Fachhochschule Köln
Fakultät für Informatik und Ingenieurwissenschaften

L O G B U C H

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

1 Einleitung

Das folgende Dokument befasst sich mit dem Werdegang, Abwägungen und Problemen des Projektes in chronologischer Reihenfolge und soll, neben der eigentlich Dokumentation, als zusätzliche Informationsquelle zur Arbeitsweise und dem Verlauf de Projektes dienen.

2 15.April.2013

Phase 2 begann damit, eine Idee zu einem interaktiven System zu finden, welches als verteiltes System mit synchronen und asynchronen Komponenten realisierbar ist.
Unsere Idee handelt von einem System, welches im Grunde genommen auf alle Bereiche anwendbar ist, die Bestellungen auszuführen haben - sprich Lieferanten-Kunden-Beziehungen beinhalten. Wir werden das System als Beispiel im Bereich der Gastronomie realisieren, in dem Lieferanten und Gastronome miteinander interagieren.

2.1 Die synchrone Komponente des Systems:

- Abfrage über vorhandene Produkte des Lieferanten mit Preisangaben
(ohne Lagerbestand)

2.2 Die asynchronen Komponenten des Systems:

- Anfrage an Lieferanten über die Verfügbarkeit des gewünschten Produktes, welche vom Lieferanten bearbeitet wird
Durch das Fehlen des angeforderten Produktes ergibt sich ein weiterer asynchroner Aspekt, wobei der Restaurantbetreiber informiert wird, sobald das jeweilige Produkt wieder verfügbar ist.
Des Weiteren hat der Lieferant die Möglichkeit dem Gastronom aktuelle Angebote oder Neuerscheinungen zukommen zu lassen sowie auch der Gastronom die Möglichkeit hat Gesuche aufzugeben, auf die Lieferanten oder auch andere Gastronomen reagieren können.

Bei der Ideenfindung traten mehrere Probleme auf:
Da die Idee relativ schnell gefunden wurde, war der Umfang des Systems zu Beginn eher gering. Nach Rücksprache mit den Tutoren wurden die Lücken und Schwachstellen der Idee schnell deutlich. Nun musste abgewägt werden:
- Was genau ist synchron und asynchron bei unserer Idee?
- Reicht der Umfang?
- Wie kann man die menschliche Komponente mehr mit einbeziehen? (mehr Interaktion)

2.2.1 Probleme

Mehrere Probleme machten sich bemerkbar: Vor allem wurde die unzureichende Interaktion bemängelt und stand somit im Vordergrund der Überlegungen. Es wurde sich schwer getan, eine zusätzliche Funktion zu finden, welche den Anforderungen entspricht. Letzten Endes wurde folgende Funktionen zum Konzept hinzugefügt:
Zum Einen kann der Gastronom eine Anfrage über ein spezielles Produkt rausgeben, auf diese verschiedene Lieferanten und Supermärkte reagieren können. Zum Anderen haben die Lieferanten oder Supermärkte die Möglichkeit bestimmte Produkte anzubieten, die beispielsweise aufgrund des Ablaufdatums in nächster Zeit verkauft werden müssen.
Die Anwendung bietet letzten Endes folgende Funktionen und Ressourcen:

2.3 Funktionen

- Der Kunde kann sich die angebotenen Produkte des Lieferanten ansehen
- Der Kunde kann Bestellungen aufgeben und sich über die Verfügbarkeit der einzelnen Produkte informieren
- Der Kunde hat die Möglichkeit Daueraufträge aufzugeben
- Falls das gewünschte Produkt zum Anforderungszeitpunkt nicht lieferbar ist, wird der Kunde informiert, sobald dieses wieder möglich ist
- Der Lieferant hat die Möglichkeit seinen Kunden Angebote und neue Produkte vorzustellen
- Der Gastronom kann eine Anfrage über ein spezielles Produkt rausgeben, auf diese verschiedene Lieferanten und Supermärkte reagieren können
- Die Lieferanten oder Supermärkte können bestimmte Produkte anbieten, die beispielsweise aufgrund des Ablaufdatums in nächster Zeit verkauft werden müssen

2.4 Ressourcen

Für das System ergeben sich folgende Ressourcen:
- Person:
Es gibt zwei verschiedene Arten von Personen(Nutzern) - Kunden und Lieferanten, wobei ein Lieferant auch Kunde sein kann.
- Liste aller Kunden
- Liste aller Lieferanten
- Betrieb
- Lister aller Produkte und ihren Preisen und Artikelnummern
(Produkt, Lieferdatum, Menge, Zahlungsart, Betrag)
- Bestandsliste
- Liste der Kundenbestellungen beim jeweiligen Lieferanten

- Bestellung einzeln und als Übersicht
(Lieferdatum, Menge, Produkte mit Preis, Zahlungsart, Gesamtbetrag)
- Liste aller Daueraufträge - Börsenübersicht
- Börseneintrag

siehe Dokumentation.

3 22.April.2013

3.1 Kommunikationsabläufe und Interaktionen

Für den konzeptioneller Meilenstein war es unsere Aufgabe die Kommunikationsabläufe und Interaktionen unseres Systems darzustellen. Dazu haben wir zunächst ein Informationsflussdiagramm erstellt, durch welches die synchronen und aysnchronen Abläufe in unserem System ersichtlich werden. Umgesetzt wurde das Informationsflussdiagramm mit Hilfe von Adobe Photoshop. Als Beispiel wählten wir eine übliche Bestellung, welche vom Gastronom getätigt wird.

3.2 Informationsflussdiagramm

3.3 Synchrone und Asynchrone Komponenten

Die synchrone Komponente unseres Systems ergibt sich durch den Lieferanten, der seine Produktliste im Webserver speichert, die von den Gastronomen jederzeit sofort abrufbar ist. Als Beispiel der asynchronen Seite 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 die Benachrichtigung erfolgt erst, sobald das Produkt verfügbar ist und die Möglichkeit besteht Dieses zu bestellen. Der Gastronom bestellt dieses Produkt direkt beim Lieferanten und Dieser gibt eine Bestätigung raus.

