Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Mireille Manto

Vergleich von AJAX unter Java und ASP.NET anhand einer Fallstudie

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Mireille Manto

Vergleich von AJAX unter Java und ASP.NET anhand einer Fallstudie

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Olaf Zukunft
Zweitgutachter: Prof. Dr. Bettina Buth

Eingereicht am: 12.September 2011

Mireille Manto

Thema der Arbeit

Vergleich von AJAX unter Java und ASP.NET anhand einer Fallstudie

Stichworte

AJAX, Google Web Toolkit, ASP.NET, Metriken, FCM Modell, GQM Modell, RMI, Softwarequalität, Webframework, Java, C-Sharp, Web-Test

Kurzzusammenfassung

Bei der großen Anzahl an Webframeworks auf dem Markt besteht die Schwierigkeit für Unternehmen darin, ein gutes Framework zu finden. In dieser Arbeit wird die AJAX-Technologie anhand zweier Webframeworks bewertet. Die Bewertung erfolgt Anhand eines im Rahmen dieser Arbeit entwickelten Informationssystems in den beiden Webframeworks. Die Informationssysteme werden mit den vorher definierten Metriken sowohl lokal als auch entfernt getestet. Außerdem werden die Testergebnisse analysiert und ausgewertet.

Mireille Manto

Title of the paper

Comparison of AJAX under Java and ASP.NET based on a case study

Keywords

AJAX, Google Web Toolkit, ASP.NET, Metrics, FCM model, GQM model, RMI, Software quality, Web application framework, Java, c sharp, web testing

Abstract

Frameworks have been pioneering innovative and blended solutions for some of the largest companies in the world. However, the complexity and diversity of these frameworks make the selection of the best framework, a non-trivial task. This work provides an evaluative study of two different Web frameworks offered by the AJAX Technology. A set of information systems, designed in this paper, will serve as basis to the evaluation of those two web-frameworks. The quality of the proposed information systems will be locally and remotely tested using some predefined metrics. Furthermore, test results will be analyzed and interpreted.

Danksagung

An dieser Stelle möchte ich meinen Dank denjenigen aussprechen, die mir bei der Realisierung dieser Arbeit zur Seite standen.

Als erstes möchte ich mich bei Prof. Dr. Olaf Zukunft für seine hervorragende Betreuung während meiner Bachelorarbeit bedanken.

Für das Korrekturlesen möchte ich mich bei Herrn Hermand Dieumo Kenfack, Till Gerken und Markus Beck bedanken.

Des Weiteren möchte ich mich bei meinen Eltern für die moralische Unterstützung bedanken.

Ohne diese Menschen wäre diese Arbeit nicht zu Stande gekommen.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einführung

Vor zwanzig Jahren als das Projekt „Advanced Research Project Agency“ (ARPA) des US-Verteidigungsministeriums entstand, konnte man sich nicht vorstellen, dass das Internet heutzutage einen so großen Einfluss im Leben der Menschen haben würde. Es wurde zunächst für die Vernetzung von Universitäten und Forschungseinrichtungen verwendet und hatte zum Ziel, die knappen Rechenkapazitäten sinnvoll zu nutzen, erst in den USA, später weltweit. Heutzutage ist das Internet sowohl eine Wirtschaftsquelle als auch ein Kommunikationsmedium, das aus dem Leben der Menschen nicht weg zu denken ist. Unternehmer benutzen das Internet um Kunden zu gewinnen. Dafür wurden zunächst Webanwendungen mit den Standard-Werkzeugen wie HTML (Hypertext Markup Language), CSS (Cascading Style Sheets) und JavaScript entwickelt. Im Laufe der Zeit reichten diese Standard-Werkzeuge nicht mehr aus, um komplexe Webanwendungen umzusetzen. So entstanden Webframeworks, mit dem Ziel bessere und schnellere dynamische Webseiten zu entwickeln. Sie bieten dem Entwickler ein Programmiergerüst an. Sie geben dem Entwickler Entwurfsmuster und fertige Bibliotheken vor. Es wurden ständig neue Webframeworks entwickelt mit dem Ziel, besser zu sein als die Konkurrenz auf dem Markt und die Anforderungen der Kunden besser zu erfüllen. So sind zahlreiche Frameworks entstanden, welche auf verschiedenen Wegen den Programmierer unterstützen, dynamische und komplexe Webanwendungen zu entwerfen. Bei dieser großen Anzahl an Frameworks besteht die Schwierigkeit für Unternehmen darin, ein gutes Framework zu finden. So stellen sich die folgenden Fragen: Welche Frameworks sollen benutzen werden? Wurde für das Projekt ein gutes Framework gewählt? Diese Fragen sind nur ein paar von vielen. Die Antworten auf diese Fragen können durch einen Vergleich von Frameworks herausgefunden werden. So kommt diese Arbeit ins Spiel. In dieser Arbeit werden zwei Frameworks verglichen. Das erste ist das GWT-Framework von Google. Das zweite ist das ASP.NET

AJAX-Framework von Microsoft. Um diesen Vergleich mit gleichen Voraussetzungen realisieren zu können,wird in Rahmen dieser Arbeit ein Informationssystem in den beiden Frameworks entwickelt. Dadurch entstehen zwei Web-Anwendungen. Im Anschluss daran werden Vergleichskriterien mit Hilfe von wissenschaftlichen Modellen festgelegt. Nach der Festlegung der Kriterien, werden Metriken entwickelt, die für den Vergleich zugrunde gelegt werden. Durch spezielle Werkzeuge werden die beiden Anwendungen getestet und die Testergebnisse verglichen.

1.1 Zielsetzung

In dieser Arbeit werden zwei AJAX-Anwendungen mit unterschiedlichen Technologien entwickelt. Für eine Anwendung wird das Google Web Toolkit eingesetzt. Die zweite Anwendung wird mit Microsoft ASP.NET-AJAX implementiert. Diese beiden Anwendungen sind Implementierungen eines Ausleihsystems. Anschließend werden diese getestet und auf den Entwicklungsaufwand und die Performance hin untersucht. Die Testergebnisse werden miteinander verglichen und ausgewertet. Die Komplexität

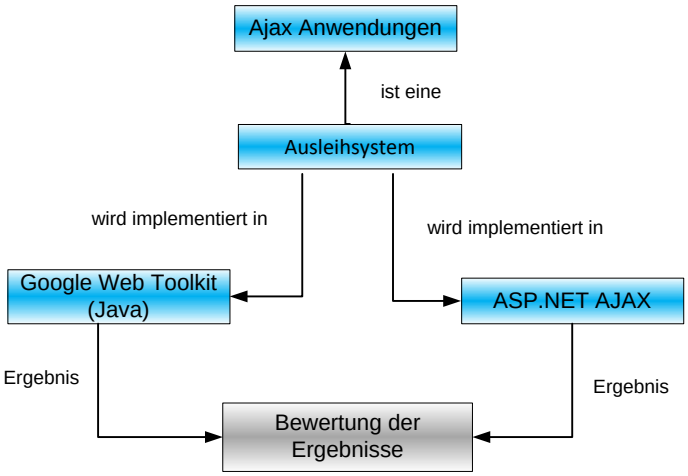

Abbildung 1.1: Übersicht über die Arbeit

dieser Arbeit liegt nicht nur an der Entwicklung des Systems, sondern auch in der Bewertung der beiden Technologien. Abbildung 1.1 gibt eine Übersicht über die gesamte Arbeit.

1.2 Aufbau der Arbeit

In Kapitel 2 wird eine Einführung der beiden Frameworks gegeben. Da die beiden Frameworks auf AJAX basieren, wird zunächst etwas über die Entstehungsgeschichte sowie die Motivation für den Einsatz von AJAX geschrieben. Die Beschreibung der beiden Frameworks widmet sich der Architektur sowie der Kommunikationsart zwischen Server und Client.

Das Kapitel 3 beschäftigt sich mit der Festlegung der Anforderungen des Informationssystems, mithilfe dessen die beiden Frameworks verglichen werden. Dafür werden die funktionalen und nicht- funktionalen Anforderungen sowie Use-Cases entwickelt.

Das Kapitel 4 widmet sich dem Design und der Architektur der beiden Anwendungen. Hier wird die fachliche und technische Architektur der Anwendungen dargestellt. Weiterhin wird das System anhand eines Klassendiagramms und eines Zustandsdiagramms verständlich gemacht.

In Kapitel 5 geht es um die Implementierung des Systems in beiden Frameworks. Mithilfe eines Flußdiagramms und kurzen Quellcode-Ausschnitten wird der interne Aufbau der beiden Anwendungen präsentiert.

Im Kapitel 6 geht es um die Vergleichsmethode. Hier werden Metriken, die für den Vergleich relevant sind, anhand von zwei Modellen abgeleitet. Diese Modelle sind: das Factor Criteria Metric (FCM) und das Goal Question Metric (GQM) Modell.

Das Kapitel 7 beschäftigt sich mit der konkreten Bewertung der beiden Systeme. Diese beiden Anwendungen werden sowohl lokal als auch entfernt getestet und die Test-Ergebnisse in grafischer und tabellarischer Form dargestellt und bewertet.

Abschließend wird die Zusammenfassung der Arbeit im Kapitel 8 gegeben

2 Technologien

In Diesem Kapitel wird eine Einführung der beiden Frameworks gegeben. Die Beschreibung der beiden Frameworks widmet sich der Architektur sowie der Kommunikationsart zwischen Server und Client.

2.1 AJAX

2.1.1 Definition

AJAX steht für **Asynchronous JavaScript and XML** und wurde ab Februar 2005 durch das Dokument „Ajax: A New Approach to Web Application“ von Jesse James Garrett bekannt. Jesse James Garrett stellt die Kombination DHTML (Dynamic HTML), JavaScript und XML (Extensible Markup Language) unter den Begriff AJAX vor. Die Idee, die dem AJAX-Konzept zugrunde liegt, gibt es seit März 1999. Das Microsoft-Outlook Entwickler-Team hat im Microsoft Internet Explorer 5 eine Technologie eingeführt, um im Hintergrund HTTP (Hypertext Transfer Protocol) Abfragen zu senden und die Rückgabe auszuwerten. Das Team benötigte diese HTTP-Abfragen im Hintergrund, um beispielweise ohne permanentes Neuladen zu prüfen, ob neue Mails vorliegen (vgl. Wenz, 2006). Allgemein ermöglicht AJAX einen asynchronen Datenaustauch zwischen einem Browser und einem Webserver. Dies erlaubt es, HTML-Anfragen durchzuführen während einer HTML-Seite angezeigt wird. Weiterhin kann die Seite verändert werden, ohne dass sie neugeladen wird. Viele Anwendungen von AJAX werden dazu benutzt, um im Webbrowser ein desktop-ähnliches Verhalten zu simulieren, wie beispielsweise Popup-Fenster (vgl. Wikipedia, 2011a). In den folgenden Abschnitten werden die Nachteile von klassischen Webanwendungen, die Motivation für den Einsatz von AJAX und das Konzept von AJAX-Anwendungen beschrieben.

2.1.2 Nachteile von klassischen Webanwendungen

Die Abbildung 2.1 zeigt die klassische Datenübertragung zwischen einem Browser und einem Webserver. Zunächst liegt eine Aktivität auf der Seite des Clients vor. Dies kann z.B. der Aufruf eines Hyperlinks sein. Dies erfordert eine Datenübertragung zum Server, durch die der Server informiert wird, welche Seite zum Client übertragen werden soll. Der Server wertet die Anfrage aus und sendet die Antwort zum Client zurück. Dieses Übertragungsmodell hat mehrere Nachteile: Durch die synchrone Kommunikation wird die Benutzeraktivität blockiert oder unterbrochen solange der Client auf die Antwort des Servers wartet (siehe unterbrochene Linie der Abbildung 2.1).

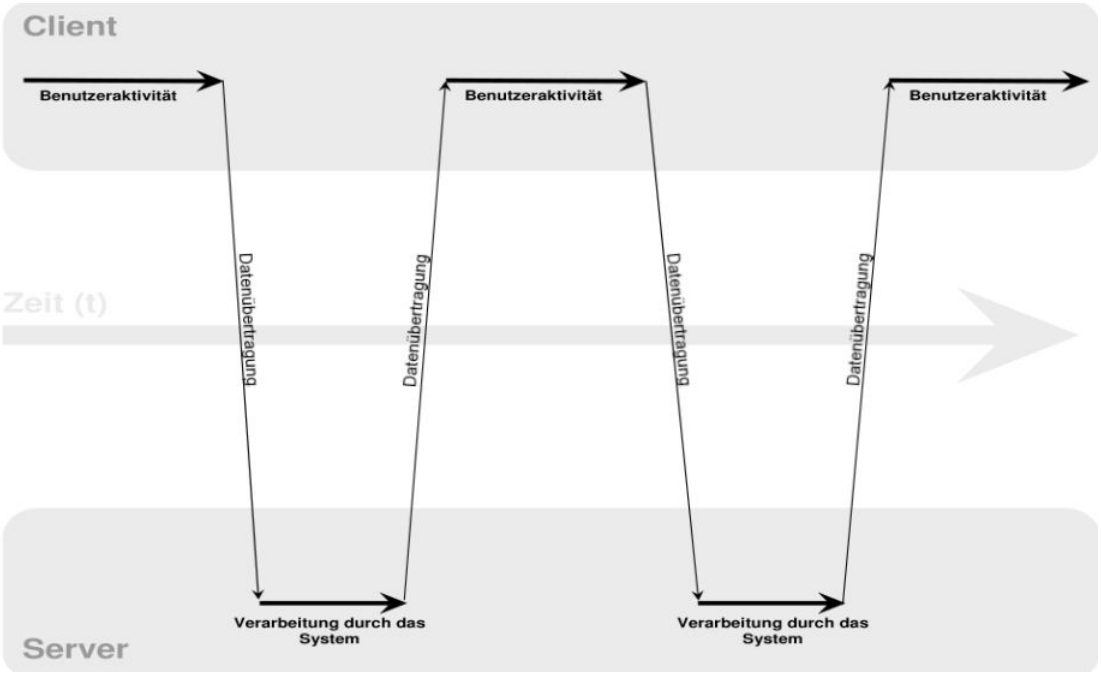

Abbildung 2.1: Klassisches Modell einer Webanwendung (synchrone Datenübertragung) (vgl. Wikipedia, 2011b)

Des Weiteren ist ein hoher Overhead bei der Übertragung zu erwarten. Unnötige Daten werden bei der Datenübertragung mitgesendet: Möchte ein Benutzer eine E-Mail-Adresse eines Teilnehmers aus einer Webseite erfahren, wird neben der eigentlichen Information auch der ganze HTML-Code der Seite mitgesendet. Die eigentliche

Information ist nur die E-Mail-Adresse an sich (vgl. Barthel u. a., 2005). Das gewünschte Verhalten ist, wenn die Benutzeraktivitäten nicht unterbrochen oder blockiert werden. Aus der Anwendersicht soll eine Web-Applikation vergleichbar mit einer Desktop-Anwendung sein. Des Weiteren sollen nur benötigte Daten zwischen Client und Server übertragen werden. Dieses gewünschte Verhalten wird von der AJAX-Technologie gewährleistet.

2.1.3 Konzept von AJAX-Anwendungen

AJAX bezeichnet ein Konzept, welches aus mehreren Technologien besteht: JavaScript, HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), DOM (Document Object Model) und XHR (XMLHTTPRequest). Die Funktionsweise dieses Konzepts wird durch die Abbildung 2.2 veranschaulicht.

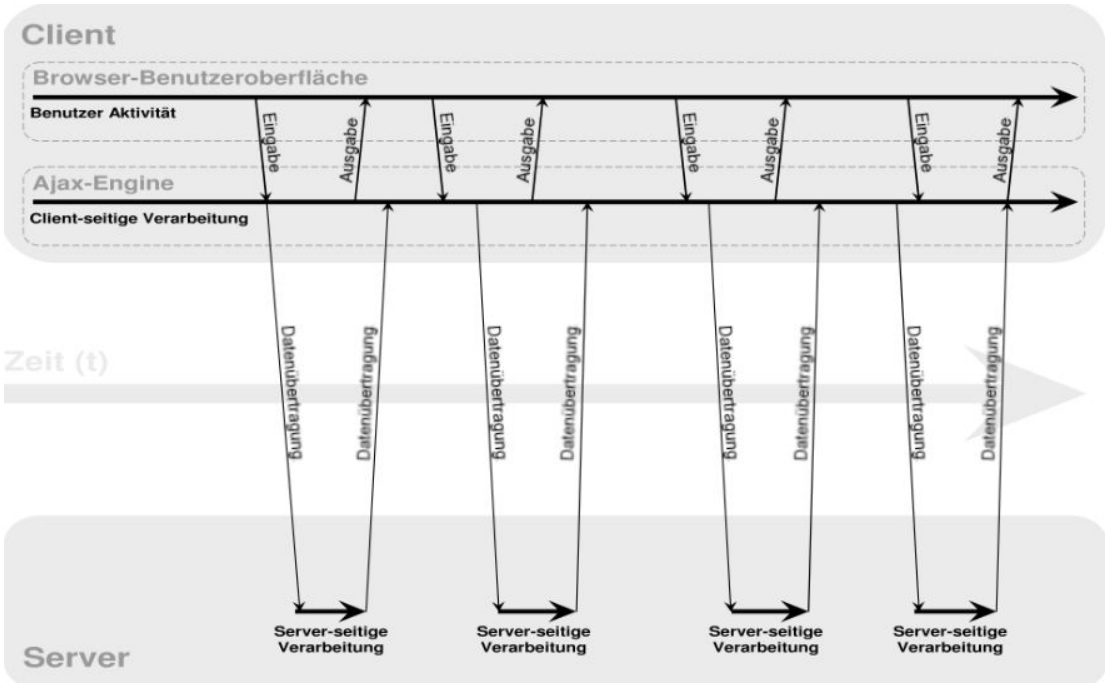

Abbildung 2.2: AJAX Modell einer Webanwendung (asynchrone Datenübertragung) (vgl. Wikipedia, 2011b)

Der Client besteht aus einer Browserbenutzeroberfläche und einer AJAX-Engine. Die AJAX-Engine ist eine JavaScript-Komponente, welche die Eingaben des Browsers

entgegennimmt und mit dem Webserver kommuniziert. Die AJAX-Engine fordert die benötigten Daten an und gibt sie nach Erhalt an den Browser weiter (vgl. Wikipedia, 2011a). Vor dem Empfang der Antwort vom Server kann die AJAX-Engine dem Benutzer eine Rückmeldung geben. Dadurch fühlt sich der Benutzer als ob er mit einer Desktop-Anwendung arbeitet anstatt mit einer klassischen Webanwendung. Um dieses Konzept mehr zu verdeutlichen wird das Beispiel von Microsoft Outlook-Webmail untersucht. Outlook-Webmail basiert auf dem AJAX-Konzept. Öffnet der Benutzer eine E-Mail in seinem Posteingang, fordert die AJAX-Engine beim Webserver den entsprechenden Inhalt der E-Mail an. Bevor die Antwort vom Server zurückkommt, öffnet die AJAX-Engine das Fenster für die Darstellung dieser E-Mail. Sobald die Antwort vom Server eingetroffen ist, wird der Inhalt der E-Mail in dem geöffneten Fenster angezeigt. Dadurch hat der Benutzer den Eindruck mit einer „nonstop“ Anwendung zu arbeiten (vgl. Barthel u. a., 2005).

2.1.4 Vor- und Nachteile von AJAX

Der größte Vorteil der AJAX-Technologie liegt darin, dass Daten verändert werden können, ohne dass die komplette Seite von Webbrowser neu geladen werden muss. Dadurch reagiert die Webanwendung schneller auf die Benutzereingaben. Zudem wird vermieden, dass statische Daten permanent über das Internet übertragen werde müssen. Dies gibt den Benutzer den Eindruck, mit einer Desktop-Anwendung zu arbeiten. Die AJAX-Technolgie ist frei zugänglich und wird auch unabgängig vom Webbrowser unterstüzt. So wird kein Browser-Plugin für den Einsatz von AJAX benötigt.

Die Tatsache, dass AJAX auf mehreren Technologien basiert, kann jedoch auch als Nachteil interpretiert werden. Möchte man eine professionelle AJAX-Anwendung schreiben, braucht man viele Domain-Experten. Darüber hinaus wird sowohl eine client- als auch eine serverseitige Sprache benötigt, was teilweise dazu führt, dass AJAX aufwändig zu implementieren ist. Das nächste Problem von AJAX ist bei der Indizierung der Informationen für die Suchmachinen. Hierbei werden diese Informationen nicht automatisch indiziert. Sie werden erst in einer XML-Datei oder Datenbank gespeichert und dann über spezielle Anfragen oder nach mehreren Benutzerschritten

auf der Webseite angezeigt werden. Dies führt dazu, dass die Roboter von Suchmaschinen die Informationen nicht einfach wie statische HTML-Seiten auslesen können (vgl. Barthel u. a., 2005).

