SBB Fahrplan als Web Service

Florian Amstutz <florian@amstutz.nu>

27. Mai 2013

Zürcher Hochschule für Angewandte Wissenschaften, Seminar
Webprojekte mit ASP.NET

Inhaltsverzeichnis

1 Einleitung

Das Seminar Webprojekte mit ASP.NET wird im Rahmen des Wahlpflichtmoduls Programmierung mit dem .NET Framework besucht und hat die selbstständige Formulierung und Lösung eines Projekts mit ASP.NET zum Ziel hat.

1.1 Auftrag

Im Rahmen des Seminar soll selbstständig ein Projekt zu einem ausgewählten Thema erarbeitet werden. Diese Seminararbeit hatte ursprünglich zum Ziel den SBB Fahrplan mittels des Open Data Protocols zur Verfügung zu stellen. Auf Grund der limitierenden SBB.ch Fahrplan-Webseite wurde die Aufgabenstellung so abgeändert, das anstelle eines OData Services ein Web Service zur Verfügung gestellt wird.

Das Seminarresultat besteht aus einer Fahrplanwebseite analog SBB.ch, welche als Modul in Orchard Project implementiert ist sowie einem Web Service, über welchen Fahrplaninformationen angefragt werden können. Die Fahrplanwebseite benutzt diesen Web Service um die gewünschten Fahrplaninformationen anzuzeigen. Der Web Service selbst greift auf SBB.ch um die Fahrplaninformationen weiterzugeben.

2 Hauptteil

Diese Kapitel beschreibt die Umsetzung der Seminararbeit nach dem Wasserfallmodell. Nach [Kuhrmann(2012)] ist das Wasserfallmodell insbesondere für einfache Projekte mit klar definierten Anforderungen einsetzbar. Da der Umfang dieser Seminararbeit überschaubar ist und der bearbeitende Student der einzige Stakeholder ist hat sich das Wasserfallmodell als ideales Vorgehensmodell für die Bearbeitung von Seminararbeiten herausgestellt. Die einzelnen Unterkapitel des Hauptteils folgen dem Modell (siehe auch [Hung(2007)]).

2.1 Projektplanung

Nach Reglement der ZHAW (siehe [Stern(2012)]) steht der Seminararbeit 50 Stunden zur Verfügung. Das Kick-Off der Seminararbeit fand am 20. März 2013 statt und die Präsentation der Arbeit findet am 12. Juni 2013 statt. Die Seminararbeit muss dabei eine Woche vor der Präsentation abgegeben werden, sprich am 5. Juni 2013. Dadurch dass diese Arbeit im Wasserfallmodell bearbeitet wird, folgt eine Phasenplanung auf Basis dieser Daten im nächsten Kapitel.

2.1.1 Phasenplanung

Nach dem Wasserfallmodell wird das Projekt in einzelne Phasen eingeteilt, die zu einem bestimmten Zeitpunkt mit vordefinierten Endergebnissenen enden. Bei Erreichen des Endzeitpunkts und bei Lieferung aller Endergebnisse geht das Projekt in die nächste Phase über.

Neben dem geplanten Start- und Enddatum werden auch das effektive Start- und Enddatum eingetragen.

Phase	Start (geplant)	Ende (geplant)	Start (effektiv)	Ende (effektiv)
Erfassen der Anforderungen	21.03.2013	04.04.2013	24.03.2013	01.04.2013
Erarbeiten des Konzepts	04.04.2013	21.04.2013	01.04.2013	18.04.2013
Implementierungs des Konzepts	21.04.2013	01.05.2013	18.04.2013	12.05.2013
Überprüfen und Test des Konzepts	01.05.2013	03.05.2013	12.05.2013	14.05.2013
Fertigstellung Seminarbericht	03.05.2013	01.06.2013	14.05.2013	29.05.2013
Erarbeiten der Präsentation	01.06.2013	05.06.2013	29.05.2013	31.05.2013

Tabelle 2.1: Phasenplan

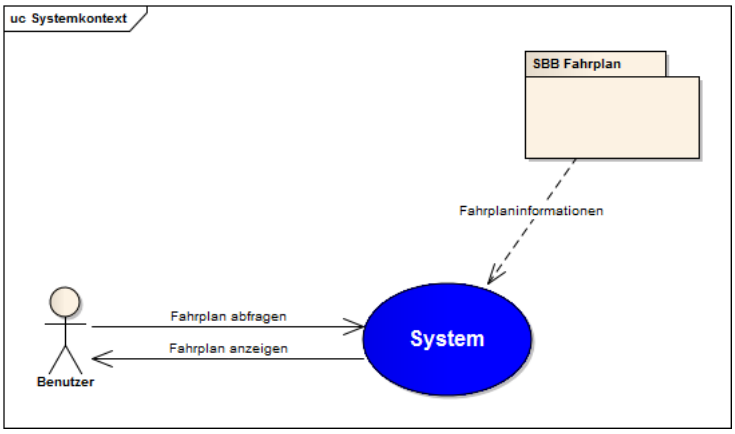

Abbildung 2.1: Systemkontext

2.2 Anforderungen

Der erste Schritt des Wasserfallmodells besteht darin die Anforderungen an das zu entwickelnde System festzuhalten. Aus Gründen der Einfachheit wurde dabei ein einziger Use-Case identifiziert, der im Rahmen dieser Arbeit implementiert werden wird (siehe Kapitel 2.2.2).

Um überhaupt erst Use-Cases spezifizieren zu können, beschreibt das erste Unterkapitel den Systemkontext des geplanten Systems OpenTimeTable.