4 29. April 2013

4.1 XSD-Schema

Zur Erstellung des XSD-Schemas haben wir zunächst die einzelnen Elemente aufgelistet und abgewägt. Im Folgenden werden diese Elemente erläutert.

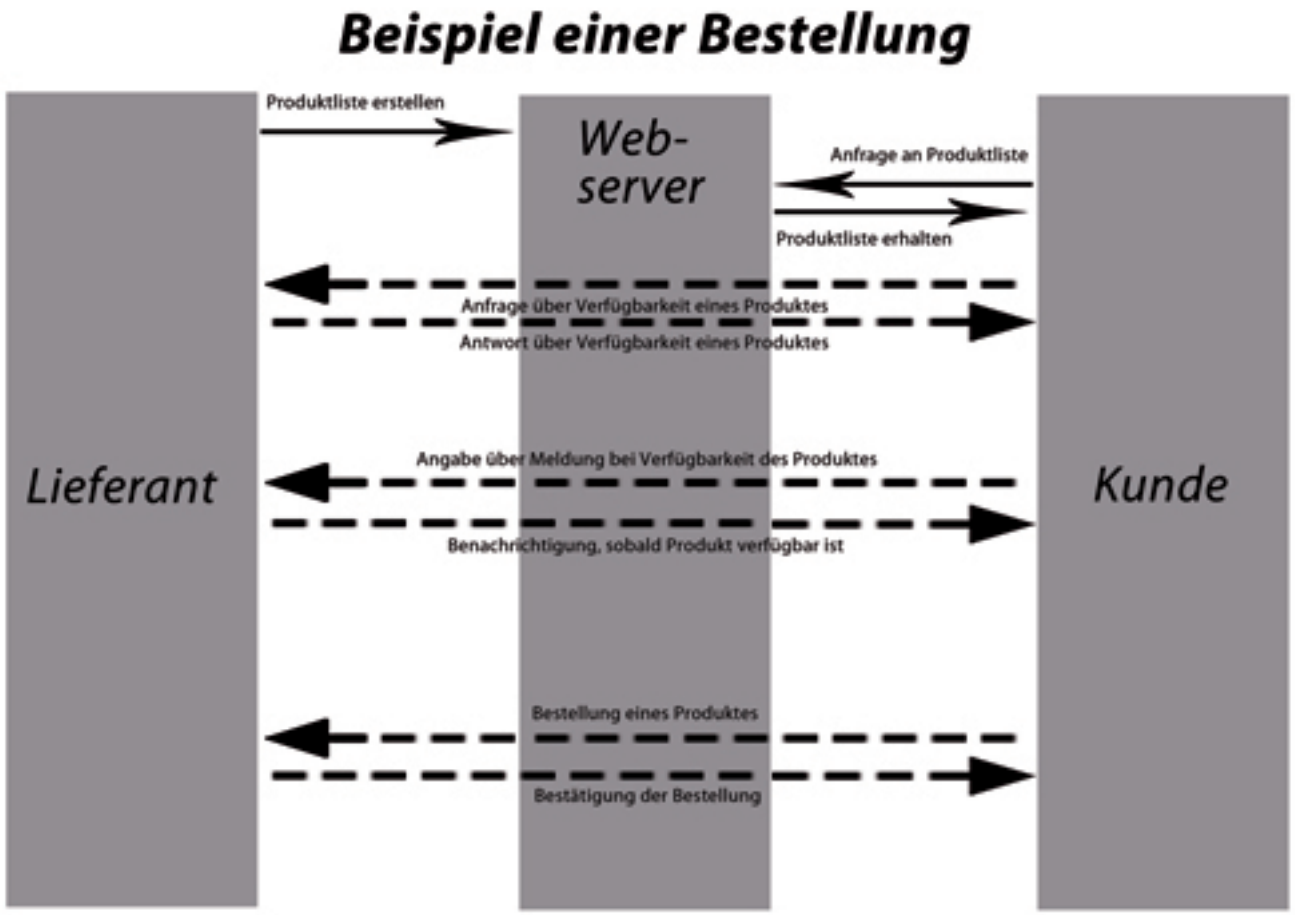

Abbildung 1: Informationsflussdiagramm anhand eines Beispiels einer Bestellung

4.2 Angaben zur Person

4.2.1 Angaben zum Lieferant

- Name *öffentlich*
- Alter(Mindestalter 18?)
- Betrieb mit Adresse *öffentlich*
- Produktliste mit Preisen *öffentlich*
- Bestandsliste
- Kundenliste(mit den Bestellungen des jeweiligen Kunden)
- Bestellungslisten

Was soll der potentielle Kunde über den Lieferanten erfahren können und umgekehrt?
Wie kann ein Lieferant ausgewählt werden?

4.2.2 Angaben zum Gastronom

- Name *öffentlich*
- Alter(Mindestalter 18?)
- Betrieb mit Adresse (Lieferadresse) *öffentlich*
- Art der Küche *öffentlich*

(welche Art an Produkten ist gefragt)

- Liste aller Lieferanten
- Liste seiner Produkte
- Liste seiner Bestellungen beim jeweiligen Lieferanten
- Bestandsliste

4.3 Angaben zur Bestellung

- Lieferant
- Kunde
- Lieferdatum

(regelmäßiger Dauerauftrag oder einmalige Bestellung?)