2.2 Google Web Toolkit

2.2.1 Definition

Das Google Web Toolkit (GWT) wurde am 17. Mai 2006 von Google veröffentlicht. Beim Google Web Toolkit handelt es sich um ein kostenloses Opensource-Toolkit zur Erstellung von AJAX-Applikationen für das Web 2.0. Man kann mit GWT auf einfache Weise desktop-ähnliche Webanwendungen in Java schreiben und testen, welche durch einen GWT-Compiler in JavaScript-Anwendungen übersetzt werden. Die Entwicklung einer solchen Anwendung lässt sich mit Hilfe einer Java-Umgebung (wie z.B. Eclipse) realisieren. Weiterhin ist GWT mit einem XML-Parser ausgestattet. Dieser XML-Parser erlaubt das Auslesen und Verarbeiten von XML-Dateien. Die Schnittstelle für einen Remote Procedure Call ist bereits in GWT integriert und ermöglicht die Kommunikation zwischen Clients und Servers. Das Ziel von GWT ist die Komplexität von JavaScript zu reduzieren und dabei dynamische Webanwendungen zu implementieren, die jeder Browser versteht. Dafür werden die GWT-Komponenten, die durch Abbildung 2.3 dargestellt werden, angewendet (vgl. Steyer, 2007).

2.2.2 Architektur von GWT

Die Architektur des GWT-Frameworks besteht aus den vier grundlegenden Komponenten, die durch Abbildung 2.3 veranschaulicht werden.

GWT-Compiler bzw.Java-zu-JavaScript-Compiler

Der GWT-Compiler ist der Kern von GWT. Seine Aufgabe besteht darin, aus einer Java-Applikation eine äquivalente Applikation zu erzeugen, welche aus JavaScript, HTML und CSS besteht. Diese wird von jedem Browser unterstützt. Dadurch kann

ein erfahrener Java-Programmierer Webanwendungen schreiben ohne JavaScript beherrschen zu müssen.

JRE Emulation Library

Die Java Runtime Environment (JRE) emulation library beinhaltet eine Implementierung der Java-Standardklassenbibliothek. Diese Bibliothek ermöglicht die Übersetzung von Java-Code in JavaScript. JRE Emulation Library beinhaltet die Java.lang- , Java.util- und Java.io-Pakete.

Web UI class Library

Die Web UI class Library besteht aus benutzerdefinierten Interfaces und Klassen mit denen man im Webbrowser Widgets wie z.B. Buttons, Textboxes darstellen kann.

GWT Hosted mode Browser

Der GWT Hosted mode Browser ist ein spezieller Browser, der GWT-Applikationen im „hosted mode“ (Entwicklungsmodus) darstellen kann. Die Applikation wird in „hosted mode“ implementiert, da die Java Debugging-Funktionen angewendet werden können. Sie wird als Bytecode in einer Java Virtual Machine (JVM) ausgeführt.

2.2.3 Kommunikationsarten zwischen Client und Server

Eine Möglichkeit mit dem Server zu kommunizieren ist durch Remote Procedure Call Die Schnittstelle für diese Kommunikation ist bereits in GWT integriert. Das Konzept basiert auf Java Servlets . Als Servlets bezeichnet man Java-Klassen, deren Instanzen innerhalb eines Java-Webservers, Anfragen von Clients entgegen nehmen und beantworten (Vgl. SUN). Der Server muss in Java implementiert werden. Dafür hat Google fertige Interfaces zur Verfügung gestellt (Siehe Abbildung 2.4). Wenn der Server also eine entfernte Methode des Clients aufrufen möchte, muss er die Basisklasse „RemoteServiceServlet“ extendieren. Hiermit erbt er die Funktionalität, womit ankommenden Objekte des Clients deserialisiert und die ausgehenden Antworten serialisiert werden können. Der Client muss die „RemoteService“ implementieren, um mit dem Server

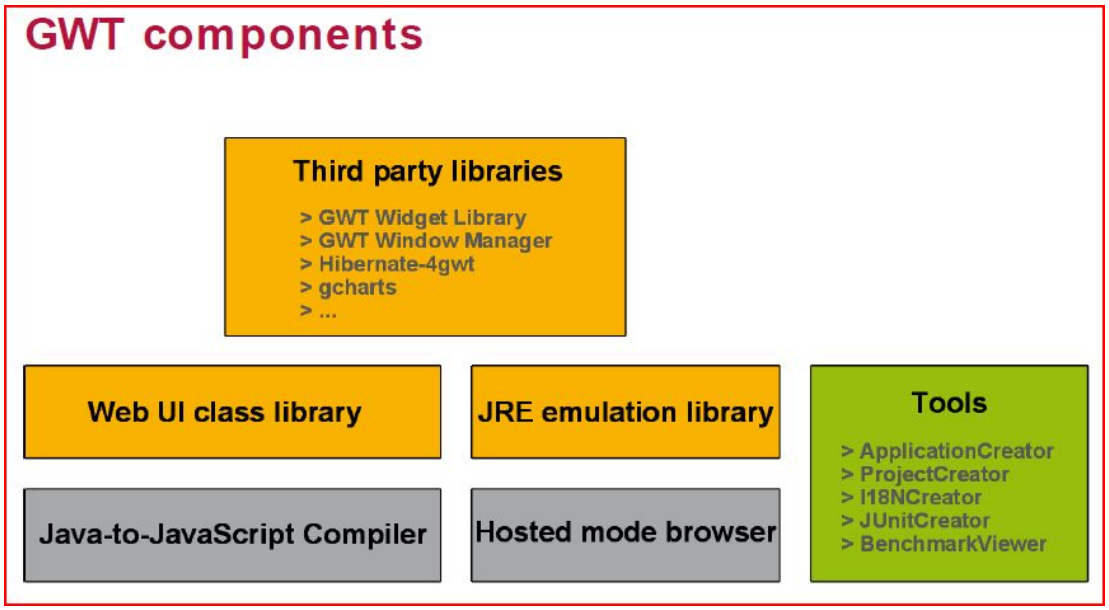

Abbildung 2.3: Architektur von Google Web Toolkit

erfolgreich kommunizieren zu können. Ein Vorteil dieser Kommunikationsform ist der Austausch von Java-Objekten zwischen Clients und Servern (vgl. Alexander Brosch, 2009).

2.2.4 Vor- und Nachteile von GWT

Die Besonderheit von GWT ist die Tatsache, dass der clientseitige Code komplett in Java geschrieben werden kann. Dies ermöglicht dem Programmierer, auf Java-Ebene den Code zu debbugen und mit Hilfe der bereits integrierten JUnit-Konponente zu testen. Ein weiterer Vorteil von GWT ist die Wiederverwendung durch JAR-Dateien (Java Archive). Die Module oder JAR-Dateien können von anderen Komponenten referenziert werden. Außerdem bietet GWT die Nutzung des Objektorientierten Designkonzepts. Jedoch hat GWT auch Nachteile. Die Integration von GWT-Komponenten in einer bestehenden Infrastruktur ist schwer. Außerdem dauert die Compilierung sehr lange.

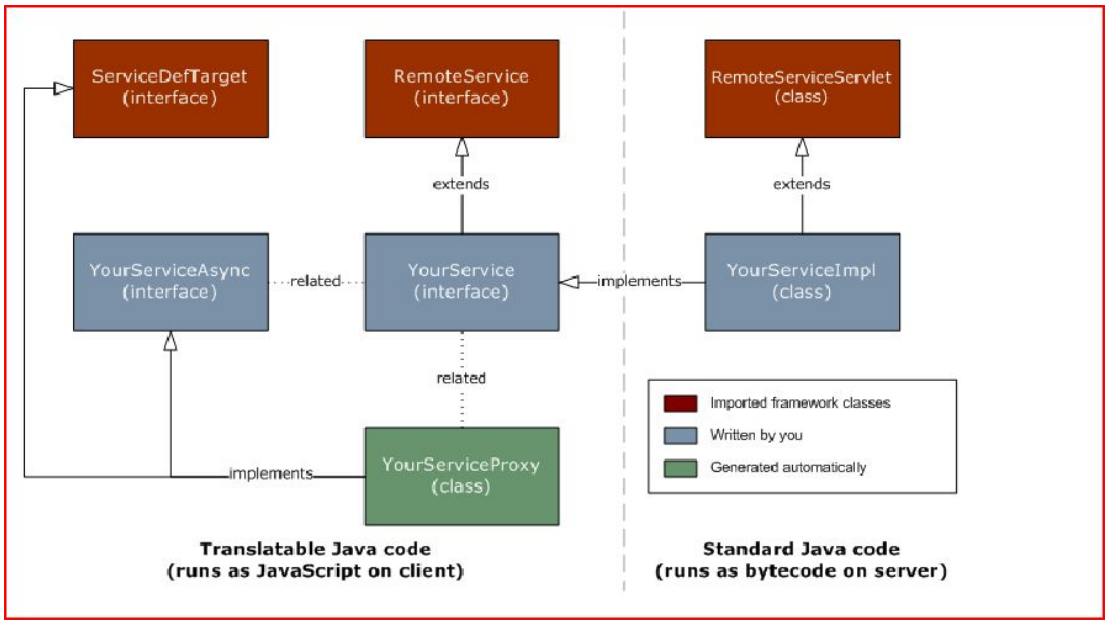

Abbildung 2.4: GWT-RPC Klassen (vgl. Google-Web-Toolkit)

2.3 ASP.NET AJAX

2.3.1 Definition

Active Server Pages .NET (ASP.NET) ist eine serverseitige Technologie von Microsoft, welche die Erstellung von Webapplikationen auf der Grundlage des Microsoft-.NET-Frameworks ermöglicht. Das Microsoft AJAX-Produkt kam zum ersten Mal bei der Professional Developer Conference (PDC) im September 2005 In Los Angeles unter dem Codename „Atlas“. Später erschien ASP.NET-AJAX als Erweiterung von ASP.NET als Framework, welches es dem Entwickler erlaubt, interaktive Webanwendungen mit AJAX-Funktionalität unter der Microsoft-Entwicklungsumgebung zu implementieren. Es hat ein integriertes client- und serverseitiges AJAX-Framework und besteht aus den Komponenten, die in den nachfolgenden Abschnitten beschrieben werden (vgl. Schwichtenberg, 2007).

2.3.2 Architektur von Microsoft AJAX-Anwendungen

Eine Microsoft AJAX-Anwendung besteht aus einer Client-Projektmappe oder aus einer Projektmappe mit Client- und Server-Projektmappen. Eine Client-Projektmappe benötigt die Microsoft AJAX Library, aber keine ASP.NET-Serversteuerelemente. Mit Hilfe der Microsoft AJAX Library wird die gesamte Verabeitung der AJAX-Webanwendung auf dem Client stattfinden. Die Microsoft AJAX Library ist der Kern des Client-Frameworks und arbeitet unabhängig von der Server-Seite. Sie stellt die JavaScript-Bibliotheken, AJAX-Funktionen und JavaScript-Erweiterung bereit. Eine Projektmappe mit einer Client- und Server-Mappe besteht aus einer Microsoft AJAX Library und ASP.NET-Serversteuerelemente.

Die Abbildung 2.5 zeigt die Grundfunktionalität der Clientseite der Microsoft AJAX-Bibliothek, die für das Erstellen von Clientkomponenten, die Browserkompatibilität, Netzwerk, Basisdiente sowie Debbugen und Fehlerbehanlung zuständig ist. Außerdem zeigt diese Abbildung auch die Serverseite der Microsoft AJAX-Bibliothek. Dazu sind Unterstüzung für Skript, Webdienste und Anwendungsdienste, Serversteuerelemente integriert. In den folgenden Abschnitten wird die Abbildung ausführlich beschrieben.

Microsoft AJAX-Client Architektur

Clientkomponenten sind die ersten Elemente vom Microsoft AJAX-Client. Sie werden in drei Hauptkategorien unterteilt: Die beiden ersten Kategorien sind Komponenten, die entwerder keine visuellen Objekte sind oder Komponenten mit Verhalten, welche das Verhalten von DOM-Elementen (Document Object Model) erweitern. Die dritte Kategorie von Clientkomponenten sind Steuerelemente, die ein neues DOM-Element mit benutzerdefiniertem Verhalten darstellen. Alle diese drei Clientkomponenten erlauben viele Aktionen ohne Postback (Absenden einer Seite zum Server mit dem HTTP-Verb "Post") im Browser.

Die Microsoft AJAX-Skriptkompatibilität ist in den am häufigsten benutzten Browser (Microsoft Internet Explorer, Mozilla Firefox und Apple Safari) bereits integriert. Dadurch können Skripte geschrieben werden unabhängig davon welcher Browser benutzt wird.

Die Netzwerkebene spielt eine Rolle in der Kommunikation zwischen Skripten im

Abbildung 2.5: Architektur von Microsoft AJAX-Anwendungen

Browser und webbasierten Anwendungen. Sie verwaltet den Remote-Methodenaufruf und unterstützt den Zugriff auf eine webbasierte Formularauthentifizierung, Rolleninformationen und Profilinformationen im Clientskript.
Die Basisdiente sind JavaScript-Dateien, die Funktionen zur Objektorientierten Entwicklung bereitstellen. Diese Funktionen können JavaScript-Erweiterungen sein. Dazu gehören zum Beispiel Klassen, Datentypen und Objektserialisierung. Sie können auch Basisbibliothek, die die Komponenten Zeichenfolge-Generatoren und Fehlerbehanlung enthalten (vgl. MSDN).

ASP.NET AJAX-Server Architektur

Die Serverseite von Microsoft-AJAX-Produkten besteht aus ASP.NET-Websteuerelementen und Komponenten, die die Benutzeroberfläche einer Webanwendung verwalten. Dazu gehören die Skriptunterstützung, Webdienste, Applicationdienste und Steuerelemente. In diesem Abschnitt werden nur die Steuerelemente ausführlich beschrieben, weil diese für die Implementierung der Anwendungen relevant sind. Die folgenden beschriebenen Komponenten sind die am häufigsten verwendeten ASP.NET AJAX-Steuerelemente.

ScriptManager

Der ScriptManager ist das zentrale Steuerelement der AJAX-Anwendung. Jede Webapplikation, die die AJAX-Funktionalität erfüllt, muss einen ScriptManager haben. Ohne einen ScriptManager kann ASP.NET ein AJAX-Framework nicht verwenden. Jede ASP.NET-Seite mit der AJAX-Funktionalität soll genau eine Instanz des ScriptManagers haben und diese soll vor allen anderen Steuerelementen positioniert werden. Der ScriptManager sorgt dafür, die synchrone und asynchrone Funktionalität besser zu trennen. Weiterhin ermöglicht der ScriptManager die teilweise Aktualisierung der Seite zwischen dem Client und dem Server und das Erkennen der Ereignisse, die eine solche Aktualisierung erfordern.

UpdatePanel

Das UpdatePanel ist ein Steuerelement, welches es ermöglicht, ein Teil der Seite zu wählen und diesen unabhängig vom Rest zu aktualisieren. Diese Aktualisierung

kann auch asynchron sein. Dafür wird ein „AsyncPostBackTrigger“ instanziiert. Um dieses Steuerelement in einer Webseite verwenden zu können, muss die Eigenschaft „EnablePartialRendering“ des ScriptManagers auf True (Standard) gesetzt werden. Eine Webseite kann ein oder mehrere UdaptePanels haben. Weiterhin ermöglicht das UdaptePanel, eine Webanwendung zu schreiben, die dem Verhalten einer Desktop-Anwendung näher kommt.

UpdateProgress

Das UpdateProgress-Steuerelement gibt dem Benutzer eine Rückmeldung über das Ergebnis auf dem Server. Es wird benutzt, wenn eine Aktion etwas länger dauert. Der einfachste Fall dieses Steuerelement zu benutzen ist, es auf der Seite hinzuzufügen, wo die Update-Meldung erscheinen soll.

Timer

Das Timer-Steuerelement hilft bei dem regulären Laden einer Webseite oder ein Teil der Seite in kollaboration mit dem UpdatePanel-Steuerelement (vgl. Fischer und Krause, 2010).

AJAX Control Toolkit

Das AJAX Control Toolkit ist kein Microsoft Produkt sondern ein Open Source-Projekt von CodePlex Foundation (vgl. CodePlex-Foundation), die zusammen mit Microsoft arbeitet. Das AJAX Control Toolkit enthält Steuerlemente zum Erstellen von ajaxfähigen Webanwendungen ohne JavaScript oder AJAX beherschen zu müssen. Mit dem AJAX Control Toolkit können ASP-MVC-Webanwendungen (MVC: Model View Controller) erstellen werden, indem die Steuerelemente aus dem Toolkit von Visual Studio in einer Seite verschoben werden. Das AJAX Control Toolkit ist vergleichbar mit dem GWT-Designer (vgl. MSDN).

2.3.3 ASP.NET AJAX Funktionen

ASP.NET AJAX arbeitet nach folgenden Funktionen:

Partielle Seitenerzeugung (Partial Page Rendering- PPR)

Durch eine Partielle Seitenerzeugung können Seitenfragemente zwischen dem Client und dem Server ausgetauscht werden. Jedoch wird die Darstellung für diese Seitenfragmente vom Server erzeugt. Der Client bekommt dann ein Paket mit dem HTML-Seitenfragment und Informationen wie die Fragmente auf dem Bildschirm dargestellt oder positioniert werden.

Entfernter, asynchroner Codeaufruf vom Browser zum Webserver

Bei einem entfernten, asynchronen Codeaufruf wird die Darstellung der Seite nicht vom Server erzeugt sondern der Client ruft einen Webservice mit bestimmten Parametern auf und bekommt später eine Antwort vom Webserver. Die Ausgabe dieser Antwort auf den Bildschirm wird dann vom Client bestimmt. Beim Vergleich der beiden ersten Funktionen ist anzumerken, dass die Partielle Seitenerzeugung den Server mehr belastet als der entfernte, asynchrone Codeaufruf, denn der Server muss mehr Code für die Darstellung des Seitenfragments generieren. Jedoch ist die Partielle Seitenerzeugung einfacher zu realisieren als der entfernte asynchrone Codeaufruf (vgl. Schwichtenberg, 2007).

3 Anforderungsspezifikation

Im Rahmen dieser Arbeit werden zwei Webanwendungen zur Bewertung der AJAX-Technologie in zwei unterschiedliche Plattformen (ASP .NET und GWT) entwickelt. In diesem Kapitel werden die Anforderungen dieser Anwendungen angegeben. Hier geht es nicht um die Entwicklung von komplexen Anwendungen sondern von Anwendungen mit bestimmten Komponenten, Funktionen und Anforderungen, die für die Bewertung notwendig sind.

3.1 Einleitung

Die zu entwickelnde Software ist vergleichbar mit einem Web-Informationssystem (WebIS). Ein Web-Informationssystem ist ein strukturiertes geordnetes System, welches Informationen über einen Dienst zur Verfügung stellt. Ein konkretes Beispiel für ein Web-Informationssystem ist ein Nachrichtenportal. Die multimedialen Inhalte werden hier verwaltet und für die Benutzer zur Verfügung gestellt. Funktionen wie Suchen in der aktuellen Ausgabe, Archiv oder Foren sind vorhanden. Aus Sicht der Architektur steht im Hintergrund eine große multimediale Datenbank, welche eine große Anzahl von Benutzern gleichzeitig speichern kann. In dieser Arbeit geht es um ein Ausleihsystem, welches in zwei Anwendungen mit den Namen Biblio-GWT und Biblio-ASP.NET-AJAX implementiert wird. Jedoch ist das zu entwickelnde Ausleihsystem nicht vollständig, enthält aber die Basis-Funktionen und Komponenten jedes Web-Informationssystem. Dazu gehören das Login, die Verwaltung der Benutzerdaten, Produktanzeigen, eine Datenbank, eine Suchfunktion und die Navigation. Dies sind schon die wesentlichen Komponenten eines echten Web-Informationssystems mit den Unterschied, dass die Funktionalitäten nicht sicher und komplex sind. Zum Beispiel wird bei der Anmeldung keine Verschlüsselungsmethode angewendet. Die

Benutzerdaten (Benutzername und Passwort) dienen als Zugangsdaten zum System. Sie werden in der Datenbank gespeichert und bei der Anmeldung wird überprüft, ob sie miteinander übereinstimmen. In dieser Arbeit wird auch kein HTTPS-Protokoll für eine sichere Kommunikation und Datenübertragung benutzt, weil dieses nicht Bestandteil dieser Arbeit ist. Der eingeloggte Benutzer kann seine Daten bearbeiten, ein Produkt suchen und Informationen über die Produkte abrufen. Der Administrator kann neue Produkte hinzufügen und Benutzer verwalten. Im folgenden Abschnitt werden die Anforderungen an das System ausführlich beschrieben.

