Bachelorarbeit

Sergej Becker

Ressourcenoptimierung von Webapplikationen am Beispiel einer Rich Internet Applikation

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Sergej Becker

Ressourcenoptimierung von Webapplikationen am Beispiel einer Rich Internet Applikation

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Olaf Zukunft
Zweitgutachter : Prof. Dr. Michael Neitzke

Abgegeben am 25. Juni 2010

Sergej Becker

Thema der Bachelorarbeit

Ressourcenoptimierung von Webapplikationen am Beispiel einer Rich Internet Applikation

Stichworte

Webapplikationen, Rich Internet Applikationen, Web Services und Web-APIs, Webapplikationsressourcen, Domänendienstleister, Domänenkosten

Kurzzusammenfassung

In dieser Arbeit wird Ressourcenoptimierung von Webapplikationen am Beispiel einer Rich Internet Applikation beschrieben. Ausgehend von allgemeinen Betrachtungen zu Ressourcen von Webapplikationen werden Möglichkeiten für Optimierung von Ressourcen für Webanwendungen untersucht. Anhand ausgewählter Möglichkeiten und Technologien werden Anforderungen an die Optimierung der beispielhaften zu implementierenden Webapplikation erarbeitet und in einer Systemarchitektur und einem Entwurf umgesetzt. Zum Beleg der Realisierbarkeit wird die prototypische Implementierung der Webapplikation am Beispiel einer Rich Internet Applikation beschrieben. Abschließend werden kritische Punkte, Empfehlungen sowie zukünftige Anwendungs- und Weiterentwicklungsmöglichkeiten der entwickelten Webapplikation zusammengefasst und diskutiert.

Sergej Becker

Title of the paper

Resource optimization of web applications exemplified by Rich Internet Application

Keywords

Web applications, Rich Internet Applications, Web Services and Web-APIs, Resources of web applications, Domains, domain costs

Abstract

In this thesis, the author describes resource optimization of web applications exemplified by Rich Internet Application. Starting with some general views on which resources have a web application, existing possibilities for optimization of resources for web applications are analyzed. A set of possibilities and technologies is then chosen as basis for developing requirements, which are subsequently implemented in a system architecture and a design. A prototype of the optimized web application exemplified by Rich Internet Application is described, proving the feasibility of the design. Finally, the author summarizes and discusses critical issues and recommendations, as well as future applications and enhancements for the developed technology.

Inhaltsverzeichnis

Abbildungsverzeichnis

Danksagung

Ich danke Prof. Olaf Zukunft von der Hochschule für Angewandte Wissenschaften Hamburg für die gute Betreuung und das schnelle und ausführliche Feedback während der Erstellung dieser Arbeit.

Weiterhin bedanke ich mich herzlich bei Natalia Becker, Andreas Winschu und Jakob Becker für hilfreiche Anmerkungen, hilfreiches Feedback und ausführliche Korrekturarbeiten.

Sergej Becker, Juni 2010

1. Einleitung

1.1. Problemstellung

Im Rahmen der Begriffe Marketing, Verfügbarkeit und Bequemlichkeit spielt die Präsenz einer Applikation im Web eine große Rolle, sowohl kommerziell als auch privat. Webapplikationen eignen sich gut dafür, verfügbare[1] und schnellzugängliche Systeme[1] am Markt für ein bestimmtes Geschäftsproblem zu haben und haben in den letzten Jahren deutlich zunehmende Verbreitung gefunden. Anwendungsbeispiele (2.1.1) umfassen Optimierung von Geschäftsprozessen und Marketing eines Unternehmens, „Nice-To-Have"-Webapplikationen, Webspiele, etc. und das Webangebot von Webapplikationen wächst mit sehr großer Geschwindigkeit. Die Tendenz des Wachstums geht auch in die Richtungen Einfachheit, Vielfältigkeit und Schönheit, besonders bei privaten Webapplikationen. Diese Tendenz erhöht Ressourcenanforderungen und ist leider proportional zu den Entwicklungs- und Betriebskosten der Applikation. Dabei spielen insbesondere Domänendienstleistung als technische Verfügbarkeit der Webapplikation im Web und Ressourcenoptimierung eine sehr wichtige Rolle.

Ressourcenaufwendige, schöne Webapplikationen verursachen einen großen und intensiven Client-Webserver-Datenverkehr, eine große Speicheranforderung und sind die Ursache der hohen Domänenkosten (z.B. Art der Domäne, Anzahl der Subdomänen, Anzahl der Postfächer, Speichergröße, Traffic, Anzahl der FTP-Zugänge und etc.) und des großen Entwicklungsaufwandes, die man allgemein als Kosten bezeichnen kann. Diese Problematik der verursachten Kosten wird meistens unterschätzt, denn die o.g. Aspekte wie Datenverkehr und Speicheranforderung ist nicht immer sofort sichtbar und schätzbar. Besonders private Webentwickler eines jungen Projektteams mit einem engen Entwicklungs- und Betriebsbudget, die bei der Entwicklung der Webapplikation nur eine Vision im Auge haben - so schnell wie möglich ins Web - müssen über diese Problematik Gedanken machen, wenn der Start ins Web erfolgreich war. Nur der erfolgreiche Start zieht mit sich automatisch die Besucherzahl der Webapplikation nach oben und sorgt allein für die hohe Präsenzqualität und hohen Kosten der Webapplikation. Die überschrittenen Leistungen des Domänenanbieters (z.B. o.g. Traffic bzw. Datenverkehr im Sinne monatliches Übertragungsvolumen und Speicheranforderung) oder zur Verfügung stehenden Betriebsbudgets eines Unternehmens, das selbstständig den Betrieb des Webservers und Vereinbarung mit dem ISP (Internet Service Provider) übernimmt, können dazu führen, dass die zur Verfügung stehende im Web Webapplikation offline ist bzw. ihre Betriebskosten sehr teuer werden können. Denn für die ressourcenaufwendigen erfolgreichen Webapplikationen im Sinne hoher Besucherzahl ist ein großer Client-Webserver-Datenverkehr unvermeidbar, weil der Traffic dabei größer und intensiver

[1] Der Begriff verfügbar wird hier im Sinne „online" bzw. „immer" und der Begriff schnellzugängliche Systeme im Sinne „keine Installation nötig" gebraucht.

bezüglich der Häufigkeit der Benutzerbesuche wird. Deswegen ist dabei Ressourcenoptimierung von Webapplikationen einer der zentralen Aspekte der Webanwendungsentwicklung. Unabhängig von sich ständig weiterentwickelnden Technologien und wechselnden Paradigmen der Websoftwareentwicklung ist Ressourcenoptimierung Mittel der Wahl, um Performanz, Reduzierung der Implementierungs- und Domänenkosten von ressourcenaufwendigen und schönen Webapplikationen zu realisieren.

Webapplikationen für kleine mobile Geräte wie Mobiltelefone, PDAs (Personal Digital Assistants) und die so genannten Smartphones (Kombinationen aus Mobiltelefon und PDA), haben seit letzten Jahren eine explosive Verbreitung erfahren.[2] Ressourcenoptimierung von Webapplikationen spielt auch hier eine wichtige Rolle bezüglich der zur Verfügung stehenden Ressourcen der kleinen mobilen Geräte im Vergleich zu PCs oder Laptops und der Tatsache, dass sie nicht ununterbrochen online sind. Die Konnektivität von Mobiltelefonen ist durch lange, gewünschte (z.B. nachts) und kurze, unerwünschte (z.B. in einem Eisenbahntunnel) Offline-Zeiten gekennzeichnet. Der Ausdruck „kleine mobile Geräte“ wird in dieser Arbeit verwendet, um Ressourcenoptimierung von Webapplikationen für diese Geräte zu betonen.

1.2. Zielsetzung

Im Rahmen dieser Arbeit soll Ressourcenoptimierung von Webapplikationen am Beispiel einer Rich Internet Applikation erarbeitet werden, um Performanz, Reduzierung der Implementierungs- und Domänenkosten von ressourcenaufwendigen Webapplikationen zu realisieren.

Ausgehend von Betrachtungen der Ressourcenoptimierung von Webapplikationen ist das Ziel, notwendige Ressourcen von Webapplikationen vorzustellen, Performanz, Implementierungsaufwand und Domänenkosten bzw. Betriebskosten kritisch zu betrachten und mit mathematischen Berechnungen (2.1.6) darzustellen. Zudem soll am Beispiel der Implementierung einer Rich Internet Applikation zur Verdeutlichung der o.g. Betrachtungen die Machbarkeit des angestrebten Ziels demonstriert werden (mit Hilfe einer analytischen Komponente). Dabei ist nicht die möglichst vollständige Implementierung das Ziel, sondern Analyse und Entwurf der benötigten Technologien, möglichen neuen Komponenten und deren Kernfunktionalitäten.

Am Beispiel folgender ressourcenaufwendigen einfachen Webapplikation (Rich Internet Applikation), welche im Laufe dieser Arbeit kontinuierlich entwickelt wird, soll das angestrebte Ziel erreicht werden:

Heutzutage ist es sehr hilfreich, besonders für weibliche Personen, die über eine große Menge von Kleidungsstücken in ihren Garderoben verfügen und großes Bedürfnis für die Veranschaulichung verschiedener Kombinationen dieser Kleidungen an sich haben, einen Kleidungsassistenten in der Rolle einer Webapplikation zu haben, der das anbietet. Da Kleidungsstücke, die aus der technischen Sicht in Bildformaten einen riesigen Datensatz und

---

[2] Siehe z.B. entsprechende Berichte auf http://www.cellular.co.za/analysts/07182001-emc_forecasts_subscribers_to_top.htm

somit Datenverkehr verursachen können, für die Simulation aufwendiger Ressourcen sehr gut passen, soll anhand oberer Beschreibung eine ressourcenaufwendige Webapplikation bezüglich der Problemstellung (1.1) und Zielsetzung (1.2) für das folgende Szenario implementiert werden.

Szenario: Es gibt ein System (Kleidungsassistent), wahrscheinlich mit mehreren Subsystemen, das für eine unbegrenzte Anzahl der Nutzer zugänglich ist. Jede Person, die dieses System nutzen möchte, muss sich für den Zugang autorisieren. Wenn der Authentifizierungsvorgang erfolgreich war, steht das System wahrscheinlich mit allen seinen Subsystemen für den Benutzer zur Verfügung. Der Benutzer kann seine Garderobe (Inventar) in Form der Darstellung all seiner Bilder sich anschauen und an seinem Körperprofil zur Laufzeit diese Bilder kombinieren lassen bzw. sich für die Entscheidung, wie ich mich heute am besten anziehen kann, anziehen. Der Assistent unterstützt den Benutzer mit einem Subsystem, das dem Benutzer die aktuelle Wetterinformation der nächsten zwei Tage im System anzeigt. Der Ort der Wetterinformation ist für den Benutzer interaktiv und kann nach der Benutzeranfrage für den gewünschten Ort angezeigt werden. Die Kombination der Kleidungsstücke an dem Profil ist auch interaktiv und wird zusätzlich mit einem weiteren Subsystem unterstützt, das für die Bildbearbeitung zuständig ist. Wenn Benutzer seine Sitzung im System beenden möchte, meldet er sich im System ab. Die Sitzung kann auch ohne Abmelde-Vorgang beendet bzw. unterbrochen werden.

Aufwendige Ressourcen der zu implementierenden RIA werden durch die riesigen Benutzer- und Bilderdatenbanken für einzelne Benutzer abgebildet. Diese Start-Webapplikation mit sehr speicheranspruchsvollen Anforderungen und typischen wiederkehrenden Komponenten bietet folgende Funktionen und Lösung eines Geschäftsproblems an:

- Lösung des Geschäftsproblems
    - Der Anwender hat Möglichkeit, sein Inventar der Sachen visuell an seinem Profil anschauen und kombinieren zu lassen, indem die Bilder mit Hilfe von Web Services und Web-APIs (Cloud-APIs) persistiert werden.
- Funktionen (Benutzerschnittstelle bzw. UI)
    - Login-Logout (nicht angemeldete Benutzer haben keinen Zugang zur Webapplikation und somit zu folgenden Funktionen)
    - Wetteranzeige (ein Subsystem, das für die aktuelle Anzeige der Wetterinformation zuständig ist)
    - Verwaltung des Inventars des angemeldeten Benutzers, der an seinem Profil die gewünschten Bilder der Sachen kombinieren und anschauen kann
    - Analytische (eine für die Benutzer unsichtbare Funktion, deren Informationen nur für den Entwickler bzw. Inhaber der Webapplikation oder Leute, die eine Zugangsautorisierung haben, zur Verfügung steht)
    - Bildbearbeitung (ein Subsystem, das für die Bildbearbeitung zuständig ist)

Alle o.g. ressourcenaufwendigen Funktionen der Webapplikation werden aus der technischen Sicht an die kostenlosen Web Services und Web-APIs delegiert, die durch jeweilige Webapplikationskomponenten abgebildet werden (Abb. 1.1). Die Implementierung der Webapplikation findet am Beispiel einer Flash- bzw. Flex-Applikation, die man aus der

technischen Sicht mit einem Überbegriff „Rich Internet Applikation“ bezeichnen kann. Die analytische Delegation dient zur Analyse der angestrebten Zielsetzung und hat für den Anwender keine Bedeutung.

Die Webapplikation wird im 2D-Modus implementiert. Es wird besonders auf die angestrebte Funktionalität der Applikation geachtet und nur versucht, ein schönes und benutzerfreundliches UI zu implementieren. Die entsprechenden Dienste (Web Services und Web-APIs) werden vom Benutzer manuell gewechselt, wenn einer der Web Services oder Web-APIs nicht erreichbar ist oder der Benutzer vorhat, einen anderen Dienst zu verwenden. Die entsprechenden Mitteilungen werden im „Fehlerfall“ von der Webapplikation an den Benutzer weitergeleitet.

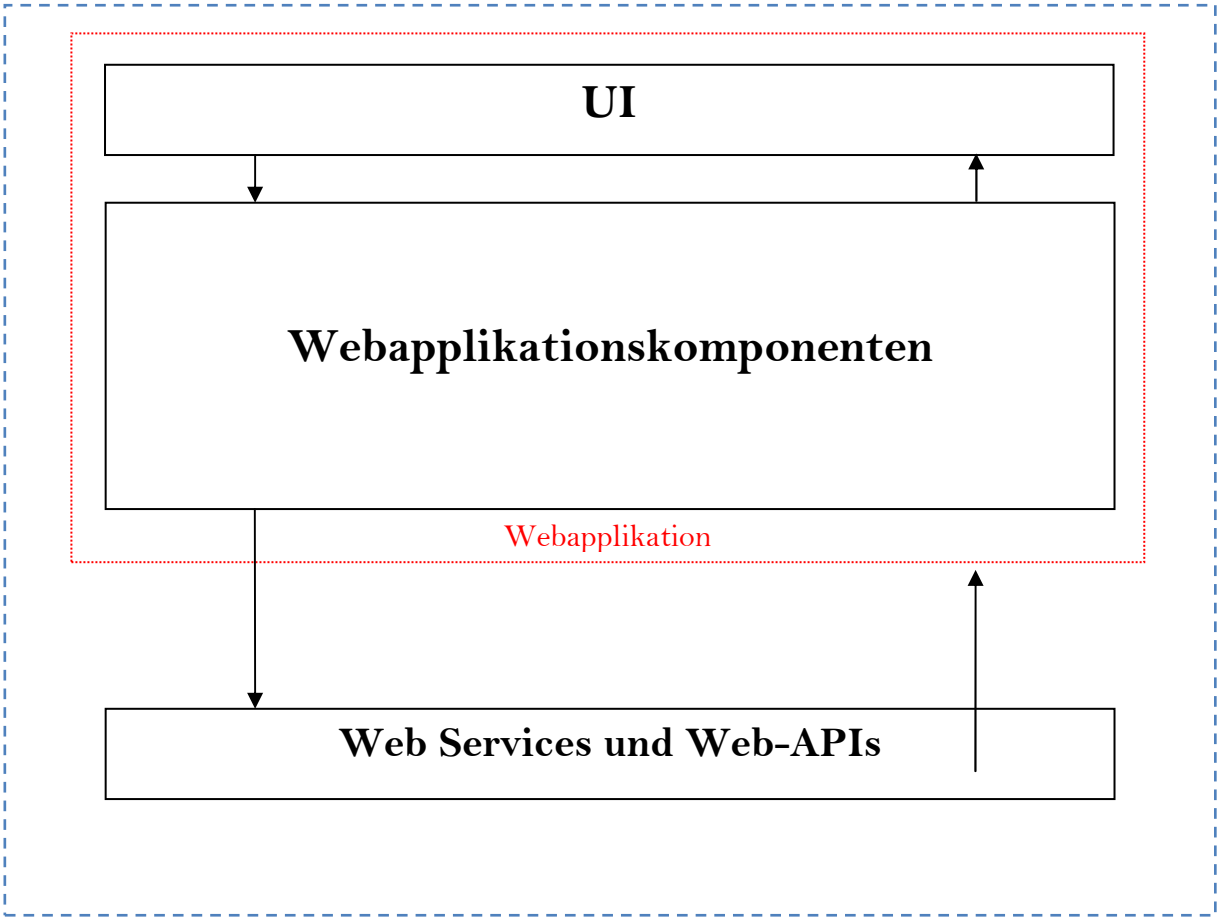

Abbildung 1.1.: Die zu implementierende Webapplikation aus der technischen Sicht (eigene Darstellung)

Diese Webapplikation soll schließlich im Rahmen einer prototypischen Implementierung die Realisierbarkeit des zu erstellenden Entwurfs demonstrieren.

1.3. Zielgruppen

Ausgehend von der Zielsetzung bietet diese Arbeit Informationen für Personen, die sich mit der ressourcenaufwendigen Websoftwareentwicklung unter Einbeziehung der Aspekte Performanz, Reduzierung der Implementierungs- und Domänenkosten und allgemein Ressourcenoptimierung befassen. Die technische Ausrichtung bedingt das Vorhandensein von Grundwissen über Webapplikationen und die zugrunde liegenden Spezifikationen und Technologien. Folgende Zielgruppen sollen angesprochen werden:

- Webentwickler und andere Mitglieder eines Projektteams, die die Konzepte und Zusammenhänge der für diese Arbeit maßgeblich wichtigen Bereiche kennen lernen wollen, finden in Kapitel 2 Informationen.

- Webentwickler, besonders private Webentwickler eines jungen Projektteams mit einem engen Entwicklungs- und Betriebsbudget, die eine Ressourcenoptimierung von Webapplikationen, ggf. am Beispiel einer Rich Internet Applikation, erreichen möchten, finden im Entwicklungsteil dieser Arbeit in den Kapiteln 3, 4 und 5 Informationen.

Da der Entwicklungsteil dieser Arbeit auf der dynamischen Programmiersprache Action Script (Shupe u.a. 2008), (Lott u.a. 2007) und einem Framework „Flex“ (Adobe LiveDocs 2009) für Rich Internet Applikationen aufbaut, ist es sinnvoll, sich zunächst mit diesen Technologien vertraut zu machen.

1.4. Inhaltlicher Aufbau der Arbeit

Kapitel 2, „Grundlagen“, legt erforderliche Grundlagen aus den Bereichen „Webapplikationen“, „Leistungen der Domänendienstleister“, „Rich Internet Applikationen“, „Webapplikation-Webserver-Kommunikation“, „Frameworks für RIAs“ sowie „Web Services und Web-APIs“. Es wird in den einzelnen Abschnitten auch auf die Vorbereitung des Erarbeitens der Ressourcenoptimierung von Webapplikationen eingegangen und die Arbeit gegen diese abgegrenzt. Weiterhin wird ein Markt von für Rich Internet Applikation geeigneten RIA-Frameworks vorgestellt. Abschließend werden weitere Ziele vor dem Hintergrund der Grundlagen definiert, um auf die folgenden Kapitel vorzubereiten.

Kapitel 3, „Analyse“, beschreibt die softwaretechnische Analyse (Anforderungen, Anwendungsfälle) der zu implementierenden Rich Internet Applikation für Demonstration der Ressourcenoptimierung von Webapplikationen.

In Kapitel 4, „Entwurf“, werden Architektur und Komponenten der Rich Internet Applikation genauer beschrieben.

Implementierung der Komponenten wird in Kapitel 5 vorgestellt, bevor Fazit und Ausblick in Kapitel 6 die Arbeit abschließen.

2. Grundlagen

In diesem Kapitel wird der Einsatz der in dieser Arbeit grundlegenden Konzepte und Technologien – Webapplikationen, Leistungen der Domänendienstleister, Rich Internet Applikationen, Web Services und Web-APIs sowie Frameworks für RIAs – erläutert. Die in 1.2 kurz beschriebene zu implementierende Rich Internet Applikation wird dabei im Abschnitt über Ressourcen von Webapplikationen, Rich Internet Applikationen und Web Services und Web-APIs (2.1.7, 2.2, 2.3) aufgegriffen. Weiterhin werden für das Verständnis und die Realisierung der Zielsetzung notwendige Grundlagen erarbeitet. Es werden auch Literaturhinweise sowie Hinweise auf existierende Arbeiten in den berührten Themenfeldern gegeben.

2.1. Webanwendungen

2.1.1. Überblick

Wie in 1.1 angedeutet, haben Webanwendungen in den letzten Jahren deutlich zunehmende Verbreitung gefunden und ihre Anzahl im Web wird immer größer und größer. Jeder Mensch, zumindest in westlichen Industrienationen, hat mittlerweile die Möglichkeit, eine Vielzahl interessanter Anwendungen im Web zu nutzen, die von überall aus leicht zugänglich sind und immer anspruchsvollere Geschäftsprobleme bzw. Workflows übernehmen.

Wie in 1.1 auch bereits herausgestellt wurde, hat sich Webpräsenz nicht nur bei der Klasse von Geschäftsanwendungen als Tendenz für alle Systeme bewährt, sondern auch bei der „Nice-To-Have“ Klasse. Diese Klasse der Webanwendungen ist im Rahmen dieser Arbeit besonders interessant, da sie aufwendige Ressourcen voraussetzt und somit ein guter Repräsentant für die beispielhafte Ressourcenoptimierung darstellt. Dazu ist es nicht nur notwendig, ressourcenaufwendige Webanwendungen der „Nice-To-Have“ Klasse auf Endgeräten zu betreiben, sondern ressourcenaufwendige Webanwendungen müssen auch auf den kleinen mobilen Geräten betrieben werden können, die über mangelnde Hardware-Ressourcen verfügen. Die Ressourcenoptimierung ist nicht nur für die mangelnden Hardwareressourcen wichtig, sondern auch für Kosten der Webanwendung, die bei der Implementierung und besonders beim Betrieb im Web (z.B. Traffic- und Speicherkosten) entstehen.

Im Rahmen dieser Arbeit wird eine Webanwendung der Klasse „Nice-To-Have“ implementiert. Wie in 2.1.2 herausgestellt wird, eignet sich gut dafür die clientseitige Architektur bzw. eine Anwendungsklasse „Rich Internet Applikation“, die eine Untermenge der Gesamtmenge „Webanwendungen“ darstellt.

In Abschnitt 2.2 und in Kapitel 2 wird noch einmal näher auf die Wahl der Anwendungsklasse „Rich Internet Applikation“ und auf Frameworks und Entwurfsmuster eingegangen, am deren Beispiel eine in 1.2 beschriebene Anwendung bezüglich der vorgenommenen Res-

sourcenoptimierung (1.2) implementiert wird. Im Rahmen des Entwurfs (Kap. 4) wird beschrieben, wie die Webanwendung bzw. Komponenten der zu implementierenden Webanwendung bezüglich der Ressourcenoptimierung unter Benutzung bestimmter Entwurfsmuster realisiert werden können, so dass die in der Analyse (Kap. 3) erarbeiteten Anforderungen erfüllt werden.

2.1.2. Beispiel-Szenarien

Webanwendungen haben nicht nur den Vorteil, dass sie jederzeit und überall nutzbar sein können, sondern sie sind außerdem, wenn man vom Webzugang abstrahiert, plattformunabhängig. Endgeräte sind höchst heterogen: Die Geräte verwenden unterschiedliche Betriebssysteme und verfügen über unterschiedliche Netzwerkschnittstellen und unterschiedliche Softwareausstattungen. Es gilt nun, diese Webanwendungen den Clients zur Verfügung zu stellen, die nicht nur unter bestimmten Software- und Hardwarevoraussetzungen auf den Endgeräten (z.B. Computer, PDAs, Mobiltelefone, Laptops und andere kleine mobile Geräte) ausgeführt werden können, sondern auf allen Endgeräten, die nur über einen Browser aus der fachlichen Sicht verfügen sollen. Dafür gibt es folgende, in Abb. 2.1 Arten bzw. visualisierte Möglichkeiten unter Verwendung von Webanwendungen[3]:

1. serverseitig: der Server bzw. Webcontainer stellt für die Webanwendung die Laufzeitumgebung dar und antwortet auf die Clientanfragen. Jede logische Anfrage an die Webanwendung ist somit eine Anfrage an den Server.

2. clientseitig: Die Webanwendung läuft vollständig auf dem Client und stellt somit keine Anfragen an den Server.

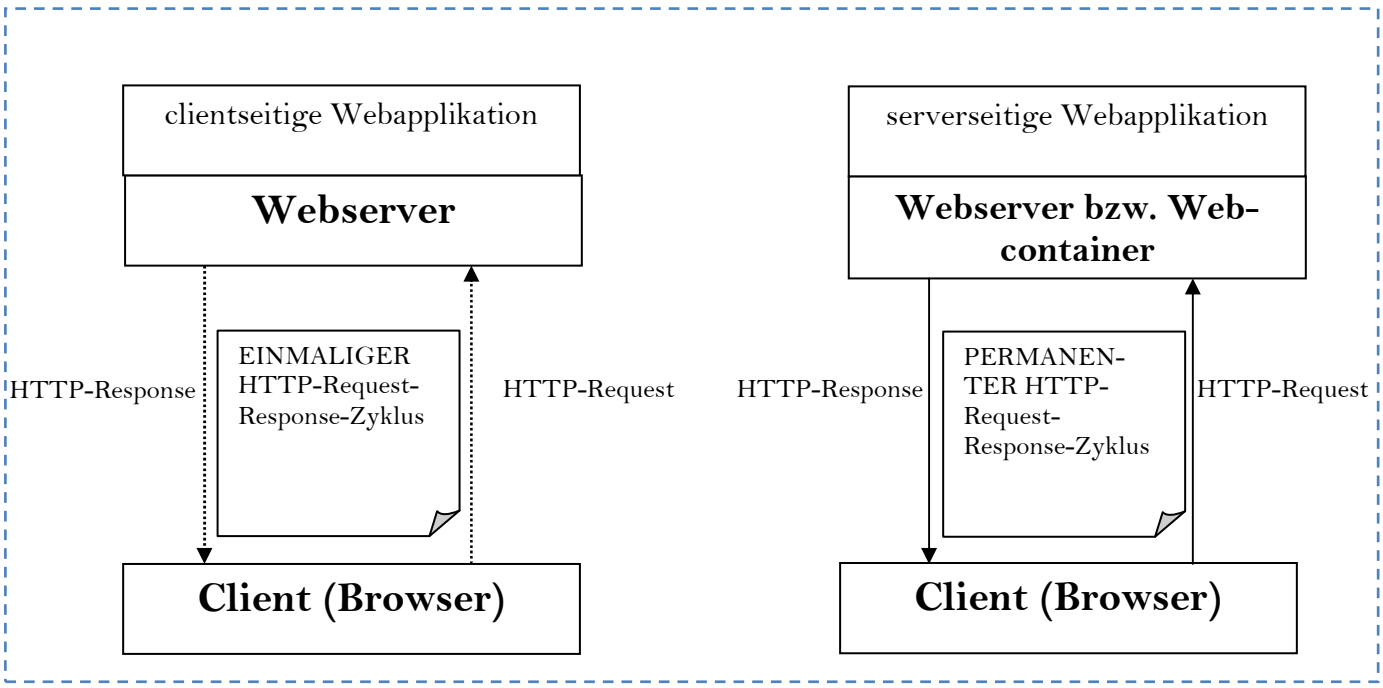

Abbildung 2.1.: Serverseitige vs. clientseitige Webanwendungen (eigene Darstellung)

Möglichkeit 1 ist dabei vor allem für Webanwendungen interessant, die in kurzen Abständen die verwendeten Daten aktualisieren müssen und sie in kurzen Abständen von anderen Stellen aus abfragen. Dies könnten z.B. variable Daten (Temperatur, Aktienkurs, Währungskurs, etc.) sein.

Möglichkeit 2 ist interessant für Webanwendungen, die ihre Daten gezielt anfragen und sie nicht sehr oft benötigen, wie z.B. Multimedia-Daten (Töne, Bilder, kurze Videos). Weiterhin können so auch Teile der Webanwendung auf dem Endgerät angestoßen werden, wie z.B. eine Benutzeraufforderung, bestimmte Daten einzugeben und diese Daten dann an einen anderen Dienst (Web Service oder Web-API) weiterleitet.

Die in 1.2 zu implementierende Beispiel-Webanwendung fällt unter die Möglichkeit 2 „clientseitig". Es soll wie in Abb. 2.2 veranschaulicht für den Rest dieser Arbeit als Beispiel dienen. Die Abbildung stellt die zu implementierende Webapplikation im eingeloggten Zustand dar.

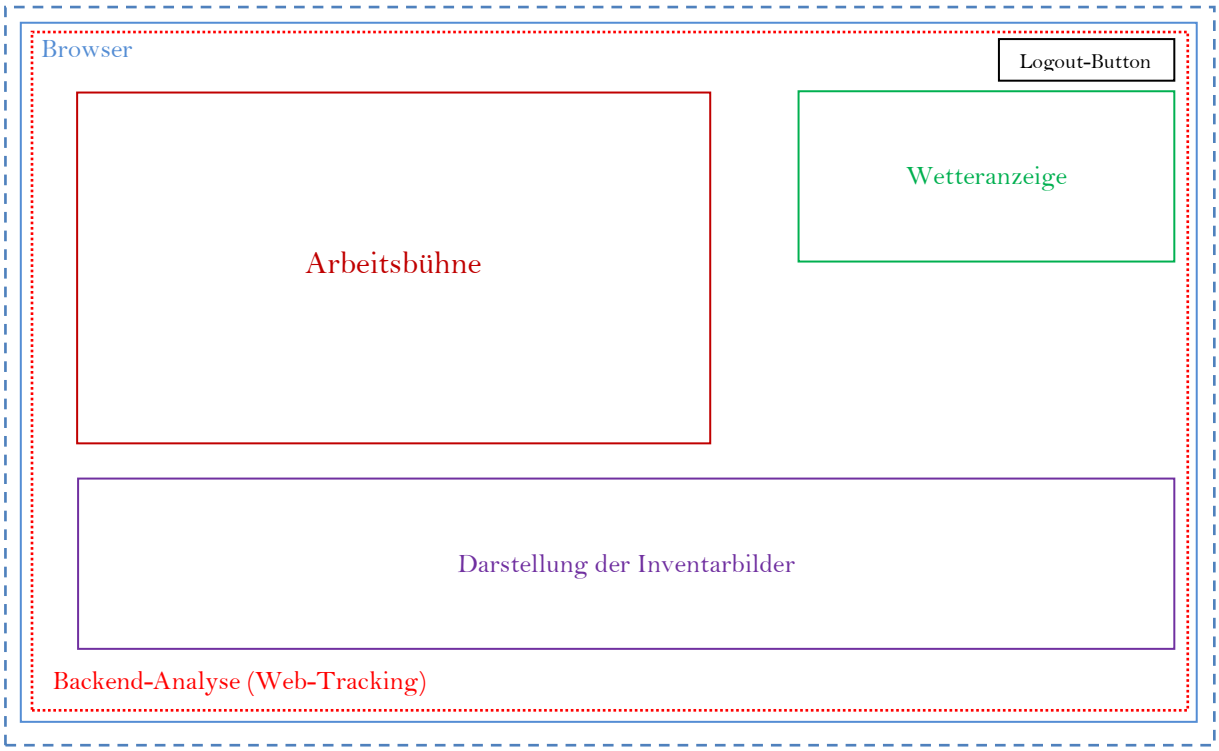

Abbildung 2.2.: Die zu implementierende clientseitige Webanwendung (eigene Darstellung)

2.1.3. Webapplikationen

Webanwendungen oder Webapplikationen, wie in 1.1 erwähnt, eignen sich gut dafür, verfügbare bzw. online und schnellzugängliche Systeme, die keine Installation nötig haben, am

[3] Der Begriff Webapplikation wird hier im Sinne der Beschreibung der Struktur und Eigenschaften vom Computer-Programm, das im Web verfügbar ist, gebraucht.
Unter http://de.wikipedia.org/wiki/Webanwendung ist eine mögliche Definition des Begriffs zu finden.

Markt für ein bestimmtes Geschäftsproblem zu haben und haben in den letzten Jahren deutlich zunehmende Verbreitung gefunden. Webanwendungen lassen sich von anderen Anwendungen gut abgrenzen, da die Interaktion mit dem Benutzer ausschließlich über einen Browser erfolgt.

Webapplikation ist nichts Anderes als eine Applikation, die im Web zugänglich ist[3]. Viel interessanter ist, was eigentlich eine Webapplikation von einer gewöhnlichen statischen Webseite (HTML-Seite), die normalerweise als Willkommensseite der meisten Domänennamen jeder Besucher sieht, unterscheidet. Letztendlich ist jede Antwort (Response) eines Webservers auf eine Anfrage (Request) des Clients jeder Webapplikation eine HTML-Seite, abgesehen von der Rich Internet Applikation, die clientseitig in ihrer eigenen Laufzeitumgebung (meistens als Plug-In im Browser integriert) laufen. Alle serverseitige Webapplikationen benötigen eine zugehörige Laufzeitumgebung, die im Endeffekt für das Rendering der Response in die HTML-Response zuständig ist. Wie auch zu mehreren Begriffen gibt es auch zum Begriff Webapplikation verschiedene, mehr oder weniger klare Definitionen. Für diese Arbeit ist die Definition nach (Kappel u.a. 2003 S.3), die auf Spezifikationen des World Wide Web Consortium (W3C 2010) beruht, ausreichend, in der Webapplikation folgendermaßen dargestellt wird:

- Eine Webanwendung ist ein Softwaresystem, das auf Spezifikationen des World Wide Web Consortium (W3C) beruht und webspezifische Ressourcen wie Inhalte und Dienste bereitstellt, die über eine Benutzerschnittstelle, den Web-Browser, verwendet werden.