- Menge
- Produkte mit Preis
- Zahlungsart
- Gesamtbetrag

4.4 Angaben zur Börse

- Angebote
- Gesuche
- Gastronom
- Lieferant
- Informationstext
- Startdatum
- Ablaufdatum
- Kommentare

4.5 Schwierigkeit Referenz

Als Schwierigkeit stellt sich die Erstellung von Referenzen heraus. Die Idee ist es, die Angaben beim Kunden-Element aus dem zuvor definierten Element Gastronom zu übernehmen. Somit sollen nur Kunden übernommen werden können, welche als Gastronomen registriert sind.
Dasselbe gilt für die Produkte und die Lieferanten. Die Bestellungen wurden einheitlich definiert, da sie für alle Bestellungsabwicklungen gleich aussehen und aufgebaut sein sollen. Die Funktion, ob eine Bestellung als Dauerauftrag abgewickelt werden soll, ist noch nicht gelöst worden.

5 30. April und 2. Mai 2013

5.1 XSD-Erstellung

Bei der Erstellung der XSD-Datei fiel auf, dass es sich als sinnvoller erweisen würde, die Eigenschaften des Restaurants des Gastronoms als eigenes Element zu gestalten.
Dies geschah aus dem Grund da es Gastronomen gibt, welche mehr als einen Gastronomiebetrieb besitzen. Dabei kann es sein, dass sich die verschiedenen Betriebe komplett voneinander unterscheiden und unterschiedliche Eigenschaften aufweisen
(beispielsweise Standort und Art der Küche). Dies wirkt sich natürlich auch auf die Bestellungen aus, welche unterschiedlich ausfallen. Somit soll der Gastronom mehrere Gastronomiebetriebe einfügen und zu jedem Betrieb den Bestand, die Liste aller Lieferanten, die Liste der Produkte und die Liste seiner Bestellungen beim jeweiligen Lieferanten auflisten können. Noch nicht festgelegt ist die Option, ob der Gastronom zusätzlich zu den einzelnen Bestellungen seiner jeweiligen Betriebe eine Übersicht über alle Bestellungen als eine große Bestellung generieren kann. Dies würde dem Gastronom beispielsweise bei der Verwaltung seiner Finanzen zu Gute kommen. Allerdings könnte dies auch zur Verwirrung führen. Eine weiter Idee wäre es, zu jedem Betrieb die Summe der Ein- und die der Ausgaben auszurechnen und diese dann als Gesamtumsätze aufzulisten.

5.2 Veränderung des Aufbaus

Bei der Erstellung der XSD-Datei wurde der Aufbau noch einmal verändert: Gastronom und Lieferant finden sich nun unter dem Element "Mitglieder"wieder, um sowohl Gastronom als auch Lieferant gleichzeitig ansprechen zu können.

5.2.1 Lieferant

- Vorname
- Nachname
- Alter
- Betrieb

Betriebname
Adresse

- Produktliste

Name
Preis
Beschreibung

- Bestand

Artikel(Anzahl,Produkt)

- Kundenliste
- Bestellungslisten

5.2.2 Gastronom

- Vorname
- Nachname
- Restaurant

Restaurantname
Adresse
Art der Küche

- Liste seiner Lieferanten mit jeweiligen Bestellungen
- Bestand

Artikel (Anzahl, Produkt)

- Kaufabwicklung bei Börse
- Referenz auf andere Elemente

5.3 Angaben zur Bestellung

- Lieferant
Auswahl aus Lieferantenliste
- Kunde
Auswahl aus Kundenliste
- Lieferdatum
(regelmäßiger Dauerauftrag oder einmalige Bestellung?)
- Menge
- Produkte mit Preis
aus Produktliste
- Zahlungsart *optional*
- Gesamtbetrag

5.4 Angaben zur Börse

Angebote und Gesuche sollen zwar gleich aufgebaut werden, jedoch eigenständige Elemente sein.
Gastronom und Lieferant können sowohl beide Gesuche und Angebote aufgeben als auch auf Diese reagieren.
Ungeklärt bleibt, wie die Kaufabwicklung angebunden werden soll.

- Angebote bzw. Gesuche
- Erstellungsdatum
- Gastronom bzw. Lieferant
- Informationstext
- Startdatum
- Ablaufdatum
- Kommentare
Verfasser
Gastronom bzw. Lieferant
Kommentieren

6 3. Mai 2013

Die XSD wird verfeinert und optimiert. Gleichzeitig wird die Dokumentation angepasst und ein Layout für Diese erstellt. Es besteht die Frage ob Bestellung, Börse, Gastronom und Lieferant in verschiedene XSD-Dateien gespeichert werden soll. Die Entscheidung fällt gegen die Aufteilung auf mehrere XSD-Dateien, da dies zu kompliziert und zu unübersichtlich zu werden scheint. Die Elemente werden wie zu Beginn in einer XSD mit dem Namen OrderHero gespeichert.

7 4. Mai 2013

7.1 Modifikationen

Es werden weitere Feinarbeiten durchgeführt, in dem Elemente wie 'Produkte', 'Gastronom' und 'Restaurant' eigene Types bekommen, sodass man sich innerhalb der XSD-Datei darauf beziehen und weitere Elemente von deren Typ definieren kann.
Des Weiteren stellt sich die Frage, ob das Informationsflussdiagramm noch verfeinert werden müsste, da nur begrenzte Funktionen des Systems aus diesem Diagramm ersichtlich werden. Nach Rücksprache mit den Tutoren entschieden wir uns dagegen, da nur ein Beispiel dargestellt werden sollte.

7.2 Listen

Für die Ressourcenplanung ergeben sich weitere Listen, die wir zuvor nicht beachtet haben. Zum Einen sollen bei der Bestellung die Lieferanten aus einer Lieferantenliste auswählbar sein sowie die Kunden aus der Kundenliste. Auf diesem Wege ist es nicht mögliche willkürliche Namen bei der Bestellung anzugeben, sondern man kann nur zuvor eingetragene Kunden und Lieferanten auswählen, die man somit vorher in die Kartei aufnehmen muss. Dies verhindert, dass man aus Versehen falsche Lieferanten angibt, was zur Folge hat, dass die angegebene Bestellung nicht ankommt.
Zum Anderen sollten Daueraufträge, die man als Gastronom bestellt und als Lieferant ausliefert, ebenfalls in einer Liste eingespeichert sein, um einen Überblick zu bewahren.

