische Annotation wurden verschiedene Datentypen verwendet. Sowohl standardisierte Datentypen wie String und Integer als auch Komplextypes, diese aber aus Standarddatentypen bestehen. DateTime wurde beispielsweise verwendet um den XML-Gregorian-Calendar als Java Objekte verwendbar zu machen, der im weiteren Umgang mit zeitorientierten Methoden Einheitliche Datenverarbeitungen ermöglicht.

Restriktionen

Neben der Wahl der richtigen Datentypen besteht der Bedarf, diese mit den notwendigen Restriktionen zu versehen um Inkonsistenzen und Fehler bei Benutzerinteraktionen zu vermeiden. Dies verhindert Programmabstürze und garantiert den XML-Basierten Datensätzen eine inhaltlich korrekte Erfassung.

Web-basierte Anwendungen 2: Verteilte Systeme

Es wurden bestimmte Restriktionen vorgenommen jedoch bei manchen die Restriktion über den Client vorgenommen bzw. noch nicht implementiert vorgenommen um uns Änderungen und, Unschlüssigkeiten bei der Entwicklung im Teststadium offenzuhalten.

Eine Beispielliste um eine Vorstellung des Grades der Restriktionen zu erhalten.

Attribut/Element	Restriktion	Begründung
ID	Non negative(R)*	Vermeidet Inkonsistenz
Zustand	Enumeration	Müssen eindeutige Werte sein

(R) Required, muss bei neuem Datensatz angegeben werden

Ressourcen und HTTP-Operationen

Ressourcen beschreiben in dem Kontext Formalismen, diese via HTTP-Operationen abgerufen und manipuliert werden können.

Diese Ressourcen sind direkt von der XML-Struktur abgeleitet und repräsentieren einzelne Elemente in einem XML-Dokument. Aus der Analyse der notwendigen Elemente ergab folgende Schematisierung:

URI	Ressource	Methode
/user	Alle Benutzer auflisten	GET
/user/{id}	Benutzer aufrufen	GET
/user/{id}/ticket	Die vom Benutzer bearbeiteten Tickets aufrufen	GET
/user/add	Benutzer hinzufügen	POST
/user/{id}/delete	Benutzer löschen	DELETE
/ticket	Alle Tickets auflisten	GET
/ticket/{id}	Ticket aufrufen	GET
/ticket/add	Ticket erstellen	POST
/ticket/{id}/edit	Ticket bearbeiten	PUT
/ticket/{id}?zustand=wert	Priorität des Tickets ändern	PUT
/ticket/{id}/delete	Ticket löschen	DELETE

Die URI „/user“, „/user/{id}“, „/user/add“ „/user/delete“ beziehen sich auf das Dokument nach dem XML-Schema Kundenliste. User werden in einem XML-Dokument zusammengefasst.

Die URI „/user/{id}/ticket“ bezieht sich auf das Dokument: nach dem XML-Schema „Ticketliste“.

Es werden alle Tickets in der Ticketliste zusammengefasst bzw. indiziert.

Tickets werden nach dem XML-Schema Tickets in jeweils einem XML-Dokument gespeichert.

Die URI „/ticket/{id}“ bezieht sich auf das Dokument nach dem XML-Schema „Ticketliste“

Alle weiteren URIs die mit dem/ticket/ aufgerufen werden beziehen sich auf die jeweiligen Tickets und auf den jeweiligen Eintrag in der Ticketliste.

Die Ticketliste haben wir aus Gründen der Performance und der Realisierbarkeit zusätzlich integriert um große Übertragungsvolumina zu vermeiden.

RESTful Webservice

Der Restful Webservice wird als Server in form eines Java Plug-Ins Realisiert.
Im konkreten Einsatz steht das Grizzly Framework.

Es werden die bekannten HTTP-Operationen verarbeitet.
Über Die JAXB-Plugin werden Aus den XML-Dokumenten, Klassen in Java-Code Erzeugt
und können anschließend weiterverarbeitet werden.