2.1.4. Softwarearchitektur einer Webapplikation

In der Regel läuft eine Webapplikation auf einem oder mehreren, insbesondere im professionellen Bereich, wo eine Webapplikation im Hintergrund auf einen oder mehrere Applikationsserver ausgelagert sein kann, Webserver. Dabei unterscheidet man grob zwei Architekturen:

1. Standalone
2. Integriert

Bei der ersten Architektur ist die Webapplikation ein eigenständiges Binär-Programm bzw. ein von einem eigenständigen Binär-Programm interpretiertes Skript (z.B. CGI-Programm), welches für jede Anfrage bzw. Request neu gestartet wird.

Bei der zweiten Architektur ist die Webapplikation Teil des Webservers oder ein vom Webserver interpretiertes Skript (z.B. durch ein Modul des Webservers interpretiert), welches im Zusammenspiel mit einem im Hintergrund stehenden Applikationsserver bearbeitet wird. Es muss nicht mehr für jede Anfrage ein Prozess gestartet werden. Dies könnten z.B. PHP-, Perl-, Python-, Java Servlet-, Java Server Pages-, Java Server Faces-Webapplikationen sein.

Zu den zentralen Bestandteilen von Webapplikation, wenn man Webanwendungen als Schichtenarchitekturen betrachtet, gehören der „Browser“, der „Webserver plus Applikati-

onsserver (z.B. Webcontainer)“ sowie „Datenbank“. Sie weisen folgende Eigenschaften auf bzw. sind für die folgenden Eigenschaften zuständig:

- Browser: grafische Benutzeroberfläche der Webanwendung, die der Präsentationsschicht entspricht
- Webserver plus Applikationsserver: zuständig für die Anwendungslogik und entsprechen der Logikschicht
- Datenbank: verantwortlich für die Persistenz der Daten, die der Datenhaltungsschicht entspricht

Besonders hervorzuheben wie in 2.1 erwähnt ist hierbei, dass Webapplikationen genau wie RIA (laut 2.1 sind clientseitige Plug-Ins (Laufzeitumgebungen) von Rich Internet Applikationen ja konzeptionell mit serverseitigen Laufzeitumgebungen vergleichbar) eine Laufzeitumgebung benötigen. Ohne Laufzeitumgebungen sind Webapplikationen nicht benutzbar. Z.B. Webentwicklungen wie J2EE (Java) und .NET/COM (C++, C#) bieten Umgebungen, in denen Webapplikationen in so genannten Web Containern laufen („leben“), wobei die Container nur das Laufzeit- bzw. Lebenszyklusmanagement übernehmen und der Webentwickler sich fast ausschließlich mit der Anwendungslogik befassen muss. Im Falle von Java Web Entwicklung kann als J2EE Web Container z.B. Tomcat (Tomcat 6 2010) verwendet werden.

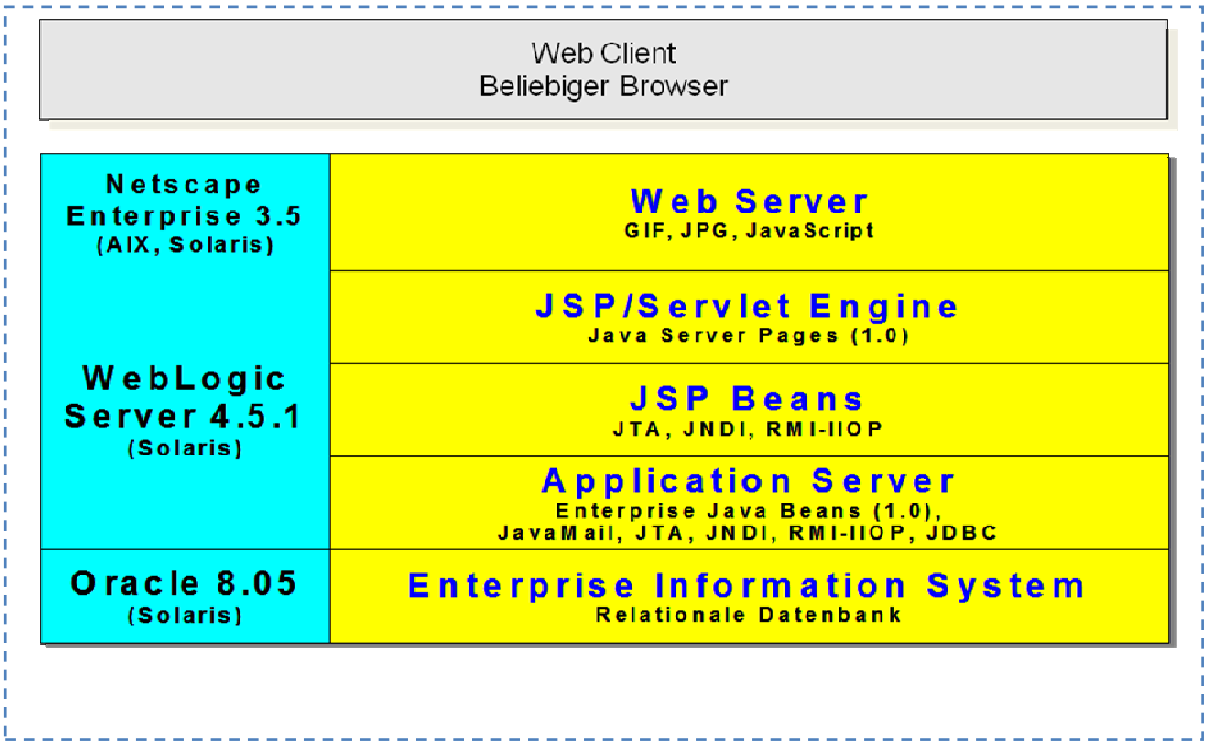

Abbildung 2.3.: mögliche Softwarearchitektur einer Webapplikation aus der Schichtenarchitektursicht nach (Merbeth 2001)

Für allgemeine Betrachtungen sowie detaillierte Beschreibungen von Webapplikationen siehe (Lubkowitz 2007), (Münz 2005) und zu deren Architektur (Wöhr 2004) und (Kappel u.a. 2003). Beispiele für Webapplikationen in der Praxis finden sich in (Flex-Applikationen 2010) und (Web-Applikationen 2010).

2.1.5. Leistungen der Domänendienstleister

Heutige Hosting-Kosten bzw. Domänenkosten (allgemein Betriebskosten) der Domänendienstleister für ressourcenaufwendige Webapplikationen, die dadurch einen großen und intensiven Datenverkehr (z.B. eine Bildbearbeitungswebapplikation, in der zu persistierende Bilder in einer Datenbank gespeichert werden) verursachen, sind nicht günstig. Sie setzten sich meistens aus mehreren Kosten zusammen und stellen nur einen begrenzten Rahmen von Leistungen für eine Webapplikation zur Verfügung.

Relevante für diese Arbeit Leistungen der Domänendienstleister, die eine Ressourcenoptimierung beeinflussen kann, umfassen Folgendes:

- Anzahl der Subdomänen
- Speicherplatz
- Monatlicher Traffic bzw. monatliches Übertragungsvolumen

Nämlich nur diese Leistungen der Domänendienstleister sind für die hohen Betriebskosten und Verfügbarkeit der ressourcenaufwendigen Webapplikation zuständig. Wenn z.B., wie in 1.1 bereits erwähnt, ein gegen eine Gebühr von einem Domänendienstleister zur Verfügung stehendes vereinbartes monatliches Übertragungsvolumen erreicht wird, dann wird die „deployte“ Webapplikation nicht mehr im Web verfügbar sein. Das ist für den Inhaber der Applikation und deren Benutzer selbstverständlich nicht schön und verlangt eine höhere Investition in den Betrieb.

Da große Webapplikationen einen großen und intensiven Datenverkehr zwischen einem Webserver und Client verursachen, ist es proportional zu den Betriebskosten.

2.1.6. Kosten einer Webapplikation

Die Kosten einer Webapplikation setzen sich aus Entwicklungs- und Betriebskosten zusammen. Da Betriebskosten die Hauptkostenverursacher sind, besonders die Traffic- und Speicherkosten, die immer wieder beim Betrieb der Anwendung im Web entstehen, wird hier hinsichtlich der zu implementierenden Webapplikation und bereits vorgestellten Leistungen der Domänendienstleister (2.1.5) eine beispielhafte Kostenberechnung der ressourcenaufwendigen „Anzieh-Anwendung“ ohne die in der Zielsetzung vorgenommenen Ressourcenoptimierung bezüglich der Traffic- und Speicherkosten dargestellt.

Nimmt man folgende durchschnittliche Daten für die Kostenberechnung nur bezüglich der Traffic- und Speicherkosten als Überbegriff Betriebskosten an, so entstehen unvorstellbare Kosten (für kleine private Webapplikationen), wenn sich die Besucherzahl der Webapplikation steigern würde:

Durchschnittliche Daten der Webapplikation

- Größe der Webapplikation, die beim Besuch immer wieder pro Benutzer heruntergeladen wird: 500 KB
- Durchschnittliche Größe eines hochauflösendes Bildes (Kleidungsstück): 1 MB
- Anzahl der Bilder im Inventar pro Benutzer: 50 Stück
- Besucheranzahl pro Tag: 50 Benutzer
- Datenverkehr, der bei nur einem einzigen Aufruf der Webapplikation entsteht, wenn man nur die Hauptverursacher (Bilder der Sachen) betrachtet: 500 KB + 1MB * 50 Stück = 50,5 MB (pro ein Aufruf der Webapplikation eines Benutzers)
- Speicheranforderung für die Bilder eines Benutzers: 50 Stück * 1MB = 50 MB
- Datenverkehr, der pro Tag entsteht: 50,5 MB * 50 Benutzer = 2,525 GB
- Speicheranforderung für die Bilder aller Benutzer: 50 MB * 50 Benutzer = 2,5 GB

Wenn man sich an die Kosten eines der bekanntesten Domänendienstleister „host4free.de" (Host4Free 2010) orientiert, dann entsteht somit folgende Berechnung, wenn man das fast optimale Paket mit Flatrate für Traffic und 2 GB Speicherplatz auswählen würde, obwohl es nicht ganz passt und wäre sehr ungünstig, wenn sich die Besucherzahl steigern würde:

Kosten des Paketes p.a.: 9 EURO pro Monat (Stand 22.05.2010) * 12 = 108 EURO

Hier wurden aber die günstigsten Vorbedingungen dargestellt. Es wurde gar nicht behandelt, welche Kosten entstehen würden, wenn man aus den schlimmsten Vorbedingungen (aus der Kostensicht) ausgeht, dass sich die Besucherzahl und die Anzahl der Bilder im Inventar kontinuierlich steigen. Wenn man aber die durchschnittliche Besucherzahl der gut besuchten Webinhalte nimmt und eine Kostenberechnung macht, dann entstehen wirklich ein unvorstellbarer großer Datenverkehr, Speicherbedarf und große monatliche Kosten (für kleine private Webapplikationen aber unvorstellbar), wo die Ressourcenoptimierung unvermeidlich ist:

- Besucheranzahl pro Tag: 1000 Benutzer
- Datenverkehr (Traffic), der pro Tag entsteht: 50,5 MB * 1000 Benutzer = 50,5 GB
- Speicheranforderung für die Bilder aller Benutzer: 50 MB * 1000 Benutzer = 50 GB

Z.B. der o.g. Domänendienstleister „host4free.de“ (Host4Free 2010) stellt zurzeit keine Pakete zur Verfügung, die die o.g. Anforderungen erfüllen würden. Man ist hier gezwungen, an professionelle Domänendienstleister anzuwenden, die im Vergleich dazu deutlich höhere Betriebskosten aufweisen.

2.1.7. Ressourcen von Webapplikationen

Im Folgenden werden Ansätze zur Ressourcenoptimierung von Webapplikationen kurz dargestellt, um dann daraus die wesentlichen Eigenschaften auf die Implementierung der Rich Internet Applikation zu übertragen.

Im Rahmen dieser Arbeit werden nur diejenigen Ressourcen von Webapplikationen betrachtet und am Beispiel einer in 1.2 beschriebenen Rich Internet Applikation optimiert, die einen Einfluss auf die Leistungen der Domänendienstleister (2.1.5), Performanz bezüglich der Interaktivität mit dem Benutzer und den Aufwand der wiederkehrenden Arbeiten bei der Implementierung einer Webapplikation haben, um möglichst niedrigere Entwicklungs- und Betriebskosten von ressourcenaufwendigen Webapplikationen zu erreichen.

In Abschnitt 2.1.5 wurde betrachtet und erarbeitet, welche Leistungen der Domänendienstleister für die Ressourcenoptimierung von Webapplikationen relevant sind bzw. welche Ressourcen der Webapplikation bezüglich der Betriebskosten genauer betrachtet werden sollen, um eine Optimierung am Beispiel einer Rich Internet Applikation zu erarbeiten. Im Folgenden werden Ansätze zur Ressourcenoptimierung bezüglich der Reduzierung der Domänen- und Implementierungskosten kurz dargestellt, um dann daraus die wesentlichen Eigenschaften auf die Webapplikation zu übertragen.

Wie in 2.1.3 bereits dargestellt, benötigen auch Webapplikationen, besonders ressourcenaufwendige, ein „Back-End“ bzw. die Logik- und Datenhaltungsschicht, für dessen Persistenzanforderung der Applikationsdaten ein möglichst unbegrenzter und schneller Speicherplatz erwünscht ist und dessen saubere Konzeption, Umsetzung und Implementierung einen großen zeitlichen Aufwand hat (bspw. Schemata einer Datenbank, Anwendungslogik einer Webapplikation, Einarbeitung und Konfiguration der bekannten Frameworks wie Hibernate (Hibernate 2010), Spring (Spring 2010), Struts (Struts 2010), Sitemesh (Sitemesh 2010), Velocity (Velocity 2010), OSCache (OSCache 2010), die meistens bei einer Implementierung verwendet werden, etc.). Aber die zeitanspruchsvollen Arbeiten der Implementierung einer Webapplikation sind meistens wiederkehrende Arbeiten (bspw. Sign-In- und Sign-Out-Komponente[4]), die durch eine Delegation an Web Services und Web-APIs erspart werden können. Dadurch wird erheblich der Aufwand der wiederkehrenden Arbeiten bei der Implementierung einer Webapplikation reduziert und die Speicherung bzw. Persistenz der Daten an das Dritte delegiert, was sehr deutlich den Traffic und benötigten Speicherplatz (Leistungen der Domänendienstleister) auf dem Webserver (Logikschicht) wiederrum reduziert.

---

[4] Mit der Komponente wird hier ein Teil der implementierten Software gemeint, der im Zusammenhang mit Backend (ggf. eine Datenbank der registrierten Benutzer) für eine Funktion bzw. einen Dienst der Webapplikation zuständig ist (bspw. ständiges Prüfen bzw. Validierung der Eingabe-Daten und anhand einer Anfrage aus der Datenbank eine Entscheidung treffen, ob der Login-Prozess erfolgreich war oder der Client überhaupt kein Konto hat und muss sich deshalb registrieren, wenn der aufgerufene Dienst der Webapplikation einen registrierten Benutzer voraussetzt).

Diese Delegation an Web Services und Web-APIs und somit die Optimierung der Betriebs- und Implementierungskosten von Webapplikationen ist nur dann bezüglich der o.g. Ressourcenoptimierung sinnvoll, wenn das clientseitig, d.h. ohne Mitwirkung des Webservers, stattfindet. Eine Rich Internet Applikation ist Mittel der Wahl, um das zu realisieren.

Also, mit Hilfe von Web Services und Web-APIs und am Beispiel einer Rich Internet Applikation wird folgende Ressourcenoptimierung von Webapplikationen durchgeführt:

- Betriebskosten: mit Hilfe von Web Services und Web-APIs inkl. RIA werden Traffic (Übertragungsvolumen), Speicherplatzanforderungen für Daten und Webapplikation selbst erheblich reduziert
- Performanz: durch asynchrone Delegation bzw. asynchrone Nachrichten an Web Services und Web-APIs wird die Performanz bezüglich der Interaktivität mit dem Benutzer erhöht
- Implementierungskosten: wiederkehrende Arbeiten werden an Web Services und Web-APIs delegiert

2.2. Rich Internet Applikationen (RIAs)

Im Rahmen dieser Arbeit wird eine Rich Internet Applikation mit Hilfe von Web Services und Web-APIs gemäß der in 1.2 beschriebenen Ressourcenoptimierung von Webapplikationen implementiert. Laut (Wikipedia 2010) setzt eine RIA per Definition ein höheres Maß an Programmlogik auf dem Client voraus, mit dem beispielsweise Berechnungen anstatt auf dem Server nunmehr auf dem Client durchgeführt werden können. Eine Rich Internet Applikation ermöglicht dem Besucher einer Website bzw. Webapplikation vor allem Drag-And-Drop-Funktionen, 3D-Effekte, Animationen und Unterstützung diverser Videoformate und anderer Medien. Rich Internet Applikationen bzw. clientseitige Webanwendungen werden vom Webserver heruntergeladen und auf dem Client typischerweise im Browser ausgeführt. Die Kommunikation der Anwendung mit dem Webserver erfolgt mittels HTTP-Protokoll, REST (REST 2010) oder Webservices.

Ende der neunziger Jahre wurde mit „Surfen im Web“ gemeint, online Texte zu lesen und statische Bilder zu sehen. Aber dieses Vergnügen war primitiv und als die Anzahl der internetbasierten Geschäfte und Individuationen stieg, tauchten Bedürfnisse in Richtung reichhaltige Benutzererfahrungen auf. 2002 führte Macromedia (Adobe 2010) den Begriff Rich Internet Applikation (RIA) ein. Rich Internet Applikationen kombinieren Flexibilität, Schnelligkeit und Einfachheit der Benutzung von Desktopapplikationen mit der umfangreichen Erreichbarkeit von Web. RIAs produzieren eine dynamische Weberfahrung, die auch so reichhaltig und attraktiv wie interaktiv ist.

Rich Internet Applikationen sind Webapplikationen (Applikationen), die clientseitig laufen. D.h. sie haben die gleichen nichtfunktionalen Anforderungen wie jede andere lokale Applikation: Stabilität, Robustheit, Erweiterbarkeit und Wiederverwendbarkeit, etc. (3.3). Rich Internet Applikation wird wie jede Webapplikation von einem Webserver für einen Benut-

zer zur Verfügung gestellt und ist zu jeder Zeit im Web aufrufbar. Beim Aufrufen der zugehörigen URL der Rich Internet Applikation bzw. bei einer Request der Applikations-URL wird diese Applikation nur einmal an den Client (Benutzer) übertragen. Weiterhin läuft die Rich Internet Applikation clientseitig, abgesehen von den Anfragen an externe Web Services und Web-APIs oder der Benutzung von dynamischen Daten, die von LifeCycle Data Services (Adobe LiveDocs 2009) zur Verfügung gestellt werden. Der Benutzer arbeitet dann mit der Applikation bis zur Vollendung seiner Session (Sitzung) clientseitig. Die Abb. 2.4, die auf der Abb. 2.2 basiert ist, stellt das beschriebene Verfahren dar.

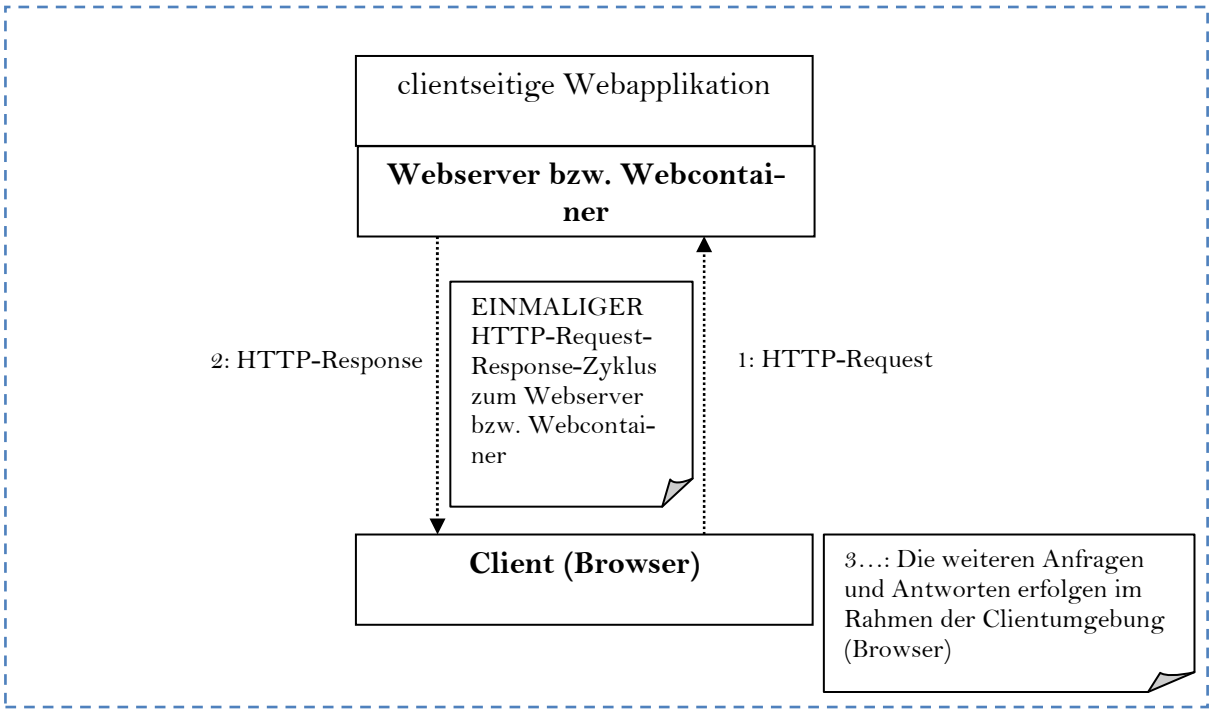

Abbildung 2.4.: Aufruf der Rich Internet Applikation (eigene Darstellung)

Für allgemeine Betrachtungen sowie detaillierte Beschreibungen von Rich Internet Applikationen und deren Arbeitsweise siehe (Adobe 2010), (Adobe wiki 2010) und zu deren Architektur (Keefe 2008). Beispiele für Rich Internet Applikationen in der Praxis finden sich in (Flex-Applikationen 2010) und (Ajax-Beispiele 2010).

In Abschnitt 2.1.7 wurde erarbeitet, dass eine der Ressourcenoptimierung, die im Rahmen dieser Arbeit durchgeführt wird, ist primär Optimierung des Traffics bzw. des übertragbaren Volumens zwischen einem Client und einem Webserver, die dadurch die monatlichen Kosten für den Traffic der Domänendienstleister sehr deutlich reduzieren kann. Rich Internet Applikation ist ein sehr gutes Mittel, um das zu ermöglichen. Vorausgesetzt ist aber, dass die benutzten Web Services und Web-APIs eine „Crossdomain-Kommunikation“ ermöglichen und zur Verfügung stellen, was in Abschnitt 4.3 ausführlicher betrachtet und erarbeitet wird.

2.2.1. Frameworks für RIAs

Der Einsatz von Frameworks kann die Anwendungsentwicklung erheblich vereinfachen.
Es gibt zurzeit viele Technologien (inkl. Frameworks), die die Entwicklung von Rich Internet Applikationen ermöglichen und mit entsprechenden Frameworks diese unterstützen. Zu den meist eingesetzten Technologien gehören u.a. JavaScript (inkl. AJAX), Java Applets, Flash, ActiveX-Plug-Ins, Silverlight u.ä..

Meistens treten clientseitige Anwendungen in einer Mischform mit einer serverseitigen Anwendung auf. In einer Clientanwendung (Webbrowser) läuft eine zweite Anwendung (z. B. ein Script in der aktuellen HTML-Seite). Eine in diesem Bereich häufig verwendete Programmiersprache ist JavaScript (JavaScript, 2010).

JavaScript ist eine Skriptsprache, die in Web-Browsern (hauptsächlich für das DOM-Scripting) eingesetzt wird. In JavaScript lässt sich objektorientiert und sowohl prozedural als auch funktional programmieren. Mit JavaScript werden Inhalte der HTML-Seite generiert und nachgeladen. Der Quellcode wird in die HTML-Seite eingebaut und durch den Browser interpretiert. Der große Nachteil aber bei der Verwendung von JavaScript liegt daran, dass die korrekte Interpretation der JavaScript-Spezifikation nach W3C client- bzw. browserabhängig ist, was dazu führt, dass man nie sicher sein kann, ob alle Features der Webapplikation, besonders ganz neue, von jedem Browser korrekt oder überhaupt unterstützt werden. Zudem führt das zahlreiche Vorhandensein der JavaScript-Frameworks wie z.B. JQuery (jQuery 2010) dazu, dass man bei der Auswahl sehr analytisch vorgehen muss.

Ajax ist ein Akronym für „Asynchronous JavaScript And XML“. Es bezeichnet ein Konzept der asynchronen Datenübertragung zwischen einem Client (Browser) und dem Server. Dieses ermöglicht es, HTTP-Requests durchzuführen, während eine HTML-Seite angezeigt wird und die Seite zu verändern, ohne sie komplett neu zu laden.
Das Ajax-Konzept findet bei clientseitigen Anwendungen in einer Mischform, die in HTML eingebettet sind, eine wachsende Bedeutung. Die Benutzung von Ajax-Konzept ist besonders bei serverseitigen Anwendungen zu finden, die mit diesem Konzept eine deutliche Performancesteigerung realisieren können. Denn wozu braucht man eine komplette Durchführung einer Server-Request bei der Webapplikation, wenn sich nur ein kleiner Teil der Webanwendung z.B. die Produktliste geändert hat, wenn man eine bestimmte Checkbox anklickt.

Ein Java-Applet ist ein Java-Programm, das normalerweise in einem Webbrowser ausgeführt wird. Die Applets wurden eingeführt, um Programme in Webseiten ablaufen lassen zu können, die auf der Client-Seite arbeiten und direkt mit dem Benutzer interagieren können, ohne Daten zum Server versenden zu müssen. Neben Applets existieren auch Servlets, die ebenfalls Java-Programme sind und auf dem Server ausgeführt werden.
Üblicherweise werden Java-Applets von HTML-Seiten aufgerufen. Um sie ausführen zu können, muss der jeweilige Client über eine entsprechende Java-Virtuelle-Maschine verfügen.
Die Applet-Technologie bietet dem Programmierer unter Berücksichtigung der Sicherheitsregeln den vollen Funktionsumfang aus der J2SE-API (J2SE 2010) und eignet sich sehr gut für Anwendungen in gepflegten Unternehmensnetzen. Ein Nachteil für den Einsatz im Internet und an langsamen Netzwerkanschlüssen ist aber die Größe der Java-Laufzeitumgebung. Ein weiterer Nachteil ist die vergleichsweise lange Initialisierungszeit für die Laufzeitumgebung, wenn diese noch nicht geladen ist sowie die Zeit, um das Applet herunterzuladen und zu initialisieren.

Zu beachten ist auch, dass Inhalte von Java-Applets nicht von Suchmaschinen erfasst werden können.

ActiveX bezeichnet ein Softwarekomponenten-Modell von Microsoft für aktive Inhalte. Es sind Softwarekomponenten für andere Anwendungen. Sie können gleichermaßen in verschiedenen Programmiersprachen und Umgebungen verwendet werden. Einige Programme nutzen zum Beispiel den Microsoft Internet Explorer (IE 2010) zur Anzeige von Informationen.
ActiveX gibt es aber nur für die Betriebssystemfamilie Windows. Da das ActiveX keine eigenen Sicherheitsfunktionen vorsieht, ist der Einsatz von ActiveX-Komponenten in Webbrowsern umstritten. Die Sicherheit muss daher von dem Entwickler der Komponente sichergestellt werden. Es lassen sich auch nicht speziell für Browser entwickelte Komponenten als ActiveX-Komponenten in Microsoft Internet Explorer benutzen, was ein weiteres Sicherheitsproblem darstellt.

Adobe Flash ist eine Entwicklungsumgebung zur Erstellung multimedialer und interaktiver Inhalte. Der Benutzer produziert mit dieser Software Dateien im proprietären SWF-Format (Small Web Format). Bekannt und umgangssprachlich gemeint ist Flash als Flash Player, mit dem man diese SWF-Dateien betrachten kann.
Der Flash Player ist einer der am meisten verbreiteten Browser-Plug-Ins und konkurriert neben W3C-Webplattformen wie HTML5 (HTML5 2010) und AJAX mit Silverlight und JavaFX.

Microsoft Silverlight ist wie Adobe Flash eine Erweiterung für Webbrowser in Form eines Browser-Plug-Ins, die die Ausführung von Rich Internet Applikationen (die für Silverlight-Plattform entwickelt wurden) ermöglicht und basiert auf einer reduzierten Version des .NET Frameworks (.NET 2010).
Silverlight ist als Plug-in für Windows und Apple Macintosh verfügbar und wird für die genglichen Browser wie Internet Explorer, Mozilla Firefox und Safari (Safari 2010) angeboten. Für Linux wird von Novell mit Zustimmung und Unterstützung von Microsoft das Plug-In Moonlight (Moonlight 2010) angeboten.
Silverlight ist hinsichtlich seiner UI-Präsentationsschicht aus der Windows Presentation Foundation (WPF 2010) abgeleitet. WPF wurde mit dem .NET Framework der Version 3 eingeführt.

Da die Technologie Flash im Vergleich zu den o.g. Technologien einer der führenden Plattformen auf dem Markt ist und sehr gut für die clientseitige Architektur geeignet ist, wird im Rahmen dieser Arbeit eine Rich Internet Applikation mit der Technologie Flash implementiert. Zudem hat Flash noch eine sehr große Verbreitung und Akzeptanz laut (Flash Player Ubiquität 2010) im Web und findet damit auf vielen Webseiten Anwendung und stellt zusätzlich ein Open-Source-Produkt dar.

Viele Web- Designer und Entwickler nutzen Adobe Flash oder Adobe Flex (Adobe 2010), die Teil der Adobe Flash Plattform sind, um Rich Internet Applikationen zu bauen. Flash ist eine autorisierte Umgebung für die Erstellung des reichhaltigen, interaktiven Inhalts für das Web. Flex ist ein Cross-Plattform Entwicklungsframework für die Erstellung von RIAs. Der mit Flash und Flex deployte Inhalt läuft im Adobe Flash Player, der meistens als Plug-In in einem Browser integriert ist und somit die Laufzeitumgebung für diesen Inhalt bereitstellt. RIAs, die in Flash oder Flex erstellt wurden, können auch mit der Adobe-AIR-Desktop-Laufzeitumgebung auf dem Desktop benutzt werden.

Im Rahmen dieser Arbeit wird mit Adobe Framework „Flex“ die in 1.2 beschriebene Rich Internet Applikation entwickelt.

In Abschnitt 4.4 wird noch einmal näher auf RIA-Framework „Flex“ eingegangen. Im Rahmen des Entwurfs (Kap. 4) wird beschrieben, wie die Rich Internet Applikation bzw. ihre Komponenten mit diesem Framework aus der Softwareengineering-Sicht sinnvoll realisiert werden können, so dass die in der Analyse (Kap. 3) erarbeiteten Anforderungen erfüllt werden.

2.2.2. Webapplikation vs. Rich Internet Applikation

In 2.1.3 und 2.2 wurde bereits das Wesentliche der Technologie Webapplikation und eine Teilmenge davon Rich Internet Applikation betrachtet. Im Folgenden werden in Form einer kleinen Zusammenfassung die wesentlichen Unterschiede vor allem hinsichtlich der Webapplikationssitzung (Session), Vorteile und Nachteile für einen kleinen Überblick kurz dargestellt.

Eine clientseitige Anwendung läuft im Unterschied zur serverseitigen nicht auf dem Server (z. B. einem Webserver oder Webcontainer wie Tomcat (Tomcat 2010)), sondern auf dem Rechner des Nutzers ab. Allerdings sind diese Clientanwendungen in aller Regel in eine Clientkommunikation eingebunden. Wichtigste clientseitige Anwendungen sind die Clients selbst, wie z. B. Browser Mozilla Firefox (Firefox 2010), Safari (Safari 2010) oder Microsoft Internet Explorer (IE 2010), welche alle HTTP-Protokoll (HTTP 2010) zur Datenkommunikation mit einem Webserver bzw. Webcontainer unterstützen.

Zwischen einem Client und einem Server gibt es zur Datenübertragung und zur Interaktion ein definiertes Protokoll (meistens HTTP), welches in der Regel eines Request- Response-Schema folgt. Durch eine clientseitige Anwendung kann der Request- ResponseSchema unterbrochen werden, indem bestimmte Funktionen wie z. B. kurzzeitige Reaktionen auf Benutzerinteraktionen direkt clientseitig durchgeführt werden, ohne einen Client- Server-Rundlauf auszulösen.

Wichtig ist im Zusammenhang auch, wie der Kontext einer Benutzersitzung gespeichert werden kann. Da HTTP-Protokoll zustandslos ist, kann also serverseitige Webapplikation grundsätzlich keine Daten zwischen zwei Datenanforderungen speichern. Eine clientseitige Anwendung kann das Cookie-Konzept problemloser verwenden, um Informationen auf der Clientseite zu speichern, welche zu einem späteren Zeitpunkt wieder ausgelesen werden können. Cookies bieten allerdings keine Form der Datensicherheit und können z. B. auch von anwendungsfremder Spyware ausgelesen werden. Diese funktionale Lücke kann durch das Konzept eines serverseitigen Session-Managements geschlossen werden. Die an sich zustandslose Client- Server-Kommunikation wird dadurch zu einem System, in dem ein Session-Zustand auf dem Server über mehrere Datenanforderungen aufrechterhalten werden kann. Hierbei kann dann auch eine passwortgeschützte Datensicherheit gewährleistet werden.