3.2 Anforderungen

Die Anforderungen beschreiben die Eigenschaften, die das System erfüllen muss. Die funktionalen und nichtfunktionalen Anforderungen werden im folgenden Abschnitt betrachtet. Die funktionalen Anforderungen beschreiben, was das System gewährleisten soll bzw. welche Funktionalität es dem Endnutzer anbieten soll. Die nichtfunktionalen Anforderungen geben an, wie gut das System die erwartete Aufgabe leisten soll.

3.2.1 Funktionale Anforderungen des Ausleihsystems

Das System soll folgende funktionale Anforderungen (FA) erfüllen.

Benutzer

FA1: Jeder Benutzer soll sich im System registrieren können und legt dadurch seine Zugangsdaten fest. Jeder Benutzer soll bei der Registrierung einen Name, einen Benutzername, ein Geburtsdatum, eine Adresse, eine Postleizahl, eine Telefonnummer und eine E-Mail-Adresse angeben.

FA2: Mit den Zugangsdaten soll sich der Benutzer im System anmelden können.

FA3: Der Benutzer soll eine Identifikationsnummer haben.

FA4: Der Benutzer soll die Möglichkeit haben, ein Produkt zu suchen, auszuleihen und zurückzugeben.

FA5: Der Benutzer soll sich abmelden können.

Produkte

FA6: Als Produkte sollen Bücher gespeichert werden.

FA7: Jedem Produkt ist ein Titel, ein Autor, ein Jahr, ein Verlag und eine ISBN zugordnet.

FA8: Das System soll um andere Produktkategorien erweiterbar sein, z.B. Computer, Fernseher.

FA10: Im System soll es die Möglichkeit geben, neue Produkte hinzufügen zu können.

Administrator

FA11: Der Administrator ist ein Benutzer mit speziellen Zugangsberechtigungen. Seine Aufgabe ist erweitert auf das Hinzufügen neuer Produkte und das Verwalten der Benutzer.

3.2.2 Nicht-funktionale Anforderungen (NFA) des Ausleihsystems

NFA12: Die Anwendung soll auf den Browser Firefox, Chrome und Internet Explorer lauffähig sein.

NFA13: Die Geschwindigkeit, mit der die Anwendung geladen wird, soll maximal sein.

NFA14: Die Navigationspunkte sollen verständlich sein und selbsterklärende Namen haben, so dass die Benutzerfreundlichkeit dadurch erhöht wird.

3.3 Use-Cases

Use-Cases sind Anwendungsfalldiagramme in der Unified Modeling Language (UML). UML ist eine graphische Modellierungsprache für objektorientierte Systeme. Sie umfasst mehrere Arten von Diagrammen, um ein System auf verschiedene Weise zu modellieren. Das Use-Case-Diagramm ist einer der 14 Diagrammarten der UML. Es wird angewendet, um mögliche Szenarien zu beschreiben, die vorkommen können,

wenn der Benutzer mit Hilfe des Systems ein bestimmtes Ziel erreichen möchte. Die Abbildung 3.1 liefert einen Überblick über die verschiedenen Anwendungsfälle im System. Im folgenden Abschnitt werden diese ausführlich beschrieben.

Tabelle 3.1: Einloggen im System

Use-Case-Name	**Einloggen im System**
Beschreibung	Der Benutzer und der Adminstrator benutzen die Zugangsdaten, um sich im System anzumelden. Bei falschen Anmeldedaten werden sie auf die Einloggen-Seite zurückgeführt.
Vorbedingung	Der Benutzer oder der Administrator ist nicht im System angemeldet.
Nachbedigung Erfolg	Der Benutzer oder der Administrator ist jetzt im System angemeldet.
Nachbedigung Misserfolg	Die Zugangsdaten sind falsch. In diesem Fall wird dem Benutzer oder dem Administrator die Einloggen-Seite angezeigt.
Akteure	Der Benutzer und der Administrator

Tabelle 3.2: Sich registrieren

Use-Case-Name	**Sich registrieren**
Beschreibung	Der Benutzer hat die Möglichkeit sich zu registrieren. Seine Daten werden in der Datenbank gesichert. Diese werden für die Benutzerverwaltung verwendet.
Vorbedingung	Der Benutzer muss sich im System angemeldet haben.
Nachbedigung Erfolg	Die Benutzerdaten sind wie gefordet im System gespeichert.
Nachbedigung Misserfolg	-
Akteure	Der Benutzer

Tabelle 3.3: Produkt suchen

Use-Case-Name	**Produkt suchen**
Beschreibung	Die Produkte sind in der Datenbank gespeichert und können nach Produkt-Name oder Produkt-Nummer gesucht werden.
Vorbedingung	Der Benutzer muss sich im System angemeldet haben.
Nachbedigung Erfolg	Der Benutzer hat das gesuchte Produkt gefunden.
Nachbedigung Misserfolg	Das Produkt existiert nicht im System. Der Benutzer bekommt eine entsprechende Rückmeldung.
Akteure	Der Benutzer

Tabelle 3.4: Produkt ausleihen

Use-Case-Name	**Produkt ausleihen**
Beschreibung	Die Produkte sind in der Datenbank gespeichert und können ausgeliehen werden.
Vorbedingung	Der Benutzer muss sich im System angemeldet haben und das Produkt gefunden haben, das er ausleihen möchte.
Nachbedigung Erfolg	Der Benutzer hat das Produkt ausgeliehen.
Nachbedigung Misserfolg	Das Produkt existiert nicht im System oder ist bereits ausgeliehen. Der Benutzer bekommt eine entsprechende Rückmeldung.
Akteure	Der Benutzer

Tabelle 3.5: Produkt vormerken

Use-Case-Name	**Produkt vormerken**
Beschreibung	Die Produkte sind in der Datenbank gespeichert und können vorgemerkt werden.
Vorbedingung	Der Benutzer muss sich im System angemeldet haben und das Produkt gefunden haben, das er vormerken möchte.
Nachbedigung Erfolg	Der Benutzer hat das Produkt vorgemerkt.
Nachbedigung Misserfolg	Das Produkt existiert nicht im System. Der Benutzer bekommt eine entsprechende Rückmeldung.
Akteure	Der Benutzer

Tabelle 3.6: Produkt zurückgeben

Use-Case-Name	**Produkt zurückgeben**
Beschreibung	Der Benutzer will das Produkt zurückgeben.
Vorbedingung	Der Benutzer muss sich im System angemeldet haben.
Nachbedigung Erfolg	Der Benutzer hat das Produkt fristgerecht zurückgeben.
Nachbedigung Misserfolg	Der Abgabetermin ist abgelaufen. Der Benutzer bekommt eine entsprechende Rückmeldung.
Akteure	Der Benutzer

Tabelle 3.7: Produkt hinzufügen

Use-Case-Name	**Produkt hinzufügen**
Beschreibung	Der Administrator hat die Möglichkeit, im System neue Produkte hinzuzufügen.
Vorbedingung	Der Administrator muss sich im System angemeldet haben.
Nachbedigung Erfolg	Der Administrator hat das Produkt hinzugefügt.
Nachbedigung Misserfolg	-
Akteure	Der Administrator

Tabelle 3.8: Benutzer verwalten

Use-Case-Name	**Beutzer verwalten**
Beschreibung	Der Administrator hat die Möglichkeit, alle Benutzer mit deren Produkten aufzulisten.
Vorbedingung	Der Administrator muss sich im System angemeldet haben.
Nachbedigung Erfolg	Die Liste aller Benutzer mit ihren Produkten ist verfügbar.
Nachbedigung Misserfolg	-
Akteure	Der Administrator

Tabelle 3.9: Ausloggen aus dem System

Use-Case-Name	**Ausloggen im System**
Beschreibung	Der Benutzer und der Administrator haben die Möglichkeit sich abzumelden.
Vorbedingung	Der Benutzer oder der Administrator ist im System angemeldet.
Nachbedigung Erfolg	Der Benutzer oder der Administrator hat sich erfolgreich abgemeldet.
Nachbedigung Misserfolg	-
Akteure	Der Benutzer und der Anministrator.

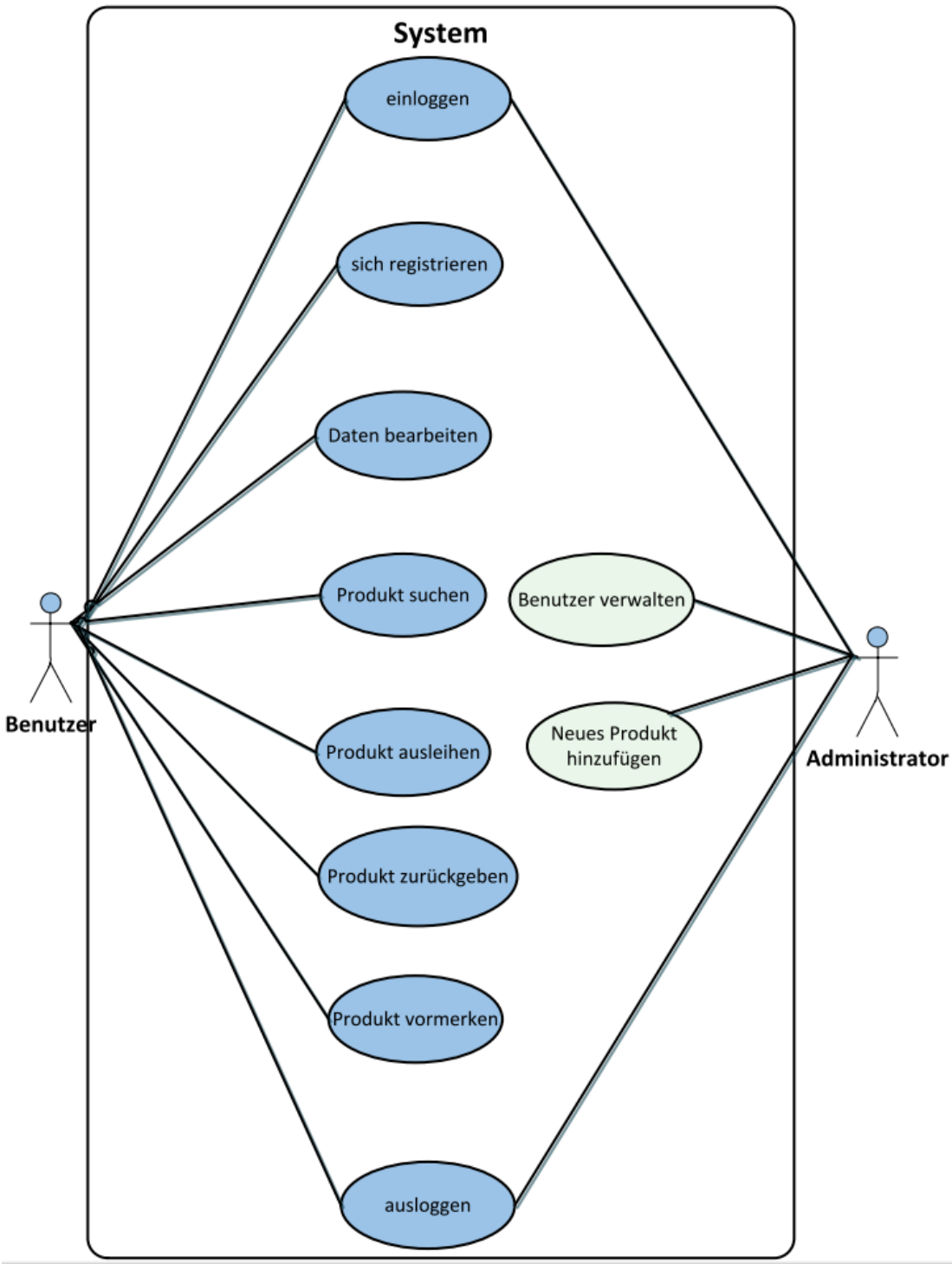

Abbildung 3.1: Use-Case Diagramm

4 Design und Architektur der Anwendungen

Die Softwarearchitektur beschreibt die Komponenten eines Softwaresystems und deren Zusammenspiel. In diesem Kapitel werden das Design und die Architektur der Biblio-GWT-und Biblio-ASP.NET-AJAX-Anwendung dargelegt.

4.1 Fachliche Komponenten

Die fachliche Architektur ist unter dem Name Quasar bekannt. Quasar steht für Qualitätssoftwarearchitektur und ermöglicht eine bessere Übersicht über die Komponenten. Das Ziel der Quasar-Architektur ist es, die Anwendung von der technischen Architektur zu trennen, um sie wiederverwenden zu können. Hierdurch lässt sich das System besser erweitern und anpassen. Die Abbildung 4.1 zeigt die fachlichen Komponenten des Systems. Um die Komponenten besser voneinander zu unterscheiden, besitzt jede Komponente eine Schnittstelle. Die Komponente Ausleihen ist eine zentrale Komponente und kann alle Interfaces benutzen.

4.2 Technische Komponenten

Die Abbildung 4.2 zeigt die technischen Komponenten des Systems. Drei Komponenten sind hier dargestellt: Die Biblio-GWT- Biblio-ASP.NET-Ajax- und MySQL-Datenbank-Komponente. Die Biblio-GWT- und Biblio-ASP.NET-AJAX-Komponenten basieren auf der Client-Server-Architektur und verwenden die gleiche Datenbank(MySQL). Um die Verbindung zu der Datenbank zu ermöglichen, werden zwei verschiedene Connectoren eingesetzt: Der erste Connector ist JDBC(Java Database Connectivity). JDBC

ist eine Datenbankschnittstelle der Java Plattform, die auf relationale Datenbanken ausgerichtet ist.

Zu den Aufgaben von JDBC gehören der Aufbau und die Verwaltung der Datenbankverbindung, die Weiterleitung von SQL-Anfragen an die Datenbank sowie die Umwandlung von Ergebnissen in eine für Java nutzbar Form. Der zweite Connector ist ADO.NET. ADO.NET ist der Nachfolger von ActiveX Data Objects (ADO), hat jedoch nichts mit ActiveX-Technologie zu tun. Unter ActiveX-Technologie versteht man ein Softwarekomponenten-Modell von Microsoft für aktive oder dynamische Inhalte. Dynamische Inhalte bezeichnen Inhalte in Webseite, die zur Laufzeit ausgeführt werden. ADO.NET ist eigentlich eine Sammlung von Klassen aus der Microsoft Plattform, die den Zugriff auf eine Datenbank ermöglichen. Zu den Klassen gehören die sogenannte Connection-Klassen, die für die Verbindung zu der Datenbank zuständig sind und Klassen, die die Tabelle im Speicher repräsentiert.
Google Web Toolkit arbeitet nach der RPC-kommunikation. Es gibt in GWT kein MVC-Framework, aber ASP.NET AJAX Komponente verwendet das MVC-Enwurfsmuster. Die Abbildung 4.3 zeigt das Konzept des Modell-View-Controller-Entwurfsmusters.

Modell

Das Modell repräsentiert eine Klasse, die für die Verarbeitung von Daten zuständig ist. Der Zugriff auf die Daten, die in der Datenbank gespeichert sind, erfolgt innerhalb des Modells. Um die Präsentation (View) über alle Änderungen zu informieren, wird innerhalb des Modells ein Beobachter (Observer) definiert.

Präsentation (View)

Die Aufgabe der Präsentationsklasse besteht darin, die Informationen darzustellen und Bedienelemente zur Benutzerinteraktion bereitzustellen. Das Modell kann eine oder mehrere Präsentationen haben. Wichtig ist, dass jede Präsentation über die Änderungen im System erfährt.

Steuerung (Controller)

Die Steuerungsklasse ist für die Steuerung von Aktionen zuständig. Aktionen der Bedienelemente aus der Präsentationsklasse werden angenommen, ausgewertet, und die Daten im Modell verarbeitet und aktualisiert. Eine Steuerung kann mehrere Präsentationsklassen bedienen.

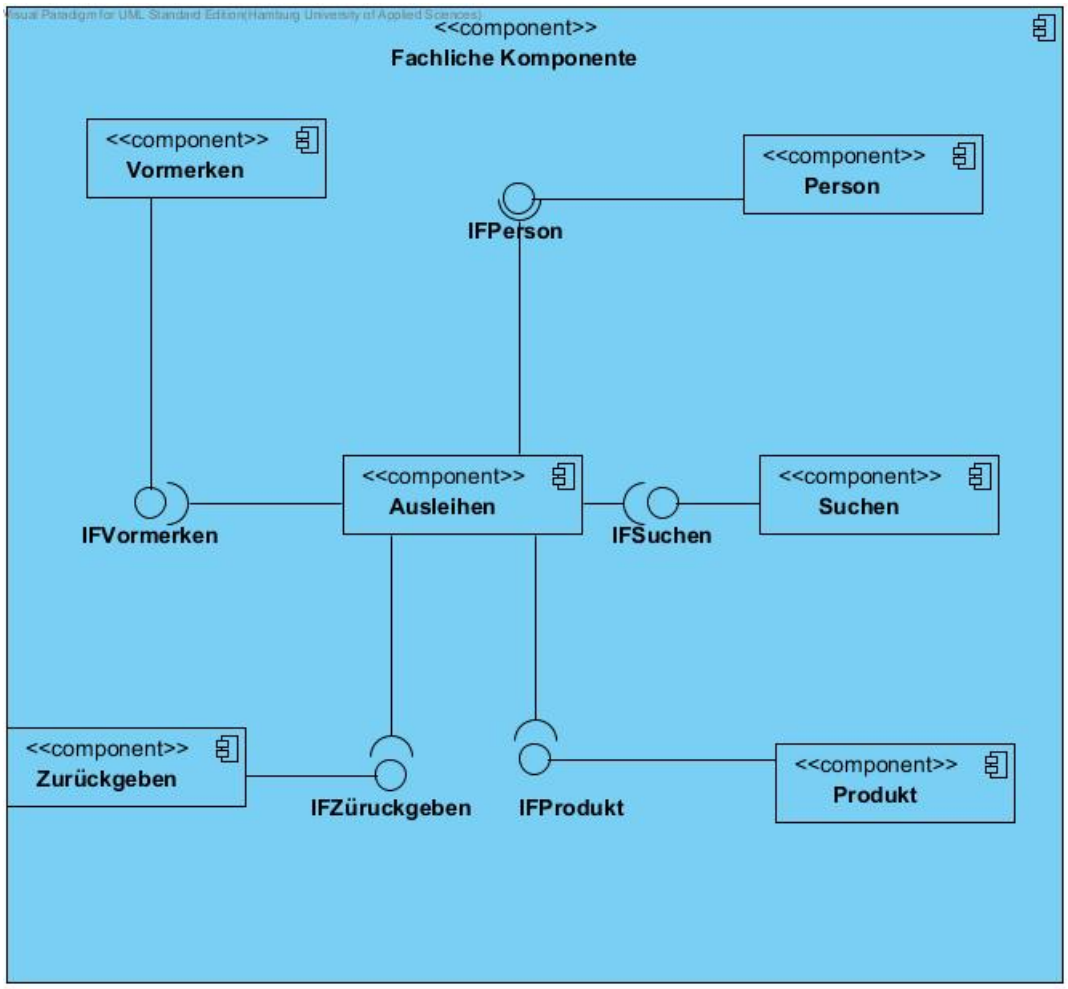

Abbildung 4.1: Fachliche Komponenten

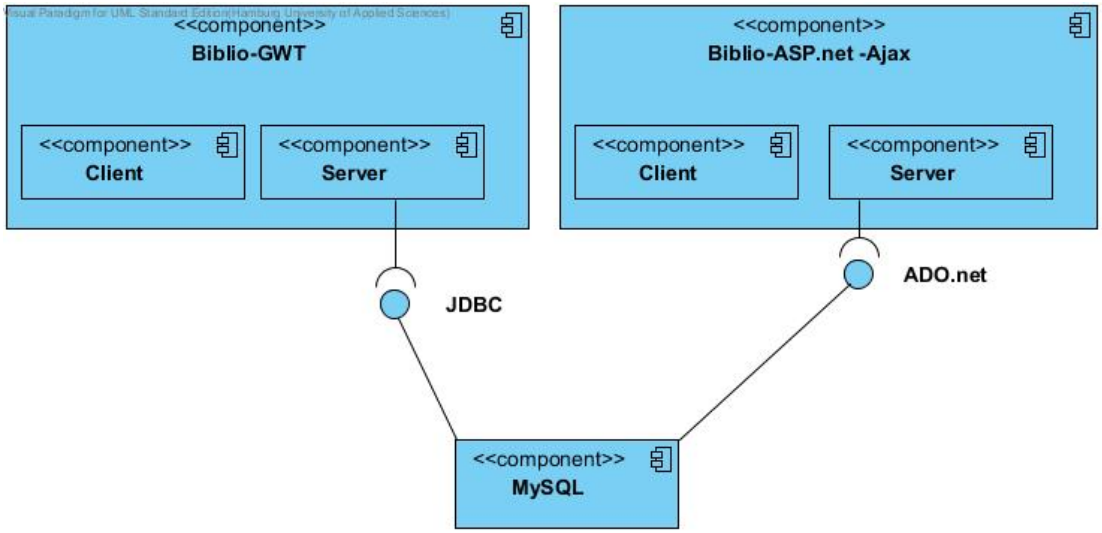

Abbildung 4.2: Technische Komponenten

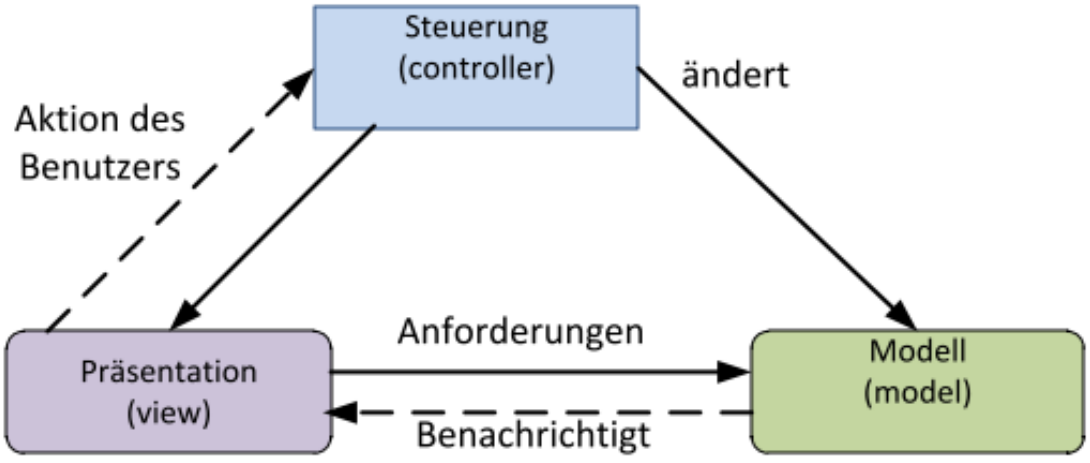

Abbildung 4.3: Model View Controller (vgl. Fischer und Krause, 2010)

4.2.1 Biblio-GWT Komponente

Die Biblio-Gwt Komponente besteht aus einem Client- und einem Server-Teil. Das Modell der GWT-Komponente befindet sich auf der Server-Seite. Die Anfragen aus dem Browser werden an den Server über Remote Procedure Call weitergeleitet. Mittels der Schnittstelle JDBC verbindet sich der Server zu der MySQL-Datenbank und verarbeitet die Datenanfragen. Die Ergebnisse der Anfrage werden in dem übergebenen Callback-Objekt gespeichert und an den Client weitergereicht. Der Client interpretiert das Callback-Objekt und verwendet ein Panel, um die Ergebnisse im Browser anzuzeigen.

