Fachhochschule Köln
Cologne University of Applied Sciences

Fachhochschule Köln

Fakultät für Informatik und Ingenieurwissenschaften

D O K U M E N T A T I O N

Webbasierte Anwendungen II

Verteilte Systeme

Vorgelegt an der Fachhochschule Köln
Campus Gummersbach
im Studiengang
<Medieninformatik>

ausgearbeitet von:
Julia Thyssen
(Matrikelnummer: 11076066)
und
Sheree May Sassmannshausen
(Matrikelnummer: 11075580)

Dozent: <Prof. Dr. Fischer>

Gummersbach, im <Juni 2013>

Inhaltsverzeichnis

Abbildungsverzeichnis **3**

1 Einleitung **4**

2 Konzept **4**

2.1 Projektidee 4

2.2 Funktionen 5

3 Ressourcen **6**

3.1 Bestellung 6

3.2 Bestellungsliste 6

3.3 Dauerauftrag 6

3.4 Börse 6

3.5 Börseneintrag 7

3.6 Betrieb 7

3.7 Betriebsliste 7

3.8 Person 7

3.9 Personenliste 8

3.10 Kunden- bzw. Lieferantenliste 8

3.11 Produkt 8

3.12 Produktliste 8

3.13 Bestand 8

4 Planung und Konzeption der XSD Schemata **9**

4.1 Planung 9

4.2 XSD-Schemata 10

4.2.1 Betrieb.xsd 10

4.2.2 Person.xsd 10

4.2.3 BörsenEintrag.xsd 10

5 HTTP-Operationen und URI **11**

5.1 GET 11

5.2 PUT 11

5.3 POST 12

5.4 DELETE 12

5.5 PathParam 12

5.6 QueryParam 12

6 URI Benennung der Ressourcen **13**

7 HTTP Operationen der Ressourcen **14**

8 RESTful Webservice **16**
8.1 Betrieb . . . 16
8.2 Person . . . 17
8.3 Produkt . . . 17
8.4 Bestellung . . . 17
8.5 Börse . . . 17

9 XMPP **19**
9.1 Leafs (Topics) . . . 19
9.2 Publish Subscribe . . . 20
9.2.1 Produktliste . . . 20
9.2.2 Bestellung . . . 20
9.2.3 Börse . . . 20
9.3 Umsetzung . . . 21
9.4 Welche Daten sollen mittels der Anwendung übertragen werden? . . . 21
9.5 XMPP-Client . . . 22

10 Client-Entwicklung **23**
10.1 Umsetzung der GUI mittels SWING . . . 23
10.1.1 Login Bereich . . . 23
10.1.2 Navigationskategorie: Personen . . . 23
10.1.3 Navigationskategorie: Betriebe . . . 23
10.1.4 Navigationskategorie: Börse . . . 24
10.1.5 GUI-Implementierung des Publisher . . . 24

11 Zusammenfassung **25**

12 Quellenverzeichnis **26**
12.1 Literatur . . . 26
12.2 Internetquellen . . . 26

Erklärung über die selbständige Abfassung der Arbeit **28**

Abbildungsverzeichnis

1 Informationsflussdiagramm anhand eines Beispiels einer Bestellung 9
2 Publish-Subscribe Prinzip 19
3 Publish-Subscribe Prinzip mit Vermittler 21

1 Einleitung

Im Rahmen der Veranstaltung 'Webbasierte Anwendungen 2' geleitet von Prof. Fischer sollte eine webbasierte Anwendung für ein verteiltes System konzipiert, geplant und in einzelnen vorgegebenen Schritten (Meilensteinen) umgesetzt werden.
Das Projekt sollte in zweier Teams durchgeführt werden.
Dieses Dokument befasst sich mit der Dokumentation des Systems und erläutert die zu den einzelnen Meilensteinen erforderlichen Endergebnissen. Zusätzlich zu diesem Dokument beinhaltet das Projekt ein Logbuch, indem der Werdegang, Abwägungen und Probleme des Projektes in chronologischer Reihenfolge dokumentiert sind. Das Projekt wurde in Eclipse erstellt und in GitHub veröffentlicht.

2 Konzept

2.1 Projektidee

Zu Beginn des Projektes musste eine Projektidee entwickelt werden, welche sich als verteiltes System mit synchronen und asynchronen Kommponenten realisieren lässt. Das im Folgenden beschriebene Projekt handelt von einem System, welches im Grunde auf alle Kunden-Lieferanten-Beziehnungen anwendbar ist. Es befasst sich mit der Abwicklung von Bestellungen und der Kommunikation zwischen Lieferant und Kunde. In Bezug darauf spielt die Börse des Systems eine wichtige Rolle.
Um detaillierter und branchenspezifischer zu arbeiten, wurde das System für den Gastronomie-Bereich realisiert, in dem Lieferanten und Gastronome miteinander interagieren. Allerdings kann das System auch in anderen Bereiche zum Einsatz kommen wie beispielsweise in Supermärkten,Apotheken etc.
Die Funktionen des Systems werden in synchrone und asynchrone Komponenten unterteilt.

Synchrone Komponente:
Um eine Bestellung zu tätigen, hat ein potentieller Kunde die Möglichkeit eine Liste aller zum Verkauf zu Verfügung stehenden Produkte samt Preisangaben anzufordern. Diese Anfrage benötigt keine Interaktion zwischen Kunde und Lieferant und kann somit synchron bearbeitet werden.

Asynchrone Komponenten
Kommt es zu einer Bestellung muss der Lieferant seinen Bestand prüfen, die Existenz des gewünschten Produkt sicherstellen und dem Kunden zu bestätigen. Ist dies nicht möglich, bekommt der Gastronom sofort eine Meldung sobald das gewünschte Produkt wieder vorhanden ist. Damit der Lieferant zu jeder Zeit die Kontrolle über seinen Lagerbestand

behält, kann keine Bestellung eines Produktes ohne die Interaktion mit dem Lieferanten getätigt werden und dem Lieferanten ist es immer möglich eine Angabe bezüglich der Verfügbarkeit von Produkten zu machen.
Zudem hat der Lieferant die Möglichkeit seinen Kunden Angebote und neue Produkte vorzustellen oder aber der Gastronom erstellt eine Anfrage über ein spezielles Produkt, auf diese verschiedene Lieferanten und Supermärkte reagieren können.

2.2 Funktionen

Im Folgenden werden die vom System bereitgestellten Funktionen erläutert.
Zunächst hat ein Kunde die Möglichkeit von einem oder mehreren Lieferanten eine Produktliste anzufordern. Daraufhin kann Dieser eine Bestellung über die gewünschten Produkte aufgeben, woraus die Verfügbarkeit der einzelnen Produkte ersichtlich wird.
Besteht ein regelmäßiger Bedarf an den Produkten können bei dem jeweiligen Lieferanten Daueraufträge erstellt werden. Sollte ein Produkt wider Erwarten nicht zur Verfügung stehen, so kann der Lieferant seinen Kunden informieren lassen, sobald das gewünschte Produkt wieder verfügbar ist.
Mit Hilfe der Börse wird das System noch interaktiver: Zum Einen haben Lieferanten die Möglichkeit Angebote zu erstellen und Diese nach Kategorien einzuordnen und zum Anderen können Gastronome Gesuche aufgeben, auf die sich Lieferanten oder auch andere Gastronome melden können. Auf ein Börseneintrag kann man Kommentare verfassen oder aber auch direkt aus dem Angebot heraus bestellen.