2.3. Web Services und Web-APIs

Der Einsatz von Web Services und Web-APIs in der Anwendungsentwicklung ist nach (Wöhr 2004) erfolgreich und empfehlenswert und die in 2.1.7 erarbeitete Ressourcenoptimierung im Rahmen dieser Arbeit ist dadurch, im Zusammenhang mit der Technologie „Rich Internet Applikation“, sehr gut realisierbar. D.h. Web Services und Web-APIs mit der Technologie „Rich Internet Applikation“ lassen folgende Ressourcenoptimierung aus der technischen Sicht durchführen bzw. es werden im Rahmen dieser Arbeit mit Hilfe von Web Services und Web-APIs am Beispiel einer Rich Internet Applikation (1.2) folgende Ressourcen optimiert, um die Zielsetzung zu erreichen:

- Größe der Webapplikation
- Traffic
- Performanz bzgl. der Interaktivität zwischen Benutzer und Webapplikation
- Implementierungsaufwand der Webapplikation

Optimierung der ersten zwei Ressourcen befasst sich mit der Optimierung der Domänenkosten und entspricht zusammen mit der Performanceoptimierung den Betriebskosten. Die letzte Art der Ressourcenoptimierung gehört zur Optimierung der Entwicklungskosten.

Die in 1.2 zum größten Teil aus der technischen Sicht beschriebene und dargestellte Webapplikation und anhand der gemachten in diesem Kapitel Überlegungen dazu soll am Beispiel einer Rich Internet Applikation mit Hilfe von Web Services und Web-APIs implementiert werden. Sie soll wie in Abb. 2.5 veranschaulicht für den Rest dieser Arbeit als Demonstration der erarbeiteten Ressourcenoptimierung dienen.

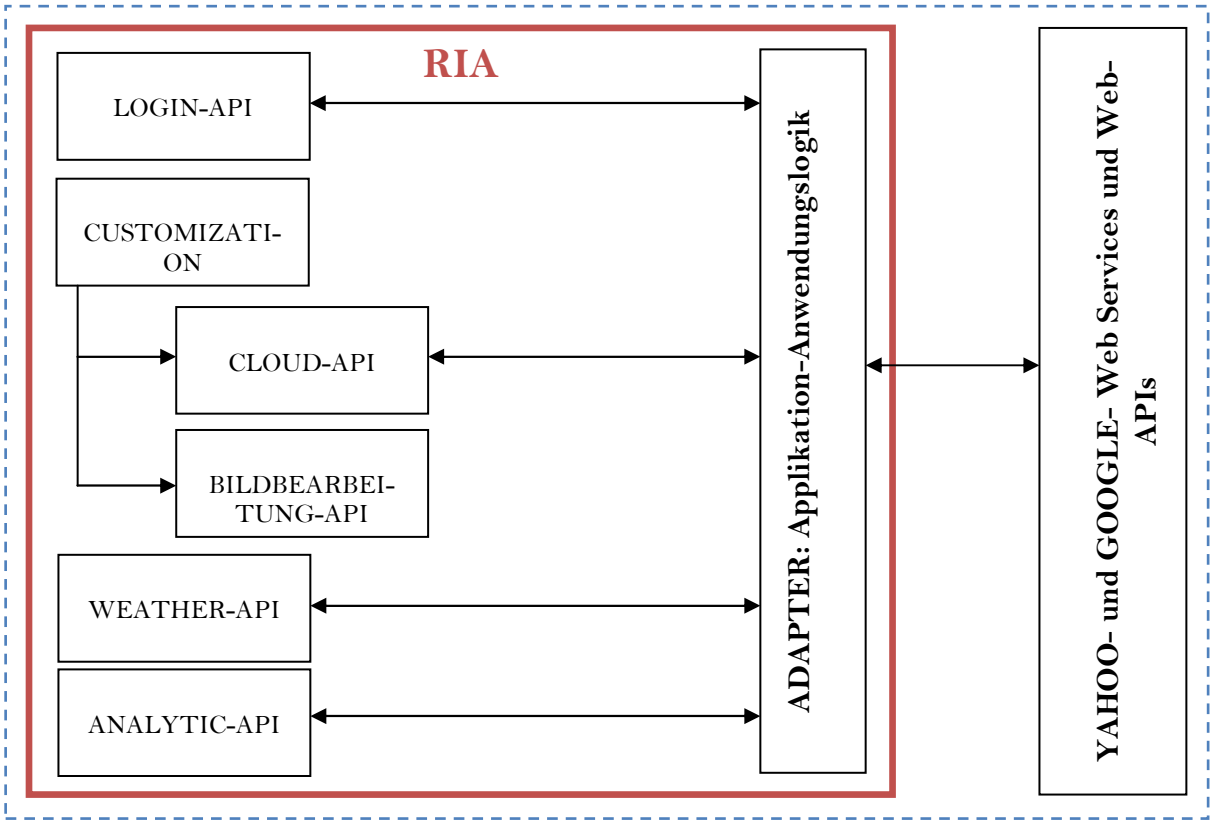

Abbildung 2.5.: Beispiel-Szenario der Webapplikation (eigene Darstellung)

2.3.1. Besondere Problemstellung Delegation

Bei der Verwendung von Web Services und Web-APIs in Webapplikationen sind folgende Rahmenbedingungen zu beachten:

- Offline-Situationen: Web Services und Web-APIs sind unter Umständen nicht ständig online.
- Kommunikationsstörungen: Es kann zu Verbindungsabbrüchen durch die Überlastung im Web kommen.
- Bandbreite: Die verfügbare Bandbreite variiert unter Umständen. Die Datenlaufzeiten variieren je nach Auslastung bei Web Services und Web-APIs, die zur Performancereduzierung führen kann.
- Kosten: Es gibt zahlreiche Web Services und Web-APIs im Web. Die Verbindungskosten sind unter Umständen hoch, wenn man nicht kostenlose Web Services und Web-APIs benutzt (z.B. (AMAZON Web-APIs 2010)). Es ist daher empfehlenswert, sich auf heutzutage kostenlose Web Services und Web-APIs zu konzentrieren (z.B. (YAHOO Web-APIs 2010), (GOOGLE Web-APIs 2010)).
- Sicherheit: Die Risiken des Diebstahls von Daten, die bei der CLOUD-Delegation in entsprechenden Web Services und Web-APIs persistiert werden, sind größer als bei lokalen, einheitlichen Systemen. (Nicht im Rahmen dieser Arbeit behandelt.)

Weiterhin sind bei der Delegation an Web Services und Web-APIs begrenzte Schnittstellen hinsichtlich der Möglichkeiten zu beachten. Die in (YAHOO Web-APIs 2010) und (GOOGLE Web-APIs 2010) beschriebenen Web Services und Web-APIs bieten interessante, kostenlose und zahlreiche Web Services und Web-APIs. Für den Entwurf einer Rich Internet Applikation werden folgende in 4.3 beschriebene Web Services und Web-APIs als Vorbild hinsichtlich der Anforderungen (3.1, 3.3) und Anwendungsfälle (3.2.2) der zu implementierenden RIA benutzt, die umfangreiche Schnittstellen und eine sehr große Akzeptanz der Webentwickler haben.

2.4. Fazit

Heutzutage sind Webapplikationen sehr verbreitet und die Tendenz der Webseiten geht in deren Richtung. Da immer mehr ressourcenaufwendige und umfangreiche Webapplikationen im Webeinsatz sind und immer mehr mobile, unter Umständen ressourcenbegrenzte Geräte ins „Web wollen“, spielt Ressourcenoptimierung von Webapplikationen hinsichtlich der dargestellten Problemstellung (1.1) eine sehr große Rolle.

Da ressourcenaufwendige Webapplikationen auch höheren Entwicklungsaufwand bzw. höhere Entwicklungskosten und Betriebskosten umfassen, vor allem Traffic und Speicherplatz, ist es wünschenswert, dass bei der Entwicklung der Webapplikationen die in 2.3 erarbeitete

Ressourcenoptimierung berücksichtigt wird. Rich Internet Applikationen und Web Services und Web-APIs sind hier zunächst geeignet, da sie zusammen weniger Ansprüche an den Traffic und Entwicklungsaufwand (allgemein Entwicklungs- und Betriebskosten) haben, was aber leider nicht immer möglich ist und einige Nachteile mit sich bringt.

Die zu entwickelnde Rich Internet Applikation muss die in 2.3 erarbeitete Ressourcenoptimierung erfüllen. Dabei werden im Rahmen dieser Arbeit nur kostenlose (2.3.1) Web Services und Web-APIs hinsichtlich der Anforderungen der zu implementierenden Webapplikation verwendet. Die benutzten Web Services und Web-APIs werden durch die Rich-Internet-Applikation-Komponenten aus der technischen Sicht abgebildet und die erreichte und vergleichbare (2.1.7) Benutzerakzeptanz von Ressourcenoptimierung der Zielsetzung wird durch die „ANALYTICS-Komponente bzw. ANALYTICS-API“ der Webapplikation darstellbar bzw. sichtbar. Diese Rich-Internet-Applikation-Komponenten als Teilmenge der zu implementierenden Rich Internet Applikation werden in Kapitel 3 analysiert und weiteren Kapiteln umgesetzt.

Zu beachten ist dabei, dass heutzutage Webapplikationen nicht nur von den „großen“ Geräten wie z.B. Laptops und PCs benutzt werden, sondern auch von kleinen mobilen Geräten.
Die mit Hilfe von Framework „Flex“ und Web Services und Web-APIs zu entwickelnde Rich Internet Applikation soll beispielhafte Demonstration bzw. beispielhafte Komponenten für die Demonstration der in 2.1.7 erarbeiteten Ressourcenoptimierung von Webapplikationen im Rahmen dieser Arbeit zur Verfügung stellen. Bei der Implementierung dieser Webapplikation sollten spezielle funktionale und nichtfunktionale Anforderungen erarbeitet und berücksichtigt werden. Ein Mittel, dies zu erreichen, ist die Verwendung von Entwurfsmustern.

3. Analyse

Die in Kapitel 2 erarbeitete Ressourcenoptimierung (2.1.7) soll am Beispiel einer Rich Internet Applikation (1.2) und beispielhaften Web Services und Web-APIs (2.3.1) implementiert werden. Im Folgenden werden dafür anhand der existierenden Spezifikationen, des existierenden RIA-Frameworks „Flex“, der existierenden Flash-APIs für beispielhafte Web Services und Web-APIs (4.3) und allgemeiner Betrachtungen funktionale und nichtfunktionale Anforderungen aus fachlicher Sicht formuliert und dementsprechend zu erstellende Komponenten und deren Schnittstellen (APIs) der zu implementierenden Rich Internet Applikation hinsichtlich der erarbeiteten Ressourcenoptimierung erarbeitet.

Sequenz-, Klassen- und Zustandsdiagramme sind dabei in einer an die „Unified Modeling Language“ (UML, s. (UML 2010)) angelehnten Notation gehalten. Bei Diagrammen ohne Quellenangabe handelt es sich in diesem Kapitel um eigene Darstellungen.

Es wird nicht mehr auf die Grundlagen der verwendeten Technologien (Webapplikation, RIA-Webapplikation, Leistungen der Domänendienstleister, Webserver, Webclient, Webserver-Client-Kommunikation, Web Services und Web-APIs, HTTP-Protokoll) eingegangen, da diese in Kapitel 2 mit Literaturhinweisen in einem für diese Arbeit ausreichenden Ausmaß behandelt wurden (s.a. Literaturhinweise).

3.1. Motivation

Ausgehend von der Zielsetzung (1.2) und Zeitgründen, die im Rahmen dieser Arbeit zur Verfügung stehen, wird eine Webapplikation benötigt, bei der man im Laufe der Implementierungsphase mit der ressourcenaufwendigen Websoftwareentwicklung unter Einbeziehung der Aspekte Performanz, Reduzierung der Implementierungs- und Domänenkosten und allgemein Ressourcenoptimierung befassen kann. Folgende wichtige Aspekte und Eigenschaften bezüglich der Ressourcenoptimierung (1.2) soll die beispielhafte zu implementierende Webapplikation aufweisen, um das Erreichen der Ziele zu ermöglichen:

- Benutzerdatenbank: Das System bzw. die Webapplikation sollte über einen autorisierten Zugang verfügen, um einen größeren Implementierungsaufwand bzw. größere Implementierungskosten und wiederholbare Tätigkeiten simulieren zu können. Mit „wiederholbare Tätigkeiten“ wird hier gemeint, dass die meisten Webapplikationen eine Benutzerdatenbank bzw. ein Autorisierungsmechanismus benötigen.

- Persistenz: Das System sollte über eine Datenbank verfügen, wo die Benutzerdaten persistiert werden können, um einen größeren Implementierungsaufwand bzw. größere Implementierungskosten und wiederholbare Tätigkeiten simulieren zu können.

- Sehr großer Speicherplatzbedarf: Das System sollte einen großen Speicherbedarf aufweisen (Benutzerdaten), um einen größeren Implementierungsaufwand, hohe Domänenkosten (z.B. Traffic- und Speicherplatzkosten) bzw. Betriebskosten simulieren zu können.

- Universalität: Das System sollte über möglichst viele Features verfügen, um einen größeren Implementierungsaufwand bzw. größere Implementierungskosten und allgemeinen Ressourcenoptimierungsbedarf (z.B. Performanz) simulieren zu können.

Wie in 2.1.1 bereits angedeutet und in 2.1.2 etwas mehr beleuchtet wurde, eignet sich gut dafür die clientseitige Architektur bzw. die Anwendungsklasse der Webapplikationen „Rich Internet Applikation“, die man mit einem Überbegriff „Webapplikationen“ bezeichnen kann und die eine Unterklasse davon bildet. Die Abb. 3.1 veranschaulicht die Unterteilung der Anwendungsklassen von Webapplikationen.

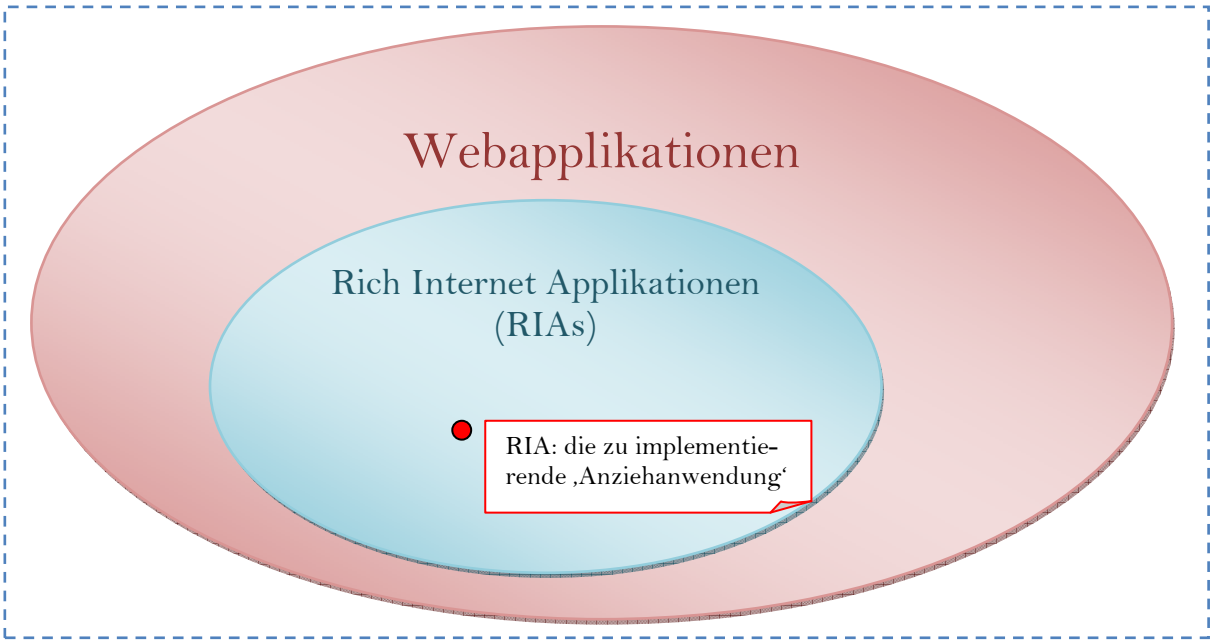

Abbildung 3.1.: Unterteilung der Anwendungsklassen von Webapplikationen (eigene Darstellung)

Die in 1.2 vorgestellte und beschriebene „Anzieh-Anwendung“ ist ein der passenden und sehr interessanten Repräsentanten der Anwendungsklasse „Rich Internet Applikation“, die alle o.g. Aspekte aufweist. Dabei, im Zusammenhang mit der Webapplikationsbeschreibung (1.2), stellen sich folgende Anforderungen an Anwendung bezüglich der in diesem Abschnitt dargestellten Eigenschaften:

- Anforderung an „Benutzerdatenbank“: die „Anzieh-Anwendung“ hat Kenntnisse über ihre Benutzer und sollte über ein Mechanismus verfügen, das für den autorisierten

Zugang zuständig ist. Diese Anforderung bezieht sich auf den Anwendungsfall „Login“ (3.2.2). Der Benutzer hat dabei die Möglichkeit, sich ein- und auszuloggen.

- Anforderung an „Persistenz“: die „Anzieh-Anwendung“ verfügt über ein Mechanismus, das für die Speicherung der Kleidungsbilder des jeweiligen Benutzers zuständig ist. Diese Anforderung bezieht sich auf den Anwendungsfall „Laden der Bilder“ (3.2.2). Der Benutzer hat dabei die Möglichkeit, seine Bilder aus der Datenbank abzurufen und sie in seinem Inventar darstellen zu lassen.

- Anforderung an „Sehr großer Speicherplatzbedarf“: die „Anzieh-Anwendung“ sollte über einen unbegrenzten Speicherplatz verfügen, der der Anforderung „Persistenz“ entspricht. Diese Anforderung bezieht sich auf den Anwendungsfall „Laden der Bilder“ (3.2.2). Der Benutzer hat dabei die Möglichkeit, seine Bilder aus der Datenbank abzurufen und sie in seinem Inventar darstellen zu lassen.

- Anforderung an „Universalität“: die „Anzieh-Anwendung“ sollte über eine erweiterbare Menge Features verfügen und zwar über die „aktuelle Wetter-Anzeige“ und „Bildbearbeitungskomponente“. Diese Menge ist beliebig erweiterbar. Diese Anforderung bezieht sich auf den Anwendungsfall „Bildbearbeitung“ (3.2.2). Der Benutzer hat dabei die Möglichkeit, seine Inventarbilder zu bearbeiten.

3.2. Funktionale Anforderungen

Jede Applikation dient dazu, um ein oder mehrere Geschäftsprobleme zu lösen. Normalerweise besteht der erste Schritt des Entwicklungslebenszyklus jeder Applikation in der Sammlung und Bestimmung ihrer Anforderungen. Gemäß 2.2 sind RIAs Applikationen, die die gleichen Anforderungen bzw. Qualitätsanforderungen wie die anderen Applikationen haben.

Aus technischer Sicht sind in dieser Arbeit nur die Webapplikationskomponenten hinsichtlich der Ressourcenoptimierung von Interesse. Die in 2.1.7 erarbeitete Ressourcenoptimierung wird am Beispiel in 1.2 beschriebenen RIA implementiert und in einem möglichen Entwurf konzipiert. Aber bevor man zum eigentlichen möglichen Entwurf der zu implementierenden Webapplikation bzw. deren Webapplikationskomponenten mit der Berücksichtigung der in 2.1.7 erarbeiteten Ressourcenoptimierung kommt, müssen zuerst die Anforderungen der Webapplikation aus fachlicher Sicht erarbeitet werden.

Funktionale Anforderungen der zu implementierenden Rich Internet Applikation betreffen primär im Endeffekt die Kommunikation zwischen Webapplikation und Web Services und Web-APIs, wobei das UI bzw. die Interaktion zwischen Webapplikation und Benutzer diese Kommunikation verursacht. Überlegungen, die in Kapitel 2 und übergeordneten Abschnitten gemacht wurden, müssen eingehalten werden. Die zu entwickelnde RIA soll dabei die im Rahmen dieser Arbeit erarbeitete Ressourcenoptimierung, Anwendungslogik (3.2.1) und Anwendungsfälle (3.2.2) der Webapplikation umsetzen, weshalb genau zu ermitteln ist, welche generischen Aufgaben die Webapplikationskomponenten übernehmen können und welche speziellen Aufgaben sie übernehmen müssen (s. 4.2). Da fast alle Hauptfunktionalitäten der zu implementierenden Webapplikation hinsichtlich der erarbeiteten Ressourcenoptimierung (2.1.7) mehr oder weniger zur Webapplikationslaufzeit von den verwendeten Web Ser-

vices und Web-APIs abhängig sind, muss das besonders betrachtet, analysiert und berücksichtigt werden. D.h. Benutzer- und Webapplikationsnachrichten (Ereignisse) müssen zur richtigen Zeit sowie zu aktuellem Zustand der Webapplikation passend gesendet werden und es muss auf die eingehenden Nachrichten (Ereignisse) ebenso passend reagiert werden, falls die Anwendungslogik (3.2.1) es erfordert.

3.2.1. Anwendungslogik der Webapplikation

Bei der folgenden Betrachtung von funktionalen Anforderungen an die zu implementierende Rich Internet Applikation muss zwischen Benutzer-Anwendungslogik und Applikation-Anwendungslogik unterschieden werden.

Die Benutzer-Anwendungslogik besteht – wie sich im Folgenden zeigen wird – aus spezifischen Flex-Klassen und -Interfaces, die eine API zur Benutzerschnittstelle (UI) anbieten und generische Aufgaben der Webapplikationskomponenten übernehmen.

Die Applikation-Anwendungslogik enthält die Geschäftslogik der Webapplikationskomponenten in Form von Web Services- und Web-APIs-Operationen (Aufrufe) bzw. spezifische Teile, die für die Hauptfunktion der Webapplikation zuständig sind (s. 4.5). Sie hat – wie sich ebenfalls im Folgenden herausstellen wird – Verantwortlichkeiten bezüglich der Einhaltung der Schnittstellen von benutzten Web Services und Web-APIs und übernimmt auch generische Aufgaben der jeweiligen Webapplikationskomponentenklasse (4.5).

Die Applikation-Anwendungslogik umfasst vor allem alle Abbildungskomponenten (4.1) und bildet somit einen Adapter („Middleware“), der das Marshalling der Kommunikation zwischen Webapplikation und Web Services und Web-APIs übernimmt (Abb. 3.2).

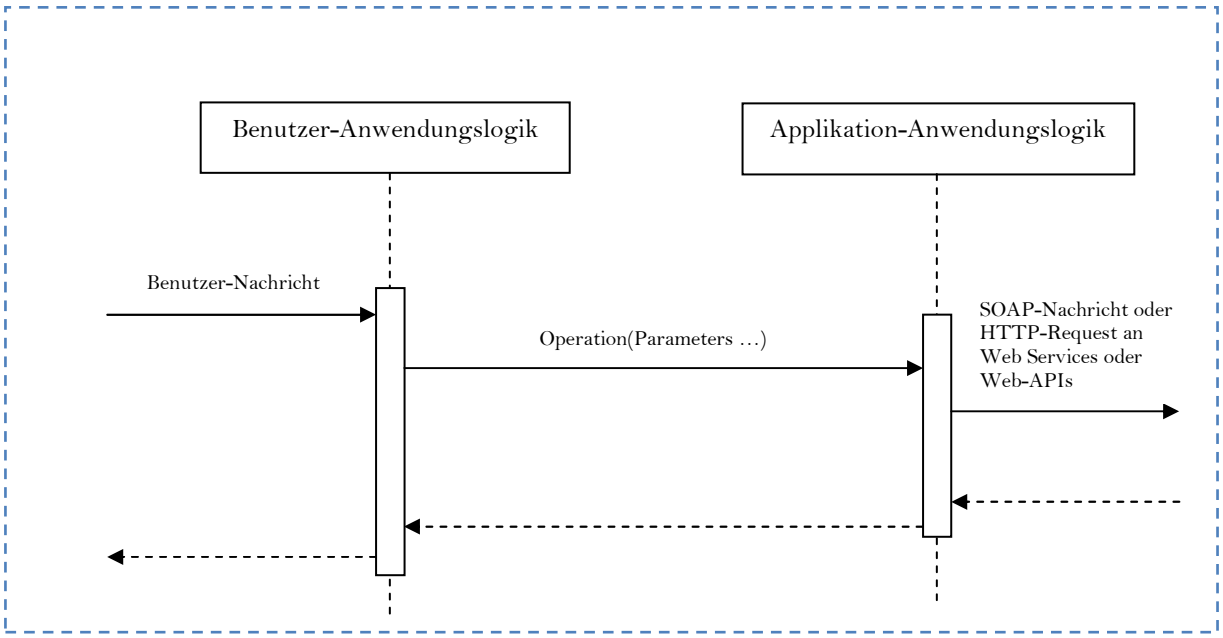

Abbildung 3.2.: Benutzer-Anwendungslogik und Applikation-Anwendungslogik (eigene Darstellung)

3.2.2. Anwendungsfälle

Bevor die einzelnen Nachrichten bzw. Ereignisse der Benutzer-Anwendungslogik und die möglichen Zustandsübergänge der Webapplikation im nächsten Abschnitt beschrieben werden, muss noch geklärt werden, welche Verantwortungen Benutzer- und Applikation-Anwendungslogik dabei haben.
Damit die zu implementierende Webapplikation dem Benutzer so viel Benutzerfreundlichkeit wie möglich anbietet und so viel Arbeit wie möglich abnehmen kann, sollte der Applikation-Anwendungslogik soviel Verantwortung wie möglich gegeben werden. Dies trägt auch zur Vermeidung von Fehlern aus der Sicht des Benutzers bei.

Es geht also um die Semantik der Nachrichten, jeweils auf Benutzer- und auf Applikation-Anwendungsseite. Die Semantik variiert dabei mit den Anwendungsfällen, für die die Webapplikation benutzt wird.
Als „Anwendungsfälle" werden hier für die Webapplikation die in Abb. 3.3 aufgeführten Kombinationen aus Akteur (aus Sicht des Benutzers) und Anwendungsfall im Zusammenhang eines Systemkontextes in Anwendungsfalldiagrammen betrachtet, wobei die Bezeichnung der Anwendungsfälle die Prosabezeichnungen tragen.

Prinzipiell muss zwischen internen und externen Anwendungsfällen (Operationen) der Webapplikation und von den mit den Web Services und Web-APIs zusammenhängenden Operationen bezüglich deren Schnittstellen und lokale Operationen der Benutzer unterschieden werden. Mit Stern gekennzeichneten Anwendungsfälle sind externe Operationen und werden an die Applikation-Anwendungslogik weiter delegiert, die in 4.2 besonders erarbeitet und betrachtet werden, um von der fachlichen Sicht besser abstrahieren zu können.

Die Aktivierung, Registrierung und Terminierung der externen Operationen sollte komplett von der Applikation-Anwendungslogik übernommen werden, sobald eine Operation vom Benutzer aufgerufen wird. Bei den lokalen Operationen hat die Benutzer-Anwendungslogik dagegen allein Verantwortlichkeiten.

Im Rahmen der zu implementierenden Webapplikation werden folgende Anwendungsfälle bezüglich der Benutzer-Anwendungslogik definiert und behandelt:

- *Sich Einloggen:* Die zu implementierende Webapplikation (1.2) sieht eine autorisierte Benutzbarkeit voraus und ist mit einem Login- und Logout-Vorgang auszustatten. D.h. beim Starten der Webapplikation „läuft" sie im nichtautorisierten Status. Die Webapplikation fordert mit der Login-Maske den Benutzer auf, sich einzuloggen. Wenn der Login-Vorgang erfolgreich war, wird der Benutzer von der Webapplikation weitergeleitet und der Status bzw. Zustand ändert sich in ACTIVE.

  Titel: **Login***
  Akteur: **Benutzer**
  Ziel: **Anmeldevorgang zur Benutzung der Webapplikation**

  Auslöser:
  Benutzer entscheidet sich zur Benutzung der Webapplikation

  Vorbedingungen:

Keine

Nachbedingungen:
Benutzer befindet sich in Besitz der Webapplikation

Erfolgsszenario:
1. Benutzer ruft Website-URL auf
2. Benutzer navigiert zu Anmelde-Button und betätigt den Anmeldevorgang
3. Das System leitet den Benutzer an den betätigten Login-API (Web-Services oder Web-API) weiter
4. Benutzer navigiert zu Formular für Anmeldedaten
5. Benutzer füllt Formular für Anmeldedaten mit den folgenden Daten: Benutzer-ID, Passwort
6. Benutzer schickt das Formular ab
7. Der Web-Service bzw. das Web-API prüft die Anmeldedaten
8. Der Web-Service bzw. das Web-API fordert die Bestätigung vom Benutzer zur Weiterleitung auf
9. Der Web-Service bzw. das Web-API leitet den Benutzer an das System weiter
10. Das System leitet den Benutzer an die Webapplikation weiter (die Webapplikation ist im Zustand ACTIVE)

Erweiterungen:
5a. Falls Prüfung Fehler aufdeckt: der Web-Service bzw. das Web-API stellt Formular dar (mit vorher eingegebenen Werten), markiert Fehler, erläutert Fehler und fordert Benutzer zur Korrektur oder Registrierung auf.

Fehlerfälle:
5a. Benutzeranmeldung schlägt dreimal in Folge fehl: Web-Service bzw. Web-API sperrt Benutzerkennung vorläufig und sendet E-Mail-Benachrichtigung an Benutzer.
8a. Falls Benutzer die Bestätigung verweigert: Der Web-Service bzw. das Web-API leitet den Benutzer an die Webapplikation mit Parametern, dass der Anmeldevorgang nicht erfolgreich war (die Webapplikation bleibt im Zustand INACTIVE).

Häufigkeit:
Pro Benutzer in Abhängigkeit von Besucheranzahl

Anforderungen:
Keine

- *Sich Ausloggen:* Die Webapplikation bietet mit dem „Logout-Button“ dem Benutzer an, sich auszuloggen und damit seine Sitzung (Session) zu beenden. Dabei wird der Benutzer von der Webapplikation in den Startzustand weitergeleitet und der Status bzw. Zustand der Webapplikation ändert sich in INACTIVE. Wenn Benutzer sich die Webapplikation weiter benutzen möchte, muss er sich wieder anmelden.

  Titel: **Logout**
  Akteur: **Benutzer**
  Ziel: **Abmeldevorgang der Webapplikation**

  Auslöser:

Benutzer entscheidet sich zum Logout

Vorbedingungen:
1. Benutzer ist eingeloggt
2. Die Webapplikation ist im Zustand ACTIVE

Nachbedingungen:
Benutzer befindet sich in Besitz der Webapplikation nicht

Erfolgsszenario:
1. Benutzer navigiert zu Abmelde-Button und betätigt den Abmeldevorgang
2. Das System leitet den Benutzer an die Webapplikation (die Webapplikation ist im Zustand INACTIVE)

Erweiterungen:
Keine

Fehlerfälle:
Keine

Häufigkeit:
Pro Benutzer in Abhängigkeit von Besucheranzahl

Anforderungen:
Keine

Die oben genannten Anwendungsfälle fassen das LOGIN-API um und der erste Anwendungsfall veranlasst nach der erfolgreichen lokalen Operation einen externen Aufruf bzw. eine externe Operation.

- *Inventarbilder laden:* Nachdem die Webapplikation im Zustand ACTIVE ist, hat Benutzer eine Möglichkeit, seine persistierten bzw. gespeicherten Bilder zu laden und zu sehen. Dafür bietet die Webapplikation folgende Elemente an: 1) Eingabemaske zur Eingabe seiner Benutzer-ID, unter dessen seine Bilder gespeichert sind 2) Submit-Button zum Starten der Anfrage 3) Container zum Anzeigen der angefragten Bilder.

  Titel: **Laden der Inventarbilder***
  Akteur: **Benutzer**
  Ziel: **Laden der Bilder für die Webapplikation von CLOUD-API**

  Auslöser:
  Benutzer entscheidet sich zum Laden der Inventarbilder

  Vorbedingungen:
  1. Benutzer ist eingeloggt
  2. Die Webapplikation befindet sich im Zustand ACTIVE
  3. Der Benutzer verfügt über einen Account bei CLOUD-API des aufgerufenen Web-Services bzw. Web-API

Nachbedingungen:
Benutzer befindet sich in Besitz der Inventarbilder

Erfolgsszenario:
1. Benutzer navigiert zu Formular für Inventarbilder
2. Benutzer füllt Formular für Bilder mit den folgenden Daten: Benutzer-ID von CLOUD-API
3. Benutzer schickt das Formular ab
4. Die Webapplikation stellt eine Anfrage (Request) an CLOUD-API
5. Der Web-Service bzw. das Web-API (CLOUD-API) prüft die Benutzerdaten
6. Der Web-Service bzw. das Web-API stellt dem System die Benutzerdaten zur Verfügung in Form einer HTTP-Response mit der Liste der Bilder-URLs
7. Das System stellt dem Benutzer der Webapplikation seine Inventarbilder (persistierte Bilder bei CLOUD-API) grafisch zur Verfügung

Erweiterungen:
7a. Falls Prüfung die angeforderten Benutzerdaten nicht findet: die Webapplikation stellt dem Benutzer keine Inventarbilder grafisch zur Verfügung

Fehlerfälle:
4a. Auf die gestellte Anfrage (Request) an CLOUD-API folgt keine Antwort (Response): Web-Service bzw. Web-API ist nicht erreichbar (OFFLINE) oder ausgelastet

Häufigkeit:
Pro Benutzer in Abhängigkeit von Besucheranzahl

Anforderungen:
Keine

- *Bilder Drag-And-Drop:* Nachdem die Bilder geladen sind und stehen zur Bearbeitung im Container zur Verfügung, hat Anwender die Möglichkeit, diese Bilder zu selektieren und für Drag-And-Drop-Funktionalität zu handhaben. Dabei verfügt die Webapplikation über ein Feld bzw. eine Arbeitsbühne, wo man die gewünschten Bilder an seinem Profil kombinieren und anschauen kann.