Ausserdem Übernimmt der RESTfull Webservice die Zeit und Client gesteuerten Aufgaben,
analysier die Nutzungsdaten und Speichert diese.

Der Benutzer gibt im Client ein neues Problem ein, also die notwendigen Daten an wie:
Titel, Priorität, Fachthemen, Beschreibung.
Anschließend Sendet der Client die Daten via HTTP-Operationen an den REST-SERVER.
Dieser Analysiert die Daten und Speichert diese Als XML-Dokument ab.

XMPP Server

Der XMPP Server dient Verwaltung von asynchronen Nachrichten auf Basis von XML .
XMPP bedeutet „**Extensible Messaging and Presence Protocol**“ und ist nach RFC-6120
Veröffentlichter Internetstandart für XML-Routing

Zum Einsatz kommt ein der OpenFire der Firma JiveSoftware und steht unter der „Open
Source Apache license“ Lizenz.

Der XMPP Server Openfire ist eine komplexe aber leicht zu bediene und zu
konfigurierende Server Architektur mit vielen Funktionen und optionalen Plug-Ins.

In unserer Projektumgebung wird er für die Benutzerverwaltung verwendet, da die
gegeben Funktionen ausreichend Vorhanden sind.
Es lassen sich Benutzer und Zugangsbedingungen sowie Rechte und Benutzergruppen
verwalten.

Außerdem realisiert der Openfire die gesamte asynchrone Kommunikation.
Das bedeutet es werden Nachrichten von den anhand von Themen(Leafs) an die
jeweiligen Abonnementen(Subscriber) .

In der asynchronen Kommunikation werden folgende Daten übermittelt:

- Ticket ID
- Typ der Notifikation (Neues Ticket, wird Bearbeitet, wieder eingestellt)
- Zeitstempel
- Fachgebiet

z.B. wäre es auch Möglich gewesen den gesamten Payload über diesen Server zu Routen,
jedoch würde dadurch entweder redundante Datenhaltung entstehen oder nicht mehr
REST konform sein. Da jede Kommunikation über das Bestimmte Problem angehangen
und gespeichert wird, würde der Standartwert eines Payloads von 5120 Bytes
überschritten.

1 Notifikation entspricht in etwa 300 Bytes. (geringes Datenvolumen).

Web-basierte Anwendungen 2: Verteilte Systeme

Eine Tabellarische Aufstellung verdeutlicht die jeweiligen Publisher und Subscriber.

Ticketnodes:

Rolle	Subscriber	Publisher	Bemerkung/Begründung
Benutzer	X	X	
Admin	X	X	
Supporter	X	X	
Systemkonto			

Fachspezifische Nodes:

Rolle	Subscriber	Publisher	Bemerkung/Begründung
Benutzer			
Admin	X		
Supporter	X		
Systemkonto		X	Sendet zeitgesteuerte und Benutzers.

Kommunikation zwischen REST und XMPP

Wird z.B. von einem Benutzer ein neues Ticket erstellt, übernimmt der REST-Server alle weiteren Aktionen zur Verarbeitung des Falls.

Es wird geprüft welche Priorität das Ticket hat und an welche Supporter es weitergeleitet wird dann werden die Themen abgeglichen und an den XMPP-Server weitergeleitet. Dort werden dann die entsprechenden Supporter informiert.

Erstellt, ändert oder Löscht der Admin ein Fachgebiet sende der REST eine Anfrage an den XMPP den zugehörigen Node entsprechend zu behandeln.