3 Ressourcen

Um einzelne XSD-Schemata erstellen zu können, mussten die einzelnen Ressourcen definiert werden. Anhand der Ressourcen und der jeweiligen Definition der einzelnen Ressource kann eine erste Struktur im Aufbau des Systems erstellt werden. Innerhalb des Systems soll es 14 einzelne Ressourcen geben.

3.1 Bestellung

Die Bestellung ist der Kern und die Hauptaktion des Systems.
Eine Bestellung wird durch mehrere eigene Eigenschaften und durch andere Ressourcen definiert. Jede Bestellung bekommt eine eigene ID um eine eindeutige Zuweisung zu ermöglichen. Des Weiteren wird festgelegt, ob eine Bestellung ein einmaliger Auftrag sein soll oder ob es sich um einen Dauerauftrag handelt. Zusätzlich können die Zahlungsart ausgewählt und der Gesamtbetrag der Rechnung ausgegeben werden. Um Angaben bezüglich des Kunden und des Lieferanten zu ermöglichen, wird die Ressource Person und somit auch die Ressource Betrieb verwendet.

3.2 Bestellungsliste

In der Bestellungsliste werden alle Bestellungen inklusive Daueraufträge aufgelistet. Dadurch können Kunde und Lieferant einen Überblick über alle Bestellungen erhalten. In einer Bestellungsliste werden nur die eigenen Bestellungen aufgelistet, sodass diese nicht für alle Personen des Systems öffentlich besteht, sondern nur für den jeweiligen Kunden und Lieferanten.

3.3 Dauerauftrag

Die Ressource Dauerauftrag kommt dann zum Einsatz, wenn eine Bestellung regelmäßig getätigt werden soll.

3.4 Börse

Die wichtigste Ressource des Systems ist neben der Bestellung an sich die Börse. Hier können Angebote und Gesuche eingetragen werden. Diese Einträge können von allen im System registrierten Personen gelesen und kommentiert werden. Zusätzlich können die in der Börse angebotenen Produkte direkt bestellt werden. Bestimmte Kategorien und deren Beiträge können innerhalb der Börse abonniert werden. Somit kann ein Kunde oder Lieferant alle Einträge eines bestimmten Lieferanten bzw. Kunden automatisch erhalten.

3.5 Börseneintrag

Die in der Börse aufgelisteten Börseneinträge werden durch eine individuelle Börseneintrags ID gekennzeichnet. Zusätzlich besitzt auch jeder Kommentar eines Beitrages eine ID. Die ID's sollen ermöglichen, dass die Beiträge und Kommentare eindeutig sind.

3.6 Betrieb

Die Ressource Betrieb definiert jeden im System bestehenden Betrieb unabhängig um was für einen Betrieb es sich handelt. Ein Betrieb wird durch eine BetriebsID gekennzeichnet, um einen eindeutigen Zugriff auf jeden einzelnen Betrieb zu ermöglichen. Desweitern wird ein Betrieb durch seinen Namen, die Adresse und die Art des Betriebes, beispielsweise Italienisches Restaurant oder Lebensmittelgroßhandel, definiert.

3.7 Betriebsliste

Um alle Betriebe in einer Ressource zusammenfassen zu können, existiert eine Ressource Betriebsliste. Diese beinhaltet eine Übersicht mehrerer Instanzen der Ressource Betrieb und weist keine zusätzlichen Eigenschaften auf. Sobald ein Betrieb erstellt worden ist, erscheint Dieser in der Betriebliste.

3.8 Person

Als Person gelten alle im System registrierten Personen, egal ob Lieferant oder Gastronom(Kunde). Die Ressourcen Gastronom und Lieferant wurden bewusst zusammengefügt, da sich diese von den Angaben innerhalb der Ressource nicht unterscheiden. Ein weiterer Grund für die Zusammenfügung besteht darin, dass ein Gastronom nicht ausschliesslich nur als Kunde fungieren und ein Lieferant nicht nur als Lieferant. Somit können Gastronom und Lieferant gleichzeitig Kunde und Lieferant sein. Bei einer Trennung der beiden Ressourcen müsste diese Person doppelt aufgelistet werden. Somit wird die Eigenschaft, ob es sich um einen Gastronom oder Lieferanten handelt als Typangabe der Person definiert.
Eine Person darf grundsätzlich nicht jünger als 18 Jahre sein, da erst ab dem 18. Lebensjahr rechtmäßige Kaufabwicklungen möglich sind.
Zusätzlich wird eine Person durch die Ressource Betrieb (siehe 3.6) definiert. Somit wird eine Verbindung zwischen den beiden Ressourcen erstellt. Es wäre möglich, auch die Ressource Betrieb durch die Ressource Person zu definieren, allerdings würde dies zu einer doppelten Abhängigkeit der beiden Ressourcen von einander führen, wodurch keine der beiden Ressourcen eindeutig definierbar wäre.

3.9 Personenliste

Die Ressource Personenliste agiert ähnlich wie die Ressource Betriebliste, in dem sie die Auflistung aller bestehenden Personen des Systems ermöglicht. Zusätzlich zu den aufgelisteten Personen besitzt diese Ressource keine eigenen Eigenschaften und auch hier erscheinen alle Personen in der Personliste sobald Diese erstellt worden sind.

3.10 Kunden- bzw. Lieferantenliste

Die Ressourcen Kundenliste bzw. Lieferantenliste wurden erstellt, um trotz der fehlenden Trennung von Kunde und Lieferant seperate Listen erstellen zu können. Somit soll es möglich sein, jeweils eine Liste mit allen Kunden und eine Liste mit allen Lieferanten anschauen zu können. Somit kann beispielsweise jeder Kunde eine Liste mit all seinen Lieferanten erhalten. Zu jedem Lieferanten werden dann die Angaben über die Ressource Person(siehe 3.8) und über die bereits getätigten Bestellungen(siehe 3.1) bereitgestellt.

3.11 Produkt

Ein Produkt wird über eine eindeutige ID und dessen Produktangaben wie Produktname, Preis, Verfügbarkeit definiert.

3.12 Produktliste

In der Produktliste werden alle zum Verkauf zur Verfügung stehenden Produkte samt Preisangabe aufgelistet. In einer Produktliste kann zu den jeweiligen Produkten eine Verfügbarkeit angegeben werden, dies ist jedoch nicht zwingend erforderlich. Die Produktliste soll dem potentiellen Kunden bei der Auswahl der benötigten Produkte helfen. Zusätzlich werden dem Kunden über die Ressource Person(siehe 8.2) Informationen über den Lieferanten und über die Ressource Betrieb (siehe 8.1) Angaben zum zugehörigen Betrieb bereitgestellt.

3.13 Bestand

Um einen Überblick über das Lager und die einzelnen Produktverfügbarkeiten zu ermöglichen, kann jeder Lieferant seine Produkte samt Mengenangaben etc. in der Ressource Bestand speichern. Somit beinhaltet die Ressource Bestand mehrere Instanzen der Ressource Produkt (siehe 3.11). Andersherum kann aber auch der Gastronom seinen aktuellen Bestand einspeichern, sodass Dieser einen Überblick erhält und weiß wann er die nächsten Produkte bestellen muss insofern Diese bald aufgebraucht sind.