4.2.2 Biblio-ASP.NET-AJAX Komponente

Die Komponente Biblio-ASP.NET-AJAX enthält eine Client- und eine Server-Komponente. Das Modell befindet sich wie bei der Biblio-GWT-Komponente auf der Server-Seite. Die Kommunikation zwischen Client und Server ist möglich mittel XML- und JavaScript-Dateien. Über den Browser gibt es das XmlhttpRequest-Objekt, welches die Kommunikation ermöglicht. Mittels der Schnittstelle ADO.NET verbindet sich der Server zu der MySQL-Datenbank und verarbeitet die Anfragen. Die Ergebnisse der Anfrage werden per normalem HTML-Text an den Client zurückgesendet.

4.2.3 MySQL-Komponente

In dieser Arbeit wird MySQL in der Version 5.5.9 verwendet. Die Datenbank enthält drei Tabellen: Die Ausleihen-Tabelle, welche alle ausgeliehene Produkte speichert. Die Person-Tabelle speichert die Benutzerdaten. Die Produkt-Tabelle ist der Katalog des Systems. Sie enthält alle Produkte, die ausgeliehen werden können.

4.3 Zustanddiagramm des Systems

Die Abbildung 4.4 zeigt das Zustandsdiagramm des Systems. Unter einem Zustandsdiagramm versteht man die Spezifikation des Verhaltens eines Systems. Beim Start geht dieses entweder in den Zustand Login oder sich registrieren. Sind die Zugangsdaten für den Zustand Login richtig, springt das System in den Zustand eingeloggt. Ist

ein Fehler aufgetreten, geht das System in den Zustand Fehler über. Dann wird eine Fehlermeldung im Browser angezeigt. Von Zustand eingeloggt kann das System zu dem Zuständen Benutzer hinzufügen, Benutzerprofil anzeigen, alle Benutzer anzeigen, Produkt hinzufügen, Produkt zurückgeben, Produkt anzeigen, Produkt vormerken, Produkt suchen, Produkt ausleihen oder ausgeloggt wechseln . Im Zustand ausgeloggt wird das System beendet.

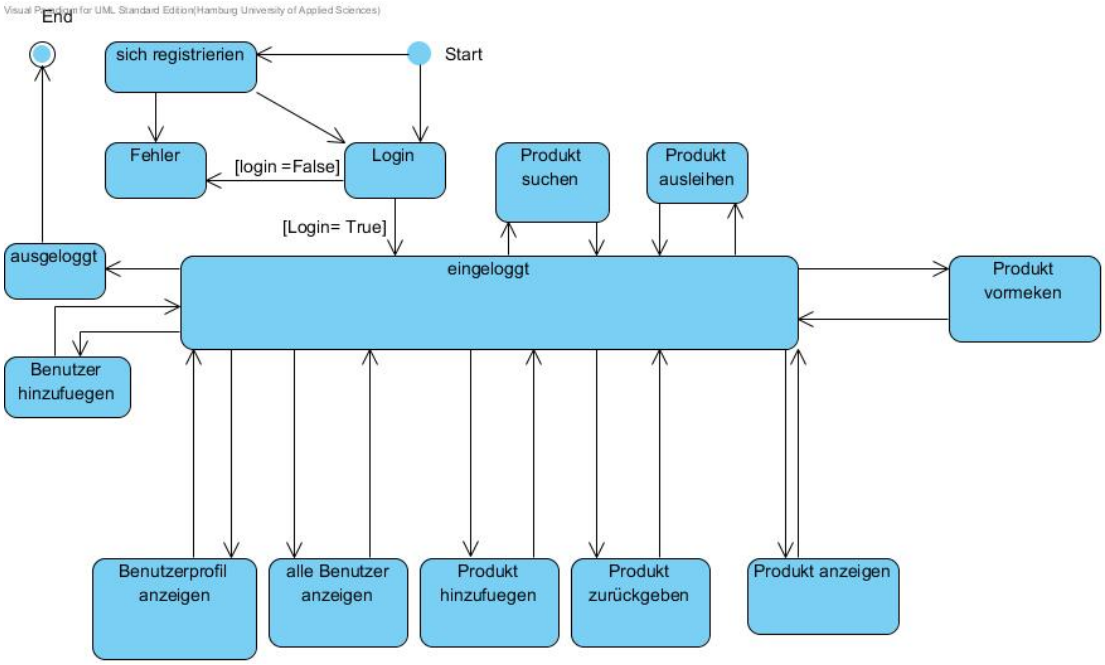

Abbildung 4.4: Zustandsdiagramm des Systems

4.4 Objekt-Modelle des Systems

4.4.1 Gemeinsames Objekt-Modell

Die beiden Anwendungen benutzen das Objekt-Modell aus der Abbildung 4.5. Zwei Hauptklassen werden dargestellt: Produkt und Person. Alle andere Klassen erweitern diese beiden Klassen.

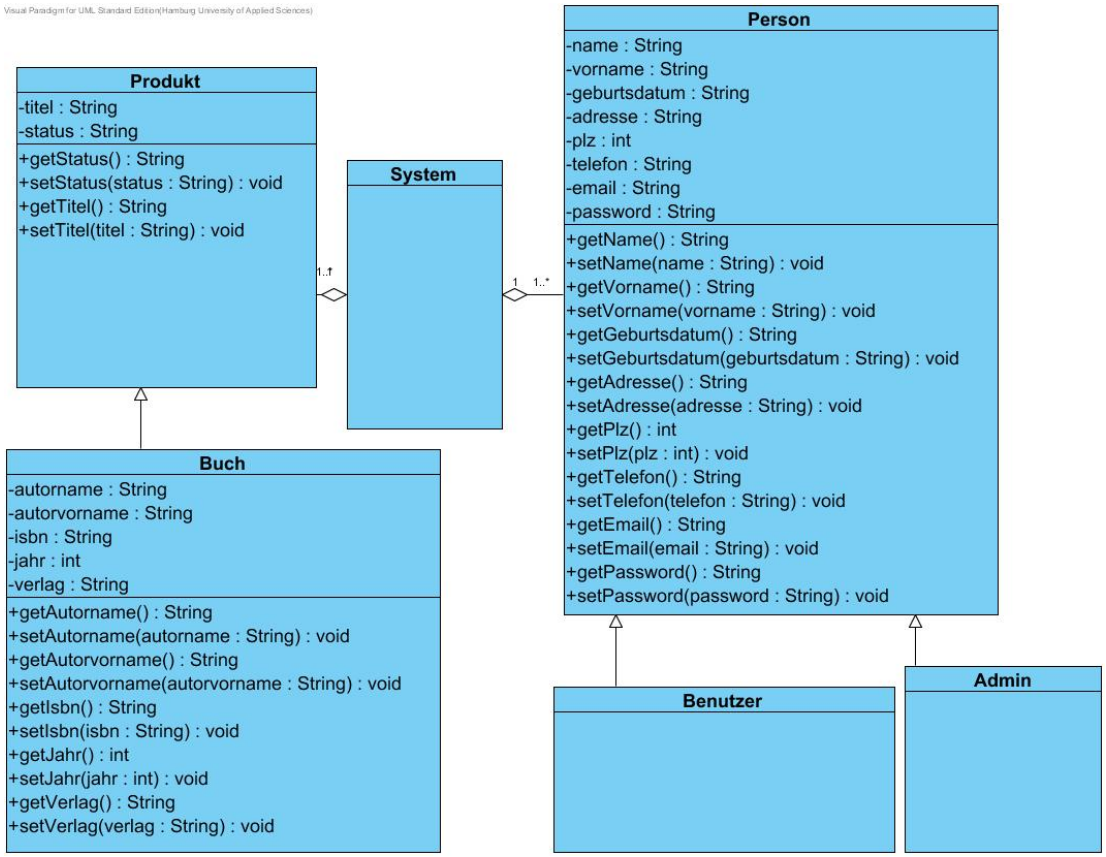

Abbildung 4.5: Objekt-Modell der beiden Systeme

4.4.2 Klassendiagramm der Biblio-GWT-Anwendung

Die Abbildung 4.6 zeigt das Klassendiagramm von Biblio-GWT. Zu der Client-Komponente gehören die Klassen Benutzer, Produkt, Buch, Benutzerservice, Produktservice, GWT-System sowie mehrere Panels. Die Klasse GWTSystem repräsentiert die Main-Klasse des Systems. GWTSystem implementiert die zwei wichtigste Interfaces: EntryPoint und HistoryListener. Das Interface EntryPoint hat die Methode onModuleLoad(), die automatisch durch das Laden des Systems die Klasse aufruft, die EntryPoint implementiert, und als Main-Klasse betrachtet. Daher darf es nur eine Klasse in System geben, die dieses Interface implementiert. In AJAX-Anwendungen taucht häufig das Problem auf, dass der Zurück-Button des Browsers nicht die letzte über AJAX aufgerufene Aktion rückgängig macht, sondern die letzte vollständige Webseite, die

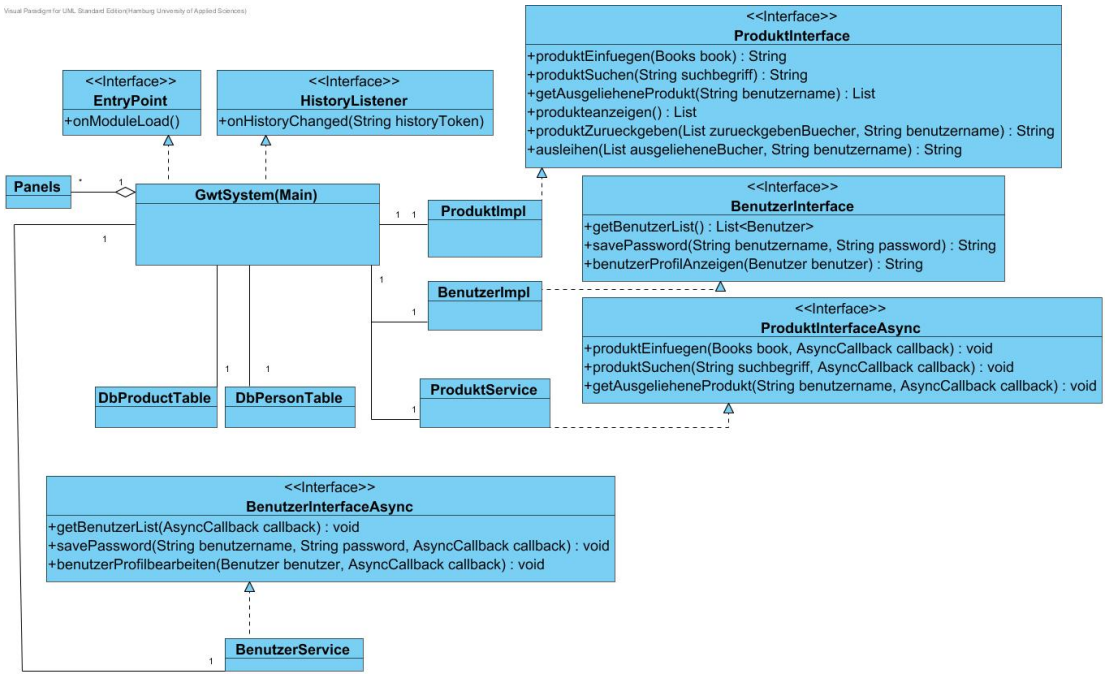

Abbildung 4.6: Biblio-GWT Objekt-Modell

aktuell im Cache gespeichert ist, angezeigt wird. Um dieses Problem zu lösen, wurde bei Google Web Toolkit die Historie-Klasse eingeführt, die für die Verwaltung der Historie der aufgerufenen Webseiten zuständig ist und deren Interaktionen. Jedes Element (Panel), das sich auf dem Stapel der GWT-Historie befindet, wird durch eine Zeichenkette repräsentiert. Um jede Änderung durchführen zu können, gibt es die HistoryListener-Schnittstelle. Dieses Interface erhält nur eine Methode: onHistoryChanged(String historyToken). historyToken repräsentiert das Element oder das Panel zu andern. Zu der Server-Komponente gehören die Klassen ProduktImpl, BenutzerImpl, DbPersonTable und DbProductTable. Das ProduktInterface und das BenutzerInterface erhalten alle Funktionen, die für die Verwaltung der serialisierten Klassen Produkt und Benutzer notwendig sind. Die beiden Interfaces werden von den Klassen ProduktImpl und BenutzerImpl auf der Server-Seite implementiert. Um die asynchrone Kommunikation zu ermöglichen, werden auch die entsprechenden asynchronen Interfaces (BenutzerInterfaceAsync und ProduktInterfaceAsync) definiert. Die asynchrone Kommunikation erfordert ein Callback-Objekt, wodurch der

Client benachrichtigt wird, wenn die Kommunikation abgeschlossen ist. Nachdem die Kommunikation abgeschlossen ist, wird die gesamte Kommunikation wieder mit dem Aufruf über das übergebene Callback-Objekt stattfinden. Der Aufrufer darf nicht blockiert werden, bis die Kommunikation beendet wird. Aus diesem Grund hat die asynchrone Methode keinen Rückgabe-Wert (void). Die interne Implementierung der Klassen und Interfaces wird im nächsten Kapitel ausführlich beschrieben.

4.4.3 Klassendiagramm der Biblio-ASP.NET-AJAX-Anwendung

Die Abbildung 4.7 zeigt das Klassendiagramm der Biblio-ASP.NET-AJAX Komponente. Die Klasse Controller ist eine statische Klasse, die die globale Steuervariable und Steuerstruktur für das ganze System speichert. Diese Klasse spielt eine große Rolle in dem Kontextwechsel zwischen den Seiten. Die Steuerelementen einer Seite können nicht von anderen Seiten zugegriffen werden. Die Klasse Benutzer-und ProduktVerwalten verwalten die Objekte der Klasse Produkt und Benutzer aus der Abbildung 4.5. Die Klasse Site.master ist ein Panel und repräsentiert die Master-Seite des Systems. Die anderen Klassen Biblio-Produktanzeigen und Biblio-Produkt-Vormerken etc. erben die Eigenschaften dieser Master-Seite.

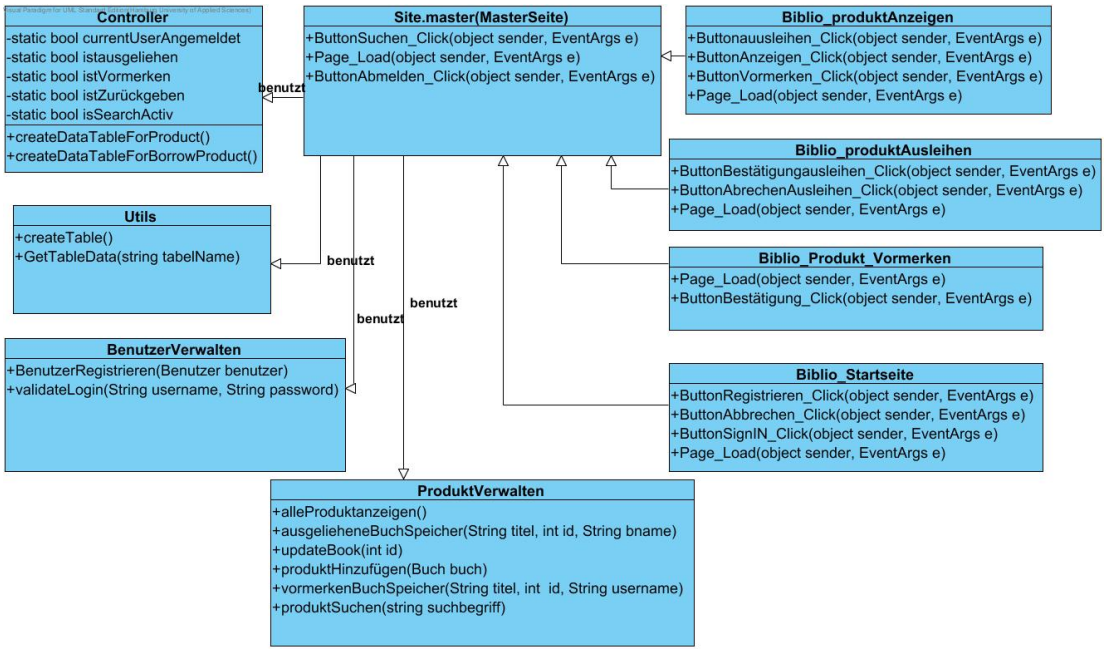

Abbildung 4.7: Klassendiagramm der Biblio-ASP.NET-AJAX -Anwendung

5 Implementierungen des Ausleihsystems

Die beiden Implementierungen des Ausleihsystems werden in diesem Kapitel beschrieben.

5.1 Implementierung von Biblio-GWT

Für die Implementierung von Biblio-GWT wurde für die Gestaltung der Benutzeroberfläche auf der Clientseite eine Swing-ähnlichen API verwendet. Die Steuerelemente werden in einem sogenannten Panel angeordnet. Der Quellcode in der Abbildung 5.1 zeigt eine anzeigenfunktion für die Darstellung aller Produkte auf einem Panel. Dieser Quellcode gehört zu der Klasse ProduktVerwaltenPanel, der die Klasse VerticalPanel aus der Standard gwt.user.client.ui erweitert.