2.2.1 Systemkontext

Der Systemkontext beschreibt die Sollperspektive, wie sich das System in die Realität integrieren wird. Hierdurch wird der Realitätsausschnitt identifiziert, der das System und damit potenziell auch dessen Anforderungen beeinflusst. Um die Anforderungen an das geplante System korrekt und vollständig spezifizieren zu können, ist es notwendig, die Beziehung zwischen den einzelnen materiellen und immateriellen Aspekten im Systemkontext und dem geplanten System exakt zu definieren. Der für die Anforderungen des Systems relevante Ausschnitt der Realität wird als Systemkontext bezeichnet (nach [Klaus Pohl(2011)]).

Der Ursprung der Anforderungen des Systems liegt im Systemkontext des geplanten Systems. Aus diesem Grund wird der Systemkontext vor Erhebung und Dokumentierung der Anforderungen festgelegt. Der Systemkontext des Systems OpenTimeTable ist in Abbildung 2.1 als Modell dargestellt.

2.2.2 Use-Case-Spezifikation

Nach [Klaus Pohl(2011)] zeigen Use-Case-Diagamme die aus einer externen Nutzungssicht wesentlichen Funktionalitäten des betrachteten Systems sowie spezifische Beziehungen der einzelnen Funktionalitäten untereinander beziehungsweise zu Aspekten in der

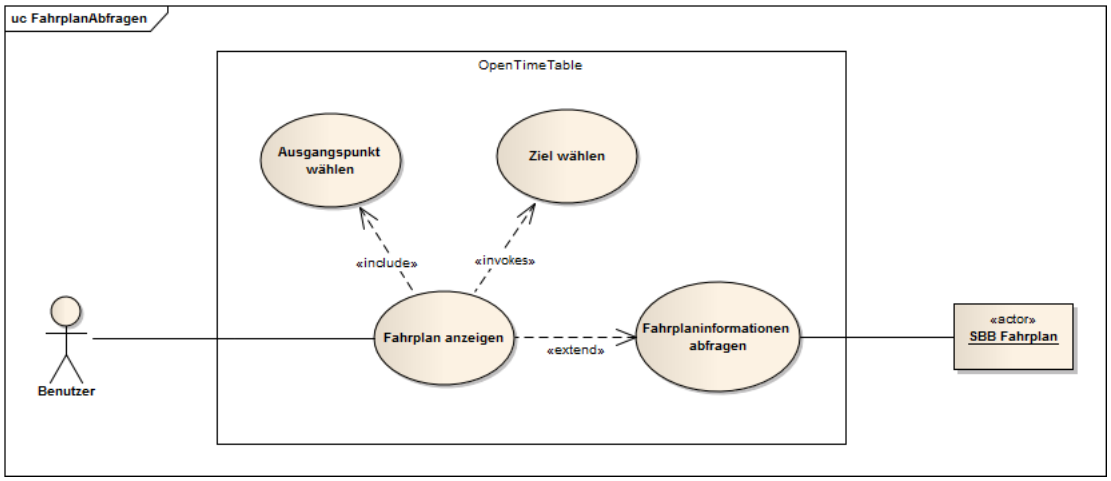

Abbildung 2.2: Use-Case Fahrplan abfragen

Umgebung des Systems. Abgesehen vom Namen des Use-Cases und dessen Beziehungen dokumentieren Use-Case-Diagramme keinerlei weitere Informationen über die einzelnen Use-Cases, wie z.B. die Systematik der Interaktion eines Use-Case mit Akteuren in der Umgebung. Diese Informationen werden unter Verwendung einer geeigneten Schablone zusätzlich zum Use-Case-Diagramm textuell als Use-Case-Spezifikation festgehalten.

Alle funktionalen Anforderungen werden als Use-Cases modelliert und spezifiziert[1].

2.2.2.1 Fahrplan abfragen

Der Benutzer möchte den Fahrplan zwischen Ausgangs- und Zielpunkt zu einem bestimmten Zeitpunkt abfragen. Das System erstellt aus den eingegebenen Daten und den Fahrplaninformationen einen personalisierten und individuellen Fahrplan und zeigt diesen dem Benutzer an.

[1] Die verwendete Schablone stammt aus [Klaus Pohl(2011)] und dient zur zweckmässigen Strukturierung von Typen von Informationen, die einen Use-Case betreffen. Die vorgeschlagenen Abschnitte der Schablone Autor, Quelle, Verantwortlicher und Qualität werden ausgelassen, da sie im Rahmen dieses Projekts keinen zusätzlichen Nutzen bringen.

Abschnitt	Inhalt
Bezeichner	UC1
Name	Fahrplan abfragen
Priorität	Wichtigkeit für Systemerfolg: hoch Technologisches Risiko: hoch
Kritikalität	Hoch
Beschreibung	Der Benutzer fragt den Fahrplan ab.
Auslösendes Ereignis	Der Benutzer möchte Informationen zum Fahrplan haben.
Akteure	Benutzer
Vorbedingung	Keine
Nachbedingung	Der Benutzer kann eine erneute Fahrplanabfrage starten.
Ergebnis	Der Fahrplan wird den Eingaben des Benutzers entsprechend dargestellt.
Hauptszenario	1. Der Benutzer gibt den Ausgangspunkt an. 2. Der Benutzer gibt das Ziel an. 3. Der Benutzer gibt die gewünschte Abfahrtszeit ein. 4. Das System erstellt die Fahrplaninformationen basierend auf den Eingaben des Benutzers und unter Einbezug des SBB Fahrplans.
Alternativszenarien	1a. Der Ausgangspunkt ist unbekannt. 1a1. Dem Benutzer wird eine Auswahlliste von passenden Ausgangspunkten angezeigt. Er wird aufgefordert einen der Ausgangspunkte auszuwählen oder erneut einen Ausgangspunkt einzugeben. 2a. Das Ziel ist unbekannt. 2a1. Dem Benutzer wird eine Auswahlliste von passenden Zielen angezeigt. Er wird aufgefordert eines der Ziele auszuwählen oder erneut ein Ziel einzugeben.
Ausnahmeszenarien	Auslösendes Ereignis: Der Benutzer kann keine Verbindung zum System herstellen.