8 5. Mai 2013

8.1 Ressourcen als seperate XSD-Dateien

Nach weiteren Feinarbeiten am gesamten XSD-Dokument hat sich die Definition aller Elemente in nur einer XSD-Datei als unübersichtlich herausgestellt. Aus diesem Grund wurde für jede einzelne Ressource, also Liste, ein eigenes XML-Schema erstellt.

Somit ergeben sich folgende XSD-Dateien:
- Bestellungsliste
- Dauerauftrag
- Lieferantenliste
- Kundenliste
- Produktliste
- Bestandsliste,

wobei sich wiederrum die Frage gestellt hat, ob die Liste der Daueraufträge explizit angegeben werden muss, da diese quasi mit der Kundenliste konform ist. Dadurch, dass beide Listen Informationen über beide Parteien - sprich Kunde <-> Lieferant - deren Sitz und die Produkte, um die es sich bei einem Dauerauftrag handelt, beinhaltet, könnte man diese Liste also in zweierlei Hinsicht verwenden.

9 6. Mai 2013

9.1 Beratungstermin Ergebnisse

9.1.1 Ressourcentabelle

Nach Rücksprache mit den Tutoren wurde schnell klar, dass die XSD-Dateien noch nicht ganz vollständig sind. Denn sobald man sich zum Beispiel in der Produktliste befindet, hat man die Möglichkeit auf das jeweilige Produkt zu klicken, welches eine eigene XSD-Datei erfordert. Gleiches gilt für den Fall, dass man sich die einzelnen Personen, ob Gastronom oder Lieferant sei frei gestellt, detaillierter anschauen kann. Diese XML Schema Dateien haben wir hinzugefügt und somit unsere Ressourcen komplettiert. Diese haben wir durch URI's eindeutig identifiziert. Somit ergibt sich eine Tabelle mit allen Ressourcen(siehe Doku Kapitel 5 und 6)

10 9. Mai 2013

10.1 HTTP-Operationen

Es wird weiter an den HTTP-Operationen gearbeitet. Zunächst geht es um eine allgemeine Recherche dieser Operationen, die wir anschließend auf unsere Ressourcen anwenden. Es fällt auf, dass jede HTTP-Operation, sprich
- PUT
- GET
- POST
- DELETE
auf jede einzelne Ressource anzuwenden ist.

Allerdings stellt sich die Frage, ob die Rechnung, die der Kunde mit seiner Bestellung bekommt, eine eigene XSD-Datei benötigt. Da die Rechnung alle Informationen der Bestellung beinhaltet, die zuvor eingegeben werden muss, generiert sich somit diese Rechnung für den Kunden aus der Bestellung, also somit aus ihrer XML heraus.

11 10. Mai 2013

11.1 Trennung ähnlicher Ressourcen

Bei der Zuweisung der HTTP-Operationen zu den einzelnen Ressourcen kam die Frage auf, ob 'Kundenliste.xsd' und 'Lieferantenliste.xsd' in eine Liste zusammengefügt werden könnten, da sich diese nicht wesentlich voneinander unterscheiden. Es wurde sich jedoch gegen diese Idee entschieden, da zum einen die Übersicht gewahrt werden sollte und zum anderen der Kunde auf eine andere Liste zugreift wenn er seine Lieferantenliste anfordert, als der Lieferant bei der Kundenliste. Die Benennung an dieser Steller unterscheidet die beiden XSD-Dateien. Eine Kundenliste soll immer eine Kundenliste bleiben und separat gespeichert werden.

12 12. Mai 2013

12.1 Ressource Bestand

Es wurde die Ressource Bestnd hinzugefügt, um dem Kunden eine Übersicht seines Bestands zu gewährleisten und sicherzustellen, dass dieser auf einen Blick weiß, ob und was er beim Lieferanten bestellen muss.

12.2 Änderungen in Person.xsd, Personverwaltung.java

In der ‘Person.xsd’ wurde das Element PersonID hinzugefügt, um bei späteren Änderungen nur über die ID auf die Person zugreifen zu können (Suchfunktion z.B.).
Selbiges Element wurde in der ‘Personenliste.xsd’ hinzugefügt.

Zusatz in der ‘Personverwaltung.java’: Änderungen einer Person per PersonID mittels Unmarshalling und Marshalling:

```
‘// Suche Person mit ID=person1 und aendere Personeninformationen
List<Person> p = person.getPerson();
for(int i = 0; i < p.size(); i++)
if ( p.get(i).getPersonID().equals("person1") )
p.get(i).setTyp("Gastronom");
p.get(i).setVorname(SSheree");
p.get(i).setNachname(SSaßmannshausen");
p.get(i).setAlter(22);’
```

13 13. Mai 2013

13.1 Beratungstermin Ergebnisse

Nach Rücksprache mit den Tutoren wurde deutlich, dass unser bisheriger Weg für die Umsetzung unseres Projektes ungünstig ist, da Eingaben/Ausgaben etc. nicht über die Konsole erfolgen sollten, sondern unsere Ressourcen sollten sich im Webserver (bspw. Grizzly) befinden, die per URL zum Beispiel in einem Browser bezogen werden können.