Kommunikation zwischen Client und XMPP

Die in Java geschriebene JabberSmackAPI von Jive Software ist eine Schnittstelle, die es ermöglicht ohne großen Aufwand Verbindungen zum XMPP-Server aufzunehmen. Sie lässt sich einfach in ein Java Projekt implementieren, ein Auszug soll die kompakte Nutzbarkeit verdeutlichen:

Java Code (Ausschnitt):

```
-----  
//Baut die Verbindung zum XMPP auf  
public class JabberSmackAPI implements MessageListener{  
  
    static XMPPConnection connection;  
  
    public void login(String userName, String password) throws  
XMPPException  
    {  
        ConnectionConfiguration config = new  
ConnectionConfiguration("localhost",5222, "Work");  
        connection = new XMPPConnection(config);  
  
        connection.connect();  
        connection.login(userName, password);  
    }  
}-----
```

Benutzerverwaltung:

Bei erstellen eines Neuen Benutzers werden die Daten aus dem Client, also Benutzername und Passwort direkt an den XMPP-Server übermittelt.

Ein Algorithmus erzeugt aus dem Vornamen und dem Nachnamen eine Benutzerkennung in form von erster Buchstabe Vorname + Nachname zum Beispiel aus Max Mustermann entsteht mmusterman oder aus David Ostolski „dostolski“.

Client Entwicklung

Die Entwicklung des Client würde in Java mit dem Java Swing Framework entwickelt. Es wurde im Vorfeld keine große Skizzierung vorgenommen, da der Umgang mit grafischen Oberflächen erst einmal ausprobiert und die Funktionale Konzeption im Vordergrund stand. Usability und optische Ansätze wurden zu dem Zeitpunkt hinten Angestellt.

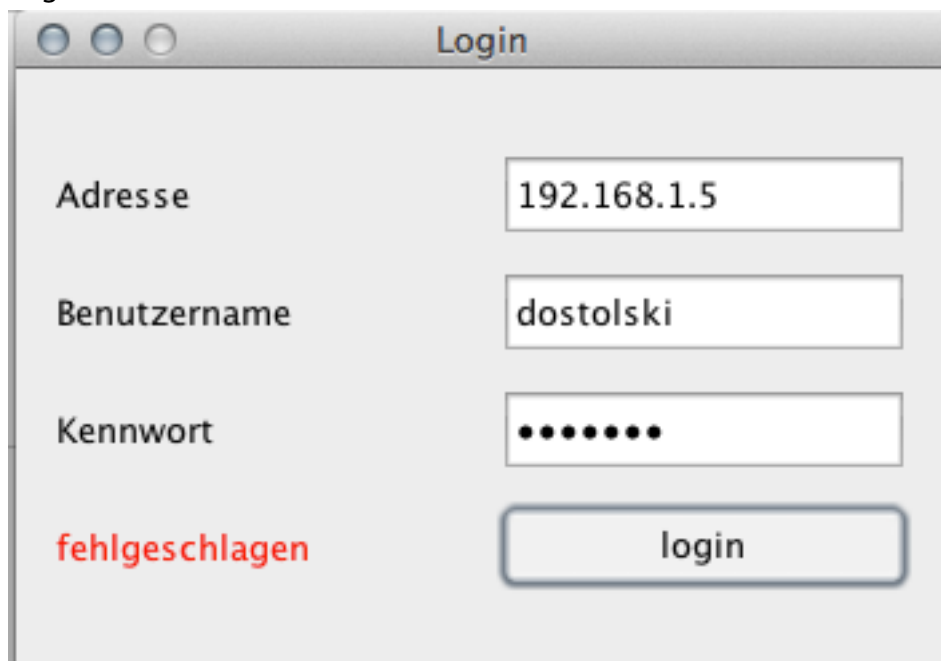
Der Client wurde Ursprünglich als drei rollenorientierte Module entworfen. Es stellt sich jedoch heraus, den Funktionsumfang je nach Benutzerrolle(Supporter, Benutzer, Admin) anzupassen.

Nach Eingabe der Benutzerdaten, also Benutzername und Passwort baut der Client die Verbindung zum XMPP-Server auf um die Benutzerrechte zu erfragen und dementsprechend Funktionen freizugeben.