Tabelle 2.2: Use-Case-Spezifikation Fahrplan abfragen

2.3 Konzept und Architektur

Die Konzeptphase[2] des Wasserfallmodells behandelt die Entwicklung eines vollständigen und umfassenden Lösungskonzepts auf Basis der dokumentierten Anforderungen. Zuerst wird das System aus der Bausteinperspektive betrachtet, es wird von der Komponentenebene bis zur Klassenebene das System modelliert und die Systemarchitektur festgelegt. Als weitere Sicht wird die Laufzeitsicht und die Verteilungssicht des Systems beleuchtet und spezifziert.

[2]auch Designphase genannt

2.3.1 Bausteinsicht

Nach [Starke(2011)] und [Peter Hruschka(2013)] lassen sich unter dem Begriff "Bausteine" sämtliche Software- oder Implementierungskomponenten zusammenfassen, die letztendlich Abstraktionen von Quellcode darstellen. Dazu gehören Klassen, Prozeduren, Programme, Pakete, Komponenten (nach der UML-Definition) oder Subsysteme.

Die Bausteinsicht bildet die Aufgaben des System auf Software-Bausteinen oder -Komponenten ab. Diese Sicht macht Struktur und Zusammenhänge zwischen den Bausteinen der Architektur explizit. Bausteinsichten zeigen die statischen Aspekte des Systemes und entsprechen in dieser Hinsicht den konventionellen Implementierungsmodellen.

2.3.1.1 Komponentendiagramm

Das Komponentendigramm in Abbildung 2.3 stellt das System OpenTimeTable aus der Vogelperspektive dar und ist die höchstabstrahierte Ansicht der Bausteinsicht, die in diesem Projekt verfügbar ist.

Die Fahrplan-Webseite SBB.ch stellt einen XML Fahrplan zur Verfügung welcher von der Serverapplikation genutzt wird um Fahrplandaten aufzubereiten und danach der Clientapplikation weiterzugeben.

Der OpenTimeTable.Service bezeichnet die Serverapplikation des Systems und implementiert die beiden Schnittstellen[3] IConnectionService und ILocationService (siehe Kapitel 2.3.1.3 auf der nächsten Seite). Er stellt die Schnittstellen als Webservices der Clientapplikation zur Verfügung.

Die Clientapplikationen OpenTimeTable.Client, die als Modul innerhalb von Orchard realisiert wird, greift auf die Serverapplikation über die beiden Schnittstellen zu und stellt die zurückgegeben Daten dem Benutzer dar.

[3] im Zusammenhang mit WCF auch Service Contracts genannt

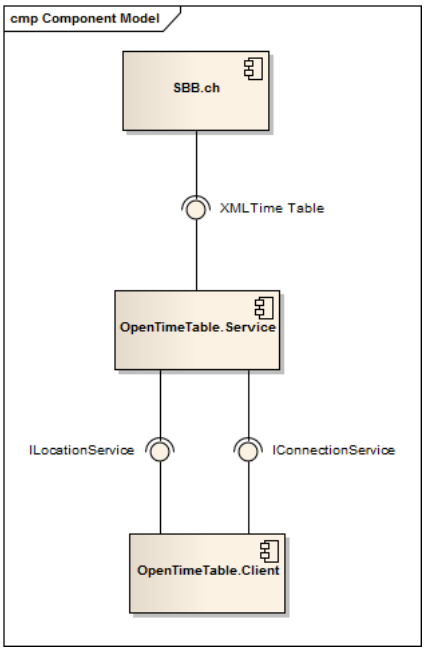

Abbildung 2.3: Komponentendiagramm

2.3.1.2 Domänenmodell

Das Domänenmodell (siehe Abbildung 2.4) umfasst nur die Klassen der Objekte, die über die Serviceschnittstelle von der Client- an die Serverapplikation beziehungsweise umgekehrt übertragen werden.

OpenTimeTable verwendet ein vergleichsweise simples und selbsterklärendes Domänenmodell, das für jeden Benutzer eines Fahrplaninformationssystems wie SBB.ch ohne weitere Erklärung verständlich sein sollte.

2.3.1.3 Service Contracts

Ein Service Contract spezifiziert eine Schnittstelle zur Kommunikation verschiedener Applikationen innerhalb eines verteilten Systems. Häufig werden diese Service Contracts als Webservices angeboten, da sie dadurch plattform- und frameworkunabhängig genutzt werden können.

Typischerweise umfasst ein Service Contract mehrere Operationen, deren Rückgabewerte als XML-Fragmente an die konsumierende Applikation zurückgegeben werden. Ausserdem ist ein Service Contract per Definition grundsätzlich zustandslos, er behandelt mehrere Anfragen (auch desselben Auftraggebers) immer als unabhängige Transaktionen. Anfragen werden ohne Bezug zu früheren, vorhergegeganen Anfragen behandelt und es werden auch keine Sitzungsinformationen ausgetauscht.