Titel: **Drag-And-Drop der Inventarbilder**
Akteur: **Benutzer**
Ziel: **Positionierung des Inventarbildes**

Auslöser:
Benutzer entscheidet sich zum Ändern der Position des ausgewählten Inventarbildes bei der Kombinierung der Bilder an seinem Profil auf der Arbeitsbühne (Stage bzw. grafischer Bereich in Form eines Rechtecks)

Vorbedingungen:
1. Benutzer ist eingeloggt
2. Die Webapplikation befindet sich im Zustand ACTIVE
3. Die Webapplikation verfügt über die Inventarbilder des Benutzers

Nachbedingungen:

Benutzer befindet sich in Besitz des Bildes auf der aktuellen Position

Erfolgsszenario:
1. Benutzer navigiert zu Inventarbild auf der Darstellungsbühne, das er skalieren möchte
2. Benutzer wählt das Inventarbild aus, indem er auf das Bild mit der linken Maustaste klickt
3. Benutzer klickt mit der linken Maustaste das ausgewählte Bild an und lässt die Maustaste nicht los
4. Benutzer zieht das ausgewählte Bild auf die Arbeitsbühne, indem er mit der Maus den Drag-Vorgang vornimmt
5. Benutzer erreicht die Arbeitsbühne und die gewünschte Position des Bildes und lässt die Maustaste los, indem er den Drop-Vorgang vornimmt
6. Das System stellt dem Benutzer der Webapplikation das Bild auf der gewünschten Position der Arbeitsbühne dar
7. Die Schritte 2-6 sind auch auf der Arbeitsbühne durchführbar.

Erweiterungen:
4a. Falls die Maustaste losgelassen wird: die Webapplikation stellt dem Benutzer das ziehende Bild auf der ursprünglichen Position dar, wenn die Arbeitsbühne nicht erreicht wurde
5a. Falls der Drag-Vorgang das Ende der Arbeitsbühne erreicht: Die Webapplikation bzw. die Rahmen der Arbeitsbühne lassen den weiteren Drag-Vorgang nicht zu

Fehlerfälle:
Keine

Häufigkeit:
Pro Benutzer in Abhängigkeit von Besucheranzahl

Anforderungen:
Keine

- *Bilder skalieren:* Nachdem ein Bild ausgewählt ist und damit über eine Drag-And-Drop-Funktionalität verfügt, muss es noch aus Benutzerfreundlichkeitsgründen skalierbar sein, was die Aufgabe der Arbeitsbühne ist. Der Anwender hat damit die Möglichkeit, seine Bilder, die eventuell für sein Profil zu groß bzw. klein sind, gemäß seiner Profilgröße zu skalieren.

Titel: **Skalieren der Inventarbilder**
Akteur: **Benutzer**
Ziel: **Ändern der Größe des Inventarbildes**

Auslöser:
Benutzer entscheidet sich zum Ändern der Größe des ausgewählten Inventarbildes bei der Kombinierung der Bilder an seinem Profil auf der Arbeitsbühne (Stage bzw. grafischer Bereich in Form eines Rechtecks)

Vorbedingungen:
4. Benutzer ist eingeloggt

5. Die Webapplikation befindet sich im Zustand ACTIVE
6. Die Webapplikation verfügt über die Inventarbilder des Benutzers

Nachbedingungen:
Benutzer befindet sich in Besitz des skalierten Bildes

Erfolgsszenario:
8. Benutzer navigiert zu Arbeitsbühne (Stage bzw. grafischer Bereich in Form eines Rechtecks)
9. Benutzer navigiert zu Inventarbild auf der Arbeitsbühne, das er skalieren möchte
10. Benutzer wählt das Inventarbild aus, indem er auf das Bild klickt
11. Benutzer navigiert zu Ecken des Bildes
12. Benutzer ändert die Größe des Bildes, indem er mit der Maus die Skalierung vornimmt
13. Das System stellt dem Benutzer der Webapplikation die geänderte Größe des skalierten Bildes dar

Erweiterungen:
5a. Falls die Maustaste losgelassen wird: die Webapplikation stellt dem Benutzer die geänderte Größe des skalierten Bildes dar
5b. Falls die Skalierung das Ende der Arbeitsbühne erreicht: Die Webapplikation bzw. die Rahmen der Arbeitsbühne lassen das weitere Skalieren nicht zu

Fehlerfälle:
Keine

Häufigkeit:
Pro Benutzer in Abhängigkeit von Besucheranzahl

Anforderungen:
Keine

- *Aktuelle Wetterinformation des gewünschten Ortes der nächsten zwei Tage anzeigen lassen:*
  Die Webapplikation bietet einen „Nice-To-Have-Bereich" an, ggf. eine Eingabemaske zur Eingabe des Ortes und Ausgabefeld für die Anzeige des aktuellen Wetters für den aktuellen und nächsten zwei Tage. Dieses Feature unterstützt den Anwender, indem sich der Benutzer gemäß des aktuellen Wetters entsprechend anziehen kann bzw. passende Bekleidung an sich anschauen kann.

Titel: **Laden der Wetterinformation***
Akteur: **Benutzer**
Ziel: **Laden der Wetterinformation von WEATHER-API**

Auslöser:
Benutzer entscheidet sich zum Laden der Wetterinformation, um bei der Anprobe der Kleidung an seinem Profil sich gemäß der aktuellen Wetterinformation anzuziehen

Vorbedingungen:
7. Benutzer ist eingeloggt

8. Die Webapplikation befindet sich im Zustand ACTIVE

Nachbedingungen:
Benutzer befindet sich in Besitz der Wetterinformation

Erfolgsszenario:
14. Benutzer navigiert zu Formular für Wetterinformation
15. Benutzer füllt Formular für Wetterinformation mit den folgenden Daten: Wetter-ID des angefragten Ortes
16. Benutzer schickt das Formular ab
17. Die Webapplikation stellt eine Anfrage (Request) an WEATHER-API
18. Der Web-Service bzw. das Web-API (WEATHER-API) prüft die Benutzerdaten
19. Der Web-Service bzw. das Web-API stellt dem System die Wetterinformation zur Verfügung in Form einer HTTP-Response mit der Wetterinformation im XML-Format
20. Das System stellt dem Benutzer der Webapplikation die Wetterinformation grafisch zur Verfügung

Erweiterungen:
7a. Falls Prüfung die angeforderte Wetter-ID nicht findet: die Webapplikation stellt dem Benutzer keine Wetterinformation grafisch zur Verfügung

Fehlerfälle:
4a. Auf die gestellte Anfrage (Request) an WEATHER-API folgt keine Antwort (Response): Web-Service bzw. Web-API ist nicht erreichbar (OFFLINE) oder ausgelastet

Häufigkeit:
Pro Benutzer in Abhängigkeit von Besucheranzahl

Anforderungen:
Keine

- *Ausgewähltes Bild bearbeiten bzw. ein Polygon im Bild selektieren und abschneiden lassen:*
  Eine weitere Funktionalität aus dem Bereich „Nice-To-Have“ der Webapplikation. Die Webapplikation sollte über ein Bildbearbeitungswerkzeug verfügen, falls sich der Benutzer wünscht, ein Bild zu bearbeiten, wenn dieses Bild das nötig hat bzw. für eine Kombinierung am Profil nicht zulässig ist. Dabei sollte dieses Bildbearbeitungswerkzeug nur folgende Funktionalität anbieten: auf dem Bild ein n-maliges Polygon aufzuzeichnen und beim Abspeichern wird nur das aufgezeichnete Polygon auf dem Bild abgespeichert. Dabei ist die gemachte Änderung nur für die aktuelle Sitzung (Session) aktiv bzw. sie wird nicht persistiert und nach der Operation „Bilder laden“ ist die Änderung nicht zu sehen, da das bearbeitete Bild nicht hochgeladen wurde.

  Titel: **Bildbearbeitung**
  Akteur: **Benutzer**
  Ziel: **Das ausgewählte Bild bearbeiten (Zuschnitte machen)**

  Auslöser:

Benutzer entscheidet sich zum Bearbeiten des ausgewählten Bildes

Vorbedingungen:
1. Benutzer ist eingeloggt
2. Die Webapplikation befindet sich im Zustand ACTIVE
3. Das Bild befindet sich auf der Arbeitsbühne

Nachbedingungen:
Benutzer befindet sich in Besitz des geänderten Bildes

Erfolgsszenario:
1. Benutzer navigiert zu Arbeitsbühne
2. Benutzer klickt mit der rechten Maustaste auf das zu bearbeitende Bild
3. Neben des zu bearbeitenden Bildes erscheint ein Optionsfenster
4. Der Benutzer wählt die Option ‚Bearbeiten'
5. Das zu bearbeitende Fenster erscheint in einem neuen Pop-up-Fenster
6. Der Benutzer klickt mit der linken Maustaste auf die gewünschten Positionen auf dem Bild und bildet somit ein Polygon, das einen gewünschten Zuschnitt darstellt
7. Die markierten Positionen lassen sich mit der Maus anpassen, um eine genauere Auswahl zu ermöglichen
8. Der Benutzer navigiert zu Button ‚Speichern'
9. Der Benutzer klickt das Button ‚Speichern'
10. Das System speichert lokal, solange die Sitzung (Session) gültig ist, den gewünschten Zuschnitt des Bildes und stellt es entsprechend auf der Arbeitsbühne dar

Erweiterungen:
8a. Falls Benutzer das Bearbeitungsmodul schließt: das System übernimmt keine Zuschnitte und stellt den ursprünglichen Zustand des Bildes dar

Fehlerfälle:
Keine

Häufigkeit:
Pro Benutzer in Abhängigkeit von Besucheranzahl

Anforderungen:
Keine

Es wären alle möglichen Bildoperationen für die Bildbearbeitung sinnvoll, aber aus Zeitgründen werden weitere mögliche Bildoperationen in dieser Arbeit nicht behandelt.

Es sind weiterhin Anwendungsfälle vorstellbar, die für die Webapplikation sinnvoll wären. Aus Zeitgründen werden Funktionalitäten der Webapplikation nicht zahlreich entworfen und umgesetzt, sondern nur gemäß der zu erreichenden Zielsetzung aus 1.2 implementiert.

Eine ausführliche Beschreibung der Anwendungsfälle mit Unterscheidungen, die für die Implementierung der Webapplikation nützlich ist, wäre hilfreich. Hier genügt aber eine Zusammenfassung der wesentlichen Punkte, da hinsichtlich der Zielsetzung primär die Res-

sourcenoptimierung von Webapplikationen im Fokus steht. Bei der Verarbeitung der Nachrichten (aus der Benutzersicht) wird wie in 3.2.1 erklärt zunächst die Benutzer-Anwendungslogik ausgeführt, welche dann veranlasst, dass die zugehörige Applikation-Anwendungslogik ausgeführt wird, sofern keine Fehler auftreten. Hier wurden die Anwendungsfälle der Benutzer-Anwendungslogik betrachtet und erarbeitet.
Externe Aufrufe der Applikation-Anwendungslogik an Web Services und Web-APIs, die hier einige Anwendungsfälle veranlassen, werden im Entwurf erarbeitet (s. 4.2.2.3).

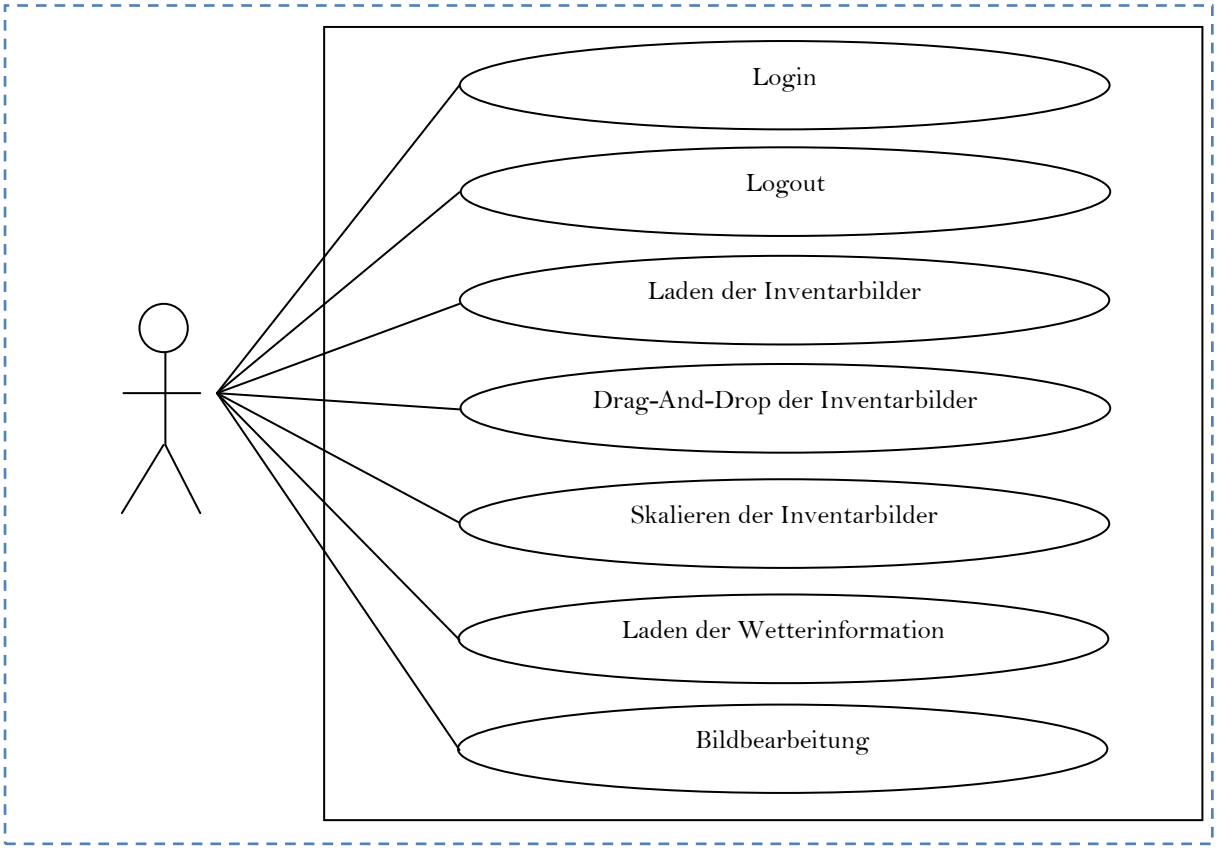

Abbildung 3.3.: Anwendungsfälle

3.3. Nichtfunktionale Anforderungen

Wie in 2.2 bereits erwähnt wurde, sind RIAs Applikationen und haben die gleichen nichtfunktionalen Anforderungen wie jede andere lokale Applikation. Nach (Coenraets 2010) sind bei der Softwareentwicklung folgende wichtige nichtfunktionale Anforderungen zu beachten:

- Robustheit (3.3.1)
- Leistungsfähigkeit bzw. Performanz (3.3.2)
- Skalierbarkeit (3.3.3)

- Verfügbarkeit (3.3.4)
- Portierbarkeit (3.3.5)
- Erweiterbarkeit (3.3.6)
- Benutzbarkeit (3.3.7)
- Vollständigkeit (3.3.8)

Robustheit und Erweiterbarkeit der Webapplikation sind im Hinblick auf Web Services und Web-APIs und Leistungsfähigkeit bzw. Performanz auf erarbeitete Ressourcenoptimierung (2.1.7) besonders wichtig, deswegen werden die einzusetzenden Spezifikationen in 3.3.1, 3.3.6 und 3.3.2 daraufhin untersucht und entsprechende konkrete nichtfunktionale Anforderungen erarbeitet. Insbesondere werden dabei die für diese Arbeit relevanten Problemstellungen der zu implementierenden Webapplikation und verwendeten Technologien (RIA, Web Services und Web-APIs) betrachtet.

3.3.1. Robustheit

Unter Robustheit werden meistens Fehlerbehandlungsmechanismen und kein Vorhandensein von schweren Softwarefehlern gemeint.

Absolute Abwesenheit von Softwarefehlern kann im Rahmen dieser Arbeit aus Zeitgründen sicher nicht erwartet werden.

Das Softwareengineering weist auf, dass ein System hinsichtlich der Robustheit sicher, zuverlässig, wiederherstellbar und verfügbar sein muss. Da Webapplikationen an das Web angewiesen sind, muss dabei für die Applikation-Anwendungslogik besonders der Aspekt Zuverlässigkeit betont und betrachtet werden, weil er am größten zu beeinflussen ist: Die Webapplikation sollte die Zuverlässigkeit in Fehlerfällen unterstützen, in dem ein ausreichendes für den Benutzer Fehler-Handling betrachtet werden sollte (s. 4.6).

3.3.2. Leistungsfähigkeit

Wie bereits mehrmals erwähnt wurde und in 4.3 komplett erarbeitet wird, sind RIA und Delegation an Web Services und Web-APIs die wichtigsten Technologien, die in einem für diese Arbeit ausreichenden Ausmaß behandelten Ressourcenoptimierung verwendet werden. Web Services und Web-APIs werden durch RIA-Komponenten abgebildet, wo „Abhängigkeit“ der kritische Punkt bei Delegation an Web Services und Web-APIs ist.

Die Verarbeitung von Nachrichten der Web Services und Web-APIs, sowohl HTTP-Requests als auch SOAP-Nachrichten, lassen sich sehr effizient mit den vorhandenen Bibliotheken im Framework „Flex 3.2.0“ (Adobe LiveDocs 2009) verarbeiten. Weiterhin sollte nur die nötigste Information aus den Nachrichten extrahiert werden, um der Erhöhung der Leistungsfähigkeit beizutragen.

3.3.3. Skalierbarkeit

Es ist davon auszugehen, dass Webapplikationen aus der Clientsicht (Benutzer) aufgrund ihrer Architektur (2.1.4) relativ zur lokalen Applikationen gut skalieren werden. Übermäßig viele gleichzeitige Zugriffe auf Webapplikationen zu fordern, scheint daher angemessen. Eine moderne DSL-Verbindung aus der Serversicht (Webserver) wird einige (tausend) gleichzeitige Verbindungen sehr gut verkraften, da gemäß der erarbeiteten Ressourcenoptimierung (2.1.7) der Webserver der Webapplikation vollkommen nicht ausgelastet werden soll, da alle ressourcenaufwendige Operationsaufrufe per Delegation an unterschiedliche Web Services und Web-APIs verteilt werden. Mit Hilfe der CLOUD- und LOGIN-APIs wird sich die Größe der Webapplikation im Kilobyte-Bereich bewegen, was für die moderne DSL-Verbindung mit tausend parallelen gleichzeitigen Clientverbindungen zum Webserver kein Auslastungskriterium ist, wenn man dabei noch berücksichtigt, dass alle weiteren Aufrufe clientseitig und verteilt an Web Services und Web-APIs stattfinden. Der Webserver (Hosting-Platz der Webapplikation) ist nur beim ersten Aufruf der Webapplikation an der Client-Server-Kommunikation beteiligt.

Die unterschiedlichen Bandbreiten der einzelnen Netzwerke[5] führen zu unterschiedlich langen Datenlaufzeiten. Die Formulierung von Skalierbarkeitsanforderungen kann vollkommen vernachlässigt werden, da komplette Applikation-Anwendungslogik hinsichtlich der Betrachtung der gleichzeitigen Zugriffe der zu implementierenden Webapplikation an Web Services und Web-APIs delegiert wird.

Es bieten sich auch keine Optimierungspotenziale bezüglich der Laufzeit der Webapplikation (z.B. Betrachtung der sinnvollen Instanziierung der Webapplikation selbst und der verwendeten Web Services und Web-APIs), da RIAs clientseitig „laufen“ und absolut aus der Laufzeitsicht voneinander unabhängig sind und die Cloud-Problematik an Web Services und Web-APIs delegiert werden. Daher kann die Betrachtung solcher Problematiken wie „thread-safe“, „sinnvolle Instanziierung“, „Speicherung“, „Crash-Fehler“ und „Session“ vollkommen vernachlässigt werden.

Bezüglich der kleinen mobilen Geräte gilt das Gleiche wie oben bereits erwähnt.

3.3.4. Verfügbarkeit

Verfügbarkeit ist ein Aspekt in der Softwareentwicklung, der von mehreren nichtfunktionalen Anforderungen und zwar einem Unterpunkt von Robustheit Fehlertoleranz (3.3.1) und Portierbarkeit (3.3.5) abhängt, wenn damit gemeint ist, wie oft und wie lange ein System funktionsfähig ist oder auf welchen Plattformen Applikation sinnvoll einsetzbar und lauffähig ist.

Die Rahmenbedingungen für diese Arbeit bezüglich des Aspektes Portierbarkeit setzen Flash-Player voraus (s. 2.2.1), eine Laufzeitumgebung, die sehr verbreitet ist, sogar auf mobilen Geräten und lokal (als Adobe AIR Laufzeitumgebung). Sowie Flash-Player als auch

---

[5] Der Begriff Netzwerk wird hier im Sinne der Beschreibung des Internets, das alle Netzwerke von der Clientseite bis Serverseite umfasst, gebraucht.

Adobe-AIR-Laufzeitumgebung ist in mehreren Versionen im Web zu finden ist. Ab Flash-Player Version 9 und Adobe-AIR Version 1.5.3 kann die zu implementierende RIA ohne Einschränkungen benutzt werden, siehe dazu Abschnitt 5.1.

3.3.5. Portierbarkeit

Portierbarkeit betrifft im Falle von Applikationen die Fähigkeit, auf verschiedenen Systemen eingesetzt werden zu können, ohne dass sie neu implementiert werden müssen. Da es sich in dieser Arbeit um eine Webapplikation handelt, die aus der Sicht der Benutzbarkeit nur einen Browser mit einem Flash-Player voraussetzt, was zurzeit laut (Flash Player Ubiquität 2010) auf 98% Rechner vorhanden ist, kann die detaillierte Betrachtung dieses Aspektes vernachlässigt werden.

Flash-RIA, am deren Beispiel die erarbeitete Ressourcenoptimierung (2.1.7) implementiert wird, benötigen auf anderen Systemen identische Konfigurationen, also Browser mit dem richtigen Flash-Player als Laufzeitumgebung oder Adobe-AIR-Laufzeitumgebung, wenn man eine Flash-RIA ohne Browser als eine Desktop-Applikation benutzen möchte.

3.3.6. Erweiterbarkeit

Die zu implementierende Webapplikation sollte erweiterbar bezüglich anderer Anwendungskomponenten der Applikation-Anwendungslogik, die eine möglichst einfache Implementierung von Web Services und Web-APIs fordern, sein (s. 4.7). Weiterhin ist eine Erweiterbarkeit der Webapplikation um eventuell weitere Features gefordert. D.h. Erweiterbarkeit soll durch erweiterbaren, leicht verständlichen Code realisiert werden.

3.3.7. Benutzbarkeit

Die Benutzbarkeit einer Webapplikation kann laut Softwareengineering durch geeignete Werkzeuge und intuitiv verständliche Schnittstellen gefördert werden. Ein Beispiel hierfür ist ein Bildbearbeitungswerkzeug zur Bearbeitung von Bildern (z.B. mit einer zusätzlichen Funktion für das Hochladen des bearbeiteten Bildes per CLOUD-API). Im Falle von Flex-Applikationen sind dies Action-Script- und View-Klassen.

Um kompatibel zu benutzten Web Services und Web-APIs der CLOUD-APIs zu sein, benötigen CLOUD-APIs der zu implementierenden Webapplikation gewisse Operationen in Ihrem Interface der Applikation-Anwendungslogik, die zur Einhaltung der Upload-API von benutzten Web Services und Web-APIs beitragen. Dies betrifft alle Webapplikationskomponenten, die Web Services und Web-APIs verwenden.

3.3.8. Vollständigkeit

Die zu implementierende RIA muss vollständig bezüglich des von ihm zu erfüllenden Zwecks sein, um produktiv eingesetzt werden zu können. Im Falle dieser Webapplikation müssen also die Spezifikationen Benutzer- und Applikation-Anwendungslogik in allen Einzelheiten unterstützt bzw. eingehalten werden. Dem Benutzer der Webapplikation (Benut-

zer-Anwendungslogik) müssen entsprechende Bedienungspunkte des UIs (d.h. Möglichkeiten zum Eingreifen in die Bedienung der Webapplikation) zur Verfügung gestellt werden. Die Vollständigkeit dient hier vor allem der Umfassung von allen erarbeiteten Anwendungsfällen (3.2.2).

3.4. Fazit

Die zu entwerfende Webapplikation ist eine Implementierung der Rich Internet Applikation mit RIA-Framework „Flex“ aus (4.4), die Komponenten, Interfaces und Klassen enthält und Funktionalität bezüglich der in 3.1 und 3.3 erarbeiteten fachlichen funktionalen und nichtfunktionalen Anforderungen bereitstellt. Dazu gehören:

- Eine in 1.2 beschriebene Rich Internet Applikation mit in 3.1 und 3.3 erarbeiteten fachlichen funktionalen und nichtfunktionalen Anforderungen, die folgende konkrete APIs (Webapplikationskomponenten) implementieren sollte:
- Ein LOGIN-API,
- Ein CLOUD-API,
- Ein WEATHER-API,
- Ein ANALYTICS-API,
- Ein CUSTOMIZATION-API.

Ein weiterer möglicher zu entwerfender Teil der Webapplikation sind:

1) erweiterbare HTTP-Services bzw. Web Services (s. 4.7), die den o.g. mehreren HTTP-Services entsprechen und für jeden zu implementierenden Web Service die Rolle des Dienstes und die erforderlichen Daten für die Webapplikationslogik (3.2.1) erstellen

2) eine Bildbearbeitungskomponente bzw. ein BILDBEARBEITUNG-API, die zur Erhöhung der Benutzerfreundlichkeit beiträgt und für die Bearbeitung (Selektion eines Bereichs im Bild) der Benutzerbilder zuständig ist.

Der erste Teil ist unbegrenzt für weitere Dienste erweiterbar und unterstützt dadurch das Erweiterbarkeitskonzept der zu implementierenden Webapplikation um weitere Web Services und Web-APIs. Aus Zeitgründen werden erweiterbare HTTP-Services nicht zahlreich entworfen und umgesetzt, sondern nur gemäß der zu implementierenden Diensten aus (4.3) implementiert.

Bezüglich der nichtfunktionalen Anforderungen werden im Rahmen dieser Arbeit spezielle Sicherheiten der Komponenten (APIs) nur zur Förderung der Robustheit (Fehlertoleranz), Leistungsfähigkeit, Erweiterbarkeit (HTTP-Services-Komponente bzw. Web Services und

Web-APIs-Komponenten) und Benutzbarkeit (UI) (s. Kapitel 4) entworfen und umgesetzt. Es wird außerdem darauf geachtet, die Webapplikation so zu entwerfen, dass bezüglich der übrigen nichtfunktionalen Anforderungen die Implementierung effizient ist, um die Wartezeit der langen Übertragungen von großen Daten so weit wie möglich zu kompensieren.

Wenn möglich, werden schnell zu implementierende vorhandene Techniken des RIA-Frameworks „Flex“ (Adobe LiveDocs 2009) wiederverwendet und nur in Ausnahmefällen verändert bzw. erweitert.

4. Entwurf

Nachdem im vorangehenden Kapitel die zu implementierende Rich Internet Applikation aus fachlicher Sicht und darauf aufbauend funktionale und nichtfunktionale Anforderungen formuliert wurden, werden in diesem Kapitel Muster und Techniken ermittelt bzw. erarbeitet, um die Anforderungen umzusetzen und die ebenfalls in der Analyse zum kleinen Teil beschriebenen APIs zu implementieren.

Es wird zunächst ein Überblick über die zu implementierende Webapplikation gegeben (4.2), in dem noch einmal die wesentlichen Anforderungen mit Ergänzungen zum Analyseteil, die nicht komplett erarbeitet wurden und eher zum Entwurf gehören, und Webapplikationskomponenten der zugrunde liegenden RIA (1.2) kurz beschrieben bzw. erarbeitet werden. Dann werden die neuen zu implementierenden Komponenten der Webapplikation ausführlich beschrieben (4.1, 4.5). Es werden Muster zur Umsetzung der Spezifikationen, Anforderungen und APIs aus Kapitel 3 ausgewählt und die mögliche Umsetzung durch Klassen-, Inter- und Sequenzdiagramme verdeutlicht. Das Fehler-Handling wird als allgemeiner, komponentenübergreifender Aspekt in 4.6 diskutiert.

Alle Diagramme in diesem Kapitel sind dabei eigene Darstellungen und sind wie schon in Kapitel 3 in einer an die UML angelehnten Notation gehalten.

Nicht weiter eingegangen wird auf all jene grundlegenden Themen im Zusammenhang mit der Implementierung von RIAs, die in Kapitel 2 und hingewiesener Literatur behandelt werden. Dazu gehören u.a. Webapplikationen, Domänendienstleistungen, Ressourcen von Webapplikationen, Transportprotokolle HTTP (HTTP 2010), XML (XML 2010), SOAP (SOAP 2010) sowie Web Services und Web-APIs (Web Services 2010).

4.1. Architektur

Nach den Überlegungen über die Anforderungen der zu implementierenden Rich Internet Applikation stellt sich jetzt folgende logische Frage: „Was ist der richtige Implementierungsweg aus der Architektursicht für eine große, skalierbare und moderne Flex-Applikation“. Nach 2.2 gibt es keinen Grund, auf die geprüften Softwareengineerings- Methoden und Praktiken zu verzichten, die bereits im Laufe der letzten Jahre für die qualitative Softwareentwicklung sorgen.

Nach (Coenraets 2010) kann die Rich Internet Applikation, nachdem alle Anforderungen dieser Applikation betrachtet und analysiert wurden, in der gleichen Weise wie bei jeder anderen Objekt-Orientierte-Programmiersprache (OOP) modelliert (in dem OO-Sinn dieses Terms) und implementiert werden. D.h. eine Flex-Applikation wird wie jede andere Objekt-Orientierte-Applikation gebildet. Eine wichtige Aufgabe an dieser Stelle der Entwicklungslebenszyklus ist die Identifizierung der Klassen im System. Bei diesem Prozess ist es eine

gute Idee, die geprüften Design-Patterns wie MVC-Architektur, zu verwenden. Dabei werden die Usability-Bequemlichkeit der Applikation und die Wiederverwendung derer Klassen unterstützt und verbessert.

Dieser Abschnitt und somit das Konzept der zu implementierenden Rich Internet Applikation basiert im Folgenden auf einem bekannten Sound-Design-Pattern „Model-View-Controller-Architektur (MVC)“ und einer der besten Praktiken für Applikationskomponenten „Lose Kopplung“.

Bei der Benutzung der MVC-Architektur wird das System in drei Klassenkategorien aufgeteilt:

- Klassen, die Daten und deren Methoden kapseln (Model)
- Klassen, die für die Benutzerschnittstelle (UI) der Applikation zuständig sind (View)
- Klassen, die für die Logik der Applikation zuständig sind (Controller)

Lose Kopplung der Applikationskomponenten ist eine andere geprüfte beste Praktik in der objektorientierten Softwareentwicklung. Lose Kopplung ist eine Programmiertechnik, die die Abhängigkeiten zwischen den Klassen so viel wie möglich vermeidet. Sie verbessert somit die Wiederverwendbarkeit aller Komponenten innerhalb und außerhalb der Applikationen.

Die Grundidee besteht darin, dass je mehr eine Komponente von einer anderen abhängig ist, desto weniger wiederverwendbar sie ist. Mit anderen Worten, wenn Komponente A eine Komponente B referenziert (eine typisierte Variable), dann kann die Komponente A nur mit der Komponente B wiederverwendet werden. Diese Art der Abhängigkeit ist in der Regel nicht wünschenswert.

Man kann häufig die direkten Referenzen auf die anderen Komponenten bei der Verwendung der Ereignisnotifikation (event notification) zwischen Komponenten vermeiden. Beispielsweise eine View-Klasse hat ein Button, der dem Benutzer der Applikation irgendein Ereignis initiieren ermöglicht. Trotzdem hat die View-Klasse keine Kenntnis über die Button-Ereignis-Klasse. Wenn Benutzer auf diesen Button klickt, das Objekt der View-Klasse löst ein Ereignis der Button-Ereignis-Klasse. Die Applikation selbst ist als ein Listener für solche Ereignisse registriert und stellt die View-Klasse der Button-Ereignis-Klasse dar.

Der traditionelle Vorteil der zu verwendeten MVC-Architektur und besten Praktik „Lose Kopplung“ ist die Partitionierung der Applikation, die den duplizierten Code vermeidet und die Wiederverwendbarkeit der Applikationskomponenten verbessert. Mit anderen Worten, zwei verschiedene Views produzieren zwei verschiedene Wege zum Aussehen desselben Datums ohne duplizierten Code in der Modellogik. Man kann leicht die Anzahl der View-Klassen, die dasselbe Datum ohne Duplizierung des Models repräsentieren, beliebig erweitern.

Im Folgenden werden die fachlichen und technischen Architekturen beschrieben.

4.1.1. Fachliche Architektur

Die zu implementierende Webapplikation ist in Abb. 4.1 dargestellt.