Zu Beginn wird ein Behälter vom Typ VerticalPanel definiert. Dadurch werden die Produkte vertikal dargestellt. Danach wird das Button Anzeigen und eine Flextable zu dem vertikalen Objekt hinzugefügt. Der Button enthält ein ClickEvent, welches für das Senden von Nachrichten zum Server zuständig ist. Die Flextable erlaubt die Darstellung der Produkte in Form einer Tabelle. Um einen GWT-RPC Request an den Server anstoßen zu können, wird die Methode Produktanzeige aus der Klasse Produktservice aufgerufen (anzeigenRpc.produkteanzeigen(callback)). Der Parameter callback enthält die Methoden, die aufgerufen werden, wenn die Antwort vom Server zurückkommt. Ist ein Fehler aufgetreten wird die Methode OnFailure() aufgerufen und entsprechend darauf reagiert. Im Erfolgsfall wird die Methode OnSuccess() aufgerufen (siehe Flussdiagramm 5.2). Das Objekt result wird in eine List von Produkten gecastet und dem Flextable-Objekt hinzugefügt. Nun kommt die Startklasse GWTsystem ins

```
public void anzeigenGui(){
      anzeigenButton.addClickHandler(new ClickHandler()
            {
                  ProduktService anzeigenRpc = new ProduktService();

        public void onClick(ClickEvent event){

               AsyncCallback callback = new AsyncCallback()

               {
                  public void onFailure(Throwable caught)
                     {
                        System.out.println("Failed:");
                        sucesslabel.setText(" Failed in System");
                         System.out.println(" Failed in System
                           + caught.getMessage());

                     }

                  public void onSuccess(Object result)
                    {
                    alleProdukt = (List<Produkt>)result;
                    books.bucheranzeigen(flexTable, alleProdukt);

                    }


            };

            anzeigenRpc.produkteanzeigen(callback);


      }

});

}
```

Abbildung 5.1: Quellcode der Anzeige-Funktion

Spiel. Hier wird dann die Methode onHistoryChanged(String historyToken) aufgerufen, das RootPanel wird vorher gelöscht und die Produkte werden dann angezeigt.
Über ein großes Problem von GWT wurde bereits in einer Bachelorarbeit (vgl. With, 2010) berichtet. Zu beachten ist, dass nicht alle Java-Standard-Klassenbibliotheken in JavaScript übersetzt werden können. Die Standardmäßigen Java-Klassenbibliotheken wie zum Beispiel Java.text und Java.io fallen teilweise aus.

5.2 Implementierung von Biblio-ASP.NET-AJAX

Biblio-ASP.NET-AJAX wurde zunächst in ASP.NET und Visual C-Sharp implementiert und danach mit AJAX Funktionalität erweitert. Das ganze Objekt-Modell wird auf der Server-Seite implementiert. Die Implementierung beginnt mit der Realisierung einer Master-Seite. Die Master-Seite bietet einen Header, eine Funktion zum Suchen und eine zum Abmelden sowie ein Navigationsmenü an. Um die AJAX-Funktionalität zu gewährleisten, wurde hier das ScriptManager-Steuerelement platziert. Es darf nur einen ScriptManager pro Seite geben. Der Zugriff auf der zentral platzierte Script-Manager innerhalb der Seite ist durch einen ScriptManagerProxy zu gewährleisten, wobei der innere Aufbau des Proxies fast dem des ScriptManager-Steuerelements entspricht. Nun können alle anderen Webseiten diese Eigenschaften erben und neue Referenz hinzufügen. Desweiteren folgen die Anmeldung und die Registrierung eines Benutzers. Microsoft ASP.NET enthält ein fertiges Steuerelement Login, das für die Anmeldung und die Registrierung eines neuen Benutzers zuständig ist. Der Vorteil dieses Steuerelements liegt in der Einfachheit der Anwendung sowie in der integrierten Sicherheit wie zum Beispiel der Authentifizierung der Benutzer bei der Anmeldung. Jedoch ist dieses Steuerelement nicht erweiterbar. Es gibt keine Möglichkeit eine neue Benutzereigenschaft wie zum Beispiel Geburtsdatum in die Komponente Login hinzuzufügen. Daher wurde ein neues Login und eine neue Registrierung entsprechend dem Konzept dieser Arbeit entworfen. Die Abbildung 5.3 zeigt ein Postback für eine Anmeldung. ButtonSignIN-Click ist das Click-Event, das die Methode validateLogin() von der Klasse Benutzerverwalten aufruft, nachdem ein Benutzer seine Zugangsdaten eingeben hat und auf den Button Sign In geklickt hat. Die Rückgabe des Ergebnisses

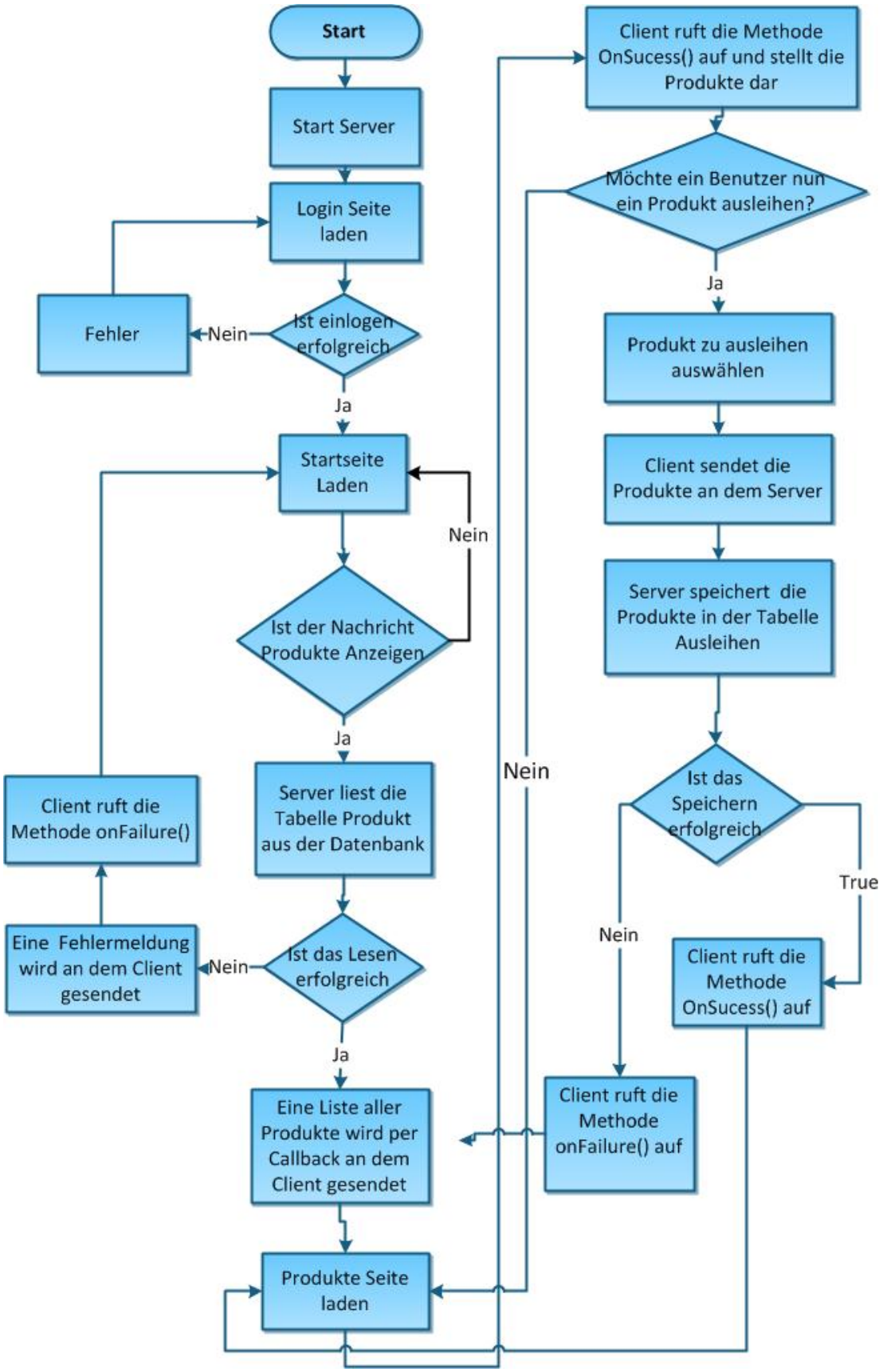

Abbildung 5.2: Biblio-GWT Haupt Flussdiagramm

wird dann von dem Callback des XmlHttp-Objekts übernommen und per normalen HTML-Text an den Client zurück gesendet.

```
2. <asp:Button ID="ButtonSignIN" runat="server" Text="Sign In"
3.  onclick="ButtonSignIN_Click" />
```

Abbildung 5.3: Postback an den Server

Nach der Anmeldung bekommt der Benutzer alle Produkte dargestellt. Im Gegensatz zu Biblio-GWT wird hier keine Flextable benutzt, sondern ein GridView-Steuerelement. Der Vorteil des GridView-Steuerelements liegt darin, dass es eine Funktion Gridview.dataSource hat, welche eine Liste von Daten speichert. Mit Gridview.dataBind() können die Daten automatisch dargestellt werden. Nun kann der Benutzer ein Produkt auswählen, ausleihen oder vormerken. Um die Daten aus einem Steuerelement serialisieren und deserialisieren zu können, muss der ViewState aktiviert werden. Nur durch den ViewState können Daten gespeichert und wiederhergestellt werden. Die Abbildung 5.3 zeigt ein Beispiel von dem Postback an dem Server. Nach der Implementierung kann die Seite mit der AJAX-Funktionalität erweitert werden. Die Abbildung 5.4 zeigt, wie eine ASP.NET-Seite, welche eine Master-Seite verwendet, zu einer ASP.NET AJAX-Seite erweitert wird. Dafür wurde in jede Seite eine UpdatePanel und ein ContentTemplate platziert. Mit einem UpdatePanel kann eine klassische Webseite einfach und elegant zu einer dynamischen Webseite erweitert werden. Der Stück-Code, das mit AJAX-Funktionalität erweitern werde soll, wird innerhalb des UpdatePanels und ContentTemplates platziert. Der Vorteil des UpdaptePanels liegt nicht an der Anforderung sondern an der Schnelligkeit der Antwort vom Server. Nur ein Teil der Seite wird vom Server aktualisiert und gerendert. Das entspricht dem Konzept des Partial Page Renderings, das im Kapitel Technologie besprochen wurde. Jedoch reicht das UpdatePanel für die Optimierung der ganzen Webseite nicht aus, weil die Netzwerk-Übertragung zwischen dem Client und dem Server genauso wie beim herkömmlichen Webseiten-Modell abläuft. Eine Anforderung entspricht einem vollständigen Postback. Beim Start der Webseite wird die ganze Seite an den Server

gesendet. Um die Seite zu optimieren gibt es die Funktion UdapteMode des UdaptePanels die auf **Conditionnal** gesetzt wird. Dies besagt, dass die Webseite neu geladen und aktualisiert wird, wenn eine bestimmte Bedingung erfüllt ist.

```
4. <asp:ScriptManagerProxy ID="ScriptManagerProsyausleihen"
   runat="server">
5. </asp:ScriptManagerProxy>
6.  <asp:UpdatePanel ID="UdaptePanelAusleihen" runat="server">
7. <ContentTemplate>
8. <asp:Button ID="ButtonAusleihen" runat="server"
     onclick="Buttonauusleihen_Click" Text="Ausleihen" />
9.  </ContentTemplate>
10.      </asp:UpdatePanel>
```

Abbildung 5.4: Erweiterung eines ASP.NET-Code zu ASP.NET AJAX

5.3 Fazit

Nach der Implementierung ist zu merken, dass die beiden Komponenten stark mit den Steuerelementen arbeiten. Für die Darstellung von Daten aus einer Datenbank kann bei ASP.NET AJAX ein GridView-Steuerelement benutzt werden, während in Java ein Flextable-Steuerelement verwendet wird. Die Umsetzung der Biblio-GWT mit AJAX Funktionalität wird automatisch durchgeführt. Bei Biblio-ASP.NET-AJAX muss das AJAX-Steuerelement manuell hinzugefügt werden. Die beiden Komponenten basieren auf der Client-Server-Architektur. Das bedeutet, die interne Kommunikation läuft nach dem Konzept der Request-und-Response-Kommunikation. Biblio-GWT verwendet die RPC-Kommunikation um den Server anzusprechen. Nach der Kommunikation kann der Client das Callback Objekt noch interpretieren, bevor er es im Browser darstellt. Aber In der Biblio-ASP.NET-AJAX wird direkt das Callback der XmlhttpRequest Objekt des Browsers, welches die Kommunikation zwischen dem Client (Browser) und dem Server ermöglicht, verwenden.

6 Vergleichsmethode

Der Kern dieser Arbeit ist der Vergleich von zwei Webanwendungen mit AJAX-Funktionalität. Um diesen Vergleich durchführen zu können, müssen die Kriterien, die verglichen werden sollen, herausgefunden werden. Weiterhin muss festgelegt werden, wie diese Kriterien verglichen werden sollen. Dieses Kapitel behandelt diese beiden Aspekte.

6.1 Allgemein

Ein optimaler Vergleich zweier Software-Systeme ist durch Anwendung von Kriterien möglich. Um Kriterien besser zu bewerten, müssen Metriken definiert werden, die die Aussage begründen können. In Softwarestechnik existieren spezielle Messtechniken die unter die Sogenannte Softwaremetrik zusammengefasst sind. „Eine Softwaremetrik ist eine Funktion, die eine Software-Einheit in einen Zahlenwert abbildet. Dieser berechnete Wert ist interpretierbar als der Erfüllungsgrad einer Qualitätseigenschaft der Software-Einheit“ (IEEE Standard 1061, 1992). Mit Software-Einheit ist der Quellecode gemeint. Die Softwaremetriken werden nach Ressourcen, Prozessen und Produkten klassifiziert (siehe Abbildung 6.1). Bei Ressourcen-Metrike kommen als interne Maße das Personal, das Produktionsmaterial sowie die Werkzeuge für die Entwicklung der Software im Spiel. Die externe Maße der Ressource-Metriken sind zum Beispiel die Erfahrungen des Personals und die Kommunikation des Personals untereinander. Die Prozess-Metriken dienen der Verbesserung der Produktqualität. Die internen Maße der Software-Prozess-Metriken sind zum Beispiel der Aufwand, die Zeit sowie die Anzahl der Fehler nach jeder Entwicklungsphase. Die Kosten hingegen gehören zu den externen Maßen der Prozess-Metriken. Die Produkt-Metriken wiederum dienen zur Verbesserung der Nutzungsqualität (vgl. Koschke, 2009). Die Produkt-Metriken

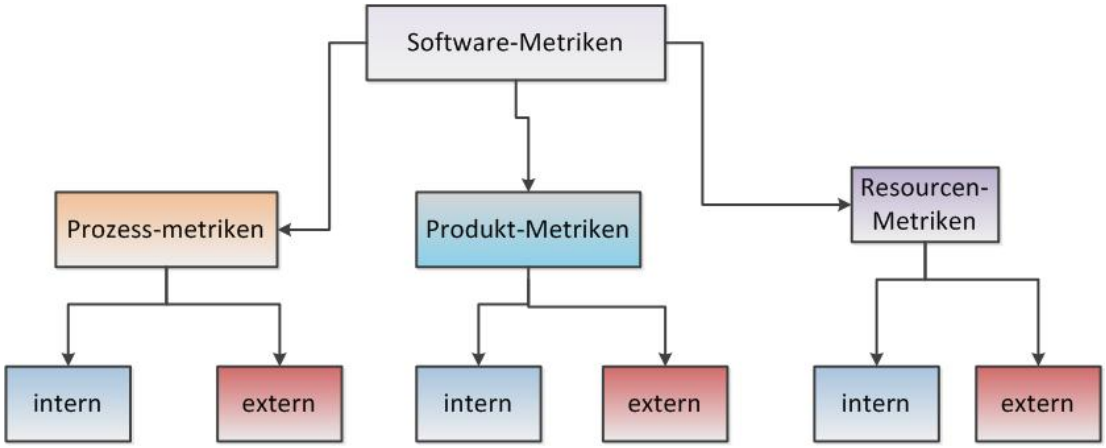

Abbildung 6.1: Klassifikation von Softwaremetriken nach Fenton und Pfeeger (1998) (vgl. Koschke, 2009)

sind die relevante Metriken für diese Arbeit. Um diese besser zu verstehen, wird das FCM-Modell (Factor-Criteria-Metrics-Modell) untersucht. Neben dem FCM-Modell gibt es das GQM-Modell (Goal-Question-Metric-Modell), das sich damit beschäftigt, wie die Metriken über Fragen ermitteln werden. Diese beiden Modelle werden im nächsten Abschnitt beschrieben.

6.1.1 Factoria-Criteria-Metric (FCM)

Das FCM-Modell misst die interne und die externe Qualität der Software mit dem Ziel die Nutzungsqualität des Softwareprodukts zu verbessern. Das Modell befasst sich mit drei Hauptpunkten der Software:

- Die Qualitätsmerkmale (factors, characteristics) dienen zur Beschreibung der Software-Qualität.
- Die Qualitätsmerkmale werden in Teilmerkmale (criterions, subcharacteristics) verfeinert.
- Die Teilmerkmale werden dann durch Maße bewertbar gemacht.

Ein konkretes Beispiel des FCM-Modells ist die Norm ISO/IEC 9126. In dieser Norm wird das FCM-Modell verwendet, um die interne und externe Qualität der Software-

produkte zu beschreiben. Das Modell besteht aus sechs Qualitätsmerkmalen (characteristics), die wiederum aus 26 Teilmerkmalen (subcharacteristics) bestehen. In drei Sätzen wird jedes Qualitätsmerkmal zusammengefasst und zu jedem Qualitätsmerkmal wird ein Beispiel eines Teilmerkmals beschrieben. (vgl. Balzert, 2008)

- **Zuverlässigkeit:** Fähigkeit des Softwareprodukts oder eines Systems, eine gewisse Funktion verlässlich in einem Zeitintervall zu erfüllen.
  - **Wiederherstellbarkeit:** Fähigkeit des Softwareprodukts, die Daten bei einem Versagen in einem spezifizierten Leistungsniveau wiederzugewinnen.
- **Funktionalität:** Fähigkeit des Softwareprodukts, Funktionen bereitzustellen sowie die Beschreibung inwiefern die Anwendung die geforderten Funktionen erfüllt.
  - **Sicherheit:** Fähigkeit des Softwareprodukts, nicht autorisierten Personen oder System zu verhindern, Informationen und Daten zu verwenden.
- **Wartbarkeit:** Fähigkeit des Softwareprodukts änderungsfähig zu sein. Wie schnell kann ein Produkt verbesssert und in eine neue Umgebung angepasst werden?
  - **Testbarkeit:** Fähigkeit des Softwareprodukts, die modifizierte Software zu validieren.
- **Benutzbarkeit:** Fähigkeit des Softwareprodukts , vom Benutzer verstanden und benutzt zu werden.
  - **Erlernbarkeit:** Fähigkeit des Softwareprodukts , den Benutzer zu befähigen, die Anwendung zu lernen.
- **Effizienz:** Fähigkeit des Softwareprodukts, ein angemessenes Leistungsniveau bezogen auf die eingesetzten Ressourcen und unter festgelegten Bedingungen bereitzustellen.
  - **Zeitverhalten:** Fähigkeit des Softwareprodukts, Antwort, die Verarbeitungszeit sowie den Durchsatz bei der Funktionsausführung sicherzustellen.

- **Portabilität:** Fähigkeit des Softwareprodukts, von einer Umgebung in eine andere übertragen zu werden.
  - **Installierbarkeit:** Fähigkeit des Softwareprodukts, in eine festgelegte Umgebung installiert zu werden.

Im nächsten Abschnitt wird ein Modell beschrieben, das verwendet wird, um diese Metriken über Fragen und aus den Messzielen abzuleiten.

6.1.2 Goal-Question-Metric (GQM)

Das GQM-Modell wurde 1984 von Basili und Weiss entwickelt. Das Modell spielt heutzutage eine große Rolle in der Entwicklung des Qualitätsmodells eines Softwareprodukts. Das GQM-Modell hilft, die Softwaremerkmale (Metriken) für die Bewertung und das Messen eines Softwareprodukts zu definieren. Das GQM-Modell besteht aus drei Hauptphasen:

- Die Ziele werden erfasst (**Goal**).
- Notwendige Fragen werden abgeleitet, um zu prüfen, ob die Ziele erreichen wurden (**Question**).
- Was muss gemessen werden, um diese Fragen zu beantworten? (**Metric**). (vgl. Koschke, 2009)

Die Ergebnisse dieses Modells können in Form eines Baum dargestellt werden. Die nächste Abbildung 6.2 zeigt ein Beispiel von GQM nach Fenton. Das Ziel ist es, die Effektivität der Codierrichtlinien zu bestimmen. Eine Frage, die aus dem Ziel abgeleitet werden kann, ist zum Beispiel: Wie ist die Produktivität der Programmierer? Als Metriken können die Erfahrungen der Programmierer mit den Entwicklungssprachen, der Umgebung geschätzt werden. Weiterhin können auch der Entwicklungsaufwand sowie die Codegröße gemessen werden. Im Bezug auf diese Arbeit zeigt die Tabelle 6.1 ein Beispiel eines GQM-Modells. Das Messziel ist es, alle Bücher von Biblio-GWT und Biblio-ASP.NET-AJAX auszuleihen. Eine Frage könnte sein: Wie lange dauert es, um alle Bücher auszuleihen. Die Metriken könnten die Ladezeit der Seite beim Ausleihen aller Bücher mit und ohne parallelem Zugriff sein.