4 Planung und Konzeption der XSD Schemata

Im ersten Meilenstein wurde sich mit der Planung und Konzeption der XML-Schemata befasst. In diesem Kapitel wird die Planung der einzelnen XML-Schemata beschrieben, die erstellten Ressourcen erläutert und die letztendliche Umsetzung der Schemata definiert.

4.1 Planung

Für die Planung wurde zunächst ein Informationsflussdiagramm erstellt, welches anhand des Beispieles einer Bestellung die synchronen und aysnchronen Abläufe innerhalb des Systemes darstellen soll.

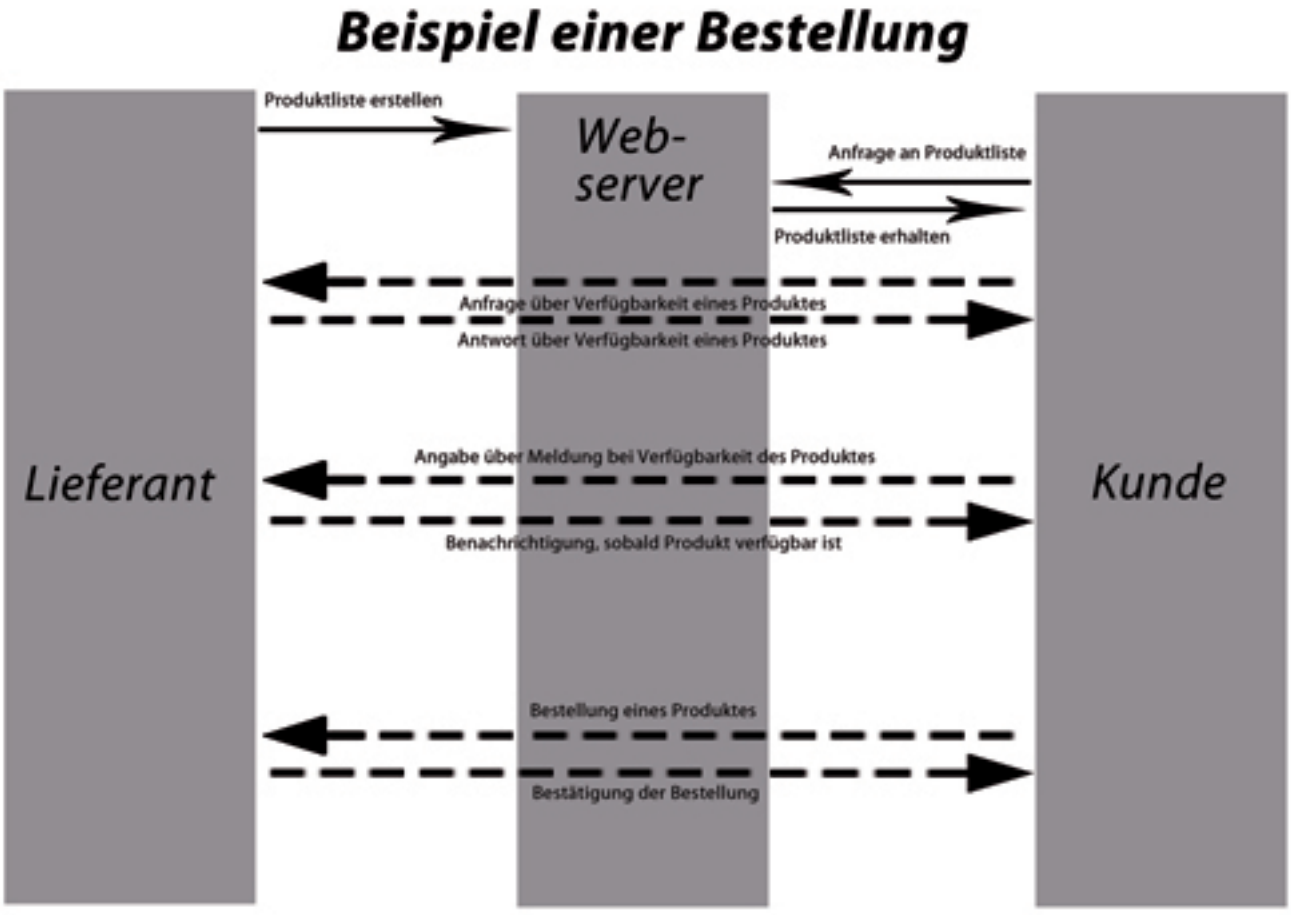

Abbildung 1: Informationsflussdiagramm anhand eines Beispiels einer Bestellung

Die synchrone Komponente des Systems ergibt sich durch den Lieferanten, der seine Produktliste im Webserver speichert, die von den Gastronomen jederzeit sofort abrufbar ist. Als Beispiel der asynchronen Komponente fragt der Gastronom, ob ein bestimmtes Produkt vorhanden ist. Der Lieferant selbst hat diese Anfrage zu bestätigen oder abzulehnen. In diesem Fall ist das Produkt nicht vorhanden und es folgt eine Benachrichtigung sobald das Produkt verfügbar ist und die Möglichkeit besteht Dieses zu bestellen. Der Gastro-

nom bestellt dieses Produkt direkt beim Lieferanten, woraufhin Dieser eine Bestätigung erteilt.

4.2 XSD-Schemata

Die Erstellung der XSD-Schemata konnte anhand der aufgelisteten Ressourcen erfolgen, die im nächsten Kapitel näher erläutert werden. Im Folgenden sollen als Beispiel drei XSD-Schemata anhand ihres Aufbaus erläutert werden, wobei die Struktur sich in den übrigen Ressourcen nicht wesentlich ändert.

4.2.1 Betrieb.xsd

Ein Betrieb besitzt die anschliessenden Elemente:
- Betriebs-ID: durch ein minOccurs='1' und ein maxOccurs='1' wird erreicht, dass pro Betrieb nur eine ID existiert
- Betriebname: durch ein minOccurs='1' und ein maxOccurs='1' wird erreicht, dass für einen Betrieb auch nur ein Name existiert
- Adresse: Eine Adresse wird durch Strasse, Hausnummer, Ort und Postleizahl definiert. Eine 'Restriction' bei der Postleitzahl verhindert die Eingabe von ungültigen Postleitzahlen.
- Art des Betriebes: beispielsweise 'Italiener', 'Mexikaner' oder Großhandel

4.2.2 Person.xsd

Innerhalb des XSD-Schema wird eine Person durch folgende Elemente definiert:
- Person-ID: diese wird mit einem minOccurs='1' und einem maxOccurs='1' definiert um eine eindeutige ID festzulegen
- Typ : für die Angabe des Typen der Person stehen durch eine 'Restriction'(= Einschränkung) die Optionen Lieferant oder Gastronom zur Verfügung
- Vorname
- Nachname
- Alter : durch eine 'Restriction' können beim Alter nur Zahlen über 18 angegeben werden
- Betrieb : Eine Referenz auf Betrieb ermöglicht die Angabe eines bereits definierten Betriebes