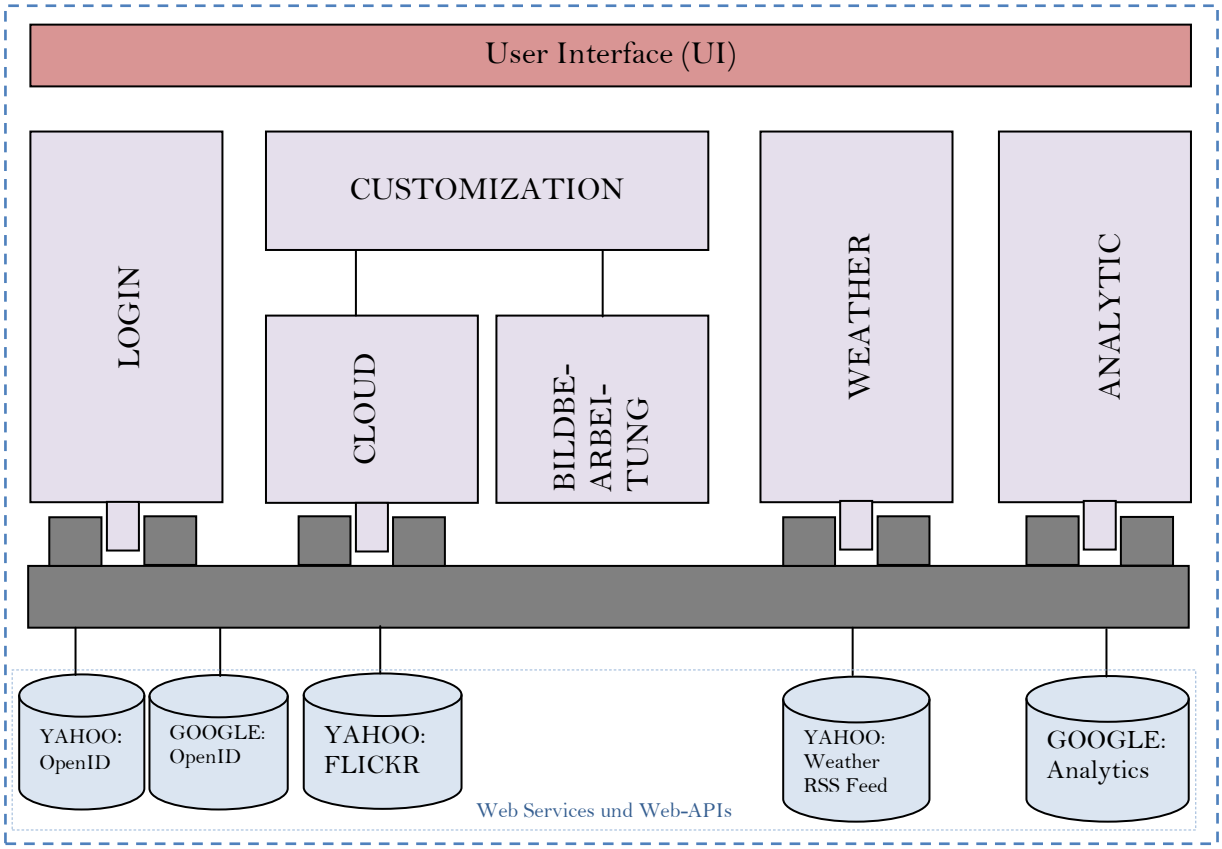

Abbildung 4.1.: Fachliche Architektur

Es folgt eine kurze Beschreibung der fachlichen Architektur. Die Webapplikationskomponenten werden dann im weiteren Abschnitt 4.5 ausführlich beschrieben.

User Interface: Alle Webapplikationskomponenten im Zusammenhang mit der Benutzer-Anwendungslogik stellen dem Benutzer die Benutzerschnittstelle zur Verfügung.

CUSTOMIZATION: Diese Komponente verwaltet die grundsätzlichen Funktionalitäten hinsichtlich der Möglichkeit, dem Benutzer sein Profil anziehen bzw. mit seinen Inventarbildern gestalten zu lassen.

LOGIN: Diese Komponente verwaltet die Login- und Logout-Funktionalitäten (s. 4.5.1), die hinsichtlich der Applikation-Anwendungslogik an die OpenID Web Services und Web-APIs von YAHOO und GOOGLE delegiert.

CLOUD: Diese Komponente verwaltet die Cloud-Funktionalitäten (s. 4.5.2), die hinsichtlich der Applikation-Anwendungslogik an die Cloud Web Services und Web-APIs von YAHOO delegiert. Sie ist für die Persistenz der Inventarbilder zuständig.

WEATHER: Diese Komponente verwaltet die Wetter-Funktionalitäten (s. 4.5.3) und ist somit zuständig für die Anzeige des aktuellen Wetters. Sie delegiert an die Weather Web Services und Web-APIs von YAHOO.

ANALYTIC: Diese Komponente ist zuständig für das Web-Tracking (s. 4.5.4) und delegiert an die Analytics Web Services und Web-APIs von GOOGLE.

4.1.2. Technische Architektur

Die in 4.1.1 beschriebene fachliche Architektur der zu implementierenden Webapplikation wird hier aus der technischen Sicht beschrieben. Die Zielplattform ist dabei wie in 2.2.1 beschrieben und in 4.4 vorgegeben wird Flash Player in der Ausprägung Flash-Laufzeitumgebung 9.0.124 oder höhere Versionen.

Die zu implementierende Webapplikation ist im Komponentendiagramm in Abb. 4.2 dargestellt.

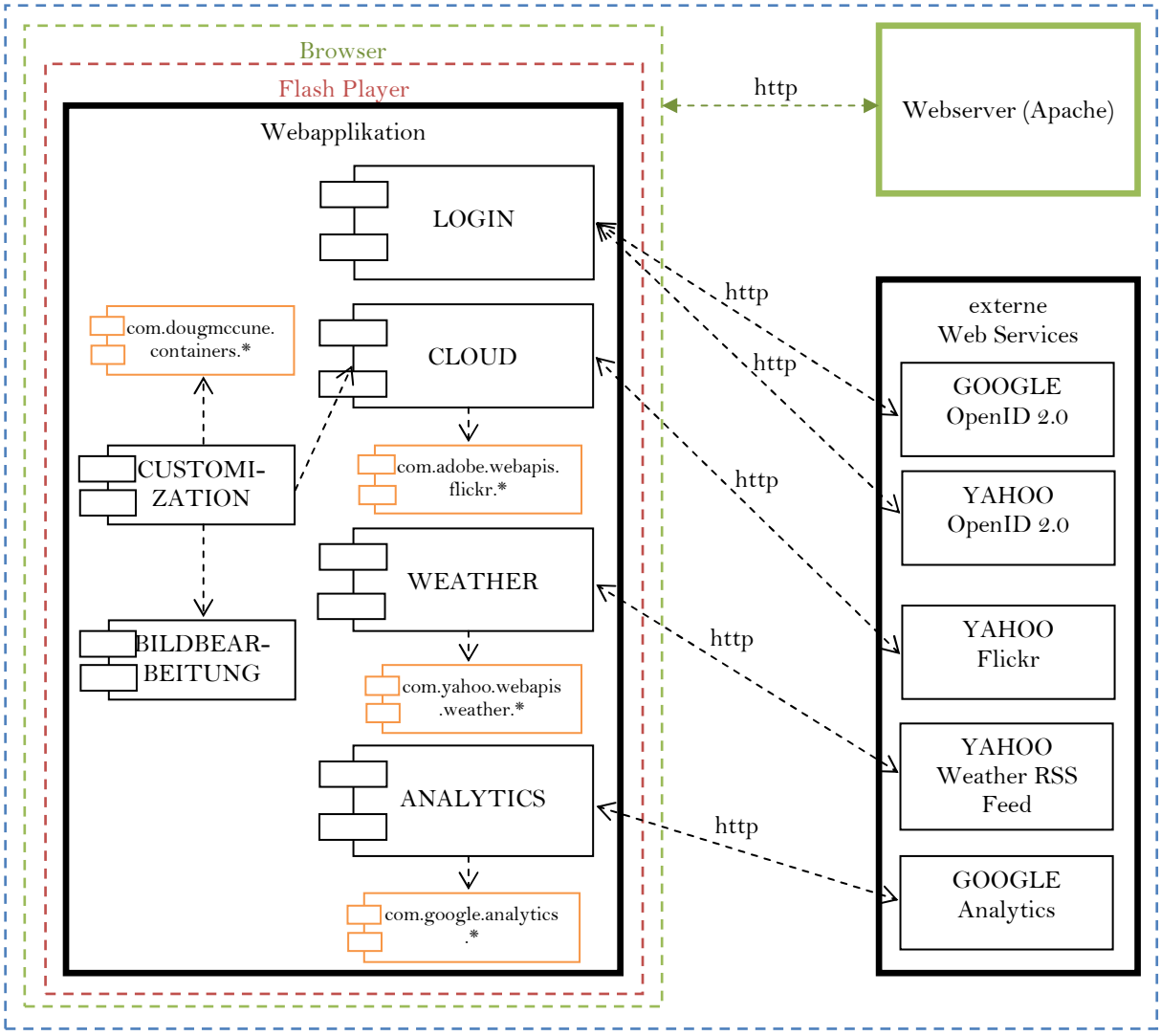

Abbildung 4.2.: Komponentendiagramm

Es folgt eine kurze Beschreibung aller Komponenten. Die Komponenten werden dann, wie bereits angedeutet, im weiteren Abschnitt 4.5 ausführlich beschrieben.

com.dougmccune.containers.* : Diese externe Bibliothek (Customization Library) ist eine der Grundlagen für die CUSTOMIZATION-Komponente.

com.adobe.webapis.flickr.*: Diese externe Bibliothek (Cloud Library) ist die Grundlage für die CLOUD-Komponente.

com.yahoo.webapis.weather.* : Diese externe Bibliothek (Weather Library) ist die Grundlage für die WEATHER-Komponente.

com.google.analytics.* : Diese externe Bibliothek (Analytics Library) ist die Grundlage für die ANALYTICS-Komponente.

LOGIN : Diese Komponente übernimmt die Logik der Webapplikation, die für das Einloggen und Ausloggen der Benutzer zuständig ist. Sie arbeitet Web Service-Aufrufe gemäß GOOGLE OpenID 2.0 (GOOGLE OpenID 2010) und YAHOO OpenID 2.0 (YAHOO OpenID 2010) ab.

CLOUD : Diese Komponente übernimmt die Logik der Webapplikation, die für das Darstellen und Abrufen der Inventarbilder der Benutzer zuständig ist. Sie arbeitet Web Service-Aufrufe gemäß YAHOO Flickr (YAHOO Flickr 2010) ab.

WEATHER : Diese Komponente übernimmt die Logik der Webapplikation, die für das Darstellen und Abrufen der aktuellen Wetterinformationen für einen Ort zuständig ist. Sie arbeitet Web Service-Aufrufe gemäß YAHOO Weather RSS Feed (YAHOO Weather 2010) ab.

ANALYTICS : Diese Komponente übernimmt die Logik der Webapplikation, die für das Web-Tracking der Webapplikation zuständig ist. Sie arbeitet Web Service-Aufrufe gemäß GOOGLE Analytics (GOOGLE Analytics 2010) ab.

CUSTOMIZATION : Diese Komponente übernimmt die Logik der Webapplikation, die für die Gestaltung der Arbeitsbühne zuständig ist. Im Bereich dieser Arbeitsbühne kann der Benutzer seine zur Verfügung stehenden Inventarbilder an seinem Profil kombinieren lassen bzw. sein Profil anziehen lassen.

BILDBEARBEITUNG : Diese Komponente übernimmt die Logik der Webapplikation, die für Bearbeitung der Inventarbilder zuständig ist. Im Bereich dieser Bildbearbeitung kann der Benutzer das ausgewählte Inventarbild folgendermaßen bearbeiten: das gewünschte bzw. selektierte Polygon ausschneiden lassen.

Flash Player : Laufzeitumgebung für die Webapplikation (Plug-In).

Browser : Laufzeitumgebung für den Flash Player.

Webserver (Apache) : Server, der die Webapplikation im Webrahmen (Internet) zur Verfügung stellt.

Externe Web Services : Web Services und Web-APIs, an die die o.g. Komponenten delegieren.

4.2. Allgemeine Beschreibung der Komponenten

4.2.1. Logik der Webapplikation als Zustandsautomat

Die Benutzer- und Applikation-Anwendungslogik können als Zustandsautomaten beschrieben werden. Die Zustände der Webapplikation sind dabei jeweils aus Sicht des Benutzers zu behandeln. Die unten aufgezählten Events (Ereignisse) bzw. Nachrichten verändern den Zustand entsprechend.

In den folgenden Abschnitten werden Nachrichten (aus Sicht des Benutzers) bzw. Events (aus Sicht der Webapplikation) und Zustände der Webapplikation beschrieben. Nachrichten bzw. Events aus der Sicht des Frameworks „Flex“ bzw. Flash können vernachlässigt werden, da diese nicht von Interesse sind. Eine detaillierte Beschreibung der Nachrichten in der Technologie Flash ist in (Adobe wiki 2010) und (Adobe LiveDocs 2009) zu finden.

4.2.1.1. Events der Benutzer-Anwendungslogik

Wie in 3.1 bereits erwähnt, müssen Funktionalitäten bezüglich der externen Aufrufe (Web Services und Web-APIs) der Applikation-Anwendungslogik (s. 4.2.2) besonders betrachtet werden.
Im Rahmen der zu implementierenden Webapplikation werden folgende Nachrichten bzw. Events bezüglich der Benutzer-Anwendungslogik definiert und behandelt:

Login: Aufforderung des Benutzers an die Webapplikation (LOGIN-API), einen Login-Vorgang zu starten und den Zustand der Webapplikation zu ändern.

LoginResponse: Antwort der Webapplikation (LOGIN-API) an den Benutzer. Die Webapplikation geht in den Zustand ACTIVE (s.u.) über, wenn die aufgerufene Operation (Login-Vorgang) bzw. das Redirect von dem angesprochenen Web Service und Web-API erfolgreich war.

Cloud: Aufforderung des Benutzers an die Webapplikation (CLOUD-API), einen Cloud-Vorgang zu starten und die Webapplikation in den Zustand CLOUD überzugehen. Dabei übergibt diese Nachricht die vom Benutzer eingegebene Benutzer-ID der angesprochenen CLOUD-API an die Webapplikation.

CloudResponse: Antwort der Webapplikation (CLOUD-API) an den Benutzer. Die Webapplikation geht in den Zustand CLOUD über, indem Bilder, die unter der aufgeforderten Benutzer-ID in Cloud Web Services und Web-APIs gespeichert sind, dem Benutzer zur Verwaltung gestellt werden. In diesem Zustand ist es aus Benutzersicht möglich, die Verwaltung des Inventars des jeweiligen Benutzers zu verwalten.

Weather: Aufforderung des Benutzers an die Webapplikation (WEATHER-API), einen Weather-Vorgang zu starten und den Zustand der Webapplikation in den Zustand WEA-

THER zu ändern. Dabei übergibt diese Nachricht die Postleitzahl an die Webapplikation der angesprochenen Web Services und Web-APIs, die Benutzer eingegeben hat.

WeatherResponse: Antwort der Webapplikation (WEATHER-API) an den Benutzer. Die Webapplikation geht in den Zustand WEATHER über, indem Wetterinformationen, die unter der aufgeforderten PLZ von den Web Services und Web-APIs dem Benutzer zur Verfügung gestellt werden. In diesem Zustand ist es aus Benutzersicht möglich, die Verwaltung des Inventars des jeweiligen Benutzers bezüglich des aktuellen Wetters zu verwalten.

Die o.g. Nachrichten sowie Zustände der Webapplikation sind asynchron und unabhängig, wenn die Webapplikation im Zustand ACTIVE ist und einen externen Aufruf verursacht. D.h. die Webapplikation kann in einen beliebigen Zustand, unabhängig davon in welchem Zustand sie gerade ist, übergehen.
Ausführliche Betrachtung von Nachrichten bzw. Events bezüglich der Benutzer-Anwendungslogik, die für die Verwaltung des Inventars selbst des angemeldeten Benutzers zuständig sind, können vernachlässigt werden, da diese hinsichtlich der erarbeiteten Ressourcenoptimierung (2.1.7) im Rahmen dieser Arbeit nicht von Bedeutung sind.

4.2.1.2. Zustände der Benutzer-Anwendungslogik

Benutzer-Anwendungslogik definiert folgende Zustände während der Interaktivität zwischen Webapplikation und Benutzer. Die Zustandsnamen wurden dabei sinngemäß gewählt:

INACTIVE: Benutzer ist nicht eingeloggt.

ACTIVE: Benutzer hat sich erfolgreich gemäß des ausgewählten Login Web Services und Web-APIs eingeloggt und wurde erfolgreich ‚redirected'.

CLOUD: Webapplikation stellt die angeforderten Inventarbilder dem Benutzer zur Verfügung.

WEATHER: Webapplikation stellt die angeforderten Wetterinformationen dem Benutzer zur Verfügung.

4.2.1.3. Zusammenfassung

Alle möglichen Zustandsübergänge von Webapplikation werden für die Benutzer-Anwendungslogik in Abb. 4.3 zusammengefasst.

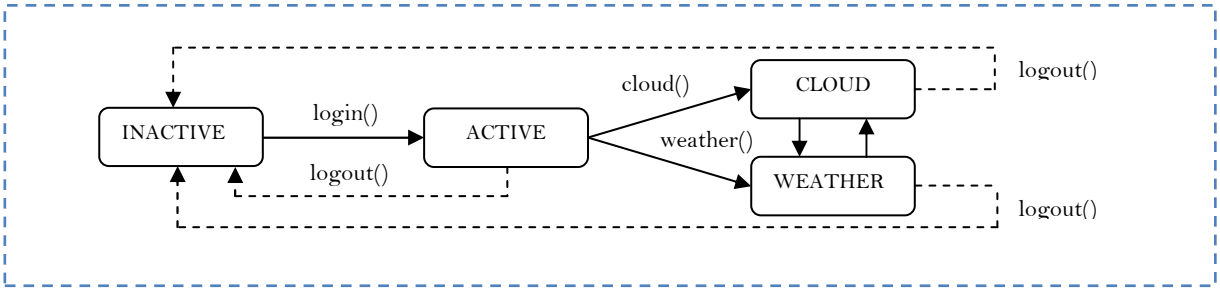

Abbildung 4.3.: Zustandsübergänge für Benutzer-Anwendungslogik (eigene Darstellung)

4.2.2. Events der Applikation-Anwendungslogik

Die Applikation-Anwendungslogik fasst alle externen Aufrufe der Webapplikation um und bildet allgemein lokal eine Schnittstelle (Interface) der zu implementierenden Web Services und Web-APIs ab, die man mit einem Überbegriff „Adapter“ bezeichnen kann. Da im Allgemeinen in dieser Logik eine Weiterleitung an die angesprochenen Dienste und Benutzer-Anwendungslogik und die automatische (ohne Kenntnis des Benutzers) Einhaltung inklusive Vervollständigung der externen Benutzernachrichten technisch stattfindet (3.2.1) und von den fachlichen Anforderungen vollkommen abstrahiert werden kann (irrelevant aus der fachlichen Sicht), wird die Behandlung der Nachrichten bzw. Events der Applikation-Anwendungslogik im Entwurfsteil (Kap. 4) detailliert aufgegriffen.

Im Rahmen der zu implementierenden Webapplikation werden folgende Nachrichten bzw. Events bezüglich der Applikation-Anwendungslogik definiert und behandelt:

Login: Weiterleitung (Redirect) der Webapplikation (LOGIN-API) an den Login-Logout-Dienst in Form einer HTTP-Anfrage (Request), einen Login-Vorgang zu starten.

LoginResponse: Redirect des Login-Logout-Dienstes an die Webapplikation (LOGIN-API) in Form einer HTTP-Anfrage (Request). Die Webapplikation geht in den Zustand ACTIVE (s.u.) über, wenn die aufgerufene Operation (Login-Vorgang) erfolgreich war.

Cloud: Aufforderung der Webapplikation (CLOUD-API) an den Cloud-Dienst in Form einer HTTP-Anfrage (Request) nach den Bildern-URLs bezüglich der Benutzer-ID. Dabei übergibt diese Nachricht die Benutzer-ID an den Cloud-Dienst der angesprochenen Cloud-API, die Benutzer eigegeben hat.

CloudResponse: Antwort des Cloud-Dienstes an die Webapplikation (CLOUD-API) in Form einer HTTP-Antwort (Response). Die Webapplikation geht in den Zustand CLOUD über, indem Bilder, die unter der aufgeforderten Benutzer-ID in Cloud Web Services und Web-APIs gespeichert sind, dem Benutzer per asynchrone HTTP-Requests der Bilder-URLs zur Verwaltung gestellt werden. In diesem Zustand ist es aus Benutzersicht möglich, die Verwaltung des Inventars des angemeldeten Benutzers zu verwalten.

Weather: Aufforderung der Webapplikation (WEATHER-API) in Form einer HTTP-Anfrage (Request) an den Weather-Dienst nach der aktuellen Wetterinformation. Dabei übergibt diese HTTP-Nachricht die vom Benutzer eigegebene Postleitzahl an den angesprochenen Weather-Dienst.

WeatherResponse: Antwort des Weather-Dienstes in Form einer HTTP-Antwort (Response) an die Webapplikation (WEATHER-API). Die Webapplikation geht in den Zustand WEATHER über, indem die Wetterinformationen, die unter der aufgeforderten PLZ von den Weather Web Services und Web-APIs der Webapplikation im XML-Format zur Verfügung gestellt werden.

Analytics: Die Webapplikation sendet Web-Tracking-Daten in Form einer HTTP-Anfrage (POST-Request) an den analytischen Dienst.

Diese Nachrichten bzw. Events bezüglich der Applikation-Anwendungslogik, die die o.g. Webapplikationskomponenten auslösen, bilden die verwendeten Web Services und Web-

APIs ab. Wie in 3.2.1 bereits angedeutet, fassen sie die Kommunikation zwischen Webapplikation und verwendeten Web Services und Web-APIs um (Abb. 3.2) und bilden als Abbildungskomponenten einen Adapter („Middleware“) der Webapplikation, der das Marshalling der Kommunikation, Information und persistierten Daten übernimmt.

4.2.2.1. Zustände der Applikation-Anwendungslogik

Applikation-Anwendungslogik definiert folgende Zustände während der Kommunikation zwischen Webapplikationskomponenten und angesprochenen Web Services und Web-APIs:

INACTIVE: Benutzer ist nicht eingeloggt.

ACTIVE: Benutzer hat sich erfolgreich gemäß des ausgewählten Login Web Services und Web-APIs eingeloggt und wurde erfolgreich von diesem Dienst ‚redirected‘.

CLOUD: Cloud Web Service und Web-API stellt auf die Anfrage die angeforderten Inventarbilder in Form einer HTTP-Responce (Antwort) der Webapplikation zur Verfügung.

WEATHER: Wetter Web Service und Web-API stellt auf die Anfrage die angeforderten Wetterinformationen in Form einer HTTP-Responce (Antwort) der Webapplikation zur Verfügung.

ANALYTICS: Die Webapplikation sendet Web-Tracking-Daten in Form einer HTTP-Anfrage (POST-Request) an den analytischen Dienst.

4.2.2.2. Zusammenfassung

Alle möglichen Zustandsübergänge von Webapplikation werden für die Applikation-Anwendungslogik in Abb. 4.4 zusammengefasst.

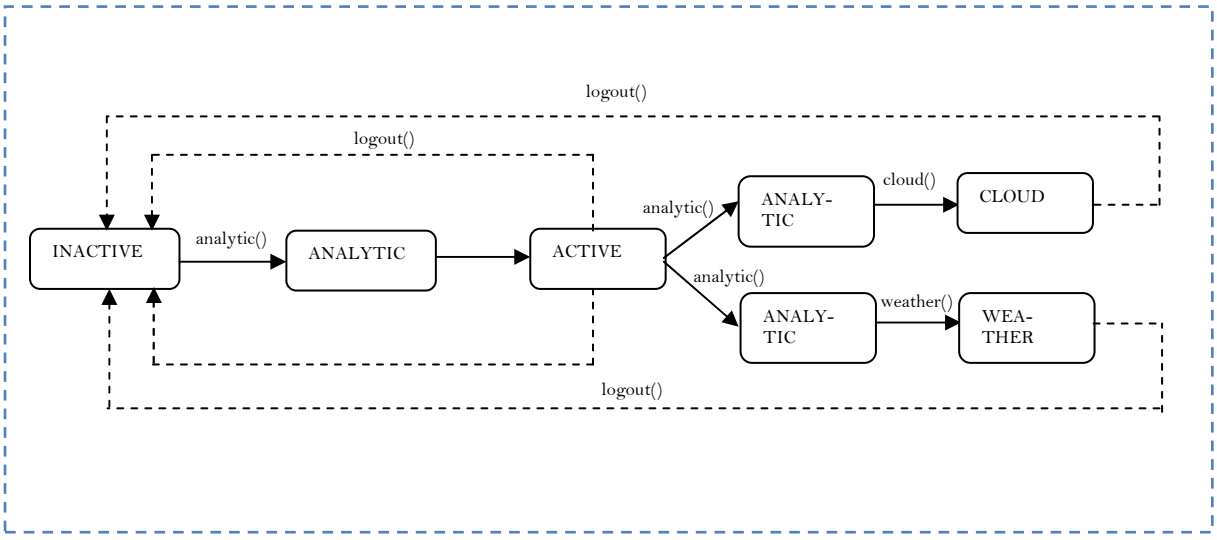

Abbildung 4.4.: Zustandsübergänge für Applikation-Anwendungslogik (eigene Darstellung)

4.2.2.3. Externe Aufrufe der Webapplikation

Nachdem die einzelnen Nachrichten bzw. Ereignisse der Applikation-Anwendungslogik und die möglichen Zustandsübergänge der Webapplikation im vorigen Abschnitt beschrieben wurden, muss noch geklärt werden, welche Verantwortungen Benutzer- und Applikation-Anwendungslogik dabei haben. Damit die zu implementierende Webapplikation dem Benutzer und Webentwickler so viel Benutzerfreundlichkeit wie möglich anbietet und so viel Arbeit wie möglich abnehmen kann, sollte der Applikation-Anwendungslogik soviel Verantwortung wie möglich gegeben werden. Dies trägt auch zur Vermeidung von Fehlern aus der Sicht des Benutzers bei.

Es geht also um die Semantik der Nachrichten, jeweils auf Benutzer- und auf Applikation-Anwendungsseite. Die Semantik variiert dabei mit den Anwendungsfällen, für die die Webapplikation benutzt wird.

Als „externe Aufrufe“ werden hier für die Webapplikation die acht in Tabelle 4.5 aufgeführten Kombinationen aus Webapplikationszustand, Nachricht (aus Sicht des Benutzers) sowie Zustand der Web Services und Web-APIs betrachtet. Je nach Kombination ergeben sich für Benutzer- und Applikation-Anwendungslogik unterschiedliche Verantwortlichkeiten hinsichtlich der Einhaltung der Nachrichten der Benutzer- und Applikation-Anwendungslogik.

Bei der Verarbeitung der Nachrichten (aus der Benutzersicht) wird wie in 3.2.1 erklärt zunächst die Benutzer-Anwendungslogik ausgeführt, welche dann veranlasst, dass die zugehörige Applikation-Anwendungslogik ausgeführt wird, sofern keine Fehler auftreten. Für den Entwurf des Zusammenspiels dieser beiden Logik empfiehlt es sich, Meta-Patterns zu benutzen (siehe auch das entsprechende Kapitel zum Entwurf von Webapplikationskomponenten 4.1, 4.5).

Webapplikationszustand	Nachricht	Dienst-Antwort
INACTIVE	login()	nicht erfolgreich
INACTIVE	login()	erfolgreich
ACTIVE	cloud()	nicht erfolgreich
ACTIVE	cloud()	erfolgreich
ACTIVE	weather()	nicht erfolgreich
ACTIVE	weather()	erfolgreich
ACTIVE	analytic()	nicht erfolgreich
ACTIVE	analytic()	erfolgreich

Tabelle 4.5.: externe Aufrufe

Prinzipiell muss zwischen Operationen (Nachrichten) des Benutzers und von den mit dem Benutzer zusammenhängenden Ereignisse und Operationen der Applikation unterschieden werden.

Der Kommunikationsaufbau (Aktivierung bzw. Hand-Shake), die Kommunikation selbst bezüglich der aufgerufenen Operationen der Dienste und Terminierung zwischen Webapplikation und Web Services und Web-APIs sollten komplett von der Applikation-Anwendungslogik übernommen werden, sobald eine Operation aus der Sicht des Benutzers bzw. der Benutzer-Anwendungslogik aufgerufen wird. Die Benutzer-Anwendungslogik hat dagegen eigene Verantwortlichkeiten und übernimmt die Rolle eines Koordinators, der eine manuelle Schnittstelle (UI) nach außen für den Benutzer darstellt.

Die Aufgabe jedes Benutzers legt zusammen mit dem Koordinator bzw. dessen Aufgabe den Nachrichtentyp fest. Der Nachrichtentyp ruft die zuständige Webapplikationskomponente (4.5) des aufgerufenen Web Services und Web-APIs auf, übergibt den vom Benutzer übergebenen Parameter und entscheidet bei der Terminierung, ob der Aufruf erfolgreich war und leitet die Information an den Benutzer weiter oder ob ein Fehler vorliegt, um eine Fehlermeldung an den Benutzer weiterzuleiten.

Die Benutzer-Anwendungslogik muss beim Empfangen von Nachrichten aus lokaler bzw. Benutzersicht und die Applikation-Anwendungslogik aus der Sicht der aufgerufenen Web Services und Web-APIs prüfen, ob die Nachricht im aktuellen Zustand gültig ist. Ist dies nicht der Fall, so muss ein Fehler protokolliert werden und der Aufrufer über den Fehler informiert werden (Benutzer).

4.3. Web Services und Web-APIs der Webapplikation

Wie in 2.3 bereits herausgestellt wurde, haben sich Web Services und Web-APIs bei der Optimierung der Betriebs-, Implementierungskosten und Performanz von Webapplikationen im Zusammenhang mit RIA als Lösung für die im Rahmen dieser Arbeit erarbeitete Ressourcenoptimierung bewährt.

Es gibt zahlreiche passende Web Services und Web-APIs im Web, die die Anforderungen der zu implementierenden RIA erfüllen können. Im Rahmen dieser Arbeit werden Web Services und Web-APIs der Anbieter GOOGLE (GOOGLE APIs 2010) und YAHOO (YAHOO APIs 2010) benutzt, da sie eine qualitative Dokumentation, zuverlässige genügende Bandbreite, Performanz, qualitative Flash-APIs und eine sehr große Akzeptanz der Webentwickler haben.

Es werden folgende beispielhafte kostenlose Web Services und Web-APIs von GOOGLE und YAHOO behandelt bzw. im Rahmen dieser Arbeit am Beispiel einer RIA (1.2) hinsichtlich der erarbeiteten Anforderungen (3.1, 3.3) und Anwendungsfälle (3.2.2) verwendet:

- YAHOO Web Services und Web-APIs
    1. Flickr Cloud API* (YAHOO FLICKR 2010)
    2. Yahoo Authentication & Sign In* (YAHOO OPENID 2010)
    3. Yahoo Weather RSS Feed* (YAHOO WEATHER 2010)

- GOOGLE Web Services und Web-APIs

    1. GOOGLE ACCOUNT API* (GOOGLE OPENID 2010)
    2. PICASA WEB ALBUMS DATA API (GOOGLE PICASAWEB 2010)
    3. GOOGLE ANALYTICS* (GOOGLE ANALYTICS 2010)

Die o.g. Web Services und Web-APIs von GOOGLE bezeichnet man als GOOGLE-DATA-APIs.

Mit Hilfe von Softwarearchitektur wird es eine komponentenbasierte Implementierung der Delegation an diese Dienste angestrebt, damit die Webapplikation leicht für andere Anbieter der o.g. Web Services und Web-APIs erweiterbar wird (3.3.6).

Am Beispiel einer Rich Internet Applikation mit Hilfe von o.g. Web Services und Web-APIs wird es durch eine Delegation an diese Dienste angestrebt, die in 2.1.7 erarbeitete Ressourcenoptimierung zu erzielen. Dadurch werden aufwendige Ressourcen und wiederkehrende, zeitanspruchsvolle Implementierungsarbeiten der Webapplikation an die Web Services und Web-APIs ausgelagert (Betriebskosten-, Performanz- und Implementierungskostenoptimierung) und durch die RIA-Technologie ein relativ sehr kleinerer Client-Webserver-Datenverkehr erreicht, der die Domänenkosten sehr deutlich reduziert (Betriebskostenoptimierung). Die im Rahmen dieser Arbeit erarbeitete Betriebskostenoptimierung ist nur mit Hilfe von RIA-Technologie erreichbar, da diese ein clientseitiges Kommunikationsverfahren voraussetzt und nur so ein Verfahren das Umgehen aller Anfragen an den Webserver (Hosting-Platz der Webapplikation) ermöglicht.

Eine Rich Internet Applikation und zwar eine Flash-Applikation ermöglicht eine Cross-Domain-Kommunikation mit mehreren Webservern. Dadurch ist es garantiert, dass die Webapplikation für seine Existenz nur einen Webserver und relativ wenig Speicherplatz zur Verfügung braucht, indem aufwendige Ressourcen auf anderen kostenlosen Webservern gelagert (Delegation an CLOUD-APIs) bzw. die bereits vorhandenen Komponente der zu implementierenden RIA durch eine Delegation verwendet werden. Da es sich um eine Rich Internet Applikation handelt, die nach dem Aufruf vom Webserver weiter clientseitig abläuft und aufwendige Ressourcen (Benutzer und Bilder) und wiederkehrende Implementierungsarbeiten der Webapplikationskomponenten, die bereits im Web vorhanden sind (ggf. Login-Logout-, Wetter- und analytische Komponente) an kostenlose Web Services und Web-APIs von GOOGLE und YAHOO delegiert werden, die wiederum clientseitig verwendet werden, wird es im Rahmen der Bachelorarbeit angestrebt, die erarbeitete Ressourcenoptimierung (2.1.7) zu erreichen. D.h. so wenig wie möglich Implementierungsaufwand, Speicherplatz der Webapplikation, Traffic, Anzahl der Aufrufe zum Webserver und so viel wie möglich Performanz haben und dadurch die

---

* Im Rahmen der Bachelorarbeit wird es angestrebt, die mit einem Stern gekennzeichneten Web Services und Web-APIs von GOOGLE und YAHOO zu implementieren. D.h. alle verschiedenen Web Services und Web-APIs, die für die volle geplante Funktionalität der Webapplikation zuständig sind, werden möglichst in einem Exemplar implementiert.