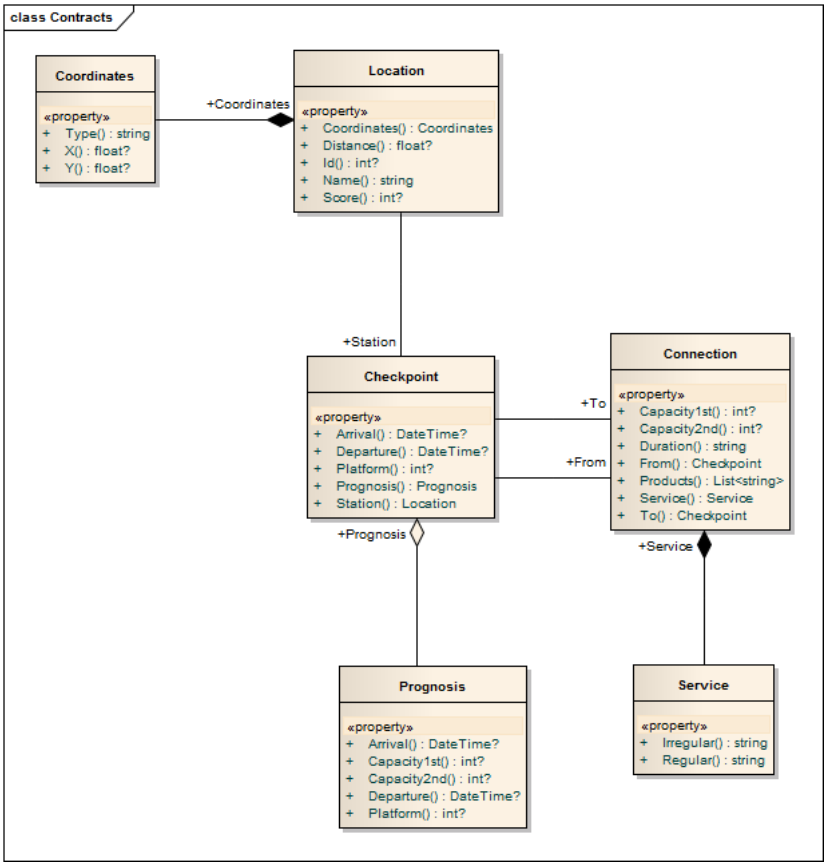

Abbildung 2.4: Klassendiagramm Domänenmodell

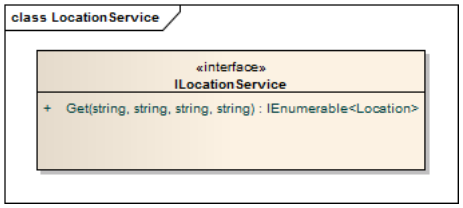

Abbildung 2.5: Klassendiagramm ILocationService

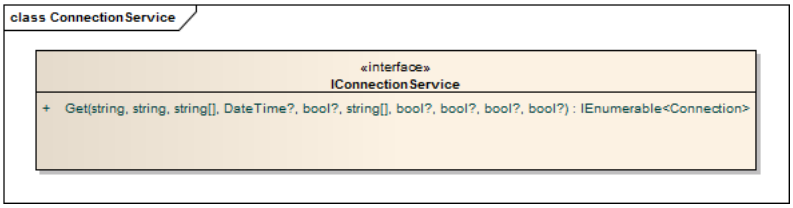

Abbildung 2.6: Klassendiagramm IConnectionService

ILocationService Die in Abbildung 2.5 gezeigte Schnittstelle ILocationService stellt Funktionalitäten zur Verfügung über welche Haltestellen und Bahnhöfe abgefragt werden können.

IConnectionService Der IConnectionService in Abbildung 2.6 gibt Verbindungen zurück, welche über verschiedene Parameter gesucht werden können.

2.3.2 Laufzeitsicht

Die Laufzeitsicht beschreibt, welche Bestandteile des Systems zur Laufzeit existieren und wie diese zusammenwirken (nach [Starke(2011)]). Dabei kommen wichtige Aspekte des Systembetriebs ins Spiel, die beispielsweise den Systemstart, die Laufzeitkonfiguration oder die Administration des Systems betreffen.

Darüber hinaus dokumentiert die Laufzeitsicht, wie Laufzeitkomponenten sich aus Instanzen von Implementierungsbausteinen zusammensetzen.

2.3.2.1 Fahrplan abfragen

Das Sequenzdiagramm in Abbildung 2.7 entspricht dem Use-Case Fahrplan abfragen (siehe Kapitel 2.2.2.1 auf Seite 6).

Der Benutzer greift auf die Connection Request View über das MVC-Pattern von Orchard zu und gibt die gewünschten Verbindungsdaten für den Fahrplan ein. Danach greift der TimeTableController des Clients auf den Server zu und generiert anhand der vom Benutzer übermittelten Daten einen SOAP-Request. Der SOAP-Request wird an den Web Server von SBB.ch gesendet, die Response wird vom Server umformatiert und dem Client zurückgegeben, welcher diese in der Time Table Result View dem Benutzer präsentiert.