4.2.3 BörsenEintrag.xsd

- Börseneintrags-ID: durch ein minOccurs='1' und ein maxOccurs='1' wird erreicht, dass pro Eintrag nur eine ID existiert
- Person: Eine Referenz auf 'Person' liefert Angaben über den Verfasser des Eintrages
- Erstellungsdatum

- Titel
- Informationstext: Inhalt des Eintrages
- Startdatum: bei Angeboten
- Ablaufdatum: bei Angeboten
- Kommentare: Kommentare werden durch eine eindeutige Kommentar-Id, das Erstellungsdatum, den Verfasser und den eigentlichen Kommentar definiert
- Typ: für die Angabe des Typen des jeweiligen Eintrages stehen durch eine 'Restriction' die Optionen Angebot oder Gesuch zur Verfügung

5 HTTP-Operationen und URI

Eingangs der zweiten Phase sollte eine theoretische Auseinandersetzung mit HTTP-Operationen und Ressourcen im Kontext RESTful Webservices erfolgen:
Insgesamt gibt es acht definierte HTTP-Operationen:
- GET
- HEAD
- PUT
- POST
- OPTIONS
- TRACE
- CONNECT.

Für die Phase 2 werden zunächst nur die Operationen GET, PUT, POST und DELETE verwendet.

5.1 GET

'Gemäß der Spezifikation dient GET dazu, die Informationen[...] in Form einer Entity abzuholen'.
Die GET-Operation ist eine Leseoperation und liest den Inhalt einer URI aus. Des weiteren besitzt diese Operation die Eigenschaft 'sicher' was bedeutet, dass diese den sichtbaren Zustand eines Servers nicht verändert. Zusätzlich ist sie 'idempotent' und lässt den Server auch bei wiederholtem Aufruf im gleichen Zustand. (Quelle:12.2)

5.2 PUT

PUT wird verwendet, um eine Ressource anzulegen oder sie zu überschreiben. 'Ein PUT wirkt sich direkt auf die Ressource aus, deren URI Ziel des Requests ist.' Auch die PUT-Operation ist ïdempotent". (Quelle:12.2)

5.3 POST

Für das Anlegen einer URI und das Anhängen von Daten an die URI wird die POST-Operation verwendet. ‘Im weiteren Sinne wird POST für alle die Zwecke eingesetzt, in denen keine der anderen Methoden passt, d.h. immer dann, wenn eine beliebige Verarbeitung angestoßen werden soll.’ (Quelle:12.2)

5.4 DELETE

Eine Ressource kann durch die DELETE-Operation gelöscht werden. Dadurch, dass mehrmaliges Löschen einer Ressource den selben Effekt hat wie einmaliges, besitzt die DELETE-Operation die ’idempotent’-Eigenschaft. (Quelle:12.2)

5.5 PathParam

Path Parameter werden verwendet, um die Parameter der Methode an ein Path Segment anzuhängen. Inerhalb des RESTful Webservices wurden Path Parameter immer dann verwendet, wenn die URI erweitert werden musste um die gewünschte Operation auszuführen. Somit ist ein Path Parameter beispielsweise bei den DELETE-Methoden zu finden.

5.6 QueryParam

Innerhalb des Webservices sollte laut Aufgabenstellung mindestens einmal ein ’QueryParam’ verwendet werden. Query Parameter sind in der Implementierung nicht notwendig, da diese beispielsweise in der Suche verwendet werden.
In der Theorie gibt es innerhalb des Systems zwar auch eine Suche, die beispielsweise in der Kunden-/Lieferantenliste oder auch in der Produktliste eingesetzt wird, aber diese Funktion haben ist aus Zeitgründen leider nicht implementieren worden, da sie nicht zu den Kernfunktionen der Anwendung gehört.
Die Query Parameter würden sich dann in der URI nach einem ‘?’ äußern, die den eingegebenen Suchbegriff beinhalten.

6 URI Benennung der Ressourcen

Ressourcen	URI
Angebot	OrderHero/Boerse/EintragID=
Bestand	OrderHero/Bestand
Bestellung	OrderHero/Bestellungen/BestellungsID=
Bestellungsuebersicht	OrderHero/Bestellungen
Betrieb	OrderHero/Betriebsliste/BetriebsID=
Betriebsliste	OrderHero/Betriebsliste
Boerse	OrderHero/Boerse
Daueraufträge	OrderHero/Daueraufträge
Gesuch	OrderHero/Boerse/EintragID=
Kunde	OrderHero/Kundenliste/PersonID=
Kundenliste	OrderHero/Kundenliste
Lieferant	OrderHero/Lieferantenliste/PersonID=
Lieferantenliste	OrderHero/Lieferantenliste
Personenliste	OrderHero/Personenliste
Produkt	OrderHero/Produktliste/ProduktID=
Produktliste	OrderHero/Produktliste

Einige URI's werden erst zur Laufzeit der Anwendung eindeutig identifiziert. Denn einige Elemente bekommen automatisch ihre ID zugeordnet wie etwa ein Börseneintrag, eine Person oder ein Betrieb. Wenn man jeweils auf diese Ressourcen klickt, enthält die URI die jeweils generierte ID. Im allgemeinen Sinne aber sind in diesen Tabellen die URI's festgelegt. Bei Erstellung eines Börseneintrags gibt man den jeweiligen Typ an, somit kann ein Börseneintrag zwei verschiedene Formen von Ressourcen annehmen: ein Angebot oder ein Gesuch. Die URI dafür ist vom Aufbau her gleich, es ändert sich lediglich die jeweilige ID in der URI. Die verschiedenen Listen sowie die Börse stellen eine Übersicht der jeweiligen Ressourcenelemente dar. So beinhaltet die Börse eine Übersicht aller Börseneinträge, sprich aller Angebote oder Gesuche, die aber jeweils je nach Wunsch gefiltert werden können. So kann man alle Gesuche oder alle Angebote anzeigen lassen, aber man hat auch die Möglichkeit eine bekannte ID einzugeben und so den jeweiligen Eintrag aus der Börsenliste heraus anzuzeigen.

7 HTTP Operationen der Ressourcen

Ressourcen	Operation	Beschreibung
Angebot	GET,PUT, DELETE	Angebot lesen bzw. aufrufen/Angebot erstellen/Angebot löschen
Bestand	GET,POST,PUT,DELETE	Bestand lesen bzw. aufrufen,Produkte ansehen/Neue Produkte hinzufügen/Eigenschaften bestimmter Produkte ändern/Produkte von Liste löschen, Produktliste löschen
Bestellung	GET,POST	Bestellung aufrufen/Bestellung hinzufügen
Bestellungsuebersicht	GET	Bestellungen aufrufen
Betrieb	GET,POST,PUT,DELETE	Betriebsinformation lesen bzw. aufrufen/Betriebsinformationen einspeichern/Informationen ändern/Betrieb löschen
Betriebsliste	GET	Betriebinformation lesen bzw. aufrufen
Boerse	GET	Gesuch/Angebot lesen bzw. aufrufen
Daueraufträge	GET,POST,PUT,DELETE	Dauerauftrag aufrufen/Dauerauftrag hinzufügen/Dauerauftrag ändern/ Dauerauftrag löschen
Gesuch	GET,POST, DELETE	Gesuch lesen bzw. aufrufen/Gesuch/Kommentar erstellen/Gesuch/Kommentar löschen
Kunde	GET,POST,PUT,DELETE	Kundeninformation lesen bzw. aufrufen/Kundeninformationen einspeichern/Informationen ändern/Kunden löschen
Kundenliste	GET	Kunden aufrufen (Kunde wird automatisch durch Bestellung in seine Liste aufgenommen)
Lieferant	GET,POST,PUT,DELETE	Lieferantinformation lesen bzw. aufrufen/Lieferantinformationen einspeichern/Informationen ändern/Lieferant löschen
Lieferantenliste	GET	Lieferanten aufrufen (Lieferanten werden automatisch bei Bestellung hinzugefügt)