1. Betriebskostenoptimierung (mit Hilfe von Web Services und Web-APIs inkl. RIA werden Traffic, Speicherplatzanforderung für Daten und für die Webapplikation selbst erheblich reduziert)

2. Performanceoptimierung (durch asynchrone Delegation an Web Services und Web-APIs wird die Performanz bzgl. der Interaktivität mit dem Benutzer erhöht)

3. Implementierungskostenoptimierung (wiederkehrende Arbeiten werden an Web Services und Web-APIs delegiert)

in einem für diese Arbeit ausreichenden Ausmaß behandelten Ressourcenoptimierung zu erreichen.

Ressourcenoptimierung 1 ist dabei vor allem für Optimierung von Traffic, Speicherplatz für Daten und Webapplikation selbst interessant. Der Speicherplatz für Daten (ggf. Bilder der Benutzer) der zu implementierenden Webapplikation wird durch eine RIA-Cloud-Komponente[6] abgebildet. Diese Abbildung findet zwischen RIA-Cloud-Komponente und zwei Web Services und Web-APIs „Flickr Cloud API“ und „PICASA WEB ALBUMS DATA API“.

Ressourcenoptimierung 2 ist interessant für Interaktivität der Webapplikation. Performanz der zu implementierenden Rich Internet Applikation hinsichtlich der Größe der Webapplikation wird durch die Delegation an die o.g. Web Services und Web-APIs erreicht und die asynchronen Aufrufe der Dienste erhöhen Performanz bzgl. der Interaktivität.

Ressourcenoptimierung 3 ist interessant für wiederkehrende Implementierungsarbeiten der zu implementierenden Webapplikation (Login-Logout-, Wetter- und analytische Komponenten). Die Login-Logout-Funktion wird durch eine RIA-Login-Logout-Komponente[6], die Wetter-Anzeige-Funktion durch eine RIA-Weather-Komponente[6] und die analytische Funktion zur Kontrolle der Ressourcenoptimierung durch eine RIA-Analytic-Komponente[6] abgebildet. Diese Abbildungen finden zwischen:

- RIA-Login-Logout-Komponente und zwei Web Services und Web-APIs: „Yahoo Authentication & Sign In“ und „GOOGLE ACCOUNT API“

- RIA-Cloud-Komponente und Web Service und Web-API “Yahoo Flickr”

---

[6] Im Folgenden werden zur Vereinheitlichung folgende Bezeichnungen der zu implementierenden externen Webapplikationskomponenten verwendet:

- LOGIN-API für „RIA-Login-Logout-Komponente“
- CLOUD-API für „RIA-Cloud-Komponente“
- WEATHER-API für „RIA-Weather-Komponente“
- ANALYTICS-API für “RIA-Analytic-Komponente”

- RIA-Weather-Komponente und Web Service und Web-API „Yahoo Weather RSS Feed“

- RIA-Analytics-Komponente und Web Service und Web-API „GOOGLE ANALYTICS“

statt.

Die Abb. 4.6 stellt die zu implementierenden Webapplikationskomponenten hinsichtlich der externen Aufrufe grafisch dar.

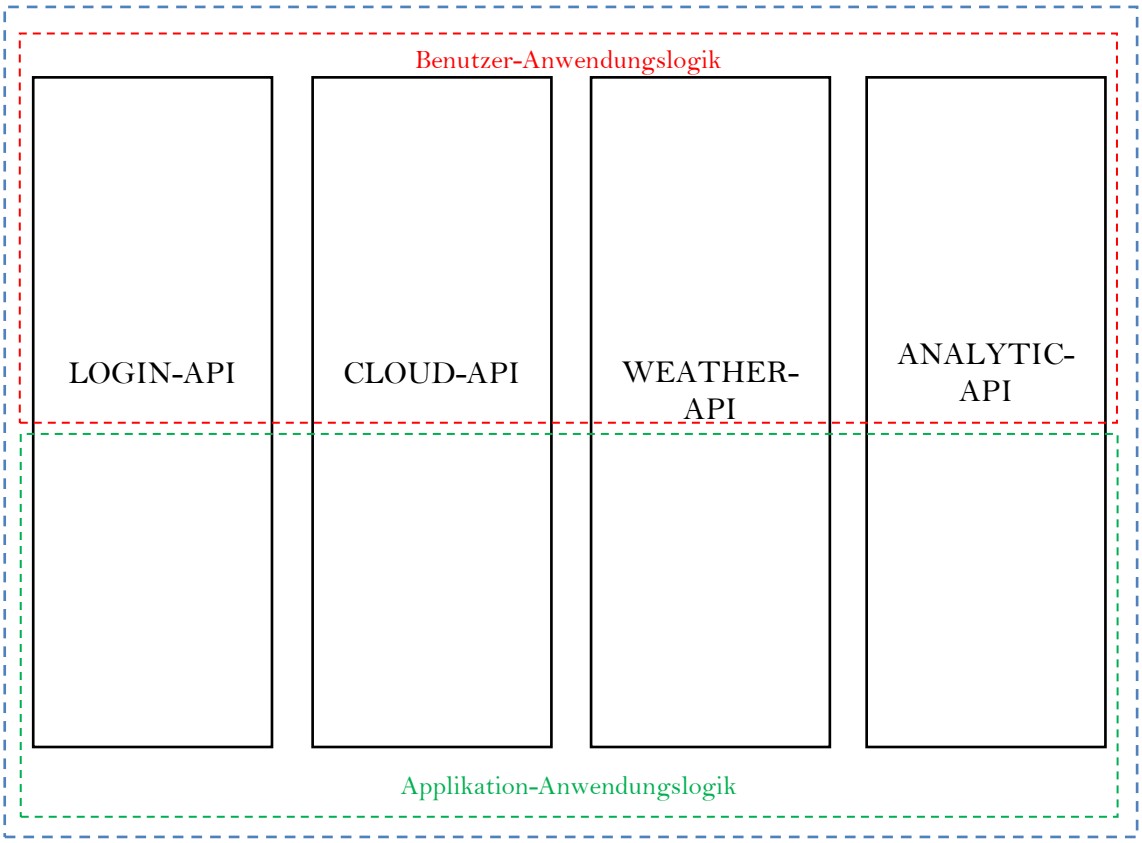

Abbildung 4.6.: Webapplikationskomponenten der in 1.2 beschriebenen Webapplikation (eigene Darstellung)

4.3.1. Asynchroner Nachrichtenaustausch mit Web Services und Web-APIs

Webapplikationskomponenten sehen als Aufrufmodell für alle Operationen mit Web Services und Web-APIs asynchrone Aufrufe vor, was zur Steigerung der erarbeiteten Ressourcenoptimierung hinsichtlich der zweiten Ressourcenoptimierung (s. 4.3) beiträgt. Um Webapplikation mit benutzerfreundlichen Benutzerinteraktion, die meistens potenziell lange andauernden Operationen der Web Services und Web-APIs verursacht, abzubilden sowie um eine möglichst lose Kopplung der beteiligten Dienste zu erreichen, eignen sich asynchrone Aufrufe, da dann nicht blockierend auf Antworten von angesprochenen Web Services und

Web-APIs gewartet wird. Diese zur Performanz beigetragene Ressourcenoptimierung kann beim Einsatz auf mobilen Geräten weitere Bedingungen hinzukommen lassen, die asynchrone Kommunikation rechtfertigen. Das Ziel ist dabei, dass keine Systemressourcen durch das Warten auf eine Antwort blockiert werden und dass keine Inkonsistenzen dadurch entstehen, dass während des Wartens Fehler auftreten (z.B. ein Verbindungsabbruch), was sehr deutlich zu der Benutzerfreundlichkeit, Interaktivität und o.g. Performanz beiträgt.

In dieser Arbeit wird daher mit den asynchronen Ausprägungen der Kommunikation zwischen zu implementierenden Webapplikationskomponenten und Web Services und Web-APIs gearbeitet. Infolgedessen müssen die Benutzer-Aufrufe von Web-Services und Web-APIs von der Benutzer-Anwendungslogik synchronisiert und an die Applikation-Anwendungslogik zu asynchronen Aufrufen der Web Services und Web-APIs weitergeleitet werden. Die Applikation-Anwendungslogik ist für den asynchronen Nachrichtenaustausch mit Web Services und Web-APIs zuständig.

Alle die in 4.3 erarbeiteten und zu implementierenden Webapplikationskomponenten bilden die Applikation-Anwendungslogik ab. Diese Komponenten der jeweiligen Ressourcenoptimierung (4.3) werden aus der Sicht verwendeten Web Services und Web-APIs die zur Verfügung stehenden Schnittstellen implementieren und müssen diese Schnittstellen einhalten (GOOGLE APIs 2010, YAHOO APIs 2010).

Die verwendeten Web Services und Web-APIs bieten unterschiedliche Varianten des Ansprechens hinsichtlich des Kommunikationsprotokolls (HTTP, SOAP, REST) an. Da letztendlich alle Varianten das HTTP-Protokoll als Transferprotokoll nutzen, einige nur einzige HTTP-Variante haben und keine Vorteile der Protokolle SOAP und REST in einem für diese Arbeit ausreichenden Ausmaß behandelten Ressourcenoptimierung genutzt werden können, wird daher in dieser Arbeit mit der HTTP-Variante von Web Services und Web-APIs gearbeitet.

Eine detaillierte Beschreibung und Behandlung der Protokolle HTTP, SOAP und REST finden sich in (HTTP 2010), (SOAP 2010) und (REST 2010).

Gemäß Spezifikationen der verwendeten in dieser Arbeit Web Services und Web-APIs (GOOGLE APIs 2010, YAHOO APIs 2010) müssen alle Nachrichten aus der Sicht zu implementierenden Applikation-Anwendungslogik den Aufbau einhalten. Im Folgenden wird die Applikation-Anwendungslogik bezüglich der zu implementierenden Nachrichten im Rahmen der erarbeiteten externen Aufrufe der Webapplikation (4.2.2.3) erarbeitet. Beim Empfang einer Nachricht von der Benutzer-Anwendungslogik müssen alle Web Services- und Web-APIs-Aufrufe zugänglich zu den aufgerufenen Web Services und Web-APIs gemacht werden. Beim Senden einer Nachricht von der Applikation-Anwendungslogik müssen notwendige HTTP-Header und Informationen automatisch gemäß Spezifikation des aufgerufenen Web Services und Web-APIs eingefügt bzw. generiert werden. Zum Beispiel können OpenID-Parameters des OpenID Web Services und Web-APIs, die für den Benutzer irrelevant sind, mit dem createOpenidParameters() jeder ausgehenden Nachricht von der Applikation-Anwendungslogik automatisch hinzugefügt werden (s.u.). Hier werden lediglich nur die benötigten Nachrichten der verwendeten Web Services und Web-APIs betrachtet und behandelt (Namen der Nachrichten sind abstrakt), so dass die folgenden Nachrichten der Webapplikationskomponenten zu implementieren sind:

- LOGIN-API (4.5.1)

redirectToOpenid() :	leitet den Benutzer mit den benötigten OpenId-Parametern an den ausgewählten OpenId Web Service und Web-API zur Authentifizierung weiter
redirectToWebapplication() :	Aktion der Webapplikation auf das Redirect von den OpenId Web Services und Web-APIs

- CLOUD-API (4.5.2)

getFrob() :	Anfrage nach dem Sicherheitstoken von dem Cloud Web Service und Web-API im Zusammenhang zur Verfügung stehenden Application- und Secret-Keys für die weiteren Anfragen
findByUsername(userId) :	Anfrage nach der Flickr-ID für die vom Benutzer eingegebene User-Id bei FLICKR
getPublicPhotos(flickrId) :	Anfrage nach den Benutzerbildern zur FlickrId des Benutzers

- WEATHER-API (4.5.3)

getWeather(zip) :	Anfrage an den Weather Web Service und Web-API nach den Wetterinformationen

- ANALYTICS-API (4.5.4)

trackPageview(name) :	Senden der Informationen an den Analytics Web Service und Web-API hinsichtlich der Seitenanzeige
trackEvent(parameters) :	Senden der Informationen hinsichtlich der Performanzanalyse

Beim Empfang einer Nachricht der Applikation-Anwendungslogik von Web Services und Web-APIs müssen Daten bzw. Informationen der Benutzer-Anwendungslogik zur Verfügung gestellt bzw. automatisch extrahiert werden.

Es ist anzumerken, dass diese Art der Zwischenbearbeitung (Applikation-Anwendungslogik) für jegliche Art von Benutzernachrichten an Web Services und Web-APIs sinnvoll ist.

4.4. Framework Flex (SDK 3.2.0)

Aus Kostengründen und Überlegungen in 2.2.1 wird für die Implementierung der Rich Internet Applikation ein programmierorientiertes und kostenloses Open-Source-Framework für Flash „Flex-SDK 3.2.0“ (Flex SDK 3.2.0) mit der Entwicklungs- und Testumgebung Flex Builder Professional 3.2 (Flex Builder 3) verwendet.

„Flex-SDK“ bietet Webentwicklern einen vertrauten und bekannten Programmiermodel: die Kombination von einer tag-basierten Sprache MXML (Adobe Flex MXML) zur Erstellung von User-Interface und einer traditionellen objektorientierten Programmiersprache Action Script zur Erstellung von „view-losen“ Klassen für die Webapplikation. Dieser mächtige Programmiermodel ermöglicht Flex-Entwicklern die Benutzung der etablierten Softwareengineeringsmethoden, Design-Patterns und besten Praktiken. In diesem Kapitel (4) werden Muster und Techniken ermittelt bzw. erarbeitet, die sich auf der MVC-Architektur (Model-View-Controller) und losen Kopplung der Webapplikationskomponente basieren, um die in Kapitel 3 erarbeiteten fachlichen funktionalen (3.1) und nichtfunktionalen (3.3) Anforderungen der Rich Internet Applikation umzusetzen und die ebenfalls folgenden in der Analyse beschriebenen fachlichen Schnittstellen (APIs) zu implementieren.

Für allgemeine Betrachtungen sowie detaillierte Beschreibungen vom Framework „Flex“ siehe (Adobe 2010), (Adobe wiki 2010), (Adobe LiveDocs 2009), (Herrington u.a. 2008) und (Reinhardt 2009), zu Action Script im speziellen auch (Hauser u.a. 2009), (Shupe u.a. 2008), (Lott u.a. 2007) und (Kersken 2006). Beispiele für Anwendungen von „Flash“ und „Flex“ in der Praxis finden sich in (Flex-Applikationen 2010).

4.5. Beschreibung der Komponenten

In 3.1 und 3.3 sind die funktionalen und nichtfunktionalen Anforderungen genau beschrieben.

Aufgrund der Überlegungen in den vorangehenden Abschnitten werden im Folgenden notwendige Webapplikationskomponenten (APIs) in Form von konzeptionellen Interface- und Klassendiagrammen dargestellt.

In diesem Abschnitt werden die zu implementierenden Webapplikationskomponenten genau beschrieben. Es werden sowohl die statischen Strukturen der Webapplikationskomponenten (Klassen und Relationen) als auch dynamisches Verhalten (Interaktionen) beschrieben.

Wenn möglich und sinnvoll, werden bewährte Entwurfsmuster verwendet (4.1), einschließlich der in 4.4 beschriebenen Muster für das zu verwendende Framework.

Neben der Erfüllung der funktionalen Anforderungen (3.1) steht von den nichtfunktionalen Anforderungen hier die Leistungsfähigkeit bzw. Performanz (s. 3.3.2) im Vordergrund. Die im Zuge von Robustheitsanforderungen (s. 3.3.1) beschriebenen benötigten Funktionen werden ebenfalls an geeigneter Stelle aufgegriffen. Aus Erweiterbarkeit (s. 3.3.6) wird im Rahmen der Beschreibung der Web Services- und Web-APIs- Komponenten (Applikation-Anwendungslogik) eingegangen. Alle weiteren nichtfunktionalen Anforderungen, auf die man bezüglich der verwendeten Architektur (4.1) haben kann, können im Rahmen der zur Verfügung stehenden Zeit hier nicht ausführlich behandelt werden.

4.5.1. LOGIN-API

Die Applikation-Anwendungslogik sollte dem Benutzer der Webapplikation möglichst viel Arbeit bei der Erzeugung der externen Operationen, der Terminierung von Operation sowie der Einhaltung der benutzten Web Services und Web-APIs abnehmen.
Die Benutzer-Anwendungslogik sollte auch möglichst transparent für die Applikation-Anwendungslogik sein und für die Einhaltung der Schnittstellen von Web Services und Web-APIs mitverantwortlich sein muss (s. 3.2.2). Dazu muss dem Benutzer die Möglichkeit gegeben werden, im Rahmen der Benutzer-Anwendungslogik auf die Ereignisse im Rahmen der Benutzung der Web Services und Web-APIs und auf die Fehler zu reagieren. Dem Anwender müssen also die Operationen gemäß Anwendungsfälle (3.2.2) und Weiterleitung der Fehlermeldungen an den Benutzer zur Verfügung stehen.

Abb. 4.7 zeigt die benötigten Interfaces bzw. was das LOGIN-API bezüglich der funktionalen (3.1) und nichtfunktionalen (3.3) Anforderungen leisten soll. Auf Benutzerseite dient das Interface *UserLogin* dazu, Zugriff auf den autorisierten Zugang für die Benutzer zu ermöglichen. Dabei werden keine Übergabeparameter vom Benutzer abgefragt. Der Anwender löst nur ein entsprechendes Ereignis, ob er sich ein- oder ausloggen möchte.

Auf Applikationsseite stehen mit dem Interface *ApplicationLogin* die Operationen zur Verfügung, die an den „Login Web Service und Web-API“ gesendet werden. Diese Operationen werden von dem Interface *UserLogin* aufgerufen.

Die Applikation-Anwendungslogik ist dafür verantwortlich, dass die jeweils richtige Methode vom angesprochenen „Login-Web Service und Web-API“ benutzt wird, da die Webapplikation beliebig erweiterbar bezüglich der verwendeten Web Services und Web-APIs des Login-Dienstes.

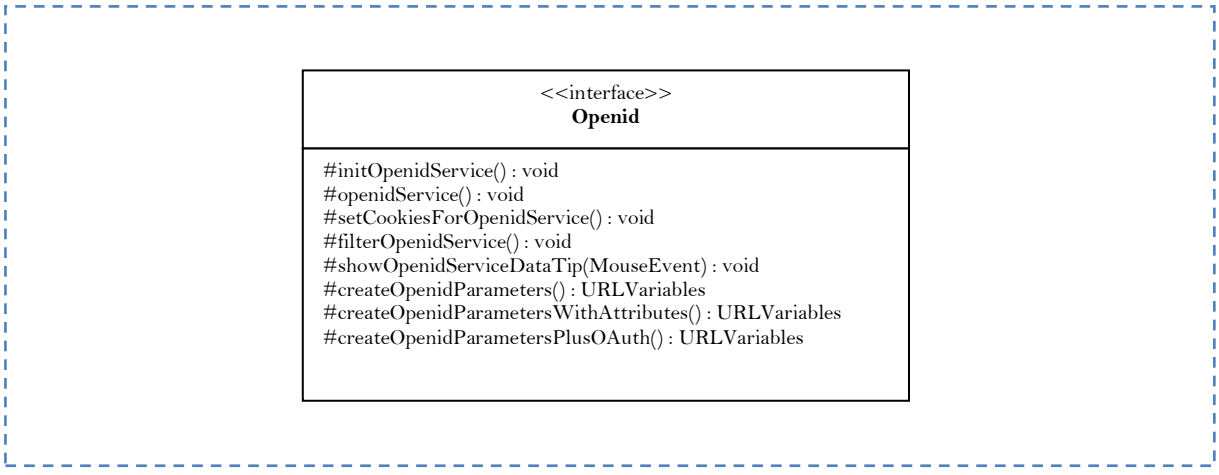

Abbildung 4.7.: Login-API

4.5.1.1. LOGIN-API – Implementierung

Die LOGIN-Komponente der Webapplikation besteht aus zwei Teilen. Der erste Teil implementiert das Interface *UserLogin* aus 4.5.1 und bildet sozusagen die Benutzerschnittstelle

für den Benutzer. Der zweite Teil bildet die Basis für die Implementierung der verwendeten „Login Web Services und Web-APIs“, die das APIs der jeweiligen Login-Dienste abbildet und spielt dabei die Rolle des sogenannten Adapters.

Das LOGIN-API wird sowohl von der Benutzer-Anwendungslogik als auch von der Applikation-Anwendungslogik benutzt. Die Benutzer-Anwendungslogik benötigt von der Applikation-Anwendungslogik unabhängige Funktionen, etwa zur Verwaltung der Ereignisse auf Benutzerseite.

Wie bereits in 4.3 angedeutet wurde, verwenden alle Webapplikationskomponenten außer CUSTOMIZATION-API und BILDBEAERBEITUNG-API entsprechende Web Services und Web-APIs, die ein API bereitstellen. Deswegen sollten alle ApplicationLogin-API Teile die Spezifikationen der jeweiligen Web Services und Web-APIs einhalten. Dazu benötigen alle ApplicationLogin-API Teile die entsprechenden Operationen in ihren Interfaces. Da fast alle Web Services und Web-APIs ihre eigene Spezifikation haben, sollten die Operationen in die Interfaces der einzelnen Services aufgenommen werden.

Zum Aufrufen von „LOGIN Web Services und Web-APIs“ und zwar (YAHOO OPENID 2010) und (GOOGLE OPENID 2010) wird von Flex-SDK 3.2.0 (Adobe LiveDocs 2009) ein API bereitgestellt. Ein Objekt der Klasse *URLRequest* wird über Setter-Methoden mit den notwendigen Informationen bezüglich der jeweiligen Spezifikationen von „LOGIN Web Services und Web-APIs“ versorgt, bevor schließlich eine Methode zur Ausführung des Aufrufs aufgerufen wird.

Die Anforderungen an das API wurden u.a. durch Interfaces (Abb. 4.7) ausgedrückt. Die Realisierung dieses APIs in Flex sollte daher diese Interfaces möglichst komplett getrennt von der Implementierung enthalten. Durch die Trennung kann die Implementierung später leichter verändert, erweitert oder ausgetauscht werden.

Die Interfaces sind bereits nach Benutzer- und Applikationslogiken getrennt. Es sollte parallel eine identische Struktur von Implementierungsklassen geben. Dabei muss zwischen Benutzer- und Applikationssicht unterschieden werden:

- Der Benutzer hat nur Zugriff auf sein eigenes UserLogin-API (UserLogin).
- Der Benutzer hat über das UserLogin-API auch Zugriff auf das ApplicationLogin-API (ApplicationLogin) aus Benutzersicht für alle implementierten bzw. zur Verfügung stehenden Login-Dienste.
- Das ApplicationLogin-API, wie oben bereits angedeutet wurde, implementiert den jeweiligen Login-Dienst des OpenID 2.0 – Standards und muss für jeden einzelnen Dienst implementiert werden. Da im Rahmen dieser Arbeit zwei Login-Dienste implementiert werden (4.3), kommt die ApplikationLogik-API-Implementierung zweimal vor, die für jeden weiteren Login-Dienst implementiert werden muss.

Im Folgenden werden diese drei API-Teile entworfen. Alle ApplilcationLogin-API Teile haben gemeinsam, dass nur diejenigen Klassen nicht abstrakt sein sollten, die den verwendeten „OpenID-Endpunkt“ (URL des OpenId-Dienstes des jeweiligen Anbieters) implementieren. Diese Struktur ist leicht erweiterbar: APIs für weitere auf ApplicationLogin aufbauende „Login Web Services und Web-APIs“ wie z.B. Amazon-OpenID können analog zu den konkreten Klassen implementiert werden.

4.5.1.1.1. UserLogin-API

Abb. 4.8 zeigt eine mögliche Implementierung des UserLogin-APIs. Eine UserLogin-Instanz wird von der Webapplikation nach der Initialisierung angelegt.

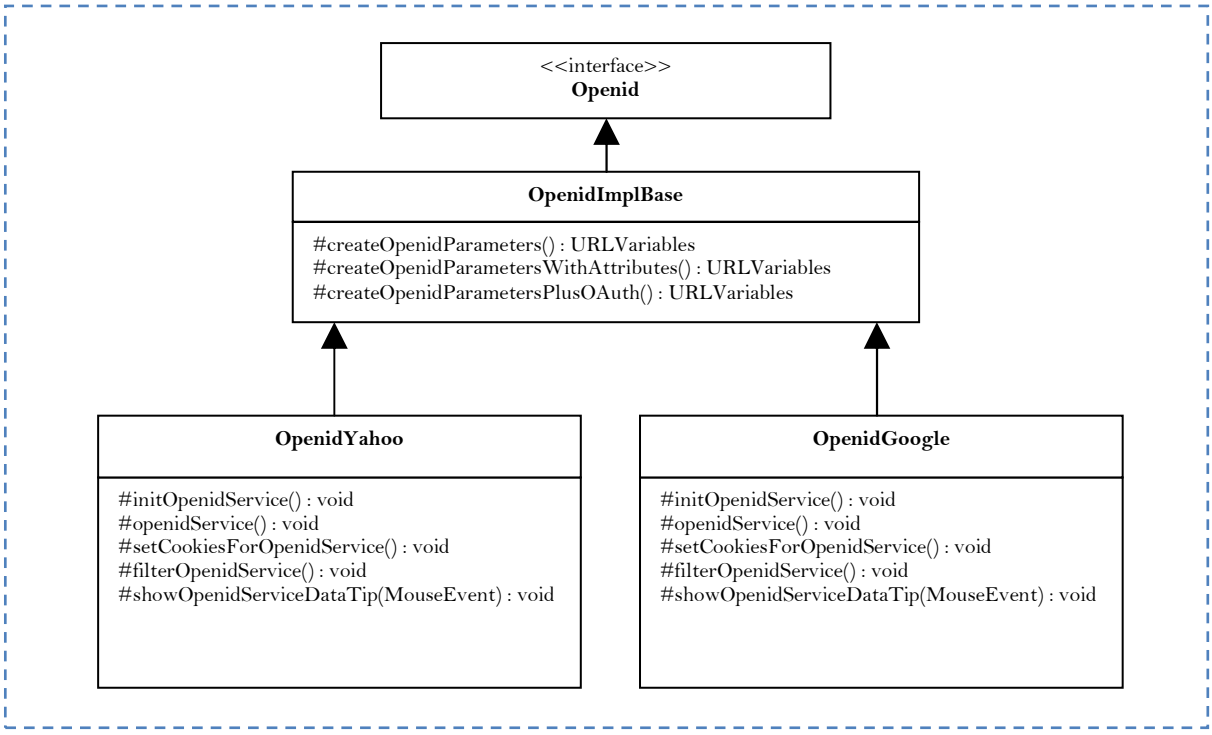

Abbildung. 4.8.: Implementierung des UserLogin-APIs

Im Falle von ApplicationLogin-Aktivität wird von der UserLogin-Instanz eine Instanz von der Implementierungsklasse der Applikation-Anwendungslogik angelegt.

4.5.1.1.2. ApplicationLogin-API aus Applikationsseite

Abb. 4.9 zeigt eine mögliche Implementierung des ApplicationLogin-APIs auf Applikationsseite. Der ApplicationLogin-Teil erlaubt der Benutzer- und Applikation-Anwendungslogik hier den Zugriff auf den Endpunkt des jeweiligen Login-Dienstes des OpenID-Standards, so dass das UserLogin Anmelde-Ereignisse (Events) an das ApplicationLogin-API schicken kann und die Applikation-Anwendungslogik die entsprechenden Methoden bezüglich der „Login Web Services und Web-APIs“ aufrufen kann.

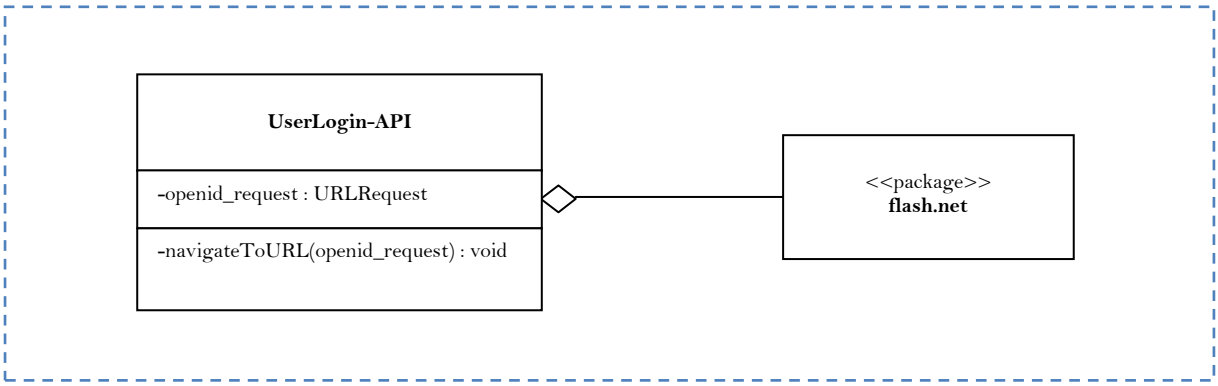

Abbildung 4.9.: Implementierung des ApplicationLogin-APIs aus Applikationsseite

Der ApplicationLogin-Teil erlaubt der Applikation-Anwendungslogik auf Applikationsseite, der Webapplikation „Login-Dienst-Nachrichten“ bezüglich des jeweiligen „Web Services und Web-APIs“ zu senden und „Login-Dienst-Antworten“ (Nachrichten) zu empfangen (YAHOO OPENID 2010, GOOGLE OPENID 2010). Die Applikation-Anwendungslogik überprüft dabei bzw. ist dafür zuständig, dass die zu sendenden Nachrichten gemäß der „Login-Dienst-APIs“ aufgebaut sind und abstrahiert die notwendige Information beim Empfang der Antworten.

4.5.1.1.3. LOGIN-Komponente

Abb. 4.10 zeigt schließlich eine mögliche Implementierung der LOGIN-Komponente in einer kompletten Darstellung. Diese Abbildung stellt ein UML-Klassendiagramm dar, das aus Übersichtlichkeitsgründen nur auf das Wesentliche grafisch beschränkt wurde. In diesem Diagramm sind die Model-Klassen gelb und die View-Klassen grün gekennzeichnet.

Modell
Die Klassen OpenidYahoo und OpenidGoogle repräsentieren das Modell der LOGIN-Komponente. Sie sind ‚viewlos‘ und in dem Action Script 3 zu implementieren.

View-Klassen
Button (UIComponent) ist die Haupt-UI-Klasse in dieser Komponente. Sie ist in MXML zu implementieren. Mit Flex ist es möglich, die MXML-Klassen von “scratch” zu erstellen oder die bestehenden Klassen zu erweitern.

Controller-Klassen
URLVariables, URLRequest und navigateToUrl() sind in Flex-SDK aufgewiesen. Sie ermöglichen die Kommunikation mit Web Services und Web-APIs (ggf. Yahoo- und Google-OpenId). Die andere Controller-Komponente, die implizit in der Webapplikation benutzt wird, ist die Binding-Klasse, die zwischen den View- und Model-Klassen sitzt: die Binding-Klasse benachrichtigt automatisch die Views, wenn eine relevante Datenänderung in den Model-Klassen stattfindet.

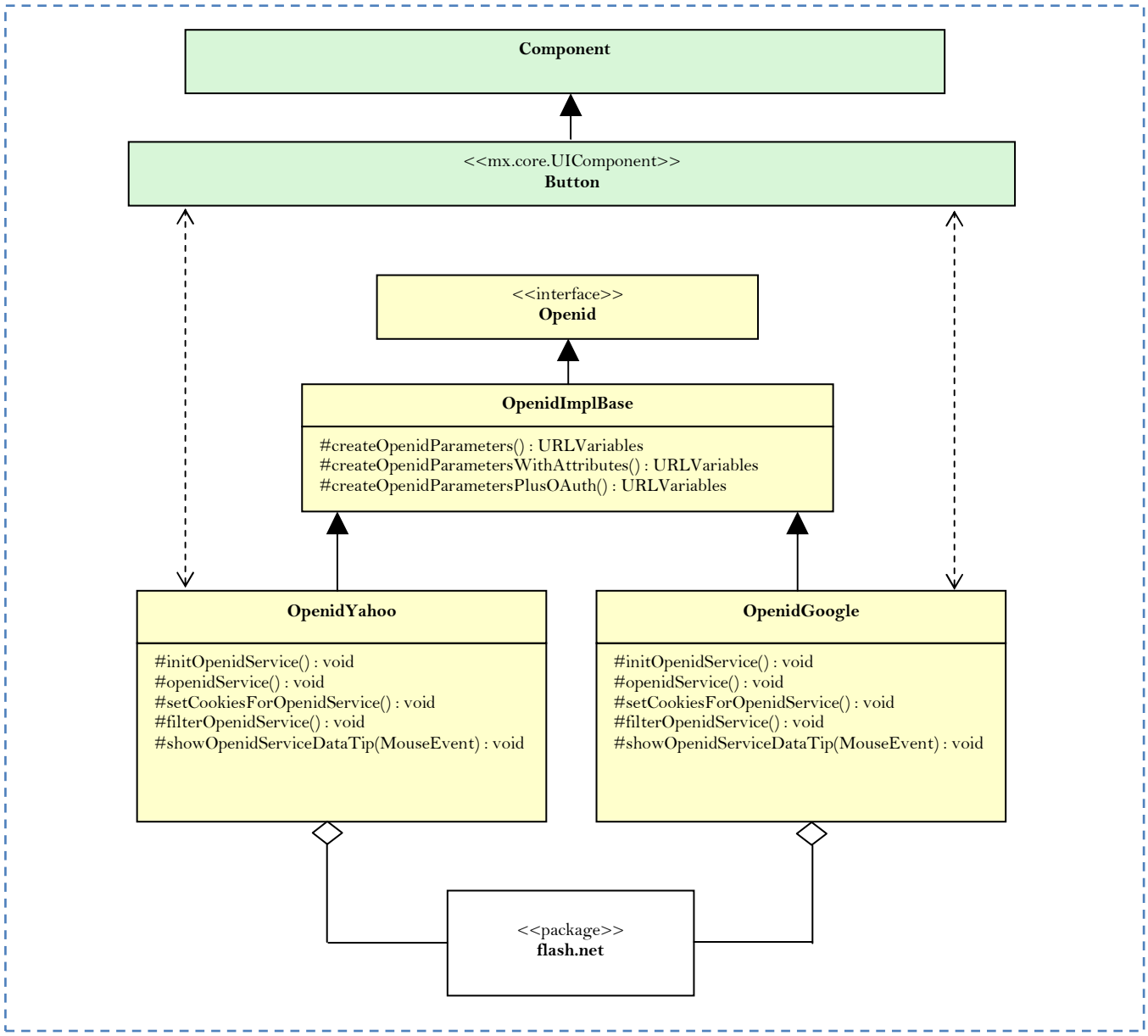

Abbildung 4.10.: LOGIN-Komponente

Implementierung

Mit dem Framework Flex-SDK ist es möglich, die Klassen mit MXML, der Flex XML-basierten Markup-Sprache, oder mit Flash ActionScript, der objektorientierten und statisch typisierten Programmiersprache, zu erstellen.
Normalerweise wird MXML benutzt, um UI-Klassen (View) der Webapplikation zu implementieren und ActionScript, um die nicht visuellen Klassen (Model und Controller) zu bilden. Die meisten Controller-Klassen sind in dem Flex-SDK vorhanden, wie in diesem Fall das Paket “flash.net”.
Die Kombination von einer tag-basierten Sprache wie MXML für das UI und einer traditionellen objektorientierten Programmiersprache für die ‚viewlosen‘ Klassen (Model oder Controller) wurde bewährt, die auch in dieser Implementierung zu verwenden ist.