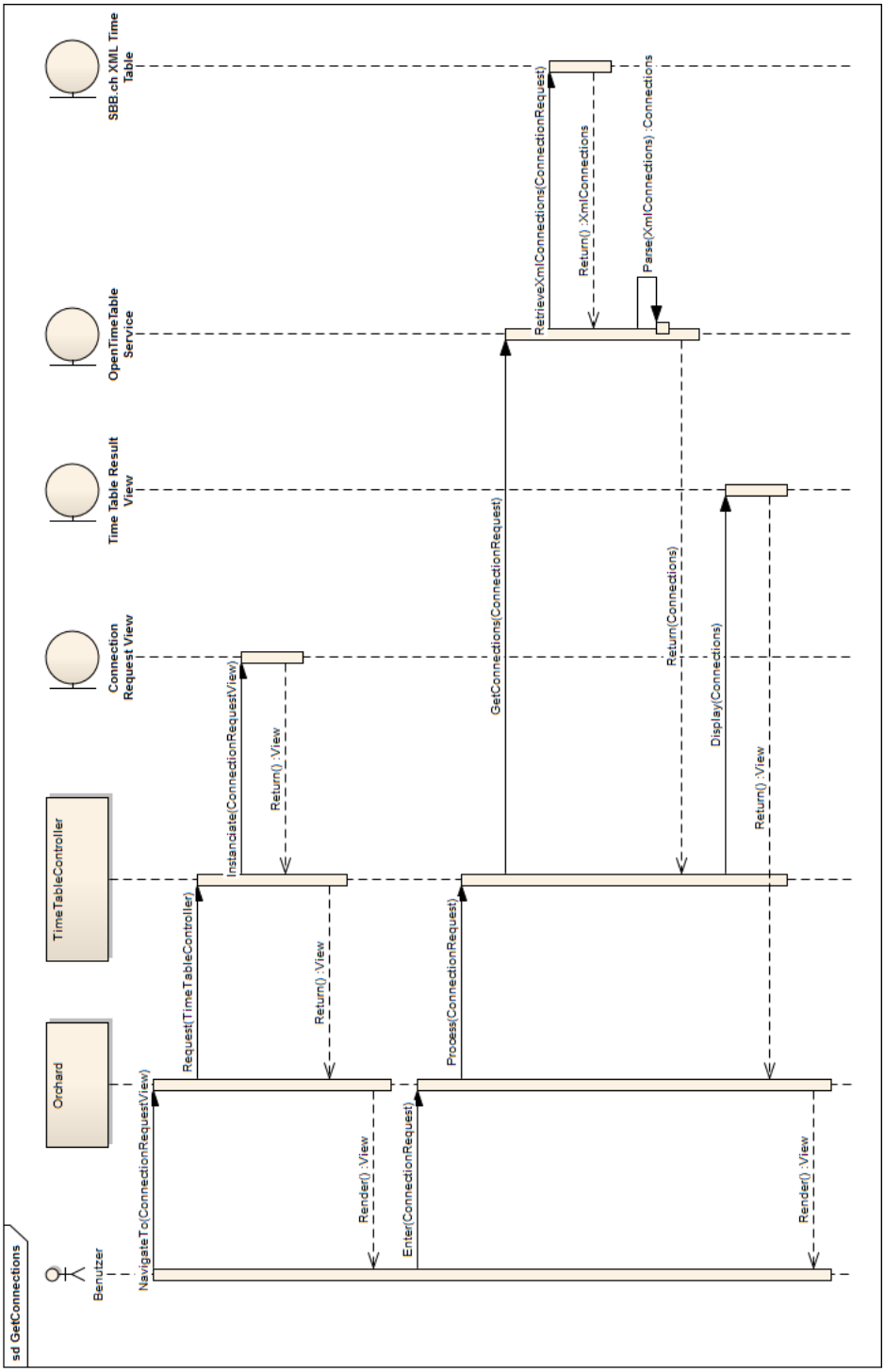

Abbildung 2.7: Sequenzdiagramm Fahrplan abfragen

2.3.3 Verteilungssicht

Nach [Starke(2011)] beschreibt die Verteilungssicht die Ablaufumgebung des Systems in Form von Hardwarekomponenten (wie Prozessoren, Speicher, Netzwerk, Router und Firewalls) sowie den beteiligten Protokollen. In der Infrastruktursicht können die Leistungsdaten und Parameter der beteiligten Elemente dargestellt werden. Ausserdem werden zusätzlich Betriebssysteme oder externe Systeme aufgenommen.

Die Verteilungssicht ist von grosser Beduetung für die Betreiber des Systems, die Hardwarearchitekten, das Entwicklungsteam sowie Management und Projektleitung (gemäss [Peter Hruschka(2013)]).

2.3.3.1 Verteilungsdiagramm

Die Verteilungssicht dieser Projektdokumentation enthält nur ein sehr rudimentär ausgearbeitetes Verteilungsdiagramm (siehe Abbildung 2.8). Dies, da kein konkretes Verteilungsszenario der Applikation innerhalb des Projekts geplant wurde. Das Projekt beeinhaltet die Erarbeitung des Konzepts sowie die konkrete Implementierung der Applikation ohne jedoch auf die Verteilung des Systems einzugehen.

Zu beachten ist, dass der OpenTimeTable.Service und der OpenTimeTable.Client beide im selben IIS und damit auf dem selben Server laufen könnten, dies jedoch auch unterschiedliche Rechner sein könnten. Der OpenTimeTable.Client wird als Modul von Orchard implementiert und wird deswegen immer innerhalb einer Orchard-Instanz angeboten.

2.4 Implementierung

Dieses Kapitel enthält Informationen zur Implementierung der beiden Teile, der Server- und der Clientapplikation. Dabei wird auf einzelne Besonderheiten in der Implementierung hingewiesen sowie die Struktur der Projekte erklärt.

Da sämtlicher Quellcode und die Dokumentation auf GitHub verfügbar[4] ist wird in diesem Kapitel nur rudimentär auf die eigentliche Implementierung der beiden Applikationen eingegangen.

2.4.1 OpenTimeTable.Service

Die Serverapplikation ist als WCF Web Service implementiert und besteht aus zwei Teilen. Die Assembly OpenTimeTable.Service.Host enthält die Service Contracts sowie deren Implementierung, die Assembly OpenTimeTable.Model.Contracts enthält das Domänenmodell (siehe Kapitel 2.3.1.2 auf Seite 9).