Die Oberfläche besteht aus Jframes(Login und Hauptfenster). Nach dem Login verschwindet das Login Fenster und das Hauptfenster öffnet sich.

Über Schaltflächen kann man durch die einzelnen Frames(JFrames) navigieren.

Login Fenster:



The screenshot shows a Java Swing window titled "Login". It contains three text input fields. The first field, labeled "Adresse", contains the IP address "192.168.1.5". The second field, labeled "Benutzername", contains the username "dostolski". The third field, labeled "Kennwort", contains masked characters represented by seven dots. Below these fields, the text "fehlgeschlagen" is displayed in red, indicating a failed login attempt. A "login" button is positioned at the bottom right of the form area.

Beispielhaft für einen Login versuch. bei Fehleingabe erscheint eine Mitteilung in Rot, ansonsten wird in grün "Erfolgreich" angezeigt, das Fenster verschwindet und es öffnet sich das Hauptfenster.

Hauptfenster aus Benutzersicht:

Profil	
ID	3
Name:	David Ostolski
Standort:	Kölle
Level:	User
Anzahl Tickets:	1

Das Hauptfenster in dem „Benutzermodus“ zeigt die nur für ihn relevanten Informationen wie Benutzerprofil und Anzahl der offenen Tickets

Die Funktionalität beschränkt sich beim Benutzer auf: „Edit Profile“ dort kann der Benutzer Sein Profil bearbeiten,

„View Tickets“ Dort kann der Benutzer sehen welche Tickets offen sind und sich über den verlauf seines Problem bzw. dessen Lösung informieren sowie „New Ticket“ um ein neues Problem zu melden.

„Edit Profile“ Fenster

The screenshot shows a web application window titled "Ticket System". Inside, there is a form for editing a user profile. The form contains the following fields and controls:

- Vorname**: A text input field containing the value "David".
- Nachname**: A text input field containing the value "Ostolski".
- Standort**: A text input field containing the value "Kölle".
- Fachgebiete**: A section with two adjacent text areas. The left area is empty, and the right area contains the text "Hardware" and "Software" on separate lines.
- Buttons**: Two buttons are located below the "Fachgebiete" section: "Aktualisieren" (Update) and "Main Menu".
- Footer**: A horizontal bar at the bottom of the window is labeled "Benachrichtigung" (Notification).

Im Profil-Editor kann Der Benutzer seine Persönlichen Daten ändern.
Die Benutzerdaten sind in diesem Entwicklungsstadium auf das Verändern von Namen Und Standort reduziert.

Im unteren Bereich lassen sich die Fachgebiete für die Supporter verwalten.
(diese sind für Benutzer nicht sichtbar)

„New Ticket“ Fenster

The screenshot shows a window titled "Ticket System" with a light gray background. It contains the following elements:

- Betreff:** A text input field containing "Keine verbindung zum Email-Server".
- Priorität:** A dropdown menu showing "hoch" with a small arrow icon.
- Fachgebiet:** A dropdown menu with "Hardware" and "Software" (highlighted in green).
- Beschreibung:** A large text area containing the text "Ich kann mich seit Heute nicht meh auf dem Email-Server Anmelden".
- Buttons:** Two buttons at the bottom left labeled "Senden" and "Abbrechen".
- Footer:** A bar at the bottom labeled "Benachrichtigung".

Im „New Ticket“ Fenster lässt sich ein neues Ticket Erstellen Betreff gibt eine kurze Betreffzeile. Die Priorität via Dropdownmenu hat Einfluss auf die Verteilung und die zeitlich gesteuerten Alert innerhalb des Systems. Und die vollständige Beschreibung des Problems.