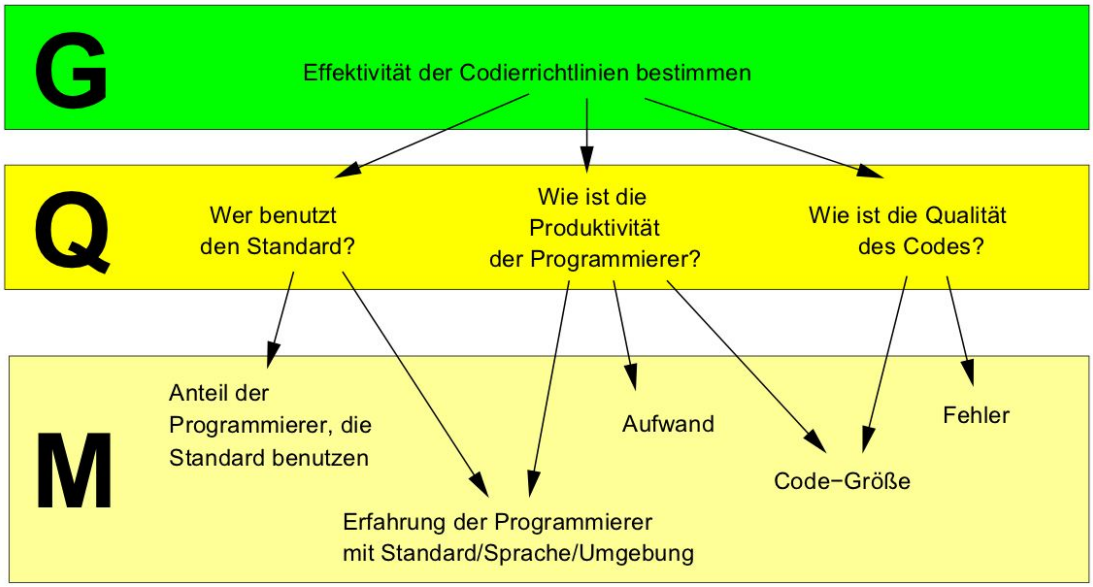

Abbildung 6.2: Beispeil eines GQM-Baums (vgl. Koschke, 2009)

Tabelle 6.1: GQM mit Biblio-ASP.NET-AJAX und Biblio-GWT

Messziele	**Fragen**	**Metriken**
Alle Bücher von Biblio-GWT und Biblio-ASP.NET-AJAX sollen ausgeliehen werden	Wie lange dauert es, alle Bücher auszuleihen?	Performance in ms: - Ladezeit der Seite - Paralleler Zugriff
	Wie viele Probleme können beim Ausleihen aller Bücher auftreten?	- Zuverlässigkeit - Fehlerrate messen
Dauer der gesamten Entwicklung von Bibio-Gwt und Biblio-ASP.NET-AJAX minimisieren	Wie lange dauert die gesamte Entwicklung?	Entwicklungsaufwand messen
	Welche Ressourcen werden gebraucht?	- Teamgröße - Werkzeuge - Erfahrungen

In dieser Arbeit wird mehr Wert auf den Entwicklungsaufwand und die Performance der beiden Anwendungen gelegt. Die entsprechenden Metriken, die in der Tabelle 6.1 präsentiert werden, werden mit speziellen Werkzeugen im Kapitel 7 untersucht.

6.2 Entwicklungsaufwand

In einem Projekt spielt die Schätzung des Entwicklungsaufwands einer Software eine sehr große Rolle für die Erstellung eines Angebots für den Kunden. Nach der Kalkulation des gesamten Aufwands kann der Hersteller den Liefertermin mit dem Kunden festlegen. Weiterhin können die benötigten Ressourcen eingeplant werden. Um diesen Aufwand optimal zu schätzen, existieren Schätzverfahren, die alle auf Erfahrungen beruhen.

6.2.1 Expertenschätzung

Die Expertenschätzung ist eine in der Praxis häufig benutzte Methode für die Aufwandschätzung. Aufgrund von langjähriger Erfahrung mit vielen verschiedenen Projekten beurteilt eine Gruppe von Experten ein Projekt mit dem Ziel, eine Voraussage über den Zeit- und Ressourcen-Bedarf sowie die Kosten der einzelnen Projektaktivitäten zu machen (vgl. Wirsing, 2006). Ein Beispiel für eine systematische Expertenschätzung ist die **Delphi-Methode**. Beim ersten Treffen zerlegt eine Gruppe von 3 bis 7 Experten das Projekt in 10 bis 20 Teilaufgaben. Jede Teilaufgabe wird dann separat von jedem Experten geschätzt und dokumentiert. Beim zweiten Treffen tauschen sich die Experten über ihre Schätzungen aus und diskutieren über die unterschiedlichen Annahme und Unklarheiten mit dem Ziel, für jede Teilaufgabe eine akzeptierte Schätzung zu bestimmen. (vgl. Balzert, 2009). Jedoch kann bei der Delphi-Methode dramatisch falsche Ergebnisse interpretiert werden, die scheinbar vertrauenswürdig sind. Neben der Expertenschätzung existiert auch die algorithmische Schätzung. Ein Beispiel für die algorithmische Schätzung ist das **Function-Point-Verfahren**.

6.2.2 Das Function-Point-Verfahren

„Das Function-Point-Verfahren (auch -Analyse oder -Methode, kurz FPA) dient zur Bewertung des fachlich-funktionalen Umfangs eines Informationstechnischen Systems, im Folgenden als Anwendung bezeichnet.“ (vgl. Wikipedia, 2011c). Die funktionalen Anforderungen aus der Sicht des Benutzers sind die Basis dieser Methode. Die Idee ist es, die funktionalen Anforderungen, die vorher schon im Lastenheft beschrieben

werden, zu verwenden und darauf eine systematische Auswertung durchzuführen und ein Bild über den gesamten Umfang des Projekts zumindest aus der Sicht der Benutzer zu erhalten. Der Vorteil dieser Methode liegt darin, dass eine frühere Aussage über den funktionalen Umfang des Projekts in der Analyse-Phase möglich ist (vgl. Balzert, 2009). Ein weiterer Punkt, der nicht zu vergessen ist, sind die eigenen Erfahrungen der Entwickler aus dem vorherigen Projekt oder einem ähnlichen Projekt. Damit können der Aufwand, die Dauer, sowie die Produktivität schneller geschätzt werden. Leider wird das Function-Point-Verfahren in der Praxis wenig benutzt.

6.3 Performance

In Allgemein versteht man unter Performance die Reaktionszeit eines Systems auf ein Ereignis: Wie schnell reagiert ein System auf ein Ereignis? Das System in dieser Arbeit ist eine Webanwendung. Um die Performance einer Webanwendung zu testen, gibt es relevante Kenngrößen wie zum Beispiel die Ladezeit, welche die Dauer für das Laden einer Webseite bestimmt. Je länger die Seite lädt, desto wahrscheinlicher ist es, dass der Benutzer die Seite wieder verlässt, bevor die Seite vollständig geladen worden ist. Daher sollte die Ladezeit der Seite idealerweise sehr kurz sein. Ein weiterer wichtiger Punkt ist der parallele Zugriff auf die Seite. Wie verhält sich die Anwendung bei mehreren gleichzeitigen Zugriffen? Wichtig ist auch der Ressourcenverbrauch, welcher durch die CPU und den Hauptspeicher bestimmt ist. Die beiden Anwendungen werden unter dem gleichen Computer getestet. Wird das Verteilen von Ressourcen gleich? Diese Fragen werden mit Hilfe eines speziellen Werkzeugs im nächsten Kapitel beantwortet.

6.4 Fazit

Im diesem Kapitel wurden die Vergleichsmethoden beschrieben, die für die Bewertung einer Software notwendig sind. Dafür wurden 2 Modelle präsentiert, die für die Entwicklung von Metriken zuständig sind. In Bezug auf diese Arbeit wurden zwei Metriken festgelegt: der Entwicklungsaufwand und die Performance. Um den Entwicklungsaufwand schätzen zu können, wurden zwei Schätzmethoden gesprochen,

die beide auf Erfahrungen beruhen. Weiterhin sind diese Schätzmethoden notwendig für die Planung der Ressourcen, der Kosten und der Zeit. In dieser Arbeit sind die Kosten und die Ressourcen nicht relevant, daher wird mehr Wert gelegt auf die Entwicklungszeit der beiden Anwendungen. Um die Performance messen zu können, werden spezielle Werkzeuge benutzt. Mit ihnen werden die CPU-Ressourcen sowie die Ladezeit der Seiten der Anwendungen gemessen. Das nächste Kapitel widmet sich den beiden Metriken.

7 Bewertung

Die beiden Anwendungen werden in diesem Kapitel sowohl lokal als auch entfernt getestet. Nach dem Test werden die Ergebnisse analysiert und bewertet.

7.1 Entwicklungsaufwand

In dieser Arbeit wird die Stunde als Bezugseinheit verwendet. Diese wird in Arbeitstage umgerechnet. Die Anzahl der Arbeitstage werden dann als Vergleichsgröße betrachtet. Bei der Umrechnung der Stunden wird ein Arbeitstag mit 7 Arbeitsstunden berechnet. Die gesamte Arbeitszeit während dieser Arbeit betrug 6 Monate. Die Bearbeitung pro Woche betrug 18 Stunden. Daraus ergeben sich 72 Stunden pro Monat. Insgesamt wurden 432 Stunden in diese Arbeit investiert. Das ist der gesamte Aufwand, der in dieser Arbeit für die Einarbeitung und die Implementierung der beiden Anwendungen geleistet wurde. Dies kann auch als der maximale Entwicklungsaufwand der beiden Anwendungen angesehen werden. Die Tabelle 7.1 zeigt die Ergebnisse des Entwicklungsaufwands

Tabelle 7.1: Entwicklungsaufwand-Ergebnisse

	Bibilo-GWT	**Bibilo-ASP.NET-AJAX**
Einheiten	Stunde/Arbeitstage	
Einarbeitung	54/7,71	72/12,28
Implementierung	90/12,85	36/5,14
Entwicklungsaufwand	144/20,57	108/15,42
Gesamter Entwicklungsaufwand	252/36	

7.1.1 Begründung der Ergebnisse des Entwicklungsaufwands

Biblio-ASP.NET-AJAX

Die Entwicklung der Webanwendung mit dem Framework ASP.NET AJAX mit der Sprache C-Sharp war für die Autorin eine neue Herausforderung. Sie musste sich in zwei neue Sprachen einarbeiten: C-Sharp und in der Skriptsprache ASP.NET. Dies hat dazu geführt, dass die Einarbeitung 72 Stunden gedauert hat. Nach der Einarbeitung ist die Implementierung einfach und schnell gegangen,weil die Sprache C-Sharp große Ähnlichkeiten mit der Sprache Java hat, mit der die Autorin vertraut ist. So konnten die bereits in Java implementierte Modelle schneller im C-Sharp umgesetzt werden. Dies hat zu einer Implementierungszeit von 36 Stunden geführt.

Biblio-GWT

Die Einarbeitung in Java war nicht nötig.Dies hat dazu geführt, dass die Einarbeitung nur 54 Stunden in Anspruch genommen hat. Aber Die Implementierung hat 90 Stunde in Anspruch genommen. Ein Grund liegt an der Komplexität von GWT-Projekt. Die Konfiguration der Entwicklungsumgebung und der Aufbau des RPC-Kommunikation mit dem Server ist nicht einfach zu realisieren.

Entwicklungsaufwand

Um den gesamten Aufwand zu berechnen, wird der Entwicklungsaufwand der Einarbeitungsphase mit dem Entwicklungsaufwand der Implementierungsphase summiert. Für Biblio-GWT ergibt sich den Gesamtaufwand von 144 Stunde, was 144/7= 20,57 Arbeitstagen entspricht. Biblio-ASP.NET-AJAX hat insgesamt 108 Stunden im Anspruch genommen. Dies entspricht 108/7 = 15,42 Arbeitstagen.

7.1.2 Fazit

Die Ergebnisse der Tabelle 7.1 zeigen, dass die Entwicklung von Biblio-ASP.NET-AJAX weniger Zeit in Anspruch genommen hat. Ein Grund dafür ist die Ähnlichkeit von C-Sharp und Java. So konnte das gemeinsame Modell, das vorher für Biblio-GWT angewendet wurde, schneller in C-Sharp umgesetzt und für Biblio-ASP.NET-AJAX

verwendet werden. Es wurde mehr Zeit für die Implementierung von Bibilo-GWT investiert. Ein Grund liegt an der konfiguration von GWT-Projekt. Dies führt zur mehr Aufwand als bei der Biblio-ASP.NET-AJAX. Aus diesen Ergebnisse kann man ableiten, dass eine Webanwendung schneller mit dem Webframework ASP.NET AJAX erstellt werden kann.

7.2 Performance-Test

Ein gute Bewertung der Performance soll mit guten Ergebnissen begründet werden. Um diese Ergebnisse zu erhalten, sollen gute Testmethoden und Werkzeuge eingesetzt werden. Für diese Arbeit wurden zwei Tests durchgeführt: lokaler und entfernter Test. Dafür wurden zwei Werkzeuge verwendet: NeoLoad für den Lokalen-und Apache JMeter für den entfernten Test.

7.2.1 Ausgewählte Werkzeuge

NeoLoad

Das erste für diese Arbeit ausgewählte Werkzeug ist eine komerzielle Software mit dem Namen NeoLoad der Firma Neotys mit dem Sitz in Gemenos bei Marseille in Frankreich. NeoLoad ist eine Java-basierte Applikation, die unter Windows, Solaris und Linux lauffähig ist.Es ist eine Lösung für Last- und Performance-Tests. Es Kann für AJAX, GWT, .NET, Oracle Form, SAP sowie viele andere Technologie angewendet werden (vgl. NEOTYS, 2011).

Apache JMeter

Apache JMeter ist ein Java-basiertes freies Werkzeug für die Durchführung von Lasttests in Client-Server-Anwendungen. Apache JMeter wird verwendet, um die Leistung von dynamischen Ressourcen (Java Objekts, Datenbanken und Abfragen, FTP-Server und mehr) zu testen. Es kann auch benutzt werden, um eine schwere Last auf einem Server über das Netzwerk zu simulieren, um seine Stärke zu testen oder die Gesamtleistung unter verschiedenen Last-Typen zu analysieren (vgl. Apache, 2011a).

Um den entfernten Test mit JMeter zu verstehen, wird zunächst ein kleiner Überblick über das Prinzip von RMI (Remote Method Invocation) gegeben.

Remote Method Invocation

JMeter verwendet RMI (entfernter Methodenaufruf), um die Kommunikation zwischen zwei Computern zu ermöglichen. RMI ist eigentlich der Aufruf einer Methode von einem entfernten Java Objekt. Ein Java-Objekt, das auf einem anderen Rechner oder einen anderen Netzwerk instantiiert ist, arbeitet transparent auf dem lokalen Rechner als ob es das Java-Objekt der lokalen Java Virtuelle Maschine (JVM) wäre. Die Abbildung 7.1 zeigt die Architektur eines RMI-Systems.

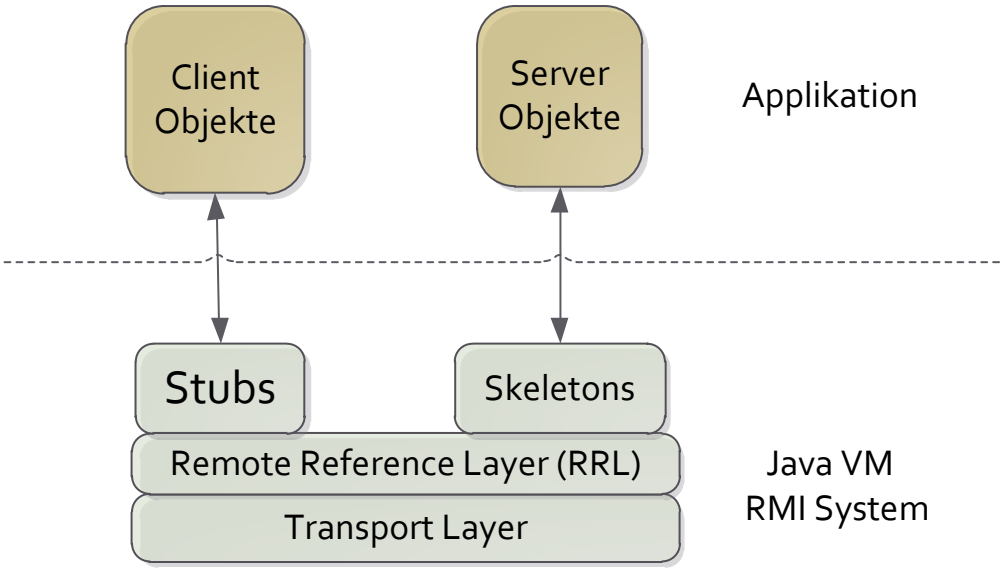

Abbildung 7.1: Architektur eines RMI Systems (vgl. André Möller, 1998)

Das RMI-System besteht aus drei schichten: der Stub/Skeleton-Layer, der Remote Reference-Layer, und der Transport-Layer. Die Stub/Skeleton-Schicht beschäftigt sich mit der Übertragung der Daten zu und von der Remote-Reference-Schicht durch das Prinzip des „Parameter Marshalling“. Möchte ein Objekt im Client vom Server aufrufen, wird im Adressraum der JVM des Clients ein sogenannter Stub erzeugt. Der Stub ist das

lokale Stellvertreterobjekt und repräsentiert ein reguläres Java-Objekt für die entfernte Kommunikation. Auf der Serverseite bekommt das entfernte Stellvertreterobjekt des eigentlichen Remote-Objektes, bezeichnet als Skeleton. Über das Skeleton kann die dortige Remote-Reference-Schicht mit dem Server kommunizieren. Die Antworten, Fehler und Exceptions werden auf dem gleichen Weg zum Client zurückgesendet. Eine weitere Aufgabe der Remote-Reference-Schichtt ist die Kommunikation mit der Transportschicht zu regeln und das Protokoll zu definieren. Im Abschluss sorgt die Transportschicht für den Aufbau und die Verwaltung der Verbindungskanäle zu dem entfernten Adressraum (vgl. André Möller, 1998).

7.2.2 Entfernter Testaufbau

Der entfernte Test mit JMeter wird nach dem Konzept von Master-Slave-Target aufgebaut. Die Abbildung 7.2 zeigt den Testaufbau für einen entfernten Test. Apachen JMeter besteht aus zwei Komponenten: Jmeter GUI und Jmeter-Server. Der Master ist der Computer, in dem die JMeter-GUI installiert ist. Der Master ist ein entfernter Client, der die Anforderungen an dem Webserver sendet. Im Java VM-Adressraum des Masters wird das lokale Stellvertreterobjekt (Stub) erzeugt. Der Slave ist der Computer, auf dem der JMeter-Server und der Webserver installiert sind. JMeter-Server ist das entfernte Stellvertreterobjekt, das mit dem lokalen Webserver kommuniziert. Damit der Master den Slave erreichen kann, muss die IP-Adresse des Slaves in der jmeter.properties-Datei vom Master eingetragen werden. Auf der Slave-Seite muss nur der RMI-Port konfiguriert werden. Der Default Port ist 1099. Für die Kommunikation zwischen dem Master und dem Slave stehen zwei Protokolle zur Verfügung: JRMI (Java Remote Method Protocol) und RMI-IIOP (Internet Inter-Orb Protocol). Die Verwendung von RMI-IIOP ist möglich ab Java 1.3 (vgl. Apache, 2011b).