Personenliste	GET	Personen aufrufen (alle angemeldeten User erscheinen automatisch in Liste)
Produkt	GET,POST,PUT,DELETE	Produkt aufrufen/Produkt hinzufügen/Produkt ändern(zB bei Preisaenderung)/Produkt löschen
Produktliste	GET	Produktliste aufrufen

Auffallend ist, dass sich quasi alle Listen automatisch generieren sobald einzelne Ressourcen angelegt worden sind. So gelangen zum Beispiel Gastronomen oder Lieferanten, die sich anmelden automatisch in einer Übersicht: in der Kunden- oder Lieferantenliste.
Alle Listen werden mittels GET aufgerufen und weisen keine weiteren HTTP-Operationen auf, da nur Änderungen oder Löschungen einzelner Ressourcen möglich sind.

8 RESTful Webservice

REST (Representational State Transfer) stellt ein einheitliches Konzept für statische Inhalte und dynamisch berechnete Informationen dar. Dahinter liegt eine konkrete Architektur, die HTTP zugrunde liegt.
Der Schwerpunkt liegt dabei auf Konzepte anstatt auf Syntax.
'Im Sinne von REST gibt es 5 Kernprinzipien:
-> Ressourcen mit eindeutiger Identifikation (mittels URI)
-> Verknüpfungen/Hypermedia
-> Standardmethoden
-> unterschiedliche Repräsentationen
-> statuslose Kommunikation
man unterscheidet zwischen Ressourcen und ihren Repräsentationen.
REST ist für die Kommunikation zwischen Systemen gut' und außerdem zustandslos, somit gibt es kein Session Protokoll.

Wie sieht der konkrete Entwurf für REST aus?
es beginnt mit der Bestimmung der Ressourcen, die das System ausmachen sowie deren Verknüpfungen untereinander.
Gruppierungen werden als Ressourcen modelliert, die Verweise auf die enthaltenen Elemente beinhalten.
Der Einstieg in den Dienst soll möglichst durch eine einzige URI erfolgen, über die Verweise auf weitere Ressourcen erreichbar sind und für jede Ressource wird festgelegt welche HTTP Methoden diese unterstützt und welche Semantik sich dahinter verbirgt.
Hinter einer URI kann sowohl eine maschinell verarbeitbare XML Darstellung als auch eine von Menschen zu interpretierende HTML Darstellung liegen

Literaturquelle: 12.1

Zur Erstellung des RESTful Webservices wurde der Fokus auf die Implementierung der Börse gelegt. Diese beinhaltet jedoch eine zusätzliche Implementierung von vier weiteren Ressourcen; dem Betrieb, der Person, dem Produkt und der Bestellung.

8.1 Betrieb

Der Betrieb muss hinzugefügt (POST) werden, um eine Person neu anlegen zu können, die jeweils mit einem oder mehrere Betriebe verknüpft ist.
Dieser Betrieb kann aufgerufen werden (GET), geändert (PUT) und auch gelöscht (DELETE) werden. Somit wurden alle vier HTTP-Operationen in der BetriebService.java implementiert.

8.2 Person

Eine Person kann jeweils in Kunde oder ein Lieferant sein, dadurch ist eine Person mit zwei Ressourcen vertreten.(Siehe: (7)) Grundsätzlich kann die Ressource Person durch alle 4 HTTP-Operationen manipuliert werden. Des Weiteren wird eine Person durch ihren Betrieb definiert, der mit registriert werden muss. Bei der Erstellung einer neuen Person wird somit über eine Identifikationsnummer auf den oder die zugehörigen Betrieb bzw. Betriebe zugegriffen. Durch diese Einschränkung kann sichergestellt werden, dass keine Personen ohne einen zugehörigen Betrieb hinzugefügt werden können.

8.3 Produkt

Innerhalb eines Börseneintrages soll es möglich sein einzelne Produkte zum Verkauf zur Verfügung zu stellen. Dafür wurde ProduktService.java erstellt. Mittels GET können alle Produkte bzw. ein einzelnes Produkt gelesen werden. Die Funktion, dass auch alle Produkte gelesen werden können beruht sich dadrauf, dass bei der Erstellung eines Eintrages innerhalb der Börse keine Einschränkungen erfolgen sollen. Somit kann ein Eintrag mehrere bzw. alle Produkte des Verfassers beinhalten.
Neue Produkte können durch POST hinzugefügt und durch PUT können diese geändert werden. Falls eine PUT-Operation auf ein Produkt angewendet werden soll, welches noch nicht besteht, so kann dies auch mit der PUT-Operation hinzugefügt werden. Gelöscht werden die Produkte durch die DELETE Methode.

8.4 Bestellung

Dadurch, dass eine Bestellung auch direkt über die Börse möglich sein soll, wurde innerhalb des RESTful Webservices auch die Bestellung implementiert. Allerdings können bezüglich der Bestellung lediglich die GET- und POST-Methoden benutzt werden und somit nur Bestellungen gelesen und erstellt werden können, da sobald eine Bestellung getätigt (POST) worden ist keine Änderung oder Löschung mehr möglich sein sollte, um den Lieferanten vor überflüssigem Abbruch der Produktzusammenstellung zu schützen.

8.5 Börse

Die Börse greift bei jedem neuen Eintrag auf zwei externe Ressourcen zu. Zunächst soll an dieser Stelle auf die Ressource Person eingegangen werden, da an mehreren Stellen innerhalb eines Eintrages auf diese zugegriffen werden kann. Zum Einen wird jeder Eintrag durch seinen Autor definiert (an dieser Stelle wird auch auf die zweite Ressource zugegriffen, dazu später mehr). Zusätzlich wird bei Kommentaren bezüglich eines Eintrages auf die Ressource Person zugegriffen. Sobald auf die Ressource Person zugegriffen wird, wird auch auf die Ressource Betrieb zugegriffen siehe Abschnitt 8.2. Durch die GET-Operation

können alle Einträge der Börse ausgegeben werden. Eine weitere GET-Operation ermöglicht unter Eingabe einer Börseneintrags-ID das Lesen eines einzelnen Eintrages mit der eingegeben ID. Mittels POST können Einträge in die Börse geschrieben werden. Da Einträge nicht mehr bearbeitet werden können sobald sie einmal in der Börse gespeichert sind, ist innerhalb der BörsenService.java keine PUT-Operation vorhanden. Allerdings können alle Einträge mittels DELETE einzeln gelöscht werden. Dazu muss jedoch die ID des zu löschenden Eintrages angegeben werden.