Hat man ein Ticket eingegeben und auf Senden gedrückt, wird der Benachrichtigungsbutton beim jeweiligen Supporter rot aufleuchten.
Settings:

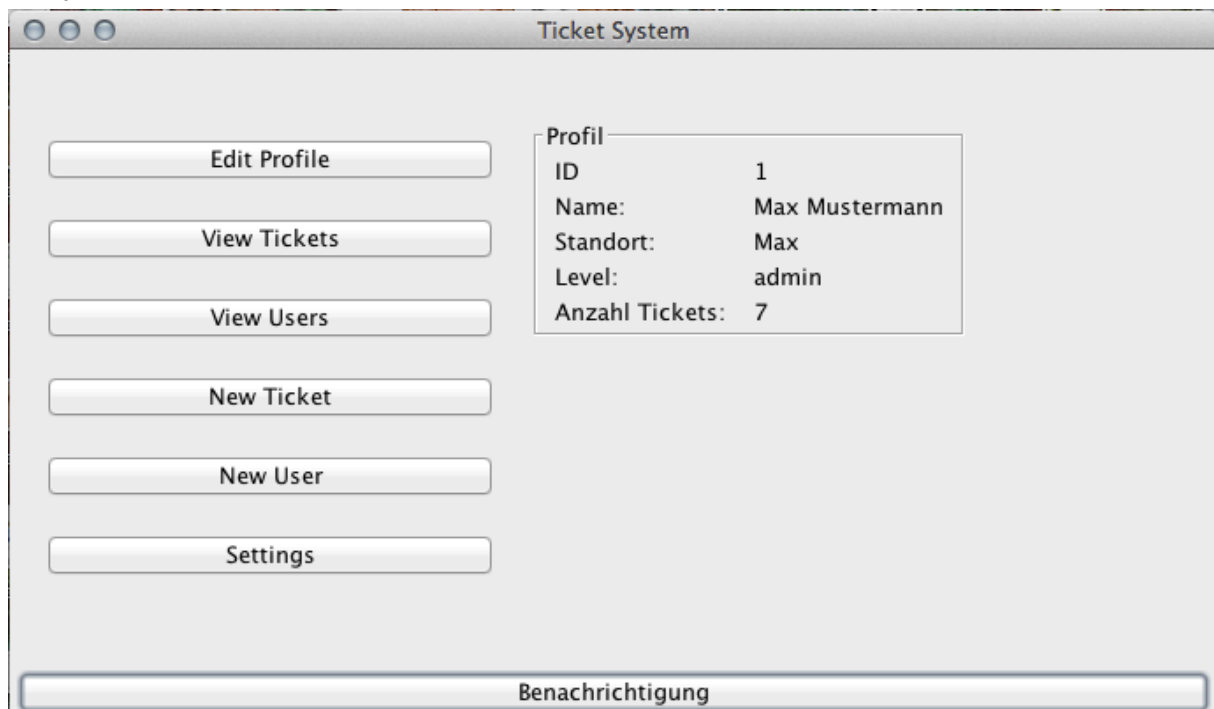
Benachrichtigungsfenster:



Ein Popupfenster öffnet sich sobald man auf den Benachrichtigungsbutton drückt und man sieht alle Tickets deren Aufmerksamkeit erforderlich ist.