Für eine erfolgreiche Kommunikation müssen folgende Punkte vor dem Testen berücksichtigt werden: die Firewalls von allen Computern müssen deaktiviert werden. Der Master und der Slave müssen in dem gleichen Subnetz sein. Die gleiche Version von JMeter und Java muss auf allen Computern installiert werden. Für diese Arbeit wurde JMeter 2.5 und Java 1.6 verwendet. Nach der Konfiguration der Testumgebung können Arbeitsschritte mit Hilfe eines Proxy-Servers gesammelt werden. Ein Proxy-

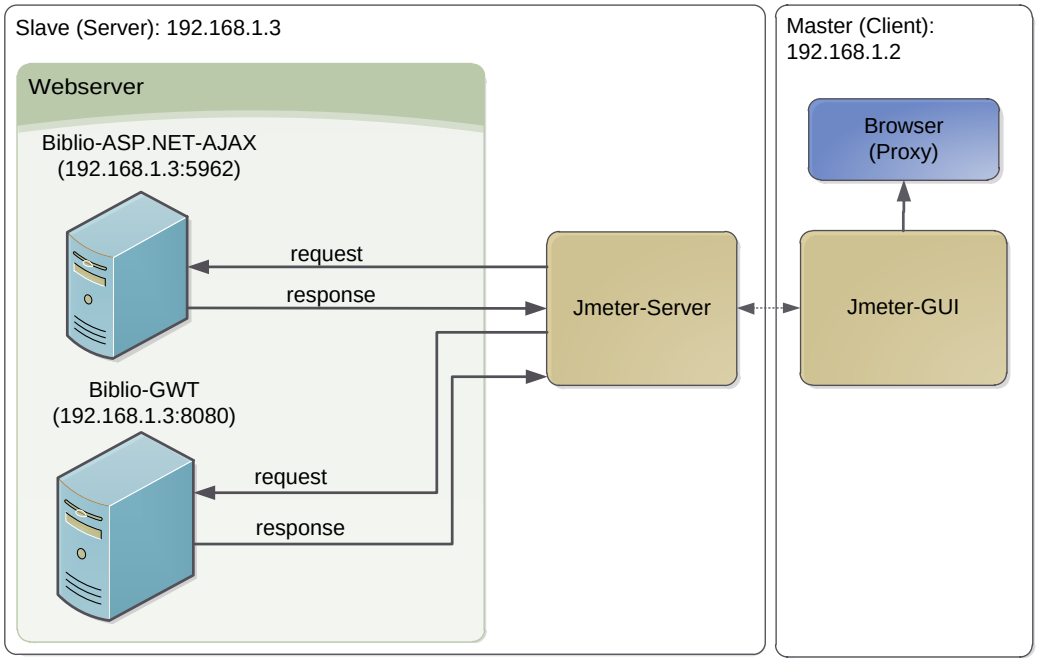

Abbildung 7.2: Testaufbau mit Tomcat und IIS Express als Webserver

Server ist ein Vermittler, der die Anfrage entgegennimmt und über seine Adresse andere Verbindung herstellen und die Daten weiterleiten kann. Ohne ein Proxy-Server ist ein entfernter Test mit JMeter sehr aufwändig. Dies ist eine Einschränkung von JMeter. Um einem Proxy benutzen zu können müssen Biblio-GWT und Biblio-ASP.NET-AJAX veröffentlicht werden. Dafür wurden zwei Webserver verwendet: Microsoft Internet Information Services (IIS 7.5 Express) und Apache Tomcat 7.

Veröffentlichung von Biblio-ASP.NET-AJAX mit Microsoft IIS

Der entfernte Zugriff auf den ASP.NET Development Server von Microsoft, der benutzt wurde, um Biblio-ASP.NET-AJAX zu entwickeln, ist nicht möglich. Der ASP.NET-Development-Server ist bereits in Microsoft Visual Studio 2010 integriert. Dafür muss die Biblio-ASP.NET-AJAX zu den Webserver Microsoft IIS Express migrieren, um der Zugriff auf die Webseite im Subnetz zu ermöglichen. Dafür wurde Internet IIS 7.5

Express und das Microsoft Visual Studio 2010 Service Pack 1 installieren. Nach der Installation kann die Anwendung mit Hilfe der IIS-Managers veröffentlicht werden.

Veröffentlichung von Biblio-GWT mit Apache Tomcat 7

Für diese Arbeit wurde ein GWT-Plug-In für Eclipse benutzt. Um Biblio-GWT in Tomcat veröffentlichen zu können, muss das Projekt kompiliert werden. Es musst sichergestellt werden, dass das Projekt im Web-Mode erfolgreich lauft. Der Inhalt des War-Ordners muss kopiert und zu einer War-Datei komprimiert werden. Diese War-Datei enthält alle benötigten Jar-Dateien der Webanwendung, Klassendateien sowie weitere Dateien sowie weiteren Datei wie CSS-Datei, HTML-Datei. Die War-Datei soll dann im tomcat/webapps/ kopiert werden. Tomcat kann jetzt neu gestartet werden, damit das ganze Projekt neu kompiliert und veröffentlicht werden kann. Wichtig dabei ist die Java Version. Tomcat muss mit der gleichen Java-Version laufen, die bei Eclipse angewendet wird, sonst wird eine „bad version Exception“ geworfen.

7.2.3 Durchführung der Tests

Durchführung des lokalen Tests

Der lokale Test wurde mit NeoLoad durchgeführt. Der Test besteht darin, die Arbeitsschritte auf die Webseite mit Hilfe einer intelligenten GUI zu sammeln. Danach kann eine „Population"(menge von Virtuellen Benutzer) simuliert werden, welche die Anforderungen an dem Lokalen Webserver senden. Der Vorteil des Tests mit NeoLoad ist, dass keine Konfiguration von einem Proxy-Server auf dem Browser benötigt wird. Für dieser Arbeit wurden 10 Benutzer simuliert, die alle Bücher im System ausleihen. Die Ergebnisse sind im nächsten Abschnitt „Ergebnisse und Bewertung“ zu finden. Wegen der Einschränkung der Testversion auf 10 Benutzer wurde NeoLoad nur für die lokale Testmethode in dieser Arbeit verwendet.

Durchführung des entfernten Tests

Bei der Durchführung des entfernten Test sollen zuerst die Arbeitsschritte erfasst werden. Dafür wurde ein Proxy-Server verwendet, welches der Browser ist. Die

konfiguration des Browsers sieht folgendermaßen aus: (Http-Proxy: localhost, Portnummer: 9999). Für die Erfassung der Arbeitsschritte werden drei Einstellungen in JMeter-GUI gemacht: Einstellung einer Threadgruppe, eines Recording-Controllers und des Proxy-Servers. Die Threadgruppe repräsentiert die simulierten Benutzer. Das Recording-Controller-Element speichert alle Arbeitsschritte des Proxy-Servers. Damit die JMeter-GUI die Arbeitsschritte des Benutzers erfassen kann, braucht sie eine Portnummer (z.B. 9999) des Proxy-Servers. Nach der Erfassung der Arbeitsschritte, wird für den Test weitere JMeter-GUI-Elemente benötigt. Eines der wichtigsten ist das „Simple Data Writer Element“. Dieses Element speichert alle Ergebnisse vom JMeter-Server in einer Datei auf dem lokalen Rechner des Clients, die später analysiert werden können. Am besten wird für jede Anforderung zum Server ein Simple Data Writer Element verwenden, um die Antwort von Server zu speichern, wenn der Client beschäftigt ist. So wird ein Schreib-Konflikt auf die gleiche Datei vermieden. Ohne dieses Simple Data Writer Element wird die Verbindung zwischen dem JMeter-Server (Slave) und der JMeter-GUI (Master) ständig unterbrochen. Beim Start des Tests ist die Reihenfolge wichtig. Mit der Funktion „Remote Start“ wird die verbindung mit JMeter-Server initializiert. Dadurch bekommt der JMeter-Server die Kontaktdaten von dem Webserver (IP-Adresse und Portnummer), mit dem er kommunizieren sollte. Mit der Funktion „Start“ werden die konkrete Daten zum JMeter-Server gesendet.

7.2.4 Ergebnisse und Bewertung

Lokale Testergebnisse und Bewertung

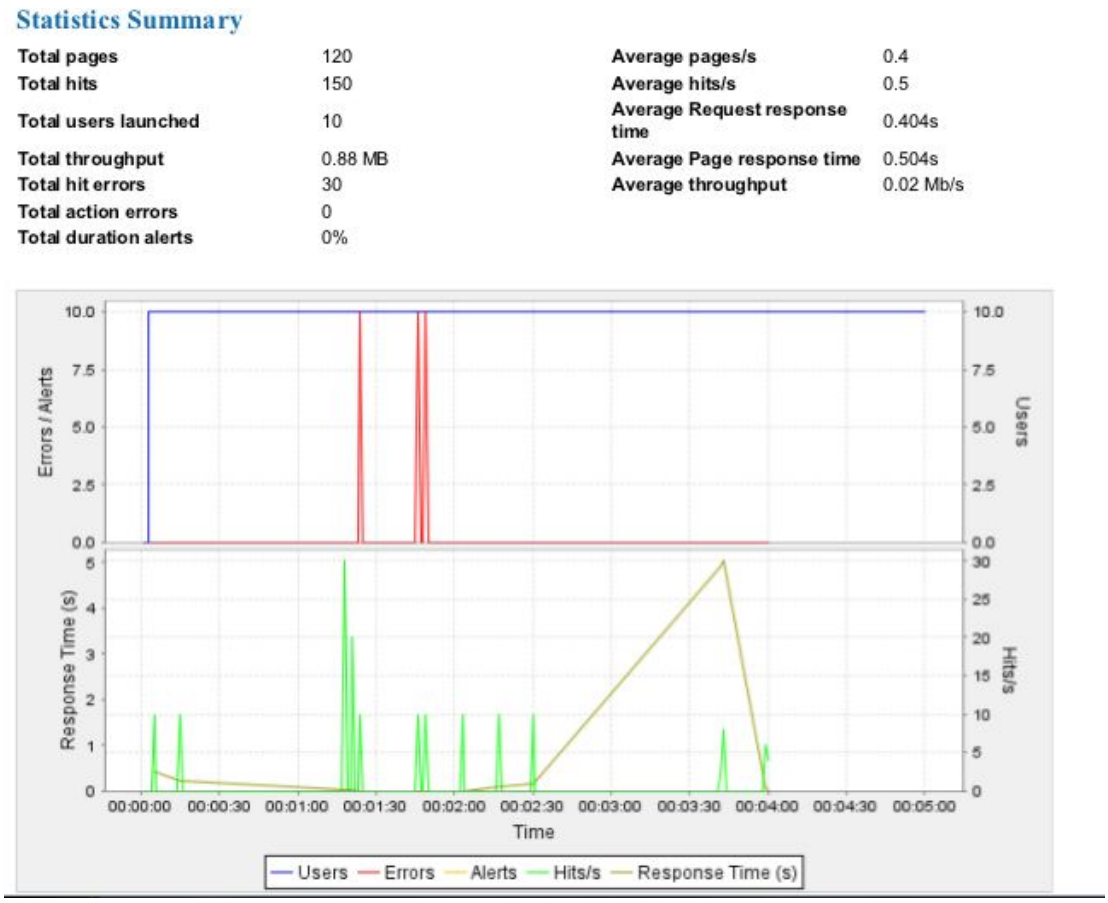

Abbildung 7.3: Statistik von Biblio-GWT mit 10 Benutzern beim Ausleihen aller Bücher

Die Abbildungen 7.3 und 7.4 zeigen die Ergebnisse des lokalen Tests von Biblio-GWT und Biblio-ASP.NET-AJAX mit Neoload. Die Abbildungen liefern Informationen über die gesamte Testdauer, die Gesamtzahl der Zugriffe, der Gesamtdurchsatz in Megabyte, die der Client vom Server erhalten hat, die Anzahl der Fehler, die durchschnittliche Reaktionszeit für HTTP-Zugriffe und für Antwortzeit für Webseiten. Bei Biblio-GWT zeigen die Ergebnisse, dass die durchschnittliche Reaktionszeit auf aller Seiten beim Ausleihen aller Büchern der Datenbank 505 ms beträgt. Insgesamt wurden 150 Re-

Statistics Summary

Total pages	140	**Average pages/s**	0.5
Total hits	142	**Average hits/s**	0.5
Total users launched	10	**Average Request response time**	3.49s
Total throughput	0.29 MB	**Average Page response time**	3.49s
Total hit errors	122	**Average throughput**	0.01 Mb/s
Total action errors	0		
Total duration alerts	0%		

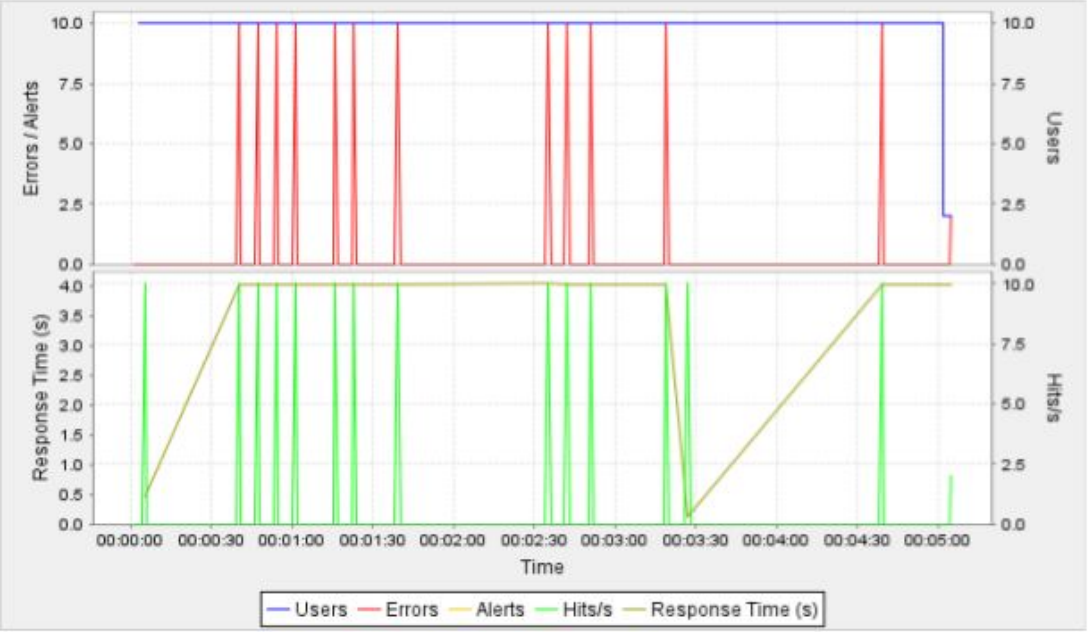

Abbildung 7.4: Statistik von Biblio-ASP.NET AJAX mit 10 Benutzern beim Ausleihen aller Bücher

quests zum Server gesendet und 30 Fehler entdeckt. Im Vergleich zu Biblio-GWT zeigen die Ergebnisse von Biblio-ASP.NET-AJAX, dass die durchschnittliche Reaktionszeit aller Seiten 3490 ms beträgt. 142 Request wurden zum Server gesendet und 122 Fehler gefunden. Mit diesen Ergebnissen lässt sich zeigen, dass Biblio-GWT schneller als Biblio-ASP.NET-AJAX wäre. Aber diese Aussage kann noch nicht bestätigt werden, weil der Server und der Client sich auf dem gleichen Rechner befinden. Dies kann auch die Ergebnisse beeinflussen. In einem realen Projekt befinden sich Client und Server auf zwei verschiedenen Rechner. Daher sollen die Ergebnisse des entfernten Tests analysiert werden.

Entfernte Testergebnisse und Bewertung

Der entfernte Test wurde mit JMeter durchgeführt. Bei dem Test wurden folgende Seiten betrachtet: Startseite, Produktanzeigen-Seite (anzeigen alle Produkte der Datenbank), Ausleihen-Seite, und Suchen-Seite. Für Jede Seite wurden die minimale, maximale und die durchschnittliche Antwortzeit des Webservers betrachtet. Die Tabbellen 7.5, 7.6, 7.7 und 7.8 zeigen die Ergebnisse. Aus diesen Abbildungen kann man erkennen, dass der erste Aufruf der Seite bei den Beiden Anwendungen fast immer unterschiedlich als der nächste Aufruf ist. Zum Beispiel bei der Startseite der Biblio-GWT dauert der minimale erste Aufruf des Clients 5 ms. Der nächste Aufruf mit 10 simulierten Benutzer dauert nur 3 ms. Das gleiche Verhalten gilt auch für Biblio-ASP.NET-AJAX. Der Grund dafür könnte am Cache liegen. Beim ersten Aufruf wird die Seite von dem Server geladen, aber der nächste Aufruf kommt aus dem Cache. Die Ergebnisse zeigen auch, dass die maximale Antwortzeit der beiden Anwendungen mit der Erhöhung der Anzahl der simulierten Benutzer steigt. Bei 500 Benutzern dauert die maximale Antwortzeit der Startseite bei Biblio-GWT 17339 ms. Bei Biblio-ASP.NET-AJAX dauert es nur 5654 ms. Um eine Aussage über die Performance der Anwendung zu machen, wird die durchschnittliche Antwortzeit der beiden Anwendungen auf allen Seiten analysiert. Beim Laden der Startseite ist Biblio-GWT schneller als Biblio-ASP.NET-AJAX. Beim Laden aller Bücher aus der Datenbank auf die Seite Produktanzeigen ist Biblio-GWT deutlich langsamer als Biblio-ASP.NET-AJAX . Dies lässt sich auch mit den Grafiken 7.9 und 7.10 bestätigen. Beim Ausleihen der Bücher

hat Biblio-GWT gekrönt. Die Ergebnisse der Suche mit dem gleichen Suchbegriff zeigen deutlich, dass Biblio-GWT vorne liegt.

	Biblio-GWT						Biblio-ASP.net-Ajax					
Anzahl Benutzer	1	10	50	100	200	500	1	10	50	100	200	500
Antwortzeit mim (ms)	5	3	3	3	3	13	294	10	318	9	11	56
Antwortzeit max (ms)	5	8	7	211	3623	17339	294	69	2058	1326	3641	5654
Antwortzeit avg (ms)	5	4	4	24	464	6880	294	29	1564	629	941	1241
Fehler (Prozent)	0	0	0	0	0	0	0	0	0	0	0	0

Abbildung 7.5: Antwortzeiten der Startseite

	Biblio-GWT						Biblio-ASP.net-Ajax					
Anzahl Benutzer	1	10	50	100	200	500	1	10	50	100	200	500
Antwortzeit mim (ms)	1441	2760	12609	21	12	965	106	27	538	23	52	523
Antwortzeit max (ms)	1441	3325	18818	49382	186375	166351	106	100	953	1407	4139	7970
Antwortzeit avg (ms)	1441	2986	16476	45164	91651	33913	106	40	721	451	1213	2336
Fehler (Prozent)	0	0	0	0	0	0	0	0	0	0	0	0

Abbildung 7.6: Antwortzeiten der Produktanzeigen-Seite

Bei der Analyse von Fehler-Quote der beiden Anwendungen zeigt, dass mit 500 Benutzer 72 Prozent von Fehler beim Ausleihen mit Biblio-GWT gefunden wurden. Ein möglicher Grund ist die Belastung vom Tomcat-Server. So könnten nicht alle Anforderungen des Clients den Server erreichen. Um diese Aussage zu prüfen, wurde die Verfügbarkeit, die Speicherauslastung von Tomcat-Server im diesem Zeitpunkt

	Biblio-GWT						Biblio-ASP.net-Ajax					
Anzahl Benutzer	1	10	50	100	200	500	1	10	50	100	200	500
Antwortzeit mim (ms)	807	53	378	21	17	17	1128	112	286	184	176	1198
Antwortzeit max (ms)	807	1277	3263	6422	29399	27538	1128	1410	3562	6554	6427	10397
Antwortzeit avg (ms)	807	524	1542	2772	5766	2490	1128	304	2984	2523	2563	4307
Fehler (Prozent)	0	0	0	2	28	72	0	0	0	0	0	0

Abbildung 7.7: Antwortzeiten der Ausleihen-Seite

	Biblio-GWT						Biblio-ASP.net-Ajax					
Anzahl Benutzer	1	10	50	100	200	500	1	10	50	100	200	500
Antwortzeit mim (ms)	38	18	17	15	13	10	31	23	260	84	779	2385
Antwortzeit max (ms)	38	91	400	552	889	2131	31	149	864	7825	10810	18480
Antwortzeit avg (ms)	38	40	80	71	163	1009	31	63	489	1976	3889	6741
Fehler (Prozent)	0	0	0	0	0	0	0	0	0	0	0	0