9 XMPP

XMPP (Extensible Messaging and Presence Protocol) beschreibt einen offenen Standard für die Echtzeitkommunikation, welcher in verschiedenen RFC's (Request for comments) und XEP's (XMPP Extension Protocol) spezifiziert ist . XMPP beschreibt eine dezentrale Client-Server Architektur und ist auch bekannt unter "Jabber", woraus sich die sogenannte Jabber ID ergibt, die die Adressen für Server und Clients definiert. Die Clients an sich kommunizieren nie direkt miteinander, aber die Server sind untereinander verbunden. Es gibt in XMPP 3 Typen der Nachrichtenprimitive:
- Message (Antwort nicht erforderlich)
- Presence (nach PubSub)
- IQ (Info/Query; Request/Response)

Literaturquelle: 12.1

9.1 Leafs (Topics)

Bei der asynchronen Kommunikation findet keine wiederholte Abfrage statt, sie handelt nach dem 'Publish-Subscribe-Prinzip'. Demnach gelangen Informationen zum Einen in das Topic und zum Anderen werden Informationen von dem jeweiligen Topic dessen Subscribern zur Verfügung gestellt.

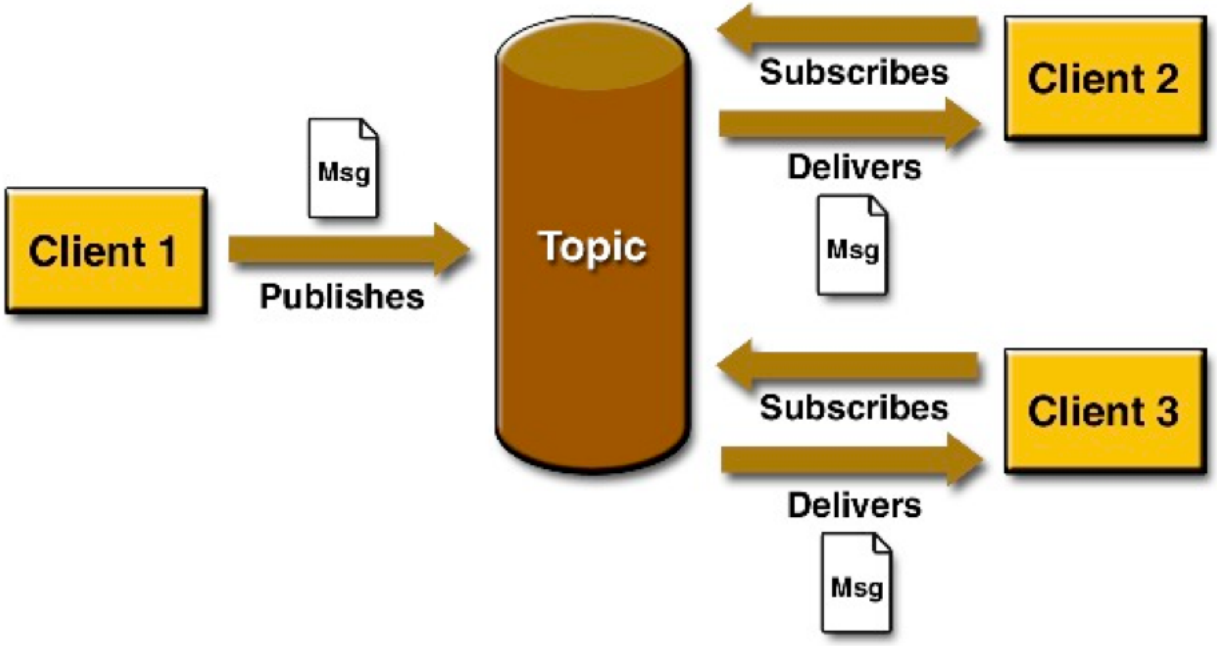

Abbildung 2: Publish-Subscribe Prinzip

Innerhalb des Systems agieren Lieferanten mit ihren Kunden, die je nach Kontext sowohl als Publisher als auch als Subscriber fungieren.
Es existieren eigene Topics für die Lieferanten, die jeder beliebige Kunde abonnieren kann sowie es auch andersherum eigene Topics für Kunden gibt, die jeder Lieferant abonnieren kann.

9.2 Publish Subscribe

Im Folgenden wird das Prinzip des Publish Subscribe anhand von Beispielen des Systems erläutert:

9.2.1 Produktliste

Zunächst speichert der Lieferant seine Warenliste in das Topic 'Produktliste', welches ihn zum Publisher macht. Der Kunde greift auf eine der Produktlisten in diesem Topic zu und agiert somit als Subscriber.

9.2.2 Bestellung

Bei einer ganz normalen Warenbestellung gelangt die Bestellung des Kunden als Publisher über das Topic 'Bestellungen' zum Lieferanten als Subscriber, der dieses Topic des Kunden abonniert hat. Für jede individuelle Lieferanten-Kunden-Beziehung sollte es somit ein Topic mitsamt allen Bestellungen geben, auf die auch kein Anderer zugreifen kann.

9.2.3 Börse

In Hinsicht auf die Börse gibt es zum Einen die Möglichkeit, dass der Lieferant als Publisher fungiert, in dem er ein Angebot aufgibt. Diese Angebote werden nach Kategorien aufgesplittet, womit sich auf diesem Wege die einzelnen Topics ergeben. Diese Topics kann der Kunde als Subscriber abonnieren und je nach Bedarf auf diese Angebote reagieren.
Andersherum verläuft dies, wenn der Kunde ein Gesuch aufgibt. Damit macht dieser sich zum Publisher und der Lieferant, der dieses Gesuch bezieht agiert als Subscriber. Bei beiden Börseneintragsformen hat man die Möglichkeit mittels Kommentare zu reagieren und Transaktionen in die Wege zu leiten zum Beispiel in Form einer Bestellung, die zuvor erklärt wurde.