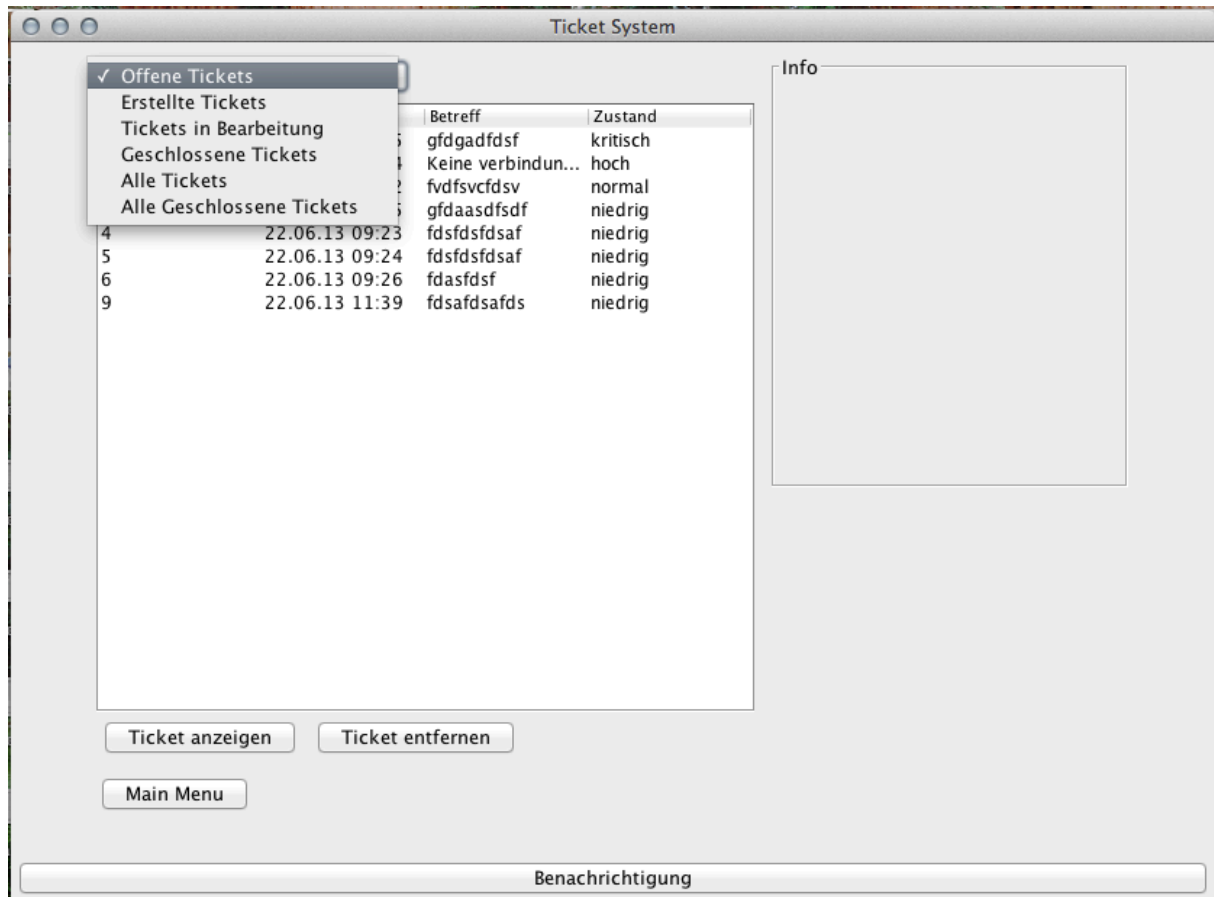
Web-basierte Anwendungen 2: Verteilte Systeme

Hauptfenster aus Administratorsicht:



Das Hauptfenster für die Administratoren erlaubt weitgehend alle Funktionen die Supportern und Benutzern zu Verfügung stehen. Darüberhinaus kann er Benutzer Anlegen und Löschen, sowie Einstellungen an dem System unter "Settings" Vornehmen (zu diesem Entwicklungszeitpunkt beschränkt sich die Funktionalität auf das erstellen von Themengebieten)

Ticketliste



Die Ticketliste stellt alle vorhandenen Tickets dar, und bietet die Möglichkeit Tickets nach bestimmten Kriterien zu Sortieren. Wählt man ein Ticket durch markieren des Eintrags aus kann man sich das Tickets Anschauen.

Web-basierte Anwendungen 2: Verteilte Systeme

Ticketview:

The screenshot shows a web browser window titled "Ticket System". Inside the window, there is a central area with a light gray background. On the left side of this area, there is a vertical list of labels: "Info", "Datum:", "Betreff:", "Ersteller:", "Standort:", "Fachgebiet:", "Zustand:", "Es bearbeitet für Sie:", "Beschreibung:", and "Ich kann mich seit Heute nicht meh auf dem Email-Server Anmelden". To the right of these labels, the corresponding values are displayed: "23.06.13 22:34", "Keine verbindung zum Email-Server", "David Ostolski", "Kölle", "Software", "hoch", and "Es bearbeitet für Sie:". Below this information, there are two buttons: "Main Menu" and "Ticket übernehmen". At the bottom of the window, there is a horizontal bar with the text "Benachrichtigung".