4.5.2. CLOUD-API – Implementierung

Im Rahmen dieser Arbeit wird für die Umsetzung der CLOUD-Komponente das kostenlose Web Service und Web-API „FLICKR“ verwendet (YAHOO FLICKR 2010). Die in Open-Source-Projekt „Flickr Library“ (Cloud Library) verfügbaren Klassen *com.adobe.webapis.flickr.events.FlickrResultEvent* und *com.adobe.webapis.flickr.FlickrService* können für die Umsetzung der CLOUD-Komponente herangezogen werden. Das „Flickr Web Service und Web-API“ ist betriebsbereit bzw. von der Webapplikation ansprechbar, indem man eine FlickrService-Instanz erzeugt und dabei den Applikationsschlüssel übergibt, der bei der Applikationsregistrierung der Webapplikation zugewiesen wurde. Zusätzlich sollte man noch den Geheimschlüssel initialisieren, der auch bei der Registrierung mitgeteilt wird (s. (YAHOO FLICKR 2010)). Diese notwendige Initialisierung sollte beim Start der Webapplikation passieren, um unnötige Wartezeiten zu vermeiden.

Abb. 4.11 veranschaulicht, welche Interfaces benötigt werden.

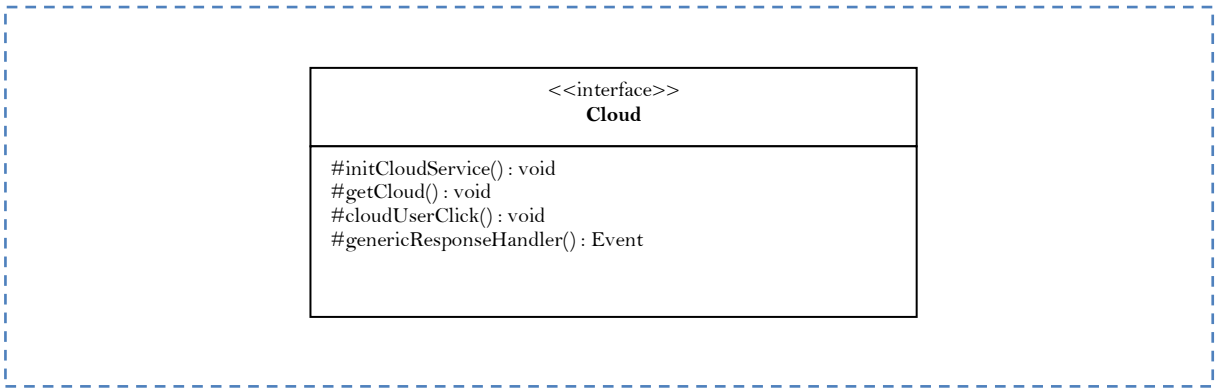

Abbildung 4.11.: Cloud-API

Im Folgenden wird dieses API-Teil entworfen. Im Vergleich zu LOGIN-API-Implementierung sollten alle ApplilcationCloud-API Teile (im Rahmen der Arbeit wird nur ein ApplicationCloud-Teil implementiert) ohne Abstraktion konkret implementiert werden, da sie keine Gemeinsamkeiten im Vergleich zu LOGIN-API aufweisen. Diese Struktur ist wiederrum leicht erweiterbar: APIs für weitere auf ApplicationCloud aufbauende „CLOUD Web Services und Web-APIs“ wie z.B. Amazon-CLOUD können analog zu den konkreten Klassen implementiert werden.

Zum Aufrufen vom „FLICKR Web Service und Web-API“ wird in (Cloud Library) ein API bereitgestellt. Ein Objekt der Klasse *FlickrService* wird über Setter-Methoden mit den notwendigen Informationen versorgt, bevor schließlich eine Methode zur Ausführung des Aufrufs aufgerufen wird.

4.5.2.1. UserCloud-API

Abb. 4.12 zeigt eine mögliche Implementierung des UserCloud-APIs. Eine UserCloud-Instanz wird von der Webapplikation nach der Initialisierung angelegt.

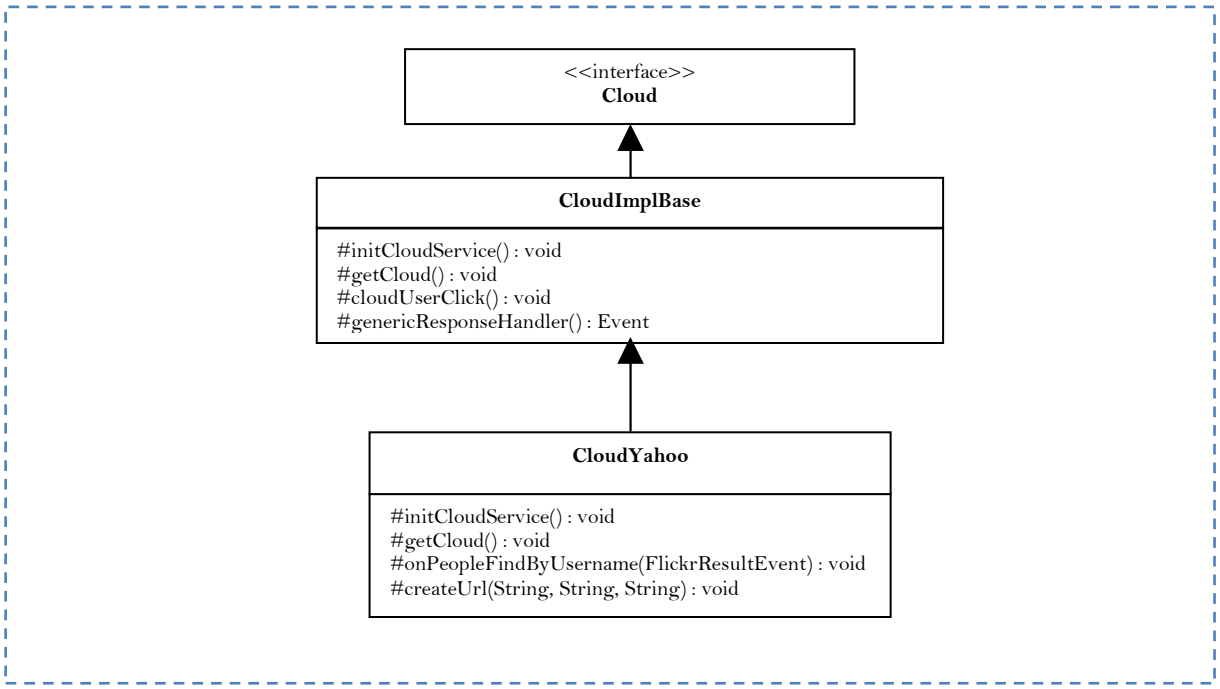

Abbildung 4.12.: Implementierung des UserCloud-APIs

Im Falle von ApplicationCloud-Aktivität wird von der UserCloud-Instanz eine Instanz von der Implementierungsklasse der Applikation-Anwendungslogik angelegt.

4.5.2.2. ApplicationCloud-API aus Applikationsseite

Abb. 4.13 zeigt schließlich eine mögliche Implementierung des ApplicationCloud-APIs auf Applikationsseite. Der ApplicationCloud-Teil erlaubt der Benutzer- und Applikation-Anwendungslogik hier den Zugriff auf den Endpunkt des jeweiligen Cloud-Dienstes, so dass das UserCloud eingegebene Benutzer-IDs des FLICKR-Services an das ApplicationCloud-API schicken kann und die Applikation-Anwendungslogik die entsprechenden Methoden bezüglich der „CLOUD Web Services und Web-APIs“ aufrufen kann.

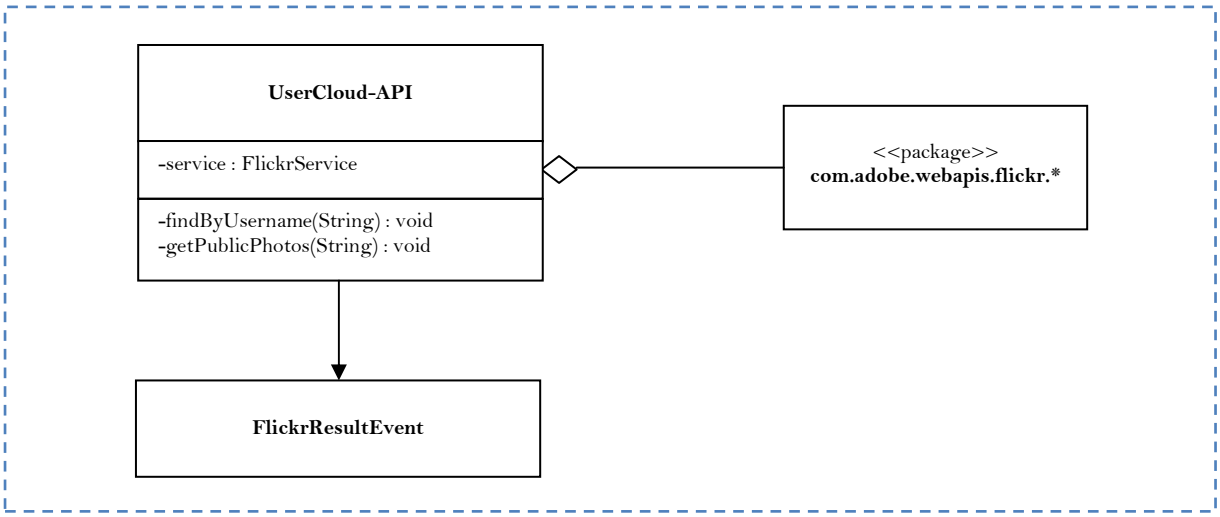

Abbildung 4.13.: Implementierung des ApplicationCloud-APIs aus Applikationsseite

Der ApplicationCloud-Teil erlaubt der Applikation-Anwendungslogik über das Listener *FlickrResultEvent* auf Applikationsseite, der Webapplikation „Cloud-Dienst-Nachrichten" bezüglich des jeweiligen „Web Services und Web-APIs" zu senden und „Cloud-Dienst-Antworten" (Nachrichten) zu empfangen (YAHOO FLICKR 2010). Die Applikation-Anwendungslogik überprüft dabei bzw. ist dafür zuständig, dass die zu sendenden Nachrichten gemäß der „Cloud-Dienst-APIs" aufgebaut sind und abstrahiert die notwendige Information beim Empfang der Antworten.

4.5.2.3. CLOUD-Komponente

Abb. 4.14 zeigt schließlich eine mögliche Implementierung der CLOUD-Komponente in einer kompletten Darstellung.

Model
Die Klasse CloudYahoo repräsentiert das Modell der CLOUD-Komponente. Sie ist ‚viewlos' und in dem Action Script 3 zu implementieren.

View-Klassen
Button und TileList (UIComponents) sind die Haupt-UI-Klasse in dieser Komponente. Sie sind in MXML zu implementieren.

Controller-Klassen
Die Methoden für die Web-Service-Aufrufe getFrob(), findByUsername() und getPublicPhotos() sind in den Open-Source-Projekt „Flickr Library" (Cloud Library) verfügbaren Klassen *com.adobe.webapis.flickr.events.FlickrResultEvent* und *com.adobe.webapis.flickr.FlickrService* aufgewiesen. Sie ermöglichen die Kommunikation mit Web Services und Web-APIs (ggf. Yahoo Flickr). Die andere Controller-Komponente, die implizit in der Webapplikation benutzt wird, ist die Binding-Klasse, die zwischen den View- und Model-Klassen sitzt: die Binding-Klasse benachrichtigt automatisch die Views, wenn eine relevante Datenänderung in den Model-Klassen stattfindet.

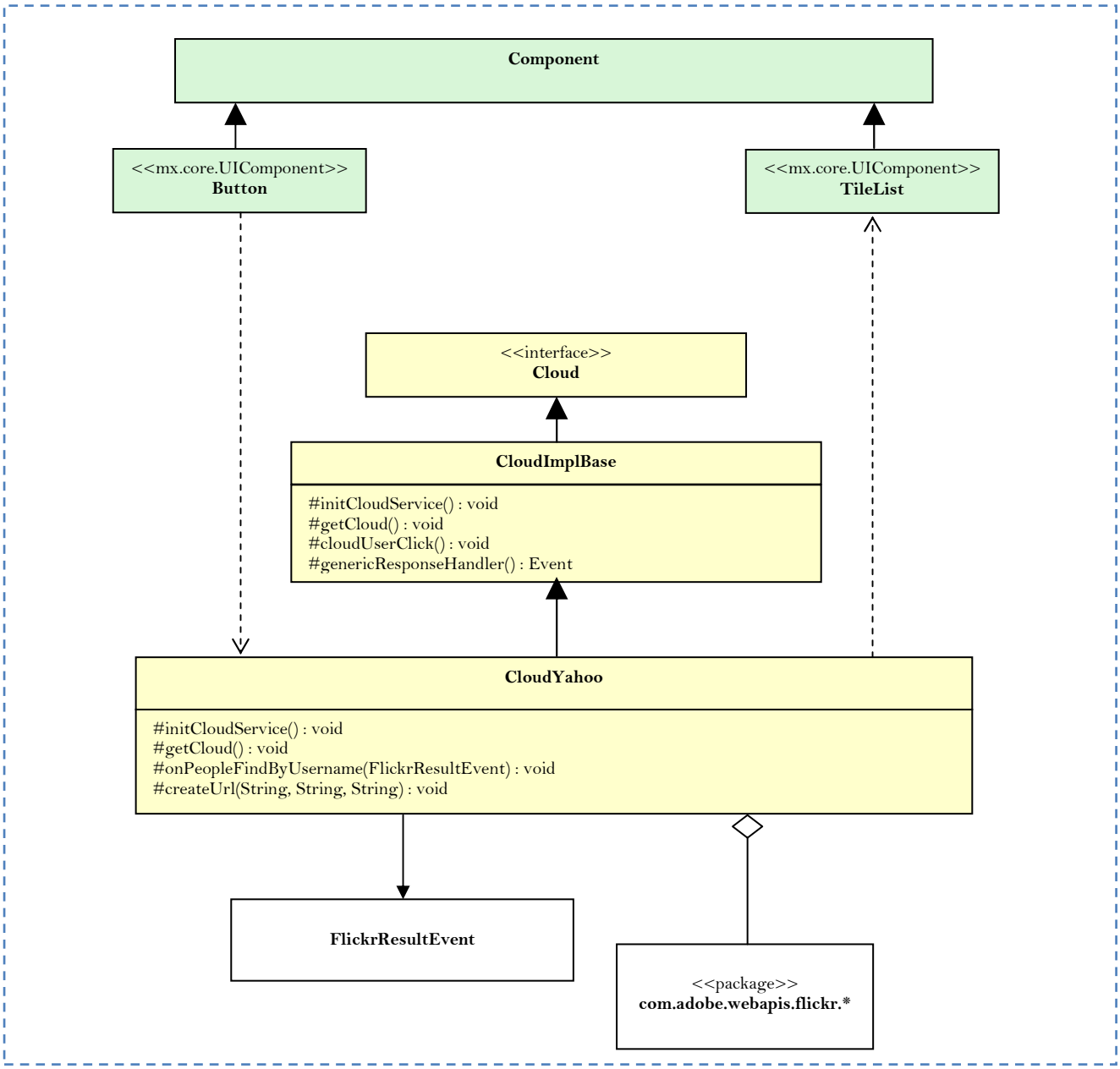

Abbildung 4.14.: CLOUD-Komponente

4.5.3. WEATHER-API – Implementierung

Im Rahmen dieser Arbeit wird für die Umsetzung der WEATHER-Komponente das kostenlose Web Service und Web-API „YAHOO Weather RSS Feed“ verwendet (YAHOO WEATHER 2010). Die in Open-Source-Projekt „Yahoo! ASTRA Web APIs library“ (Weather Library) verfügbaren Klassen *com.yahoo.webapis.weather.Weather*, *com.yahoo.webapis.weather.WeatherResultEvent* und *com.yahoo.webapis.weather.WeatherErrorEvent* können für die Umsetzung der WEATHER-Komponente herangezogen werden.

Abb. 4.15 veranschaulicht, welche Interfaces benötigt werden. Im Vergleich zu LOGIN-API-Implementierung wird hier ein Interface benötigt, weil nur ein Weather Web Service und Web-API verwendet und implementiert wird.

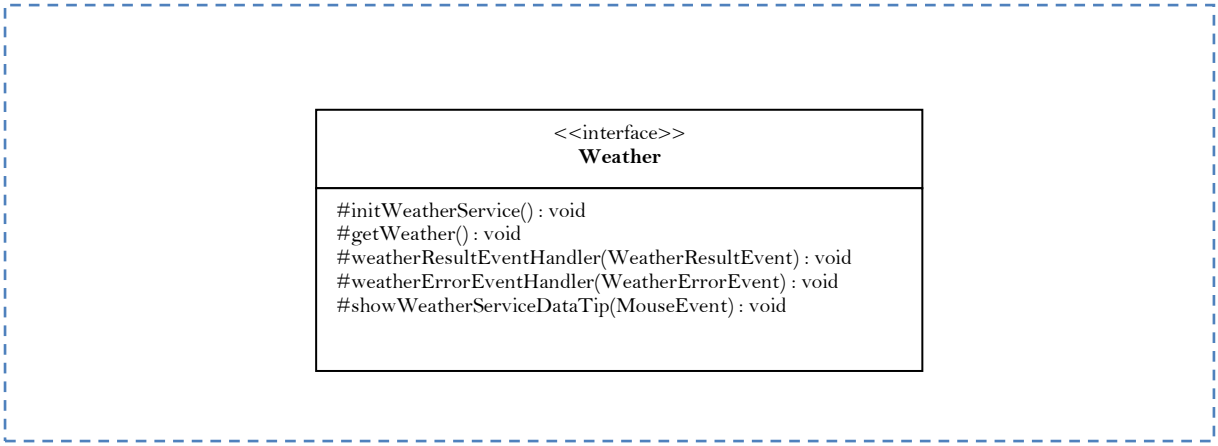

Abbildung 4.15.: Weather-API

Im Folgenden wird dieses API-Teil entworfen. Hier sollten auch alle ApplilcationWeather-API Teile (im Rahmen der Arbeit wird nur ein ApplicationWeather-Teil implementiert) ohne Abstraktion konkret implementiert werden. Diese Struktur ist wiederrum leicht erweiterbar: APIs für weitere „WEATHER Web Services und Web-APIs“ wie z.B. Amazon-WEATHER können analog zu den konkreten Klassen implementiert werden.

Zum Aufrufen vom „YAHOO! Weather RSS Feed Web Service und Web-API“ wird in (Weather Library) ein API bereitgestellt. Ein Objekt der Klasse *Weather* wird über Setter-Methoden mit den notwendigen Informationen versorgt, bevor schließlich eine Methode zur Ausführung des Aufrufs aufgerufen wird.

4.5.3.1. UserWeather-API

Abb. 4.16 zeigt eine mögliche Implementierung des UserWeather-APIs. Eine UserWeather-Instanz wird von der Webapplikation nach der Initialisierung angelegt.

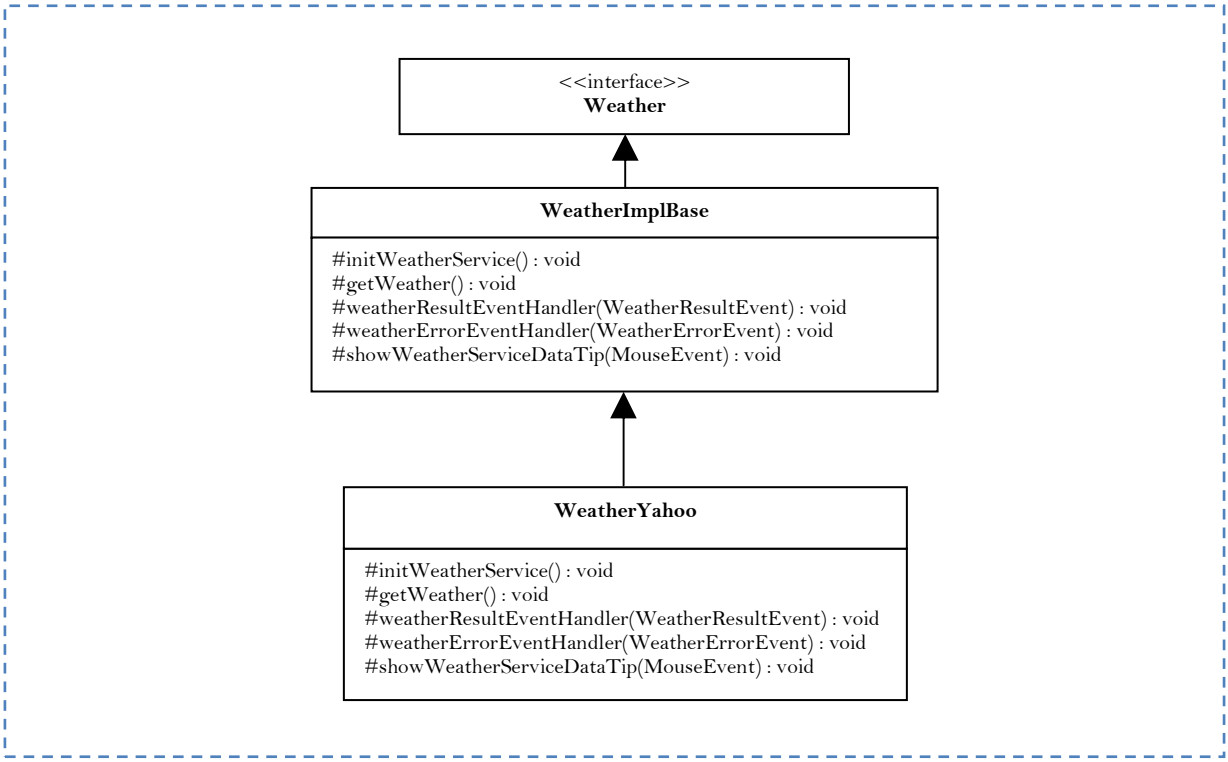

Abb. 4.16.: Implementierung des UserWeather-APIs

Im Falle von ApplicationWeather-Aktivität wird von der UserWeather-Instanz eine Instanz von der Implementierungsklasse der Applikation-Anwendungslogik angelegt.

4.5.3.2. ApplicationWeather-API aus Applikationsseite

Abb. 4.17 zeigt schließlich eine mögliche Implementierung des ApplicationWeather-APIs auf Applikationsseite. Der ApplicationWeather-Teil erlaubt der Benutzer- und Applikation-Anwendungslogik hier den Zugriff auf den Endpunkt des jeweiligen Weather-Dienstes, so dass das UserWeather eingegebene Weather-IDs des WEATHER-Services an das ApplicationWeather-API schicken kann und die Applikation-Anwendungslogik die entsprechenden Methoden bezüglich der „YAHOO! Weather RSS Feed Web Services und Web-APIs“ aufrufen kann.

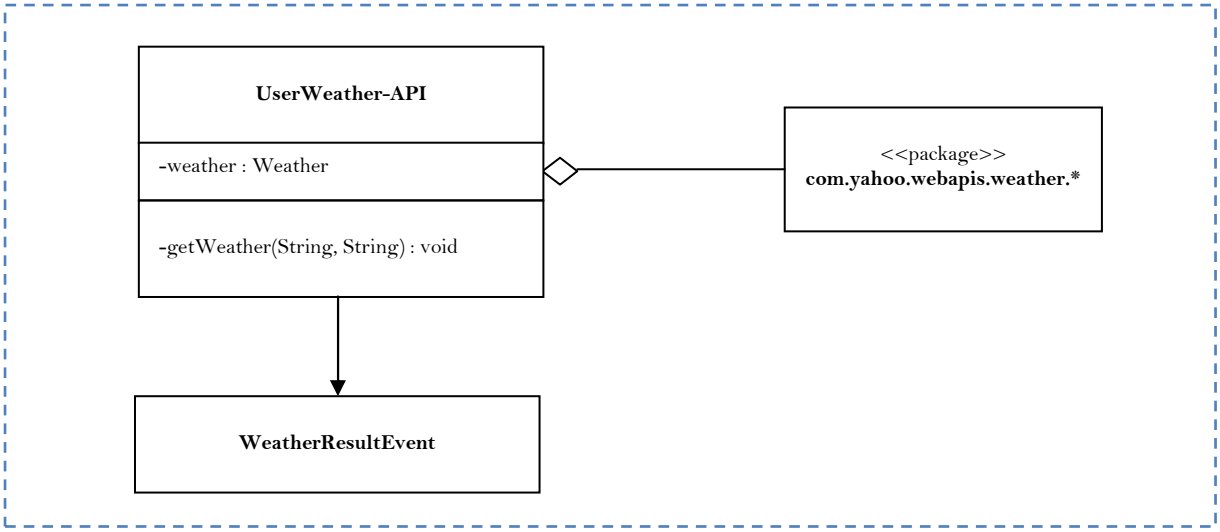

Abbildung 4.17.: Implementierung des ApplicationWeather-APIs aus Applikationsseite

Der ApplicationWeather-Teil erlaubt der Applikation-Anwendungslogik über das Listener *WeatherResultEvent* auf Applikationsseite, der Webapplikation „Weather-Dienst-Nachrichten" bezüglich des jeweiligen „Web Services und Web-APIs" zu senden und „Weather-Dienst-Antworten" (Nachrichten) zu empfangen (YAHOO WEATHER 2010). Die Applikation-Anwendungslogik überprüft dabei bzw. ist dafür zuständig, dass die zu sendenden Nachrichten gemäß der „Weather-Dienst-APIs" aufgebaut sind und abstrahiert die notwendige Information beim Empfang der Antworten.

4.5.3.3. Weather-Komponente

Abb. 4.18 zeigt schließlich eine mögliche Implementierung der WEATHER-Komponente in einer kompletten Darstellung.

Modell
Die Klasse WeatherYahoo repräsentiert das Modell der WEATHER-Komponente. Sie ist ‚viewlos' und in dem Action Script 3 zu implementieren.

View-Klassen
Button Label und Image (UIComponents) sind die Haupt-UI-Klassen in dieser Komponente. Sie sind in MXML zu implementieren.

Controller-Klassen
Die Methode für den Web-Service-Aufruf getWeather() ist in den Open-Source-Projekt „Yahoo! ASTRA Web APIs library" (Weather Library) verfügbaren Klassen *com.yahoo.webapis.weather.Weather*, *com.yahoo.webapis.weather.WeatherResultEvent* und *com.yahoo.webapis.weather.WeatherErrorEvent* aufgewiesen. Sie ermöglicht die Kommunikation mit Web Services und Web-APIs (ggf. Yahoo Weather). Die andere Controller-Komponente, die implizit in der Webapplikation benutzt wird, ist die Binding-Klasse, die zwischen den View- und Model-Klassen sitzt: die Binding-Klasse benachrichtigt automatisch die Views, wenn eine relevante Datenänderung in den Model-Klassen stattfindet.

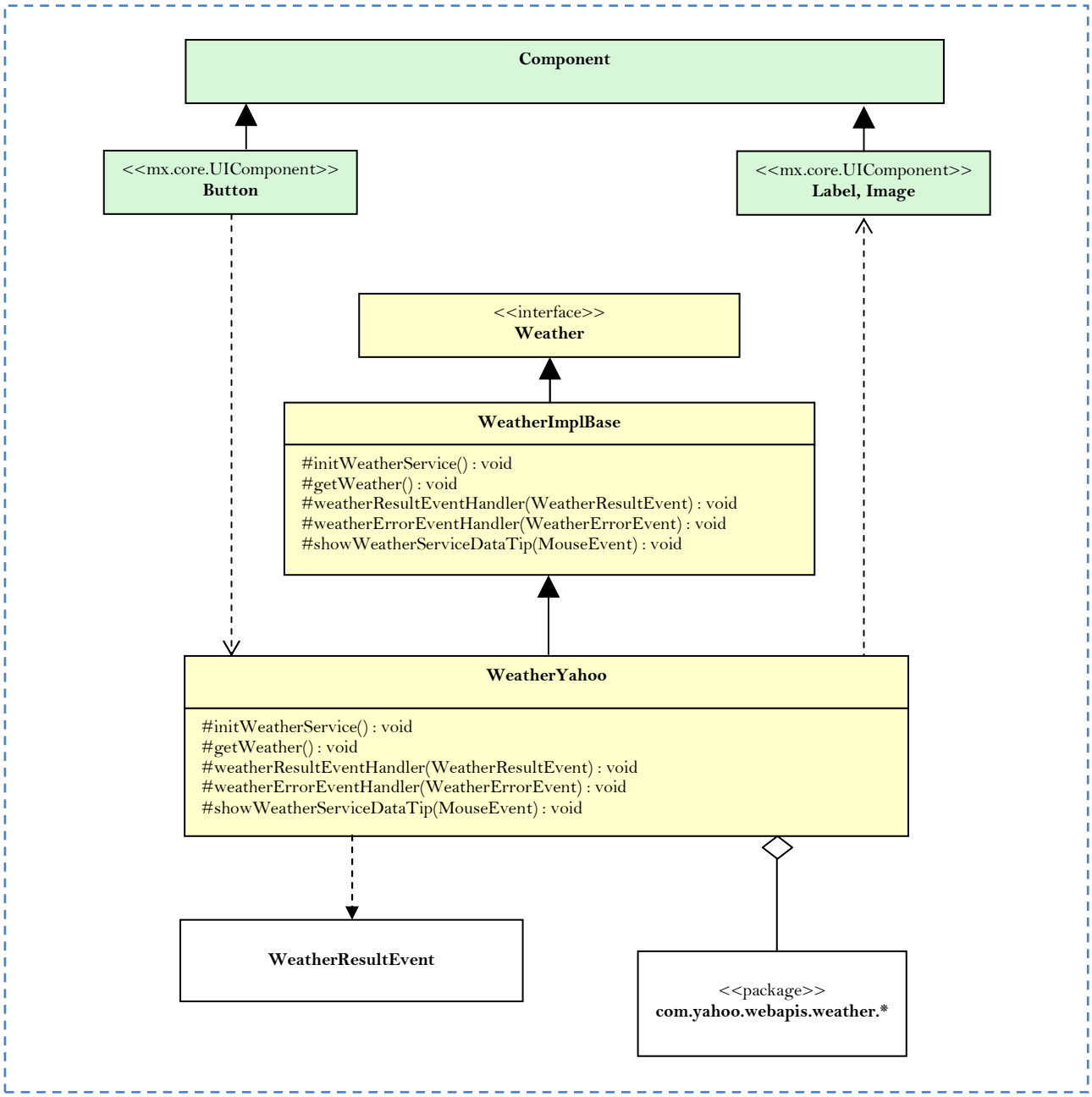

Abbildung 4.18.: WEATHER-Komponente

4.5.4. ANALYTICS-API – Implementierung

Im Rahmen dieser Arbeit wird für die Umsetzung der ANALYTICS-Komponente das kostenlose Web Service und Web-API „Google Analytics“ verwendet (GOOGLE ANALYTICS 2010). Die in Open-Source-Projekt „Google Analytics Library“ (Analytics Library) verfügbaren Klassen *com.google.analytics.AnalyticsTracker* und *com.google.analytics.GATracker* können für die Umsetzung der ANALYTICS-Komponente herangezogen werden.

Abb. 4.19 veranschaulicht, welche Interfaces benötigt werden. Im Gegensatz zu LOGIN- und CLOUD-API-Implementierung wird hier nur ein Interface benötigt, das von der Applikation-Anwendungslogik für das Webtracking verwendet wird.