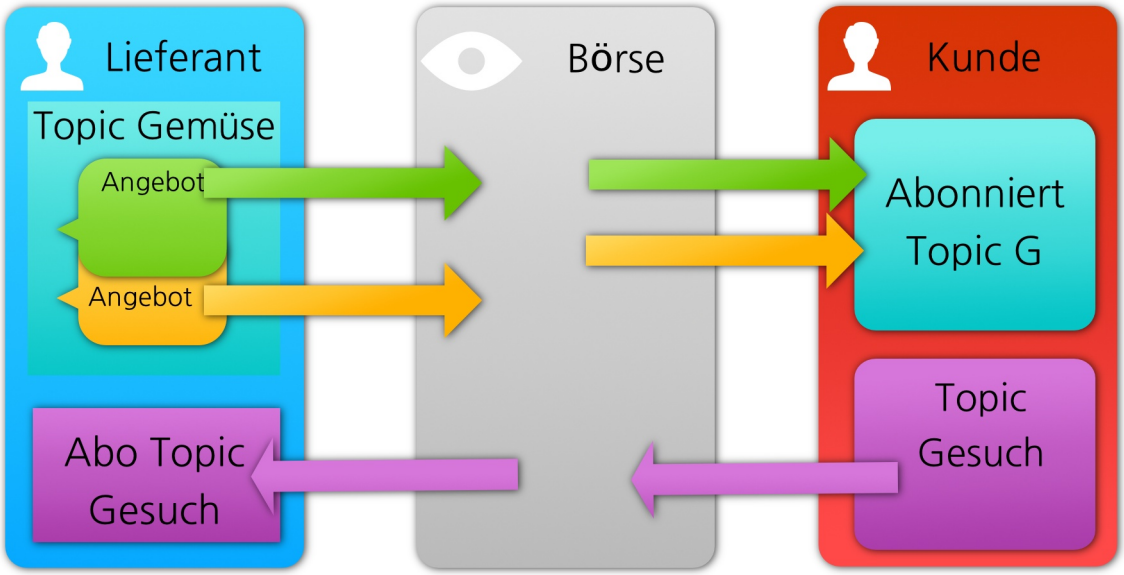

Abbildung 3: Publish-Subscribe Prinzip mit Vermittler

9.3 Umsetzung

Innerhalb des Systems wurden nicht alle möglichen Topics umgesetzt. Es wurde festgelegt, dass die Umsetzung einzelner Topics das Prinzip des Systems hinreichend demonstriert. Somit wurden nur drei Topics implementiert.
Ein Publisher kann Beiträge in die Topics Fisch, Fleisch und Gemüse speichern. Ein Subscriber kann ebenfalls nur auf Fisch, Fleisch und Gemüse zugreifen. Für jedes Topic sind alle Funktionen des Publish-Subscribe Prinzips umgesetzt worden, sodass diese uneingeschränkt verwendet werden können.

9.4 Welche Daten sollen mittels der Anwendung übertragen werden?

Die einzelnen Informationen, die in das Topic gelangen, sind die XML Dateien, die aus der XSD heraus generiert worden sind.
Somit gibt es eine Produktliste, die von dem Lieferanten (Publisher) ausgefüllt wird, sodass diese in dem Topic ‘Produktliste’ gespeichert wird und den Kunden (Subscriber) mittels der GET-Methode zur Verfügung stehen.
Das Bestellformular ‘Bestellung.xml’ kann von den Kunden, somit den Publishern ausgefüllt werden, um den Lieferanten als Subscriber zu gewährleisten, dass diese die Bestellung aus dem Topic ‘Bestellungen’ heraus aufgreifen und fertig stellen.
Für die Börse ist mittels des XML Schemas ‘Boerse.xsd’ eine XML-Datei generiert worden, die von dem jeweiligen Publisher per POST-Methode mit Daten gefüllt werden sollte. Dieses XML gelangt in das jeweilige Topic, welches von dem Subscriber mittels der GET-Methode bezogen werden kann.

Als Reaktion darauf wird die bezogene XML um Kommentare erweitert und kann darauf zu Handlungen führen wie zum Beispiel das Ausfüllen der Bestellung.xml.

9.5 XMPP-Client

Mittels der Anwendung ist es möglich, vom Publisher erstellte Leafs Fisch, Fleisch und Gemüse zu abonnieren. Ausserdem können die, vom Publisher veröffentlichten Nachrichten innerhalb der abonnierten Leafs, vom Subriber empfangen werden. Desweiteren können alle Leafs auch durch SimplePayload veröffentlicht werden. Eigenschaften des jeweiligen Topics kann durch die ServiceDiscovery eingesehen werden.
Die Datei Test.java ermöglicht das Testen des XMPP-Servers. Über die Konsole kann eingegeben werden, ob als Publisher oder Subscriber gehandelt werden soll.
Soll als Publisher gehandelt werden, so wird erst das jeweilige Topic ausgewählt. Darauf hin können nun die verschiedenen Publisher-Funktionen auf das ausgwählte Topic angewandt werden. Es können Nodes erstellt und gelöscht werden. Veröffentlicht werden die Nodes je nach Auswahl mit oder ohne Payload. Ein Publisher kann sich ausserdem eine Liste aller seiner Abonnenten ausgeben lassen.
Entscheidet sich der User als Subscriber zu agieren, so kann hier ebenfalls zunächst das gewünschte Topic ausgewählt werden. Im folgenden Schritt kann dann ein Node abonniert, deabonniert, Nachrichten ausgewählt und die Informationen über das Node aufgerufen werden.

10 Client-Entwicklung

Mit dem letzten Meilenstein sollte ein Client erstellt werden, der das Projekt repräsentiert und sowohl den REST Webservice als auch den XMPP Server nutzt.
Mittels SWING sollte ein grafisches User Interface (kurz GUI) erstellt werden, über das das System genutzt werden kann.

10.1 Umsetzung der GUI mittels SWING

Um einen kurzen Einblick darüber zu geben wie der User mit unserem System des OrderHero interagieren würde, wurde eine grafische Benutzeroberfläche ausgestaltet, die allerdings aus Zeitgründen auch nur die nötigsten Funktionen unseres Systems beinhaltet.
Diese Benutzeroberfläche wurde mittels SWING realisiert und beinhaltet folgende Funktionen:

10.1.1 Login Bereich

Der Login des Users stellt unsere Startseite dar. Dafür erscheint ein Textfeld für den erforderlichen Benutzernamen und ein Textfeld für das jeweilige Passwort, welches aus Sicherheitsgründen als Punkte dargestellt wird. (JPasswordField)
Darauf folgen zwei Buttons, um sich entweder einzuloggen oder aber auch erst zu registrieren. Insofern man sich registrieren möchte, sollte man auf ein Registrierungsformular gelangen, welches in unserem Fall jedoch nicht implementiert wurde, da die Registrierung kein wesentlicher Bestandteil unseres Systems darstellt.
Sobald man sich eingeloggt hat, werden die Login Daten als Variablen gespeichert, um innerhalb der Anwendung auf diese Benutzerdaten zuzugreifen, welches gleich nach dem Login durch die persönliche Begrüßung ersichtlich wird.
jou Ab dann hat man mit Hilfe einer Navigationsleiste (JMenuBar) verschiedene Optionen zur Interaktion mit dem System.
Es erscheinen drei Menükategorien: (JMenuItems)

10.1.2 Navigationskategorie: Personen

Darunter finden sich sowohl die Gastronomliste, die alle User beinhaltet, die sich als Gastronom registriert haben als auch eine Lieferantenliste, die alle User beinhaltet, die sich als Lieferant registriert haben.