[4] Siehe `https://github.com/famstutz/OpenTimeTable`

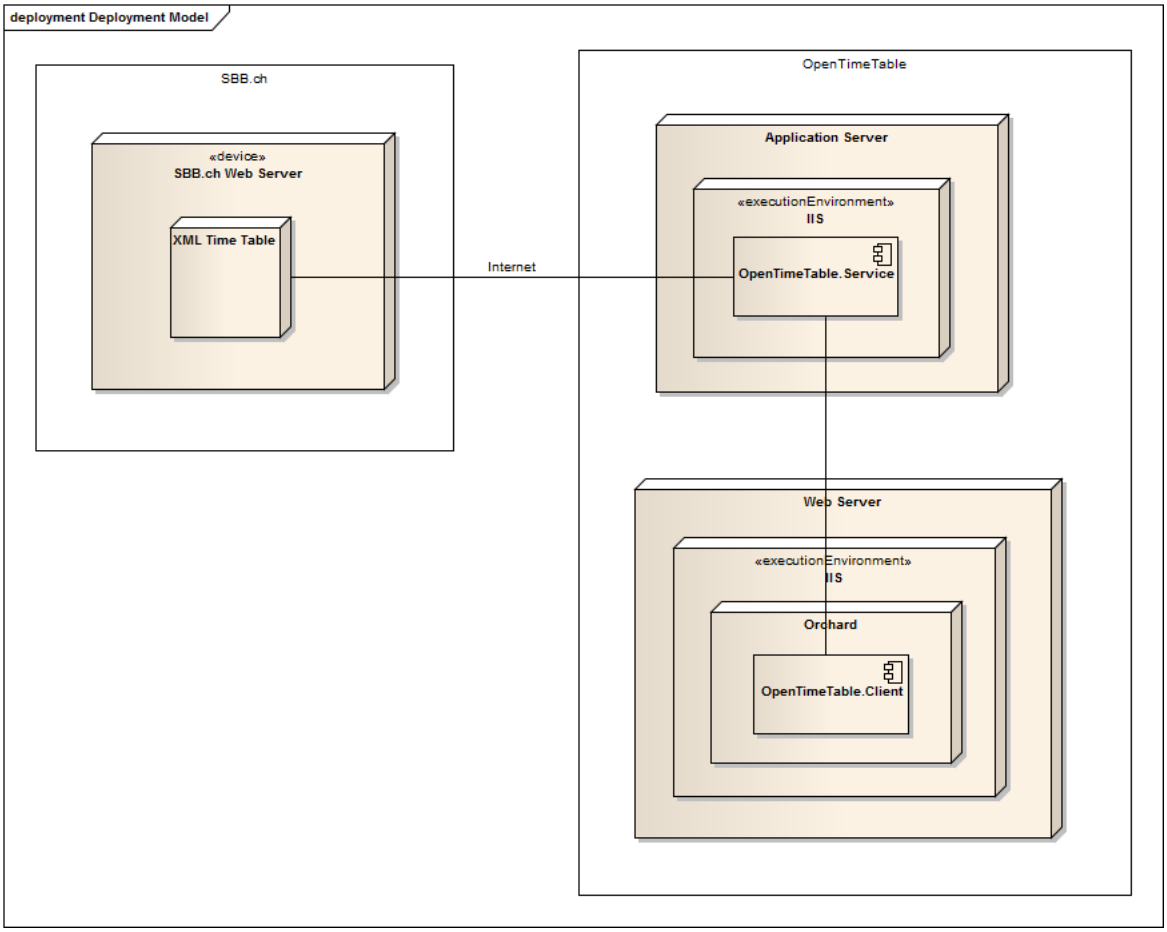

Abbildung 2.8: Verteilungsdiagramm

2.4.2 OpenTimeTable.Client

Die Clientapplikation ist als Modul für das Content Management System Orchard Project[5] geschrieben. Orchard bietet ein Plugin-System über Microsofts WebMatrix, welches eine eigene Visual Studio Solution mit dem erstellen Projekt breitstellt. Im Rahmen der Bearbeitung der Seminararbeit wurde jedoch festgestellt, dass gewisse Funktionalitäten von Orchard über die WebMatrix nicht verfügbar sind und deswegen der gesamte Quellcode von Orchard verwendet werden muss, um zum Beispiel Service References der Orchard-Instanz bekannt zu machen.

Das Modul selber wird mittels ASP.NET MVC implementiert und verwendet die Assembly OpenTimeTable.Model.Contracts aus dem vorherigen Kapitel um die gesuchten Fahrplaninformationen anzuzeigen.

2.5 Verifikation

Die Verifikation der implementierten Anforderungen (siehe Kapitel 2.2 auf Seite 5) an das System wurde manuell vorgenommen und ist im folgenden Kapitel beschrieben.

2.5.1 Manuelle Integrationstests

Um die Funktionstüchtigkeit des Systems zu überprüfen wurden drei verschiedene Fahrplanabfragen sowohl auf SBB.ch wie auch in OpenTimeTable durchgeführt. Sind die angezeigten Fahrplandaten identisch, so wird der jeweilge Testfall als erfüllt betrachtet.

2.5.1.1 Testfall 1 (Genf - Zürich, 5. Mai 2013, 08:30)

Bei einer Fahrplanabfrage von Genf nach Zürich am 5. Mai 2013 mit Abfahrtszeit 08:30 stellt SBB.ch und OpenTimeTable exakt dasselbe Resultat dar.

Der Testfall 1 ist somit als erfolgreich zu betrachten.

	Bahnhof/Haltestelle	Datum	Zeit		Dauer	Umst.	Reise mit
1	Genève	So, 05.05.13	ab	08:45	2:51	0	IC
	Zürich HB		an	11:36			
2	Genève	So, 05.05.13	ab	09:18	2:38	1	ICN
	Zürich HB		an	11:56			
3	Genève	So, 05.05.13	ab	09:45	2:51	0	IC
	Zürich HB		an	12:36			
4	Genève	So, 05.05.13	ab	10:18	2:38	0	ICN
	Zürich HB		an	12:56			

Abbildung 2.9: Resultat Testfall 1 SBB.ch

[5] Siehe `http://www.orchardproject.net/`

Station/Stop	Date	Time	Duration	Travel with	Information
Genève Zürich HB	Departure: 05.05.2013 Arrival: 05.05.2013	08:45 11:36	02:51:00	IC	
Genève Zürich HB	Departure: 05.05.2013 Arrival: 05.05.2013	09:18 11:56	02:38:00	ICN	
Genève Zürich HB	Departure: 05.05.2013 Arrival: 05.05.2013	09:45 12:36	02:51:00	IC	
Genève Zürich HB	Departure: 05.05.2013 Arrival: 05.05.2013	10:18 12:56	02:38:00	ICN	

Abbildung 2.10: Resultat Testfall 1 OpenTimeTable

2.5.1.2 Testfall 2 (Ziegelbrücke - Bellinzona, 12. April 2013, 19:15)

Auch bei der Fahrplanabfrage von Ziegelbrücke nach Bellinzona am 12. April 2013 mit Abfahrtszeit 19:15 stellen sowohl SBB.ch als auch OpenTimeTable dieselben Fahrplaninformationen dar.