Abbildung 7.8: Antwortzeiten der Suchen-Seite

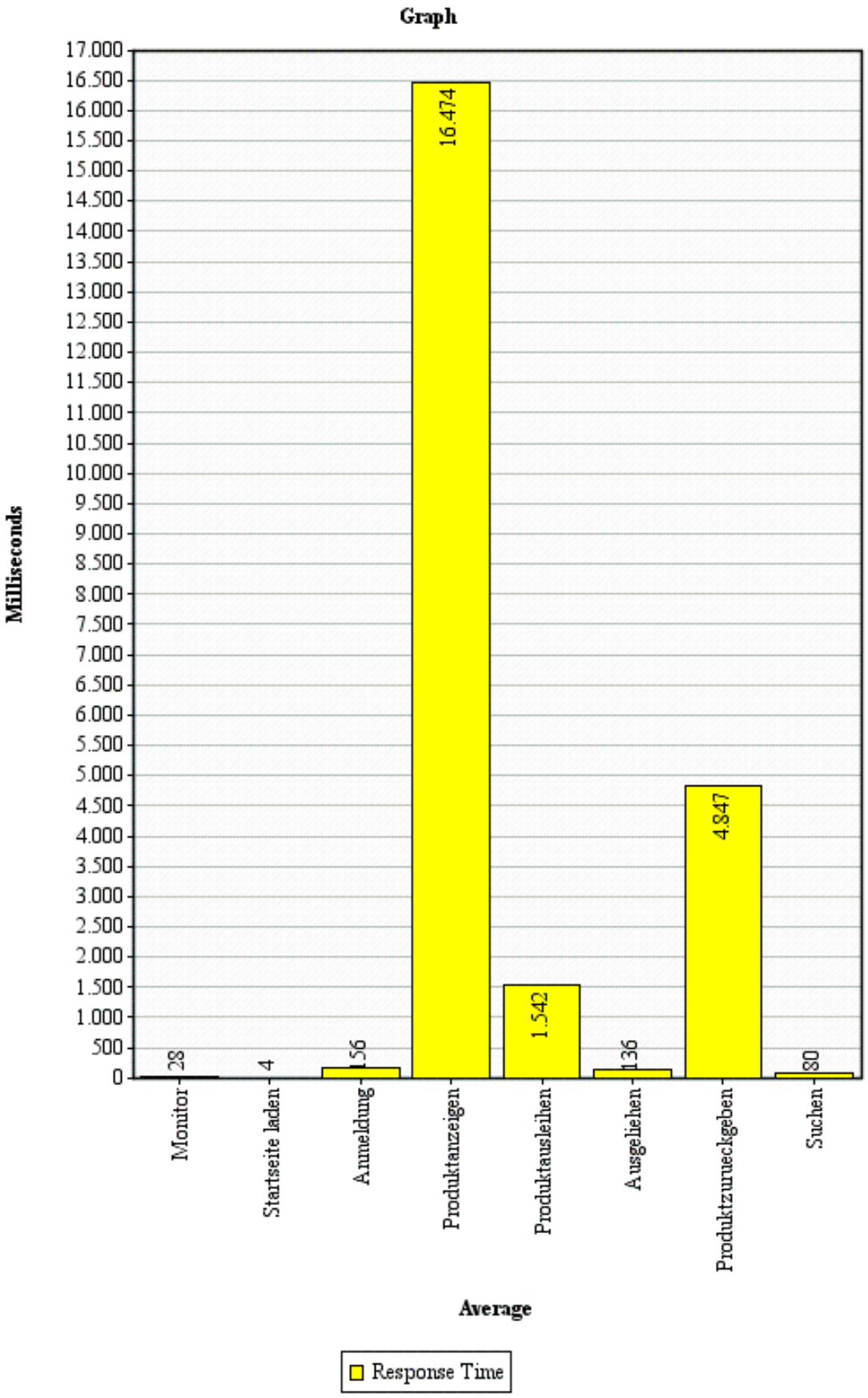

Abbildung 7.9: Durschnitt Ladezeit von Biblio-GWT mit 50 simulierten Benutzern

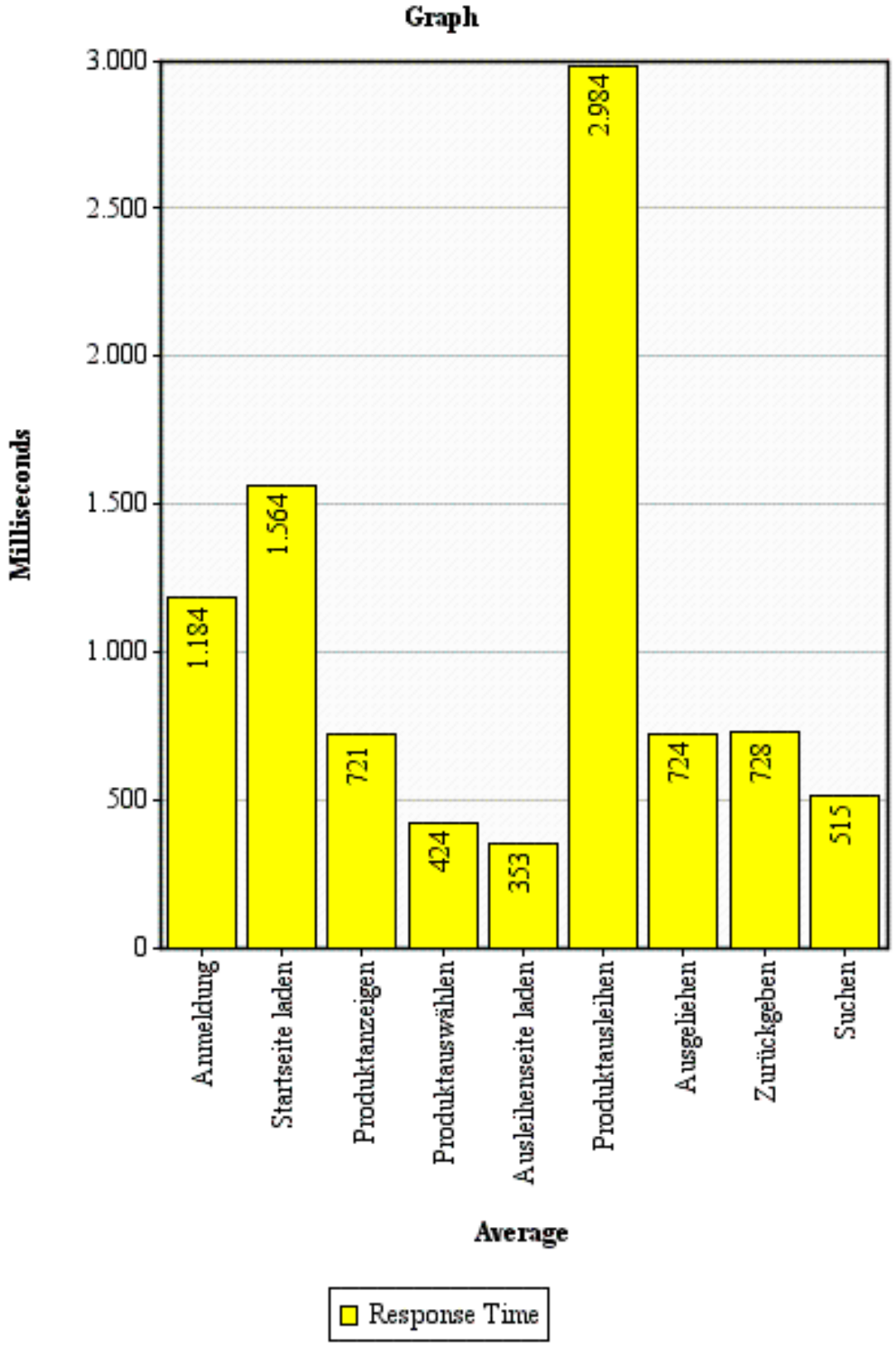

Abbildung 7.10: Durschnitt Ladezeit von Biblio-ASP.NET-AJAX mit 50 simulierten Benutzern

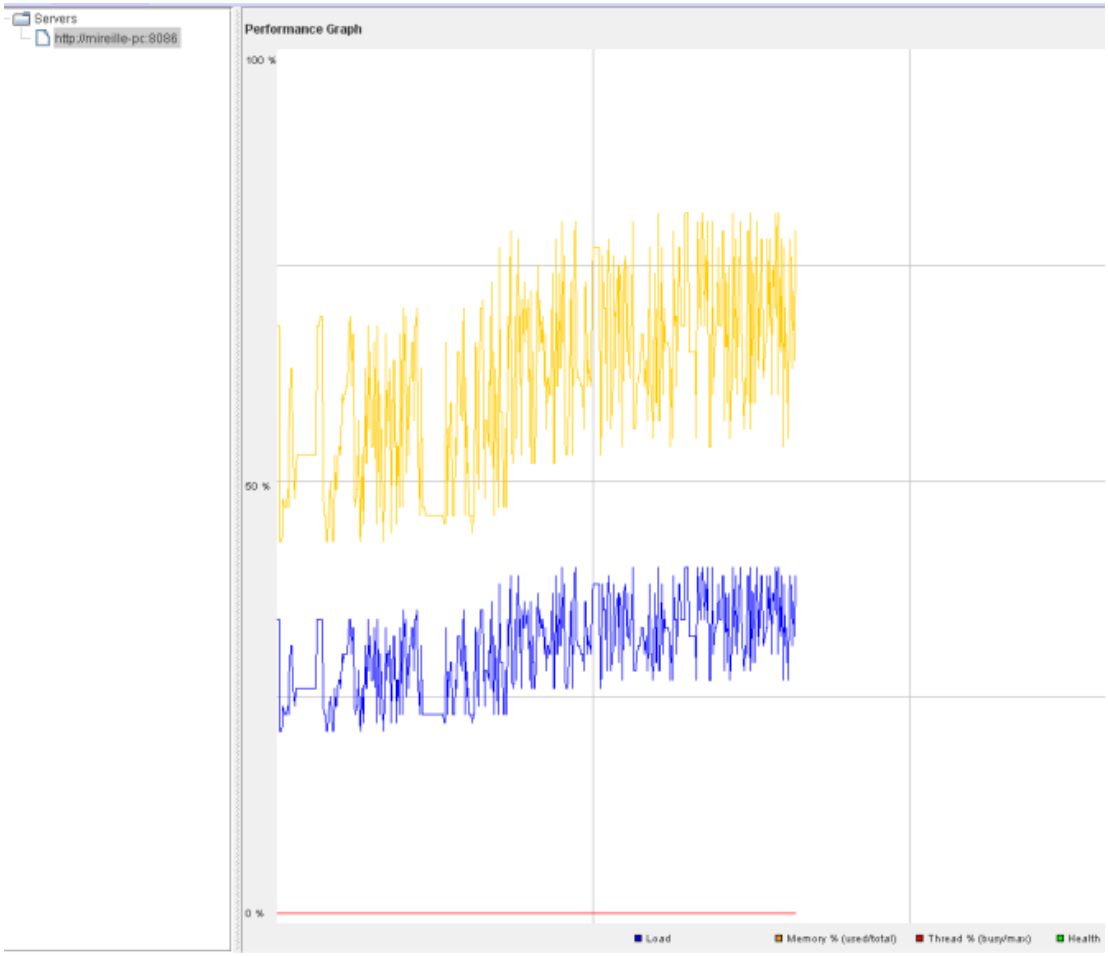

Abbildung 7.11: Monitoring der Tomcat-Performance bei 500 Benutzern

überwacht. Die Abbildung 7.11 zeigt die Ergebnisse. Diese zeigen deutlich, dass die Thread-Pools nicht ausgelastet (0 Prozent) sind, aber Tomcat verbraucht in diesen Zeitpunkt mehr als 75 Prozent der CPU-Leistung. Weniger als 50 Prozent der Client-Requests wurden verarbeitet. Diesen Test konnte nicht für IIS Express durchgeführt werden , weil die aktuelle Version von JMeter nur Tomcat überwacht (vgl. Apache, 2011b).

7.2.5 Zusammenfassung der Testergebnisse

Beim Testen auf einen lokalen Rechner betrug die durchschnittliche Reaktionszeit aller Seiten bei Biblio-GWT 505 ms. Bei Biblio-ASP.NET-AJAX lag diese bei 3490 ms. Daraus lässt sich schließen, dass Biblio-GWT für lokaler Test besser als Biblio-ASP.NET-AJAX ist. Beim entfernten Test wurde vier Seiten untersucht. Bei der Analyse der durchschnittliche Antwortzeit auf allen Seiten hat Biblio-GWT eine bessere Performance gezeigt als Biblio-ASP.NET-AJAX.

7.3 Fazit

In diesem Kapitel wurde beschrieben, wie der Entwicklungsaufwand der beiden Anwendungen geschätzt wird. Die Ergebnisse wurden vorgestellt und verglichen. Die Performancemessung wurde mit zwei Werkzeugen und zwei Testmethoden durchgeführt. Für den Lokalen Test wurde NeoLoad verwendet. Für den entfernten Test wurde Apache JMeter benutzt. Beim entfernten Test wurde zuerst eine Onlinestellung der beiden Anwendungen unter zwei verschiedenen Webserver (IIS 7.5 Express und Apache Tomcat 7) gemacht. Weiterhin wurde der Testaufbau mit JMeter vorgestellt. Um den Testaufbau zu verdeutlichen, wurde das RMI-Prinzip präsentiert. Anschließend wurden die Ergebnisse in tabellarischer und grafischer Form dargestellt und ausgewertet.

8 Zusammenfassung

In dieser Arbeit wurden zwei Frameworks anhand von festgelegten Kriterien analysiert und ausgewertet. Da die beiden Frameworks auf AJAX basieren, wurde zunächst die Entstehungsgeschichte sowie die Motivation für den Einsatz von AJAX beschrieben. Um die beide Frameworks mit den gleichen Vorraussetzungen zu bewerten, wurde ein Ausleihsystem entwickelt, in dem Bücher ausgeliehen werden können. Dafür wurden die funktionalen und nicht-funktionalen Anforderungen sowie Use-Cases entwickelt. Die Architektur wurde nicht vernachlässigt. Es wurde die fachliche und technische Architektur der Anwendungen dargestellt. Weiterhin wurde das System anhand von Klassendiagrammen, Zustandsdiagrammen und Flussdiagrammen verständlich gemacht. Nach der Implementierung der Beiden Anwendungen anhand der beiden Frameworks wurden Metriken, die für den Vergleich relevant sind, anhand von 2 Modellen abgeleitet: das FCM- und GQM-Modell. Bei der Entwicklung von Metriken wurden festgestellt, dass es viele Metriken gibt, mit denen ein Softwareprodukt bewertet werden kann. Für Diese Arbeit wurden der Entwicklungsaufwand und die Performance der beiden Anwendungen betrachtet. Um die Performance der Beiden Anwendungen zu analysieren, wurden sowohl ein lokaler als auch ein entfernter Test mit zwei Computern durchgeführt. Beim lokalen Test wurde die Software Neoload verwendet und beim entfernten Test Apache JMeter. Für den Entfernten Test musste zuerst eine Onlinestellung der beide Anwendungen gemacht werden. Dafür wurde Apache Tomocat 7 und IIS 7.5 Express verwendet. Um einen entfernten Test mit JMeter verstehen zu können, wurde das RMI-Prinzip vorgestellt, weil JMeter nach diesem Prinzip arbeitet. Abschließend wurden die Ergebnisse der beiden Anwendungen tabellarisch und Grafisch dargestellt und ausgewertet. Es kam heraus, dass GWT ein effizientes Framework ist, welches aber viel Entwicklungsaufwand erfordert. ASP.NET AJAX hat zwar schwache Ergebnisse, aber es ist für eine schnelle und einfache Erstellung einer AJAX-Anwendungen die bessere Wahl.

Abkürzungsverzeichnis

ARPA Advanced Research Project Agency

ASP Active Server Pages

CPU Central processing unit

CSS Cascading Style Sheets

DHTML Dynamic HTML

DOM Document Object Model

FCM Factor Criteria Metric

GQM Goal Question Metric

GWT Google Web Toolkit

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

IIS Internet Information Services

JAR Java Archive

JDBC Java Database Connectivity

JVM Java Virtual Machine

PDC Professional Developer Conference

RMI Remote Method Invocation

XHR XMLHTTPRequest

XML Extensible Markup Language

Literaturverzeichnis

[Alexander Brosch 2009] Alexander Brosch, Martin M.: *Google Web Toolkit*. Universität Salzburg - Department of Computer Sciences, Seminararbeit aus Informatik. 2009. – URL http://www.softwareresearch.net/fileadmin/src/docs/teaching/WS08/SaI/Brosch_Mitterbauer_paper.pdf

[André Möller 1998] André Möller, Magnus W.: *Architektur des RMI Systems*. 1998. – URL http://www.fh-wedel.de/~si/projekte/ss98/Ausarbeitung/DistributedProgramming/distr_kap03b.html

[Apache 2011a] Apache, Jmeter: *Introduction*. 2011a. – URL http://jakarta.apache.org/jmeter/usermanual/intro.html. – Zugriffsdatum: 29.08.2011

[Apache 2011b] Apache, Jmeter: *Remote Testing*. 2011b. – URL http://jakarta.apache.org/jmeter/usermanual/remote-test.html. – Zugriffsdatum: 29.08.2011

[Balzert 2008] Balzert, Helmut: *Lehrbuch der Softwaretechnik: Softwaremanagement*. Spektrum Akademischer Verlag Heidelberg 2008, 2008. – ISBN 978-3-8274-1161-7

[Balzert 2009] Balzert, Helmut: *Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering*. Spektrum Akademischer Verlag Heidelberg 2009, 2009. – ISBN 978-3-8274-1705-3

[Barthel u. a. 2005] Barthel, Stefan ; Eid-Sabbagh, Rami ; Müller, Stephan ; Tinnefeld, Christian: *AJAX - Konzept und Anwendung*. Universität potsdam - Hasso-Plattner-Institut für Softwaresystemtechnik,Ausarbeitung. 2005. – URL http://www.hpi.uni-potsdam.de/fileadmin/hpi/FG_ITS/lecturenotes/webprogrammierung/AjaxAusarbeitung.pdf

[CodePlex-Foundation] CodePlex-Foundation: *CodePlex*. – URL http://www.codeplex.com/. – Zugriffsdatum: 08.09.2011

[Fischer und Krause 2010] Fischer, Matthias ; Krause, Jörg: *ASP.NET 4.0: Konzepte und Techniken zur Programmierung von Webseites*. Carl Hanser Verlag Münschen, 2010. – ISBN 978-3-446-42238-4

[Google-Web-Toolkit] Google-Web-Toolkit: *Making Remote Procedure Calls*. – URL http://code.google.com/intl/de-DE/webtoolkit/doc/latest/tutorial/RPC.html. – Zugriffsdatum: 09.09.2011

[Koschke 2009] Koschke, Prof. Dr. R.: *Softwaretechnik*. 2009. – URL http://www.informatik.uni-bremen.de/st/lehre/swt09/metriken.pdf

[MSDN] MSDN: *Übersicht über Microsoft AJAX*. – URL http://msdn.microsoft.com/de-us/library/bb398874.aspx. – Zugriffsdatum: 08.09.2011

[NEOTYS 2011] NEOTYS: *NeoLoad*. 2011. – URL http://www.neotys.de/. – Zugriffsdatum: 19.08.2011

[Schwichtenberg 2007] Schwichtenberg, Dr. H.: *ASP.NET-AJAX (Teil 1) - AJAX für Einsteiger: Einführung in die Microsoft AJAX Library und die ASP.NET AJAX Extensions*. 2007. – URL http://www.microsoft.com/germany/msdn/webcasts/library.aspx?id=1032322560

[Steyer 2007] Steyer, Ralph: *Google Web Toolkit*. Entwickler.press, 2007. – 22–26 S. – ISBN 978-3-939084-21-1

[SUN] SUN: *Java Servlet Technology*. – URL http://www.oracle.com/technetwork/java/javaee/servlet/index.html. – Zugriffsdatum: 08.09.2011

[Wenz 2006] Wenz, Christian: *JavaScript und AJAX*. Galileo Computing, 2006. – URL http://openbook.galileocomputing.de/javascript_ajax/18_ajax_002.htm. – ISBN 3-8984-2859-1

[Wikipedia 2011a] Wikipedia: *Ajax (Programmierung)*. 2011. – URL http://de.wikipedia.org/wiki/Ajax_%28Programmierung%29. – Zugriffsdatum: 09.03.2011

[Wikipedia 2011b] Wikipedia: *Ajax (Programmierung)*. 2011. – URL http://de.wikipedia.org/wiki/Ajax_%28Programmierung%29. – Zugriffsdatum: 09.09.2011

[Wikipedia 2011c] Wikipedia: *Function-Point-Verfahren*. 2011. – URL http://de.wikipedia.org/wiki/Function-Point-Verfahren. – Zugriffsdatum: 14.08.2011

[Wirsing 2006] Wirsing, Martin: *Projektmanagement:Schätzverfahren*. 2006. – URL http://www.pst.ifi.lmu.de/lehre/WS0607/pm/vorlesung/PM-05-Schaetzung.pdf

[With 2010] With, Nicolas: *Testbett für eine Kriteriengestützte Analyse von Webframeworks*, , Bachelorarbeit, 2010. – URL http://opus.haw-hamburg.de/volltexte/2010/1074/pdf/Bachelorarbeit.pdf

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 12.September 2011 Mireille Manto