13.1.1 Ressourcenergänzung: Betriebsliste

Betriebsliste = Übersicht aller Betriebe.
Sobald man sich anmeldet und seine Informationen wie Name, Alter, Betrieb etc. angibt, erscheint dieser Betrieb automatisch in der Liste.
Man kann keinen Betrieb extra anlegen ohne, dass eine Person dazugehört.
Sprich: bei der Anmeldung greift man quasi auf ’Person.xsd’ zu, in der man Informationen über Vor-,Nachname, Alter, Betrieb etc. preisgibt.
=> zentrales Element unseres Projektes

13.1.2 Boersenspezifikationen

In der Börse werden Einträge bezüglich Angebot und Gesuch weiter spezifiziert.
-Personenliste: Operation nur GET, da diese Liste alle angemeldeten Personen beinhalten und somit alle Personen, die sich anmelden automatisch in der Liste erscheinen

14 14. Mai 2013

14.1 REST und GIT

Erstellung der Server/Client Architektur mittels REST.
Quelle dazu: Tutorialvideo von Rene -> WBA2 SoSe2012
Nach Fertigstellung gibt es Probleme mit git. Kein Pull möglich.
=> alles gelöscht, erneut synchronisiert.

15 15. Mai 2013

15.1 REST - Fehlermeldungen

Weiterarbeiten an Server/Service.java
Änderung in 'ObjectFactory.java':

public Boerse createBoerse()
return new Boerse();

um in Service.java createBoerse zu gewährleisten.
Es besteht kein Fehler im Code selbst, allerdings stoßen wir auf folgende Fehlermeldung:
'Exception in thread "main"java.lang.NoClassDefFoundError: generated/Personverwaltung
Caused by: java.lang.ClassNotFoundException: generated.Personverwaltung at java.net.URLClassLoa
202)$atjava.security.AccessController.doPrivileged(NativeMethod)atjava.net.URLClassLoader.f$
190)$atjava.lang.ClassLoader.loadClass(ClassLoader.java$: 306)$atsun.misc.Launcher$AppClassLo
at java.lang.ClassLoader.loadClass(ClassLoader.java:247)'

16 22. Mai 2013

16.1 Beratungstermin Ergebnisse

In Rücksprache mit Tutor Volker konnte die Fehlermeldung analysiert und das Problem bei der Ausgabe der XML-Datei im Browser gelöst werden.

16.1.1 Fehleranalyse

Es handelte sich um die Verwendung von Sonderzeichen innerhalb von Attributen. Bei der Ausführung trat zunächst kein Fehler auf, jedoch wenn die XML-Datei über die Ausgegebene URL vom Browser ausgegeben werden sollte gab Eclipse folgenden Fehler aus: Element type 'Stra' must be followed by either attribute specifications, '>' or '/>'.]
Dieser Fehler zog sich als Folgefehler durch sämtliche Dateien und bezog sich auf die Verwendung von 'ß' in 'Straße'. Nachdem alle 'ß' in 'ss' umgewandelt worden waren wurde die XML problemlos im Browser ausgegeben.

16.2 GET-Methoden

Mittels der URL 'http://localhost:4434/boersen' und der Methode 'getAll' greift man auf alle Börseneinträge zu und mittels der URL 'http://localhost:4434/boersen/boerse/1' und der Methode 'getOne' greift man auf den Börseneintrag mit der ID '1'zu

17 29. Mai 2013

17.1 Fokus des RESTful Webservices

Zur Erstellung des RESTful Webservices wurde der Fokus auf die Implementierung der Börse gelegt. Diese beinhaltet jedoch eine zusätzliche Implementierung zweier weiterer Ressourcen: dem Betrieb und der Person.
Deshalb wurden sowohl für die Betriebe als auch für die Personen die HTTP Operationen GET,PUT,POST,DELETE implementiert.

17.1.1 PersonService.java

In Hinsicht auf die Implementierung der 'PersonService.java' ist zu sagen, dass es zwei Arten von Personen gibt und zwar die Kunden und die Lieferanten in den jeweiligen Listen, weshalb man diese doppelt erstellen muss. Bei der Implementierung der DELETE Methode im Betrieb fiel uns ein Aspekt ein, den wir zunächst nicht beachtet hatten:
Sobald ein Betrieb gelöscht wird und eine Person nur diesen einen Betrieb für sich angegeben hatte, hat dies zur Folge, dass diese Person somit automatisch mit gelöscht wird.
Um diese Funktion zu realisieren stellte sich zunaechst ein Problem dar. Denn ursprünglich war der Gedanke, dass man im BetriebService in der DELETE Methode selbst diese Funktion mit einbringt. Da aber die Personen nicht mittels eines Betriebes angezeigt werden, sondern umgekehrt (der Betrieb durch die Personen), mussten wir also von dieser Seite aus gehen und in der PersonService.java eine Abfrage zu allen Personen hinzufügen,

die überprüft ob die BetriebsID zu den jeweiligen Personen noch existieren. Falls nicht, wird die Person mit der gelöschten BetriebsID automatisch entfernt.

18 30. Mai 2013

18.1 BoersenService.java

Zu guter Letzt wurde die Börse aufbauend auf den beiden Ressourcen Betrieb und Person durch die HTTP Operationen abgedeckt, sodass es möglich ist ein Angebot oder ein Gesuch aufzugeben, um seine Produkte anzubieten oder welche zu suchen.
Somit können nun Betriebe sowohl in der Übersicht als auch über die BetriebsID im Einzelnen ausgegeben werden. Selbiges gilt für die Ausgabe der Kunden-und Lieferantenliste mit den jeweils dazugehörigen Personen.

18.2 Konzeptionelle Planung der asynchronen Kommunikation

Bei dem vierten Meilenstein ging es darum die Planung für die asynchrone Kommunikation in Betracht zu ziehen. Dazu haben wir die Leafs, somit unsere Topics für unser System definiert und jeweils festgelegt wer im Bezug auf wesentliche Interaktionen mittels unseres Systems als Publisher bzw. als Subscriber agiert und welche Daten dabei übertragen werden.