Der Testfall 2 ist somit als erfolgreich zu betrachten.

	Bahnhof/Haltestelle	Datum	Zeit	Dauer	Umst.	Reise mit
1	Ziegelbrücke	Fr, 12.04.13	ab 19:37	2:46	2	RE, IR, ICN
	Bellinzona		an 22:23			
2	Ziegelbrücke	Fr, 12.04.13	ab 20:01	3:22	1	IR, ICN
	Bellinzona		an 23:23			
3	Ziegelbrücke	Fr, 12.04.13	ab 21:01	3:41	1	IR
	Bellinzona	Sa, 13.04.13	an 00:42			
4	Ziegelbrücke	Fr, 12.04.13	ab 21:21	3:21	2	S2, S, IR
	Bellinzona	Sa, 13.04.13	an 00:42			

Abbildung 2.11: Resultat Testfall 2 SBB.ch

Station/Stop	Date	Time	Duration	Travel with	Information
Ziegelbrücke Bellinzona	Departure: 16.04.2013 Arrival: 16.04.2013	19:37 22:23	02:46:00	RE, IR, ICN	
Ziegelbrücke Bellinzona	Departure: 16.04.2013 Arrival: 16.04.2013	20:01 23:23	03:22:00	IR, ICN	
Ziegelbrücke Bellinzona	Departure: 16.04.2013 Arrival: 17.04.2013	21:01 00:42	03:41:00	IR	
Ziegelbrücke Bellinzona	Departure: 16.04.2013 Arrival: 17.04.2013	21:21 00:42	03:21:00	S2, S, IR	

Abbildung 2.12: Resultat Testfall 2 OpenTimeTable

2.5.1.3 Testfall 3 (Zürich, Central - Zürich, Sihlpost, 26. Mai 2013, 12:00)

Bei einer innerstädtischen Fahrplanabfrage von Zürich, Central nach Zürich, Sihlpost am 26. Mai 2013 mit Abfahrtszeit 12:00 zeigten sich Unterschiede zwischen der Anzeige von

SBB.ch und derjenigen von OpenTimeTable. Im Gegensatz zu den Langstreckenverbindungen scheint SBB.ch bei innerstädtischen Verbindungen die angegegebene Ankunftszeit als spätmöglichste Ankunftszeit zu betrachten. Dies ist jedoch nur auf der Webseite der Fall, verwendet man die iOS-App der SBB so werden dieselben Fahrplaninformationen wie auf OpenTimeTable dargestellt.

Der Testfall 3 ist auf Grund der Definition in Kapitel 2.5.1 somit als fehlgeschlagen zu betrachten.

	Bahnhof/Haltestelle	Datum	Zeit		Dauer	Umst.	Reise mit
1	Zürich, Central	So, 26.05.13	ab	11:51	0:04	0	NFB
	Zürich, Sihlpost		an	11:55			
2	Zürich, Central	So, 26.05.13	ab	11:51	0:16	0	NFT
	Zürich, Bahnhofplatz/HB		an	11:53			
3	Zürich, Central	So, 26.05.13	ab	11:53	0:19	0	T
	Zürich, Bahnhofquai/HB		an	11:55			
4	Zürich, Central	So, 26.05.13	ab	11:55	0:20	0	T
	Zürich, Bahnhofstrasse/HB		an	11:58			
5	Zürich, Central	So, 26.05.13	ab	11:57	0:18	0	TRO
	Zürich, Bahnhofquai/HB		an	11:58			
6	Zürich, Central	So, 26.05.13	ab	11:59	0:05	0	NFT
	Zürich, Sihlpost		an	12:04			
7	Zürich, Central	So, 26.05.13	ab	11:59	0:16	0	NFT
	Zürich, Bahnhofplatz/HB		an	12:01			

Abbildung 2.13: Resultat Testfall 3 SBB.ch

Station/Stop	Date	Time	Duration	Travel with	Information
Zürich, Central Zürich, Sihlpost	Departure: 26.05.2013 Arrival: 26.05.2013	12:01 12:05	00:04:00	NFB	
Zürich, Central Zürich, Bahnhofplatz/HB	Departure: 26.05.2013 Arrival: 26.05.2013	12:01 12:03	00:16:00	NFT	
Zürich, Central Zürich, Bahnhofquai/HB	Departure: 26.05.2013 Arrival: 26.05.2013	12:03 12:05	00:19:00	NFT	
Zürich, Central Zürich, Bahnhofstrasse/HB	Departure: 26.05.2013 Arrival: 26.05.2013	12:05 12:08	00:20:00	NFT	
Zürich, Central Zürich, Bahnhofquai/HB	Departure: 26.05.2013 Arrival: 26.05.2013	12:07 12:08	00:18:00	TRO	
Zürich, Central Zürich, Sihlpost	Departure: 26.05.2013 Arrival: 26.05.2013	12:09 12:14	00:05:00	T	

Abbildung 2.14: Resultat Testfall 3 OpenTimeTable

2.5.2 Testresultate

Auch wenn der Testfall 3 (siehe Tabelle 2.3) als fehlgeschlagen betrachtet werden muss, lässt sich trotzdem sagen, dass die Verifikation des Systems OpenTimeTable als erfolgreich abgeschlossen wurde. Der Benutzer erhält auch in Testfall 3 die gewünschten Fahrplaninformationen, es ist dabei auf die Inkonsistenz von SBB.ch in der Anzeige von Verbindungen hinzuweisen.