Info	
Datum:	23.06.13 22:34
Betreff:	Keine verbindung zum Email-Server
Ersteller:	David Ostolski
Standort:	Kölle
Fachgebiet:	Software
Zustand:	hoch
Es bearbeitet für Sie:	
Beschreibung:	
Ich kann mich seit Heute nicht meh auf dem Email-Server Anmelden	

[Main Menu](#) [Ticket übernehmen](#)

Benachrichtigung

Dort kann man das Ticket Übernehmen. Sobald man Das Ticket übernommen hat öffnet sich der Korrespondenzteil des Tickets.

Web-basierte Anwendungen 2: Verteilte Systeme

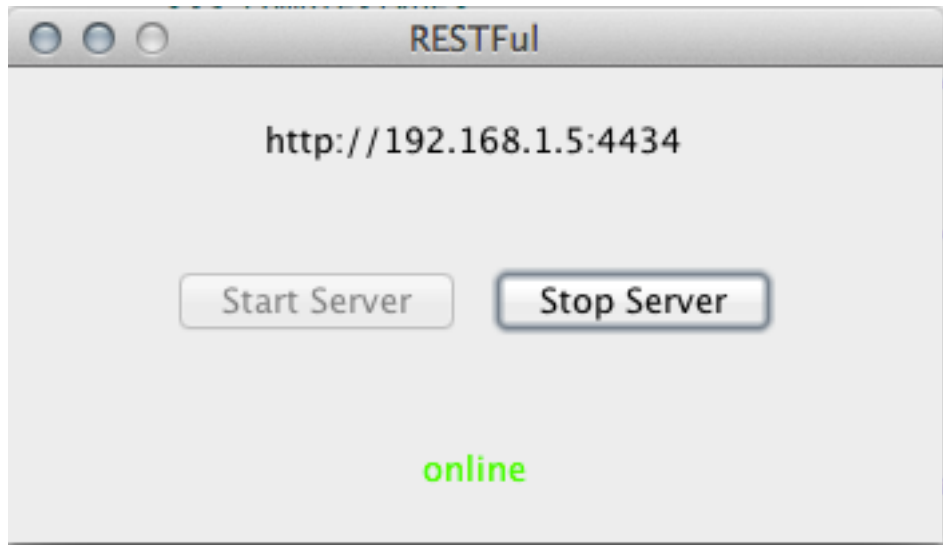
Ticketview mit Antwortpanel:

The screenshot shows a web browser window titled "Ticket System". Inside, there is a form with the following sections:

- Info:** A table-like structure with the following data:
 - Datum: 23.06.13 22:34
 - Betreff: Keine Verbindung zum Email-Server
 - Ersteller: David Ostolski
 - Standort: Kölle
 - Fachgebiet: Software
 - Zustand: hoch
 - Es bearbeitet für Sie: Max Mustermann
 - Beschreibung: Ich kann mich seit Heute nicht mehr auf dem Email-Server Anmelden
- Buttons:** Three buttons are located below the info section: "Main Menu", "Ticket freigeben", and "Ticket Schließen".
- Antworten:** A section titled "Antworten" with a sub-label "Antwort verfassen:". Below this is a large text input area containing the text "Benutzerkonto ist wieder freigeschaltet!".
- Buttons:** A "Senden" button is located at the bottom of the answer input area.
- Footer:** A bar at the bottom of the window labeled "Benachrichtigung".