10.1.3 Navigationskategorie: Betriebe

Darunter bekommt man eine Übersicht über alle registrierten Betriebe sowie deren Produktlisten, die sich ein User anfordern kann

10.1.4 Navigationskategorie: Börse

Diese beinhaltet die Bestellung und wird somit zu dem Kern des Systems. In der Börse kann man sich alle Einträge anzeigen lassen, die entweder als Gesuch oder Angebot agieren. Diese Einträge können per JComboBox ausgewählt werden und weisen die einzelnen Elemente eines Eintrags auf.
Sobald ein Eintrag ausgewählt und angezeigt worden ist, hat man von dort aus die Optionen einen neuen Eintrag oder Kommentar zu verfassen oder eine Bestellung des jeweiligen Produktes aufzugeben.
Die Option des neuen Eintrags kann man allerdings auch von der Menüleiste aus, wenn man den Menüpunkt 'Börse' auswählt.
Nach Auswahl eines Menüpunktes aus der Navigationsleiste erscheint ein neues Fenster als Pop up, welches durch Klick geschlossen werden kann und man somit zurück auf die Menüseite gelangt.
Es wurde mit unterschiedlichen Folgeseiten gearbeitet, die an sich als Methode agieren (z.B. GUI Start()), sodass auf einer Seite ein neues Objekt dieser Methode erzeugt werden kann und man somit jeweils auf die nächste Seite referenzieren kann. Sprich, zum Beispiel bei Klick auf einen Navigationspunkt gelangt man auf eine Folgeseite, die auch separat implementiert worden ist. So sind die jeweiligen Seiten ausgelagert und eventuell auftretende Fehler können besser ausgemerzt werden.
Im Hinblick auf die Börse greift man auf unsere JAXB-Klasse 'Börsen.java' zu und zieht die existierenden Börseneinträge mittels 'getBoersenEintrag()' heraus, deren Titel in einer Auswahlliste ausgewählt werden können, um anschließend den kompletten Eintrag anzuzeigen.
Für das Ausgeben des kompletten Börseneintrages musste Element für Element aus der 'Börse.java' mittels der get-Methoden aufgegriffen werden, um den Eintrag bestehend aus mehreren Elementen vollständig anzeigen lassen zu können.
Für einen neuen Eintrag sollten die einzelnen Elemente in die dafür vorgesehenen Textfelder eingetragen werden und mittels POST-Methode gespeichert werden. Daher wurde in der 'Börse.java' der Typ auf 'JTextField' geändert und die POST Methode 'erstelleEintrag()' aus der 'BörsenService.java' aufgegriffen, um den Wert jeweils als Variable speichern zu können. Allerdings kommt es dadurch zu Fehlermeldungen und diese Funktion konnte leider nicht vollständig ausgeführt werden.

10.1.5 GUI-Implementierung des Publisher

Im Bezug auf die Nodes wurde ein Publisher-Menü implementiert, das die einzelnen Topics, in die ein Eintrag gespeichert werden sollte, in einer Auswahlbox anzeigt. Der User wählt also zunächst das jeweilige Topic aus und formuliert Titel und Nachricht für den

Eintrag, der erstellt werden soll.
Die jeweils eingegeben Login-Daten zu Beginn werden als Variablen gespeichert und an den Publisher übergeben, sodass die Person unter seinen eigenen Nutzerdaten publishen und subscriben kann. Dazu wurden im Publisher die Eingaben aus dem Textfeld in der 'GUI Login' als Variable gespeichert und im Konstruktor des Publishers übergeben. In der 'GUI Node', auf der man Einträge publishen und subscriben kann, wird ein neues Element von diesem Publisher erzeugt.
Weitergehend führen unsere Überlegungen dazu, dass ein Subscriber ebenfalls die Topics anzeigen lassen kann und die gewünschten Topics einzeln abonnieren kann, um jeweilige Nachricht, die für das Topic vorgesehen sind, zu erhalten.

Insgesamt muss in Swing zunächst ein Frame erstellt werden, in dem die ganze Benutzeroberfläche angezeigt werden soll.
In Diesem werden sämtliche Elemente eingefügt, die jeweils zu sehen sein sollen. In unserem Fall wurden einige Navigationspunkte und Buttons mit Funktionen versehen (ActionListener), die dazu führen, dass man zum Beispiel bei Klick auf ein Button auf eine neue Seite gelangt.

11 Zusammenfassung

Insgesamt wurde ein System entwickelt, welches die vorgegebenen Anforderungen erfüllt. Das System befasst sich mit der Abwicklung von Bestellungen und der Kommunikation zwischen Kunden und Lieferanten. Der Fokus bei der Umsetzung des Systems wurde auf die Börse gesetzt, da diese viele Funktionen des Systems beinhaltet und sowohl synchrone als auch asynchrone Komponenten beinhaltet. Die Umsetzung des gesamten Systems wurde anhand der begrenzten Zeit nicht durchgeführt.
Es wurden zu allen Ressourcen XML-Schemate erstellt. Desweiteren wurden HTTP-Operationen und URI's den jeweiligen Ressourcen zugeteilt.
Bei der Erstellung des RESTful Webservices wurde das Funktionenspektrum auf die Funktionen innerhalb der Börse begrenzt, um ein detaillierteres Arbeiten zu ermöglichen und die Hauptfunktionen des Systems optimal darzustellen.
Der XMPP-Server befasst sich ebenfalls mit den Hauptfunktionen der Börse und ermöglicht die Erstellung von Topics durch Publisher und deren Abonnierung durch Subscriber.
Das grafische User Interface soll dem Nutzer eine Idee davon geben, wie das System möglicher Weise aussehen und funktionieren könnte.

12 Quellenverzeichnis

12.1 Literatur

[97TW17] Alonso,Gustavo; Casati, Fabio; Kuno, Harumi;Machiraju,Vijay (1998):
Web Services: Concepts, Architectures and Applications

[97TWCE] Möller, Andreas; Schwartzbach Michael (2006):
An Introduction to XML and Web Technologies

[33TVQ33] Tilkov, Stefan (2009):
REST und HTTP: Einsatz der Architektur des Web für Integrationsszenarien

[97TWCE] Saint-Andre, Peter; Smith, Kevin; Troncon, Remko (2009):
XMPP: The Definitive Guide: Building Real-Time Applications with Jabber Technologies

12.2 Internetquellen

[ilias] Präsentationsfolien 1
https://ilias.fh-koeln.de/repository.php?ref_id=393511&cmd=sendfile
(15.04.2013)

[ilias] Präsentationsfolien 2
https://ilias.fh-koeln.de/repository.php?ref_id=401583&cmd=sendfile
(12.05.2013)

[ilias] Präsentationsfolien 3
https://ilias.fh-koeln.de/repository.php?ref_id=404716&cmd=sendfile
(30.05.2013)

[dpunkt] dpunkt.verlag
http://dpunkt.de/leseproben/3574/4_Verben.pdf (09.05.2013)

[horn] Thorsten Horn
http://www.torsten-horn.de/techdocs/jee-rest.htm#
JaxRsHelloWorld-Grizzly (20.05.2013)

[youtube] WBA Workshop 2012
http://www.youtube.com/watch?v=5Jt7kihfFmA&feature=youtu.be
(20.05.2013)

[oracle] oracle
http://docs.oracle.com/cd/E19509-01/820-5892/ref_jms/index.html
(30.05.2013)

[ilias] Präsentationsfolien 4
https://ilias.fh-koeln.de/repository.php?ref_id=404716&cmd=sendfile
(08.06.2013)

[igniterealtime] PubSub Implementation; XMPP
http://www.igniterealtime.org/builds/smack/docs/latest/
documentation/extensions/pubsub.html (08.06.2013)

[oracle] Response Klasse
http://docs.oracle.com/javaee/6/api/javax/ws/rs/core/Response.html
(11.06.2013)

[tutorial] Java Tutorial zu Swing
http://www.java-tutorial.org/swing.html (14.06.2013)

Erklärung über die selbständige Abfassung der Arbeit

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht.
Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Köln, 23.Juni.2013 Julia Thyssen

Gummersbach, 23.Juni.2013 Sheree Saßmannshausen