19 31. Mai 2013

19.1 Probleme mit GIT und HTTP-Operationen

Es gibt erneut Probleme beim Synchronisieren mit GIT. Es ist weder möglich den Projektstand ins Eclipse zu ziehen noch etwas Neues hochzuladen.
Bei den HTTP-Operationen PUT, POST und DELETE innerhalb der Service-Dateien traten einige Fehler auf. Es fiel schwer diese zu ermitteln, da der im Praktikum bereit gestellte TestClient nicht auf Anhieb verständlich erschien.

20 03. Juni 2013

20.1 Beratungstermin Ergebnisse

Nach Rücksprache mit den Tutoren wurde das Problem mit GIT weitesgehend gelöst und es wurde über den bisherigen Projektstatus gesprochen.

20.1.1 BetriebService.java

Im Bezug auf den BetriebService erwies es sich als eher ungünstig, wenn die Person zu einem gelöschten Betrieb automatisch mitgelöscht wird, da die Möglichkeit besteht, dass die Person an sich möglicherweise angemeldet bleiben möchte, um im Rahmen eines neuen Betriebes weiter zu agieren.

20.1.2 @PathParam in POST- und PUT-Methoden

Unsere POST und PUT-Methoden sollten noch einmal überdacht werden, denn wir haben Diese mit PathParams implementiert, wobei es eigentlich eher unsere Aufgabenstellung war mittels der Methoden unsere Objekte direkt zu übergeben.

20.1.3 Topics

Außerdem haben wir unsere Topics überarbeitet und die bisherigen Lösungsansätze noch einmal aufgesplittet. So unterscheidet man zum Beispiel in der Börse nicht nur zwischen ‘Gesuch’ und ‘Angebot’, sondern zwischen ‘Gesuch Gemüse’ und ‘Gesuch Backwaren’ oder ‘Angebot Getränke’ und ‘Angebot Süßgkeiten’. Es bilden sich somit Kategorien, die vereinzelt nach Belieben abonniert werden können.

21 06. Juni 2013

Openfire und das Chatprogramm Adium werden installiert und miteinander verbunden. Die Kommunikation ist per Nachrichtensender möglich!

21.1 XMPP

Die Dateien 'Publisher-java', 'Subscriber.java', 'User.java' und 'ItemEventCoordinator.java' und 'Test.java' werden erstellt. Ein Benutzer kann sich somit unter User anmelden und sofor entscheiden, ob er als Subscriber oder Publisher agieren möchte. In 'Publisher.java' können Nodes erstellt und gelöscht werden. Nodes können mit und ohne Payload publiziert werden. Zusätzlich hat man die Möglichkeit eine Liste der Subscriber auszugeben. Wählt sich ein User als Subscriber ein, ist es ihm möglich Nodes zu abonnieren, bereits abonnierte Nodes zu löschen und Informationen über diese Nodes abzurufen.
Außerdem kann man dazu Nachrichten erhalten.

22 10.Juni 2013

22.1 Beratungstermin Ergebnisse

Im Gespräch mit den Tutoren wurde versucht, die Ursache für den nicht funktionierenden RESTful Webservice zu finden. Zunächst schien die Angabe des 'produzierten' bzw. 'konsumierten' MediaTypes falsch zu sein. Die Behebung dieses Problems löste nur leider nicht die Problematik der HTTP-Operationen. Nach einem langen Gespräch wurde uns empfohlen, die jeweiligen Methoden zunächst ganz simpel aufzubauen und anstatt eines Objektes nur einen String zu übergeben um Schritt für Schritt herauszufinden wo der Fehler liegen könnte. Getestet werden all unsere HTTP Methoden per vorimplementierten RESTClient (Link siehe Literatur).

22.1.1 Query Parameter

Außerdem haben wir die Frage nach den QueryParametern aufgegriffen, die laut Aufgabenstellung mindestens einmal in der Anwendung implementiert werden sollten.
In unserem Fall sind Query Parameter in der bisherigen Implementierung nicht notwendig, da diese beispielsweise in der Suche verwendet werden. In der Theorie gibt es bei unserem System zwar auch eine Suche, die beispielsweise in der Kunden-/Lieferantenliste oder auch in der Produktliste eingesetzt wird, aber diese Funktion haben wir aus Zeitgründen leider nicht implementieren können.
Die Query Parameter würden sich dann in der URI nach einem '?' äußern, die den eingegebenen Suchbegriff beinhalten.

23 11.Juni 2013

23.1 RESTful Webservice Fehlerbehebung

Bei den PUT- und POST-Methoden wurde jeweils der Rückgabewert Response gewählt. Response ist ein Rückgabewert, der die Metadaten zurückgibt,die zur Laufzeit bereitgestellt werden. Diese Klasse 'Response' bietet mehrere Methoden, die genutzt werden können, wie beispielsweise der ResponseBuilder.
In der PUT Methode kam eine if-Abfrage hinzu, die prüft, ob eine bestimmte BetriebsID bereits vorhanden ist. Falls ja, soll Diese geändert werden, falls nicht soll diese erzeugt werden.

23.2 Kommentare

Bezüglich des Kommentars war uns eine Überlegung gekommen, ob ein Kommentar eine eigenständige Ressource sein sollte, für die wir bereits eine KommentarID vorgesehen hatten. Allerdings ist es ja so, dass ein Kommentar sich auf den Infotext des jeweiligen Angebots oder Gesuchs bezieht und wenn man diesen Kommentar eigenständig anzeigen lassen könnte, würde somit der Infotext wegfallen und den Kommentar als Solches als sinnlos erscheinen lassen. Deshalb galt es nur die KommentarID aus der XSD zu löschen, da sie unnötig ist.