Testfall	Fahrplaninformationen	Resultat
Testfall 1	Genf - Zürich, 5. Mai 2013, 08:30	Bestanden
Testfall 2	Ziegelbrücke - Bellinzona, 12. April 2013, 19:15	Bestanden
Testfall 3	Zürich, Central - Zürich, Sihlpost, 26. Mai 2013, 12:00	Fehlgeschlagen

Tabelle 2.3: Testresultate

3 Schlussteil

Dieses Kapitel enthält abschliessende Betrachtungen im Rückblick auf die Seminararbeit sowie die Abweichungen von der ursprünglich eingereichten Aufgabenstellung.

3.1 Fazit

Abschliessend lässt sich sagen, dass die Bearbeitung dieser Seminararbeit aufwändiger war, als zuerst angenommen. Die Verwendung von Orchard als CMS-Container hat teilweise einige Steine in den Weg gelegt, die nicht aufgetreten wären, wenn eine eigene Webapplikation auf Basis von ASP.NET MVC erstellt worden wäre. Auch der ursprüngliche Plan, die Fahrplandaten als Open Data Service Protocol Service zurückzugeben, der sich dann als unmöglich herausgestellt hat (siehe nächstes Kapitel), hat viel Zeit gekostet.

Nichts desto trotz hat mir diese Seminarbeit gut gefallen, habe ich doch einige Zeit mit dem OData Protokoll und dem Orchard Project verbracht, was ich sicher auch in Zukunft wieder einsetzen kann.

3.2 Abweichungen von der Aufgabenstellung

Da der SBB.ch XML-Fahrplan nach einer Anzahl von zurückgegebenen Resultaten alle weiteren Anfragen blockiert, war es nicht möglich Open Data Protocol Services für die Fahrplandaten zu schreiben, da ein OData-Service jeweils die gesamte Menge aller Objekte zurückgibt und der Konsument des Services danach eine Einschränkung trifft, welches Untermenge der gesamten Objekten ihn interessieren. Aus diesem Grund wurde anstelle eines OData-Services ein Web Service erstellt.

Die Aufgabenstellung sah auch vor, dass das System mittels TDD erstellt wird sowie automatisierte Integrationstests umgesetzt werden. In Anbetracht des Kapitels 2 des Reglements Seminararbeit (siehe [Stern(2012)]), welches einen Aufwand für die Seminararbeit von 50 Stunden pro Studenten vorsieht, wurde auf diese Punkte verzichtet, da bereits ohne automatisierte Testsuite über 50 Stunden für die Bearbeitung dieser Seminararbeit aufgewendet wurden.

Glossar

ASP.NET ASP.NET ist ein Applikationsframework für Webapplikation und das Erzeugen von dynamischen Websites.

CMS Ein Content Management System (kurz CMS) ist eine Applikation die das Erstellen, Bearbeiten, Organisieren und Publizeren von Inhalten über eine zentrale Schnittstelle.

GitHub GitHub ist ein webbasierter Service für Softwareentwicklungsprojekte, welches das Git Versionsverwaltungssystem nutzen.

IIS Die Internet Information Services (kurz IIS) ist ein Webserver, welcher mit Microsoft Windows mitgeliefert wird und die Standardlösung für das Bereitstellen von ASP.NET-basierten Webseiten im Internet darstellt.

iOS iOS ist ein mobiles Betriebssystem das von Apple für das iPhone und das iPad entwickelt wurde.

OData Das Open Data Protocol (kurz OData) ist ein Datenzugriffsprotokoll, um Daten abzufragen und zu aktualisieren.

Orchard Project Das Orchard Project ist ein frei verfügbares und Open-Source CMS das in ASP.NET unter Verwendung des ASP.NET MVC Frameworks geschrieben wurde.

MVC Model-View-Controller (kurz MVC) ist ein Softwarearchitekturpattern welches die Informationsrepräsentation von der dem Benutzer möglichen Interaktion entkoppelt.

SOAP Das Simple Object Access Protocol (kurz SOAP) ist eine Protokollspezifikation für das austuaschen von strukturierten Informationen über Web Services.

TDD Testgetriebene Entwicklung (englisch kurz TDD) ist ein Softwareentwicklungsprozess das auf der Wiederholung eines sehr kurzen Entwicklungszyklus beruht: Schreiben eines Testcases, minimalen Code um das Testcase zu befriedigen, Refactoring.

WCF Die Windows Communication Foundation (kurz WCF) ist eine Laufzeitumgebung und APIs im .NET Framework für das Entwickeln von verbundenen, serviceorientierten Applikationen.

WebMatrix Die Microsoft WebMatrix ist ein Tool um Webseiten über die Cloud zu Erstellen, Verteilen und Unterhalten.

Literaturverzeichnis

[Hung(2007)] Vo-Trung Hung. Software development process. 2007. URL `http://cnx.org/content/m14619/1.2/`.

[Klaus Pohl(2011)] Chris Rupp Klaus Pohl. *Basiswissen Requirements Engineering*. dpunkt.verlag, 2011.

[Kuhrmann(2012)] Marco Kuhrmann. Wasserfallmodell. *Enzyklopädie der Wirtschaftsinformatik*, 2012. URL `http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/is-management/Systementwicklung/Vorgehensmodell/Wasserfallmodell/`.

[Peter Hruschka(2013)] Gernot Starke Peter Hruschka. arc42 - resourcen für softwarearchitekten. 2013.

[Starke(2011)] Gernot Starke. *Effektive Software-Architekturen*. Carl Hanser Verlag, 2011.

[Stern(2012)] Olaf Stern. Reglement: Seminararbeit. 2012.