Dort lässt sich die Lösung verfassen, das Ticket wieder Freigeben falls man nicht in der Lage es eine adäquate Lösung zu formulieren oder das Ticket Schließen falls die Lösung funktioniert oder das Problem nicht mehr besteht. Ist ein Ticket einmal geschlossen lassen sich keine Antworten mehr hinzufügen.

Server Gui:



Außerdem haben wir eine GUI entworfen um den REST-Server zu Starten und zu Beenden. Er besteht aus den Schaltflächen „Starten“ und „Beenden“ und zeigt die IP-Adresse des Servers.

Fazit

Die verwendeten Mittel zur Umsetzung dieses Projekts stellten sich als wirkungsvolle und umfangreiche Werkzeuge raus. Performance bei den Servern und den Clients waren im flüssig nutzbarem Bereich.

Grafische Aufbereitung von komplexen Datensätzen war ohne eine Markuplanguage wie z.B. HTML nicht ohne weiteres Umsetzbar gewesen, dafür an Funktionalität einen Grad erreichbar der die Idee hinter diesem Projekt, in vollem Umfang und Weit über den angesetzten Funktionalitätsumfang hätte erweitert werden koennen so z.B. lassen sich auch Multimediale Inhalte über XMPP nachladen. Auch lassen sich weitere Schnittstellen zu anderen Webservices integrieren.

Die Knowledgebase wurde aus zeitlichen Gründen nicht vollständig Implementiert und somit im Projekt nicht oder nur Ansatzweise Realisiert.

Es gab in der frühen Phase einige Hürden die es zu bewältigen gab:

Darunter trat besonders die Validierung der XML-Schemata hervor. Mehrmals mussten die Schemata neu erzeugt und Validiert werden da die JAXB-generierten Klassen eine geringe Abweichungstoleranz besitzt und eine Automatische Generierung verhindert hat. Und dadurch immer wieder zu Programm abstürzen führte.

Außerdem haben die Namespaces im Java Projekt Probleme mit JAXB geführt die, die exakte Deklaration des Namespaces vorausgesetzt hat.

Dagegen hatten wir sehr gute Erfahrungen mit der SMACK API der Firma Jive Software gemacht. Es ließen sich einfach, stabile Verbindungen mit dem XMPP-Server aufbauen.

Alles in allem war diese Veranstaltung ein aufschlussreiche und interessante Erfahrung. Es konnte mit komplexen Serverarchitekturen und Internetstandards experimentiert werden und lieferte einen Einblick in Abstraktions- und Modellierungsmaßnahmen in Bezug auf Verteile Systeme.

Quellen

Software/Module

- Jive Software: Openfire und SmackAPI
<http://www.igniterealtime.org/projects/openfire/>
Lizenz: Apache Software Lizenz
<http://www.apache.org/licenses/>
- JAXB
<https://jaxb.java.net/>
Lizenz: CDDL v1.1 und GPL v2.
- Projekt Grizzly (Grizzly Framework)
<https://grizzly.java.net/>
Lizenz: CDDL v1.1 und GPL v2.

Literatur:

- Brett McLaughlin: Java & XML 2. Auflage 2001, O'REILLY
- Joshua Marinacci & Chris Adamson: Swing Hacks, 1.Auflage 2005, -O'REILLY
- Stefan Tilkov: REST und HTTP, 2. Auflage 2011, dpunkt.verlag

Weblinks:

- <http://stackoverflow.com>
(Developer Community, hier wurden viele Fragen und Ansätze recherchiert)

Zum Projekt:

<https://github.com/ollsen/wba2SS13Phase2>

Erklärung

Diese Ausarbeitung wurde selbstständig verfasst und nur mit den angegebenen Hilfsmitteln erzeugt. Alle wörtlich oder sinngemäß den Schriften anderer entnommenen Stellen sind unter Angabe der Quellen kenntlich gemacht. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet und Programmcode.

Köln, den 23.06.2013

David Ostolski - Matrikelnummer:11074939

Oliver Steinle - Matrikelnummer 11075943