24 12.Juni 2013

24.1 Kommentare

Vorherige Überlegung bezüglich der Kommentare mussten wir widerlegen. Die Kommentare sollten doch über eine ID angesprochen werden, da ja auch einzelne Kommentare gelöscht werden sollten und eine Person die Möglichkeit haben sollte eine Übersicht über alle Kommentare, die sie geschrieben hat, zu bekommen. Allerdings muss man dabei beachten, dass es eine Verknüpfung zum jeweiligen Börseneintrag, zu dem der Kommentar gehört, geben muss, da ein Kommentar alleine keine Sinnhaftigkeit mehr aufweist, wenn der Kontext verloren geht. Die Kommentare werden ebenfalls über die Methdoe POST hinzugefügt.

24.2 Topics

Ein Kunde kann jeweils Publisher und Subscriber sein, ebenso wie ein Lieferant beide Rollen spielen kann. Angebote und Kommentare können vom Publisher in verschiedene Topics gespeichert werden.
Beispiel: in Doku Abbildung 3.

Mögliche Topics:
1.Lebensmittel - Gemüse - Fisch - Fleisch - Obst - Backwaren
2.Getränke - alkoholisch - alkoholfrei
3.Zubehör - Material - Gläser

Implementiert werden beispielshalber nur die Topics Fisch, Fleisch und Gemüse. Es wurde versucht, ein Collection-Node 'Lebensmittel' zu implementieren, um zusammengehörige Topics miteinander verbinden zu können. Jedoch scheiterte dies an der Umsetzung und es wurde entschieden, dass die Collection-Nodes für die Demonstration der Funktionen des Projektes nicht maßgeblich ausschlaggebend seien. Ausserdem war nicht klar, ob das Collection-Node Lebensmittel bei einer neuen Nachricht in Gemüse diese direkt an alle Abonnenten des Collection-Nodes sendet. In diesem Fall wäre ein Collection-Node nicht im Sinne der Subscriber die nur die Nachrichten des Topics Gemüse erhalten wollen.

25 13.Juni 2013

25.1 Nodes hinzugefügt

Es wurden Publisher und Subscriber hinzugefügt. Zunächst können nur jeweils die Topics Gemüse, Fleisch und Fisch verwendet werden. Ein User kann sich nun entweder als Publisher oder Subsriber anmelden. Die Datei Test.java wurde erstellt, um die implementierten Methoden auf ihre Korrektheit zu prüfen.

26 13.Juni 2013

26.1 Fehlerbehebung BestellungsService

Bei der generierten Bestellungs-Klasse fiel sofort auf, dass die notwenigen Methoden nicht generiert worden waren. Daher musste der Fehler schon in der XSD liegen. Der Fehler lag darin, dass aus Versehen zwei namensgleiche Referenzen verwendet worden waren. Einmal eine Referenz auf die Person.xsd und einmal auf die Betriebs.xsd. Dies führte dazu, dass erst die Elemente nach diesen doppelten Definition als Methoden generiert worden waren und dementsprechend mehr als die Hälfte der notwendigen Methoden fehlen. Dieser Fehler wurde behoben und in der ObjectFactory wurde manuell die Methode createBestellung/create Bestellungsliste definiert, um in der BestellungsService mittels dieser Methode eine neue Bestellung erstellen zu können.

27 14.Juni 2013

27.1 XMPP: weitere Implementierung

Der ‘connectionhandler’ wurde hinzugefügt mit Hilfe dessen man eine Verbindung zum XMPP Server aufbauen kann. Dort speichern Publisher Informationen in das Node, die von den Subscribern abonniert werden können

27.2 Fehlerbehebung Betriebsliste und Personenliste

In der Betriebliste war angegeben, dass mindestens ein Betrieb vorhanden sein muss. Aber es muss ja zunächst der erste Betrieb erstellt werden. Deshalb ‘minOccurs=0;’
Auch wenn die Betriebsliste leer ist, ist es möglich den allerersten Betrieb zu erstellen => selbiges mit Person und Produkt.

28 16.Juni 2013

28.1 Client-Entwicklung mit Swing

Mithilfe von Swing konnte eine grafische Oberfläche erstellt werden. Mit der Hilfe von zahlreichen Beispielen war die Implementierung nicht zu schwer. Es wurde ein ’Grafic User Interface’ (GUI) erstellt, bei dem bein ersten Aufrufen zunächst die Auswahl zwischen den 4 implementierten Ressourcen Börse, Person, Betrieb und Bestellung stattfindet. Ist eine der Ressourcen ausgewählt, so können mit Hilfe des RESTful Webservices die jeweiligen HTTP-Operationen angewendet werden.
An dieser Stelle gab es das Problem, dass nicht ganz klar war, wie die in den POST- und PUT-Methoden im Webservice übergebenen Objekte über die GUI übergeben werden können. Eine Überlegung war, für jedes Element des Objektes ein JTextField zu erstellen. Allerdings wäre das übergeben von Objekten im Webservice dann nicht von Nutzen gewesen. Im Gespräch mit den Tutoren soll dieses Problem gelöst werden.
Desweiteren wurde der XMPP-Server implementiert. Somit können bei der Auswahl der Börse zusätzlich zu den HTTP-Operationen die XMPP-Funktionen Publish und Subscribe ausgewählt werden. Der Server implementiert nur 3 der Möglichen Topics: Fleisch, Fisch und Gemüse. Zu den jeweiligen Topics können zusätzlich die Funktionen der importierten Klassen Publisher.java und Subscriber.java ausgeführt und verwendet werden. Nicht implementiert aber vorgesehen sind eingeschränkte Zugriffsrechte zum Beispiel bei Bestellungen, so hat nur der jeweilige Lieferant die Übersicht über die Bestellungen von Kunde XY und Kunde XY halt auch nur dieselbe Übersicht über seine Bestellungen beim jeweiligen Lieferanten.

28.2 REST-Feinschliff

Bezüglich der REST Methoden wurde noch die Funktion der Produktlisten-Anforderung hinzugefügt. Alle für uns als wichtig erachteten Funktionen sind implementiert:
- Anzeigen / Erstellen / Ändern / Löschen eines Produktes mit Anforderung der Produktliste
- Anzeigen / Erstellen / Ändern / Löschen einer Person, die entweder dem Typen 'Gastronom' oder 'Lieferant' entspricht und Anforderung der Kunden- oder Lieferantenliste
- - Anzeigen / Erstellen / Ändern / Löschen eines Betriebes mit Anforderung der Betriebliste
- Anzeigen / Erstellen / Löschen eines Börseneintrags, der entweder dem Typen 'Gesuch' oder Angebot' entspricht mit Übersicht aller Börseninträge
Wichtig ist auch, dass in jedem 'Service' der Pfad der jeweils benötigten XML bei uns als Variable quasi ausgelagert worden ist, da wir manchmal auf Fehler gestoßen sind, die aufgrund der falschen Pfadangabe aufgetaucht sind. Mittels der Pfadangabe per Variable muss also nur einmal in der Klasse der Pfad angegeben werden und die Fehlerquote wurde minimiert.

29 17.Juni 2013

29.1 Ergebnisse aus Beratungstermin

Nach Rücksprache mit dem Tutor Volker wurden kleinere Fehler behoben, sodass alle Methoden funktionieren. Sowohl in der Betrieb/Börse/Personenservice als auch nun in der ProduktService. Der Fehler lag darin, dass in der Produktliste kein minOccurs und auch kein maxOccurs angegeben war und somit einige Methoden nicht funktionieren konnten.

29.1.1 Fehler in GUI

Die zunächst erstellte GUI erfüllt nicht die geforderten Anforderungen. Die Operationen sollen im Hintergrund ablaufen, sodass der User nicht direkt auswählt, dass er die GET-Methode auf die Ressorce Person auswählen will. Statdessen soll die GUI an das System angelehnt sein und zeigen, wie das fertige System aussehen und funktionieren könnte.
Für die GUI muss man diejeweiligen Objekte erzeugen und die setMethoden übergeben. Wir konzentrieren uns auf die wesentlichen Funktionen wie die Börse aus der die Bestellung resultiert. Mit Hilfe des Window Builders ist es möglich die GUI testweise anzuschauen.

30 18.Juni 2013

Es wird nun an den zwei Hauptbaustellen des Projekts gearbeitet. Zum Einem wird die GUI erstellt und zum Anderen wird die Doku überarbeitet.

30.1 Erstellung einer neuen GUI

GUI wird komplett neugeschrieben. Es ergeben sich mehrere SWING-Dateien:
- Login
- Börse
- Start
- Node
- Nodes

Die Erstellung der neuen GUI erscheint aufwendiger zu werden als zunächst angenommen, da sehr viele Funktionen und Methoden nötig sind um das System zu veranschaulichen.

30.2 Bearbeitung der Dokumentation

Gleichzeitig wird die Dokumentation des Projektes überarbeitet und dem letztendlichen Ergebnis der verschiedenen Meilensteine angepasst. Es wird beschlossen, dass die Dokumentation lediglich die finale Version der einzelnen Komponenten des Systems beschreiben und erläutern soll. Das Logbuch hingegen soll den Verlauf und die Abwägungen im Projekt chonologisch beschreiben.

31 20.Juni 2013

31.1 GUI-Erweiterungen

POST Methode in der Börse:
- Textfeld erzeugen: Wert des Textfeldes wird in der Börse eingespeichert. Daher wird die in der Börse.java der Typ des Titel geändert: von String auf JTextField.

31.1.1 PubSub

Publish/Subscribe per GUI:
In der Datei Publisher.java werden String username und String password gelöscht. Stattdessen erfolgt der Login direkt über die GUI. Es besteht die Frage, ob es eine Liste mit

allen Topics geben soll, aus denen man per Radiobutton wählt in welches Tonic Eintrag gespeichert wird.

Bei der Erstellung von Nodes innerhalb der Publisher.java treten Probleme auf. Man kann nicht auf eine Methode der Klasse Publisher zugreifen! Die jeweils eingegeben Login-Daten 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 GUILogin als Variable gespeichert und im Konstruktor des Publishers übergeben. In der GUINode, auf der man Einträge publishen und subsciben kann, wird ein neues Element von diesem Publisher erzeugt

31.2 Dokumentation

Die Dokumentation ist bis auf den GUI-Teil fertig. Dieser soll zum Schluss erstellt werden, um wie bei den anderen Themen jeweils das Endprodukt beschreiben zu können.

32 22.Juni 2013

32.1 GUI-Börseneintrag

Aus einer Liste können Einträge der Börse ausgewählt und durch einen Klick auf den Button wird dieser Eintrag dann ausgegeben. Zusätzlich zum Eintrag können unter 'Optionen' neue Einträge und Kommentare erstellt und eine Bestellung abgesendet werden. Probleme bereitet die automatische Generierung der zugehörigen Person bei Eingabe der PersonID. Zusätzlich ist nicht ganz klar, wie die Datumseingabe in das Format von XMLGregorianCalender umgewandelt werden kann.

33 23.Juni 2013

33.1 GUI-Bestellung

Es wird versucht, über ein Formular eine Bestellung zu erstellen und in die Bestellungsliste zu speichern. Jedoch funktioniert das Speichern noch nicht.

33.2 Abschluss des Projektes

Da das Ende der vorgegebenen Projektzeit unmittelbar voraus steht, wird beschlossen sich von nun an mit der Vollendung der Dokumentation zu befassen. In der Dokumentation sollen neben den erreichten Umsetzungen auch die nicht nicht erreichten Ziele erläutert

werden. Die Arbeit an der GUI wird zunächst beendet.

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