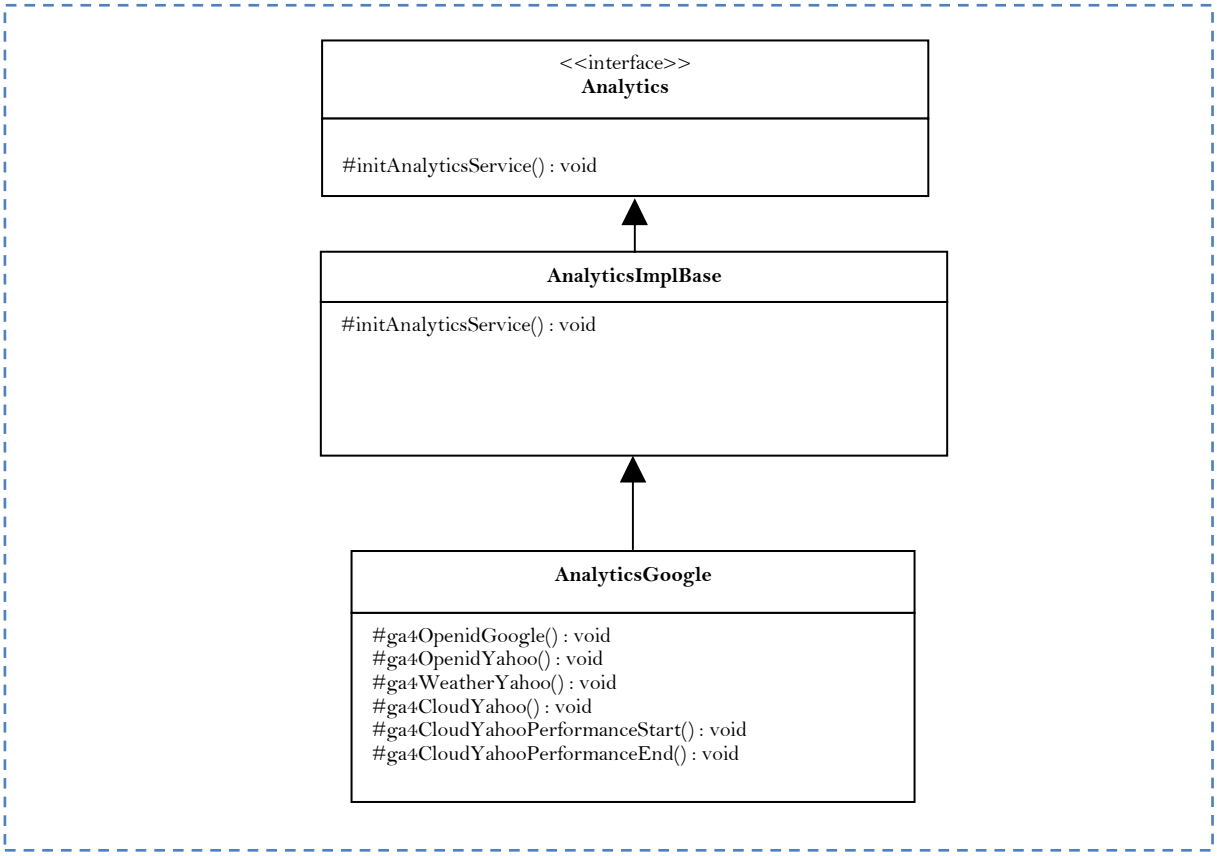

Abbildung 4.19.: Analytics-API

Im Folgenden wird dieses API-Teil entworfen. Hier sollten auch alle ApplilcationAnalytic-API Teile (im Rahmen der Arbeit wird nur ein ApplicationAnalytic-Teil implementiert) ohne Abstraktion konkret implementiert werden. Diese Struktur ist wiederrum leicht erweiterbar: APIs für weitere „ANALYTICS Web Services und Web-APIs“ wie z.B. Amazon-ANALYTICS können analog zu den konkreten Klassen implementiert werden.

Zum Aufrufen vom „GOOGLE Analytic Web Service und Web-API“ wird in (Analytics Library) ein API bereitgestellt. Ein Objekt der Klasse *AnalyticsTracker* wird über Setter-Methoden mit den notwendigen Informationen versorgt, bevor schließlich eine Methode zur Ausführung des Aufrufs aufgerufen wird.

4.5.4.1. ApplicationAnalytics-API aus Applikationsseite

Abb. 4.20 zeigt schließlich eine mögliche Implementierung des ApplicationAnalytic-APIs auf Applikationsseite. Der ApplicationAnalytic-Teil erlaubt der Applikation-Anwendungslogik hier den Zugriff auf den Endpunkt des jeweiligen Analytics-Dienstes, so

dass die Applikation-Anwendungslogik die entsprechenden Methoden bezüglich der „Google Analytics Web Services und Web-APIs“ aufrufen kann.

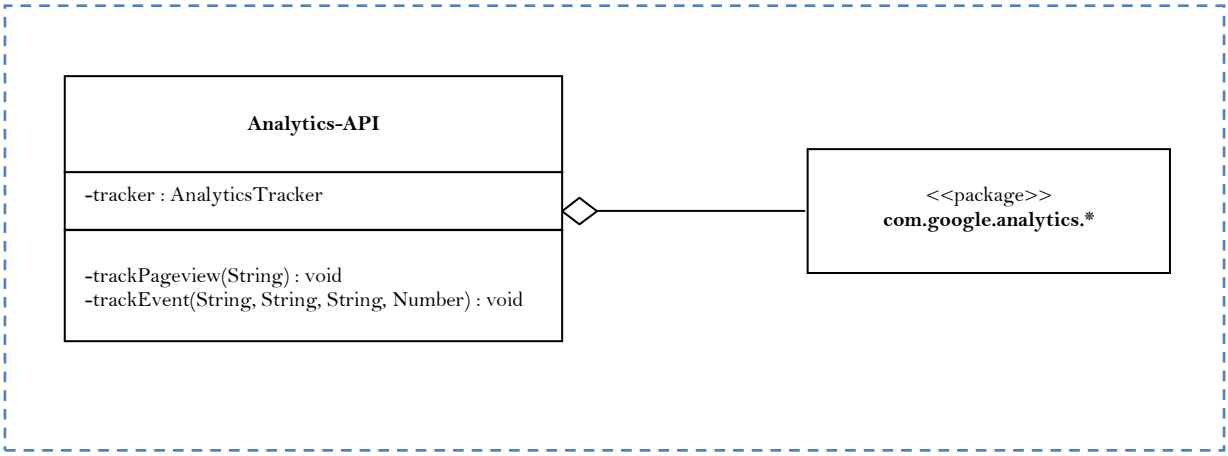

Abbildung 4.20.: Implementierung des ApplicationAnalytics-APIs aus Applikationsseite

Der ApplicationAnalytics-Teil erlaubt der Applikation-Anwendungslogik auf Applikationsseite, der Webapplikation „Analytics-Dienst-Nachrichten“ (GOOGLE ANALYTICS 2010) bezüglich des jeweiligen „Web Services und Web-APIs“ zu senden. Die Applikation-Anwendungslogik überprüft dabei bzw. ist dafür zuständig, dass die zu sendenden Nachrichten gemäß der „Analytics-Dienst-APIs“ aufgebaut sind.

4.5.5. CUSTOMIZATION-API – Implementierung

Im Rahmen dieser Arbeit können für die Umsetzung der Customization-API die in Open-Source-Projekt „Customization Library“ (Customization Library) und Flex-SDK 3.2.0 (Flex SDK 3.2.0) verfügbaren Klassen *com.dougmccune.containers.ResizableWrapper* und *mx.controls.Image*, *mx.core.DragSource*, *mx.managers.DragManager* für die Umsetzung der Customization-Komponente herangezogen werden.

Abb. 4.21 veranschaulicht, welche Interfaces benötigt werden.

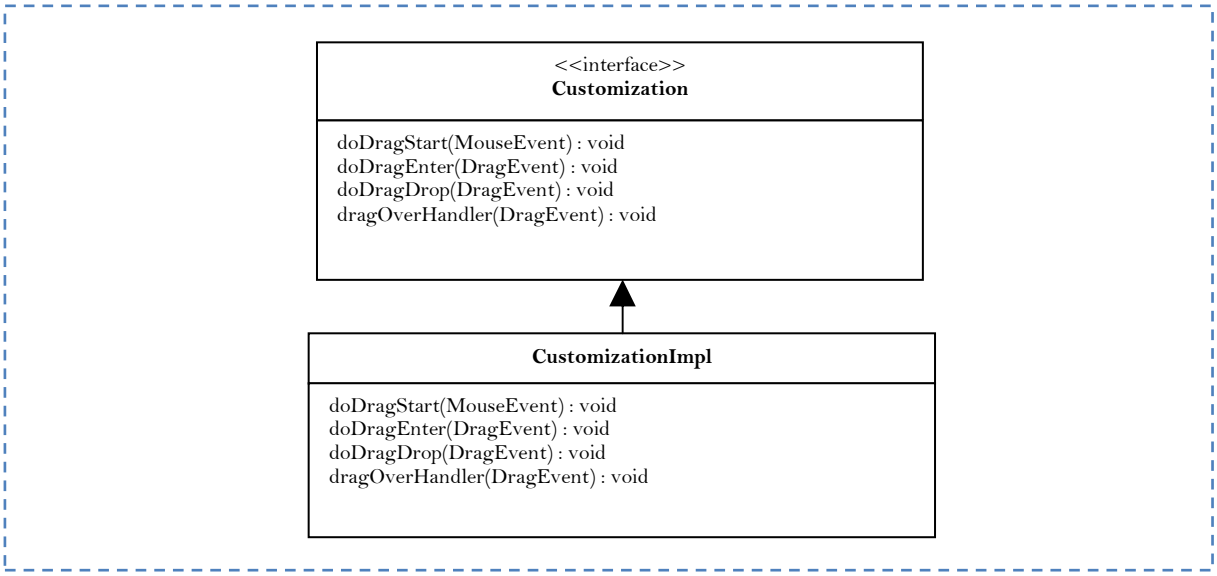

Abbildung 4.21.: Customization-API

Im Folgenden wird dieses API-Teil entworfen. Hier sollte der UserCustomization-API Teil ohne Abstraktion konkret implementiert werden.

Zum Ausführen von DragAndDrop-Funktionen auf der Arbeitsbühne inkl. der Zusammenhänge zwischen CUSTOMIZING-API und CLOUD-API wird in (Flex SDK 3.2.0) ein API bereitgestellt. Ein Objekt der Klasse *DragSource* wird über Setter-Methoden mit den notwendigen Informationen versorgt, bevor schließlich eine Methode doDrag(Parameters) zur Ausführung der Drag-Operation der Klasse *DragManager* aufgerufen wird.

4.5.5.1. UserCustomization-API aus Userseite

Abb. 4.22 zeigt schließlich eine mögliche Implementierung des UserCustomization-APIs auf Benutzerseite. Der UserCustomization-Teil erlaubt der Benutzer-Anwendungslogik hier den Zugriff auf die benötigten Klassen des Customization-APIs, so dass die Benutzer-Anwendungslogik die entsprechenden Methoden bezüglich der externen Klassen (4.5.5) aufrufen kann.

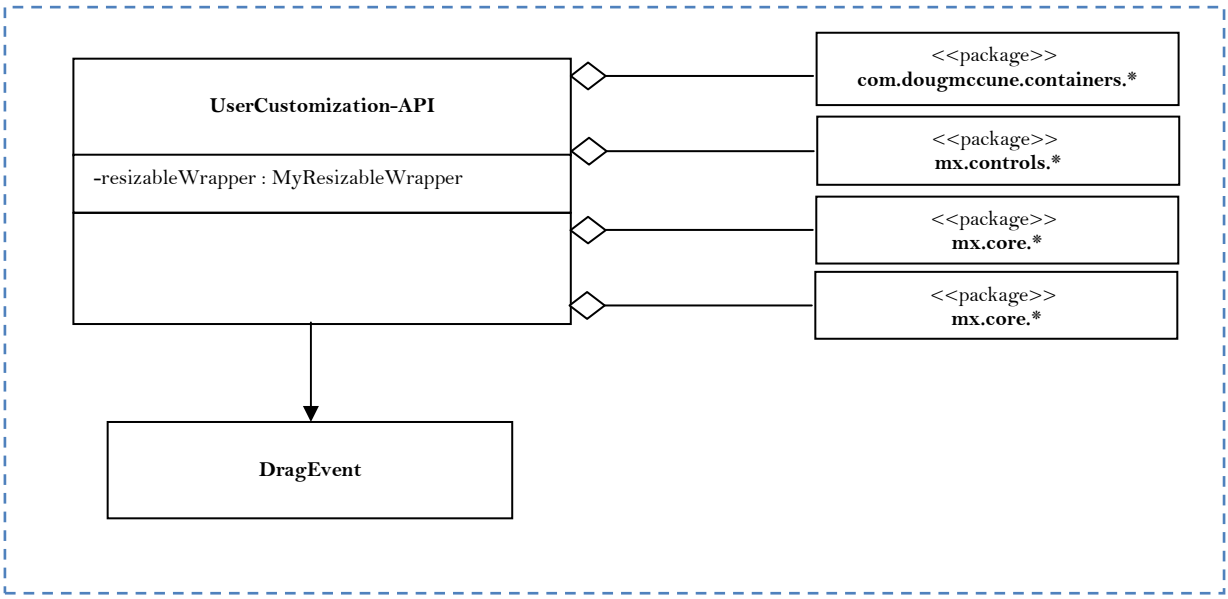

Abbildung 4.22.: Implementierung des UserCustomization-APIs aus Benutzerseite

Der UserCustomization-Teil erlaubt der Benutzerseite über die Listener *DragEvent* und *ResizableWrapper*, Nachrichten (Events) bezüglich der benötigten Bild-Operationen (DragAndDrop- und Skalierungsoperationen) zu senden. Die Benutzer-Anwendungslogik ist dafür zuständig, dass die zu sendenden Nachrichten gemäß der relevanten Anwendungsfälle (3.2.2) abgearbeitet werden.

4.5.5.2. CUSTOMIZATION-Komponente

Abb. 4.23 zeigt schließlich eine mögliche Implementierung der CUSTOMIZATION-Komponente in einer kompletten Darstellung.

Model
Die Klasse CustomizationImpl repräsentiert das Modell der CUSTOMIZATION - Komponente. Sie ist ‚viewlos' und in dem Action Script 3 zu implementieren.

View-Klassen
Canvas (UIComponent) ist die Haupt-UI-Klasse in dieser Komponente. Sie ist in MXML zu implementieren.

Controller-Klassen
Die u.a. Methoden für die Gestaltung des Benutzerprofils sind in den die in Open-Source-Projekt „Customization Library" (Customization Library) und Flex-SDK 3.2.0 (Flex SDK 3.2.0) verfügbaren Klassen *com.dougmccune.containers.ResizableWrapper* und *mx.controls.Image*, *mx.core.DragSource*, *mx.managers.DragManager* aufgewiesen. Sie ermöglichen die Operationen bzw. Funktionen, die für die Anwendungsfälle *‚Bilder Drag-And-Drop'* und *‚Bilder skalieren'* relevant sind (s. 3.2.2). Die andere Controller-Komponente, die implizit in der Webapplikation benutzt wird, ist die Binding-Klasse, die zwischen den View- und Model-Klassen

sitzt: die Binding-Klasse benachrichtigt automatisch die Views, wenn eine relevante Datenänderung in den Model-Klassen stattfindet.

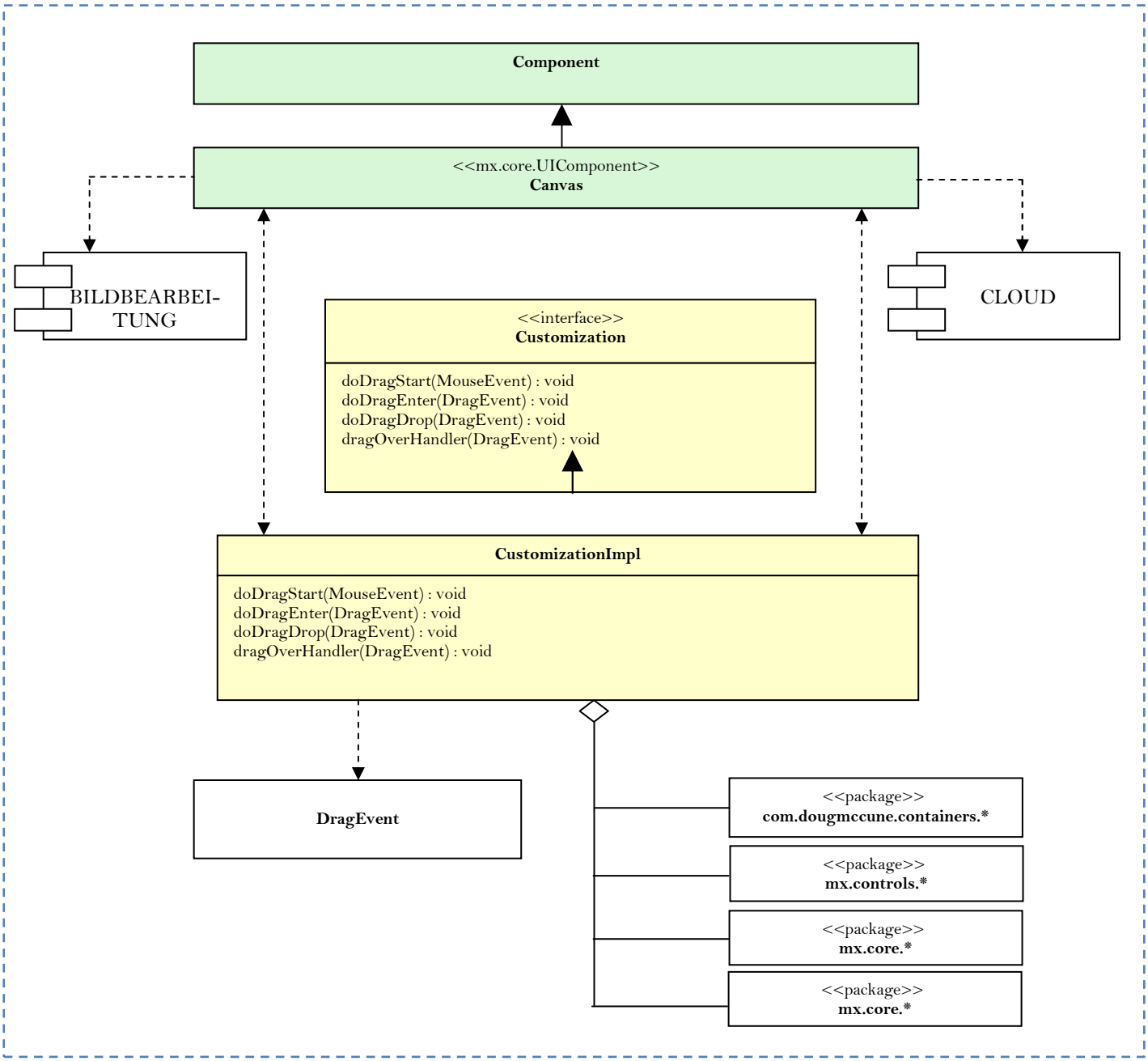

Abbildung 4.23.: CUSTOMIZATION -Komponente

4.5.6. Persistenz

Eine Persistenzschicht für die zu implementierende Webapplikation, die die Daten vorhalten muss (Benutzer- und Bilderdatenbank), wird komplett an die LOGIN-API (4.5.1) und CLOUD-API (4.5.2) delegiert.

Auf Seiten der Benutzer der Webapplikation müssen folgende Daten für die gesamte Session zugreifbar bleiben:

- Benutzerinformationen
- Bilder des Inventars des eingeloggten Benutzers
- Die während der Session relevanten gesammelten Benutzer- Änderungen und Informationen (wird im Rahmen dieser Arbeit nicht behandelt).

Aus Implementierungssicht der Webapplikation ist es eine Menge Arbeit, die durch diese Delegation erspart wird und die zu der Optimierung der drei erarbeiteten Ressourcenkategorien (4.3) beiträgt. Die dadurch möglichen Nachteile wurden bereits in (2.3.1) behandelt.

Auf kleinen mobilen Geräten ist diese Persistenzdelegierung an Web Services und Web-APIs hinsichtlich der erarbeiteten Ressourcenoptimierung (2.1.7) vom großen Vorteil, da die zur Verfügung stehenden Ressourcen der kleinen mobilen Geräte im Vergleich zu PCs oder Laptops deutlich geringer sind und bezüglich der Tatsache, dass sie nicht ununterbrochen online sind.

4.6. Fehler-Handling

An jeder Stelle der Webanwendung und des Frameworks Flex, besonders während der Web Services- und Web-APIs- Kommunikation, können Fehlerbedingungen auftreten. Sofern nicht mit diesen Fehlern gerechnet wird (etwa mit dem Fehlschlagen von Kommunikationsversuchen oder Nachrichtenbearbeitungen), führen sie in Flash-Laufzeitumgebungen zu Exceptions, die von der Stelle des Auftretens solange die Aufrufkette von Methoden zurückgereicht werden, bis sie irgendwo behandelt werden.
Das Framework Flex (Flex SDK 3.2.0) sieht vor, dass Exceptions, ggf. von der Web-Service- und Web-API- Implementierung abgefangen werden und somit für ein stabileres und zuverlässigeres Verhalten der Webanwendung sorgt. An jeder Stelle der Aufrufkette kann die Exception abgefangen werden und es kann überprüft werden, ob das Problem reparabel ist. Ist dies nicht der Fall, so können der Exception Fehlerinformationen für den Benutzer hinzugefügt werden.

Die Webapplikation muss dafür so implementiert werden, dass die Applikation-Anwendungslogik bei den externen Aufrufen ein ausreichendes Fehler-Handling bereitstellt. Tritt dann ein Fehler bzw. eine Exception auf, so wird der Fehler an die Benutzer-Anwendungslogik weiterreicht. Die Benutzer-Anwendungslogik muss dafür so implementiert werden, dass dem Benutzer nur die notwendige Handhabung und so wenig wie möglich Freiheiten bezüglich der Webapplikationsbenutzung zur Verfügung stehen (z.B. ausreichende Validierungen bei Benutzereingaben), damit das Fehler-Handling vollkommen von der Applikation-Anwendungslogik abhängig wird.

4.7. Erweiterbare HTTP-Services und Web Services

Die Implementierung der erweiterbaren HTTP-Services und Web Services dient dazu, die aus 4.3 beschriebenen und für die allgemeine Funktionalität notwendigen Web Services und Web-APIs anzusprechen. Für jeden Web Service und jedes Web-API muss eine entsprechende HTTP-Service-Komponente vorhanden sein. In (Adobe LiveDocs 2009) wird beschrieben, welche grundsätzliche Methoden und Ereignisse (Events) des HTTP-Services und Web Services vom Basis-Framework „Flex“ (SDK 3.2.0) dem Anwendungsentwickler zur Verfügung stehen. Diese Methoden sind für eine detaillierte Betrachtung irrelevant, denn nur die vorhandenen (bereits im Framework implementiert) bzw. erforderlichen (können implementiert werden) Events dieser o.g. Services sind von Bedeutung. Für die allgemeine Funktionalität der Webapplikation muss in den Controller-Methoden nur auf aufgetretene Ereignisse von den benutzten Web Services und Web-APIs eine entsprechende Logik implementiert werden.

Web Services und Web-APIs, an die in der Webapplikation delegiert wird, werden üblicherweise durch Online-Dokumente im Web durch den Anbieter beschrieben. Die darin enthaltenen Informationen werden genutzt, um folgende Implementierungsarbeit aus dem objektorientierten Paradigma dem Anwendungsentwickler zu ermöglichen:

- Implementierung der Methoden(Logik) gemäß der gewünschten Funktionalität der Webapplikation unter Einbeziehung der Information über die angesprochene Schnittstelle für jede Operation des Web Services und Web-APIs, die als Schnittstelle des jeweiligen Dienstes vom Anbieter angeboten wird.

- Klassenimplementierung für komplexe Datentypen, die den Marshalling-Unmarshalling-Prozess und Benutzbarkeit der Daten für den Model vereinfachen.

- Implementierung der eigenen, nicht im Framework vorhandenen Komponenten bzw. Softwaremodule, die schnelleres und bequemeres Darstellen der von der Web-API gelieferten Daten ermöglichen.

Das Ziel des Paradigmas von erweiterbaren HTTP-Services und Web Services im Rahmen dieser Arbeit sollte es sein, die Auswahl an angebotenen Web Services und Web-APIs für den Benutzer im Web möglichst erweiterbar zu machen.

4.8. Fazit

In den vorangegangenen Abschnitten wurde eine Möglichkeit zur Umsetzung der in Kapitel 3 erarbeiteten Anforderungen detailliert entworfen. Dabei wurde das Framework Flex-SDK in der Version 3.2.0 (Flex SDK 3.2.0) für die Realisierung der Technologien Rich Internet Applikation (Flash-Applikation) inklusive der Verwendung von Web Services und Web-APIs verwendet. Dieses Framework hat sich als leicht für die Umsetzung der Zielsetzung (1.2) erwiesen. Die zu entwerfende Webapplikation ist eine Implementierung der Rich Internet Applikation mit RIA-Framework „Flex“ aus (4.4), die Komponenten, Interfaces und

Klassen gemäß 4.5 enthält und Funktionalität bezüglich der in 3.1 und 3.3 erarbeiteten fachlichen funktionalen und nichtfunktionalen Anforderungen bereitstellt. Dazu gehören:

- Ein Model aus vier Komponenten mit jeweils eventuell mehreren HTTP-Services (Web Services und Web-APIs) und einer Customization- und Bildbearbeitungskomponente, der mit Implementierungsklassen die Hauptfunktionalitäten der Webapplikation aus 1.2 bereitstellt.

- Views, die zusätzlich durch eine eigenständige und unabhängige Implementierung nach der CSS-Syntax für das optische Aussehen bzw. Styling der Komponenten zuständig ist.

- Controller, die auch durch eine eigenständige und unabhängige Implementierung nach der Skriptsprache Action-Script 3.0 (Shupe u.a. 2008, Lott u.a. 2007) für die allgemeine Logik der Webapplikation verantwortlich ist.

Ein weiterer möglicher zu entwerfender Teil der Webapplikation sind erweiterbare HTTP-Services bzw. Web Services in Form von konkreten Implementierungen, die den o.g. mehreren HTTP-Services entsprechen und für jeden zu implementierenden Web Service die Rolle des Dienstes und die erforderlichen Daten für den Model erstellt. Dieser Teil ist unbegrenzt für weitere Dienste erweiterbar und unterstützt dadurch das Erweiterbarkeitskonzept der zu implementierenden Webapplikation um weitere Web Services und Web-APIs. Aus Zeitgründen werden erweiterbare HTTP-Services nicht zahlreich entworfen und umgesetzt, sondern nur gemäß der zu implementierenden Dienste aus (4.3) implementiert.

Bezüglich der nichtfunktionalen Anforderungen werden im Rahmen dieser Arbeit spezielle Komponenten nur zur Förderung der Robustheit (Fehlerkomponente), Leistungsfähigkeit, Erweiterbarkeit (HTTP-Services-Komponente bzw. Web Services und Web-APIs-Komponenten) und Benutzbarkeit (UI) (s. Kapitel 4) entworfen und umgesetzt. Es wird außerdem darauf geachtet, die Webapplikation so zu entwerfen, dass bezüglich der übrigen nichtfunktionalen Anforderungen die Implementierung effizient ist, um die Wartezeit der langen Übertragungen von großen Daten so weit wie möglich zu kompensieren.

Durch Verwendung der Muster (ggf. MVC) erfolgt eine saubere Trennung von Verantwortlichkeiten der einzelnen Schichten der Webapplikation. Hinsichtlich der verwendbaren Web Services und Web-APIs wird es viel Logik in prinzipiell allgemeine Logik der jeweiligen Web Service- und Web-API- Kategorie implementiert während weitere, dienstspezifische Logik, in der konkreten Implementierung des Dienstes umgesetzt wird.

Wenn möglich, werden schnell zu implementierende vorhandene Techniken des Frameworks Flex 3.2.0 (Flex SDK 3.2.0) wiederverwendet und nur in Ausnahmefällen verändert.

5. Implementierung

In diesem Kapitel wird beschrieben, welche der im vorigen Kapitel entworfenen Komponenten umgesetzt wurden und bis zu welchem Fertigstellungsgrad dies geschehen ist. Ziel konnte dabei im Rahmen der zur Verfügung stehenden Zeit nicht sein, eine vollständige (alle Werkzeuge und Web Services und Web-APIs mehrerer Anbieter), voll funktionsfähige und sehr benutzerfreundliche Version der Webapplikation zu erstellen. Vielmehr stand im Mittelpunkt, die prinzipielle Realisierbarkeit der Entwürfe zu überprüfen und Beispiel-Code für zukünftige Weiterentwicklungen und Verbesserungen der Webapplikation, besonders bezüglich der Erhöhung der Benutzerfreundlichkeit, zu liefern.

Es wird zunächst kurz auf die Entwicklungs- und Testumgebung sowie auf das Beispielszenario eingegangen (5.1), bevor die Realisierung der Webapplikation beschrieben wird (5.2). Abschließend folgt eine kurze Betrachtung technischer Probleme (5.3).

Auf der beiliegenden CD sind alle für die Einrichtung der Entwicklungs- und Testumgebung benötigten Softwarepakete, der Sourcecode in Form von archivierten Flex-Projekten sowie Installationsanleitungen zu finden.

5.1. Entwicklungs- und Testumgebung

Die Entwicklungsumgebung bestand aus einer Installation von Adobe Flex Builder 3 Pro (Flex Builder 3, Abb. 5.1). Die Entwicklungsumgebung Adobe Flex Builder 3 Pro wurde im Rahmen des akademischen Zwecks bei Adobe kostenlos registriert und benutzt.

Zur Entwicklung der Webapplikation wurde das Flex-SDK in der Version 3.2.0 verwendet (Flex SDK 3.2.0). Das Software Development Kit stellt innerhalb der Adobe Flex Builder 3 Pro Umgebung, die auf der Eclipse-DIE (Eclipse 2010) basiert, die benötigten Bibliotheken und Werkzeuge (z.B. integrierter Flex-Compiler) zur Verfügung, um Flex-Anwendungen für verschiedene Versionen der Flash-Plugins zu entwickeln.

Für die Laufzeitumgebung wurde ein Webserver Apache 2.2.3 von Apache (Apache 2010) zur Ausführung der Website verwendet, in die die implementierte Rich Internet Applikation eingebettet wurde. Der Server bzw. der Zugang zu dem Server und die Deployments-Vorgänge der Webapplikation waren durch das SFTP-Protokoll (SFTP 2010) einfach aus dem SFTP-Client WinSCP (WinSCP 2010) heraus konfigurierbar und steuerbar.

Die Teststellung bestand aus dem o.g. Webserver, der im Rahmen der Hochschule für Angewandte Wissenschaften Hamburg für Studenten mit je eigenem Scope (HAW-Webserver, Sergej Becker) frei zur Verfügung steht und wurde als Laufzeitumgebung für die deployte Webapplikation genutzt sowie einem Browser ((Firefox 2010), (IE 2010)), der die Rolle eines Clients spielte (Abb. 5.2). Man musste auf den lokalen Webserver verzichten, da die

Verwendung der Web Services und Web-APIs eine autorisierte bzw. im DNS-Verzeichnis verfügbare Domäne voraussetzt.

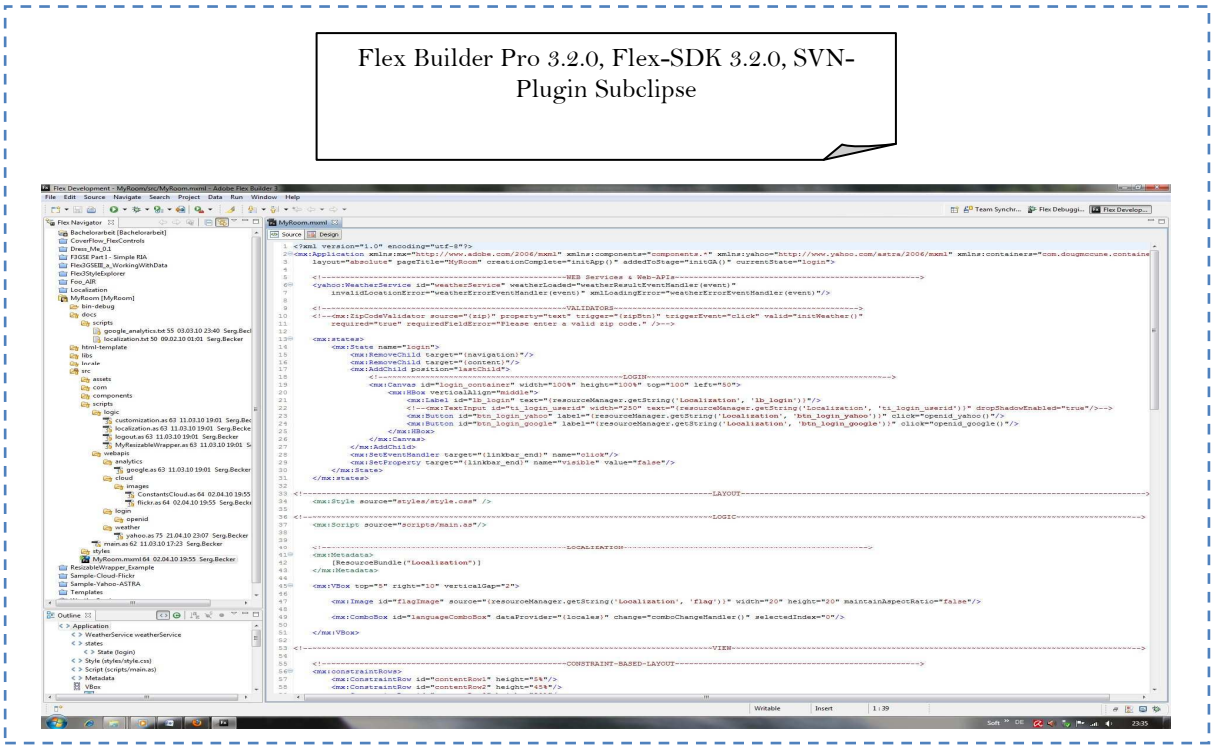

Abbildung 5.1.: Entwicklungsumgebung (eigene Darstellung)

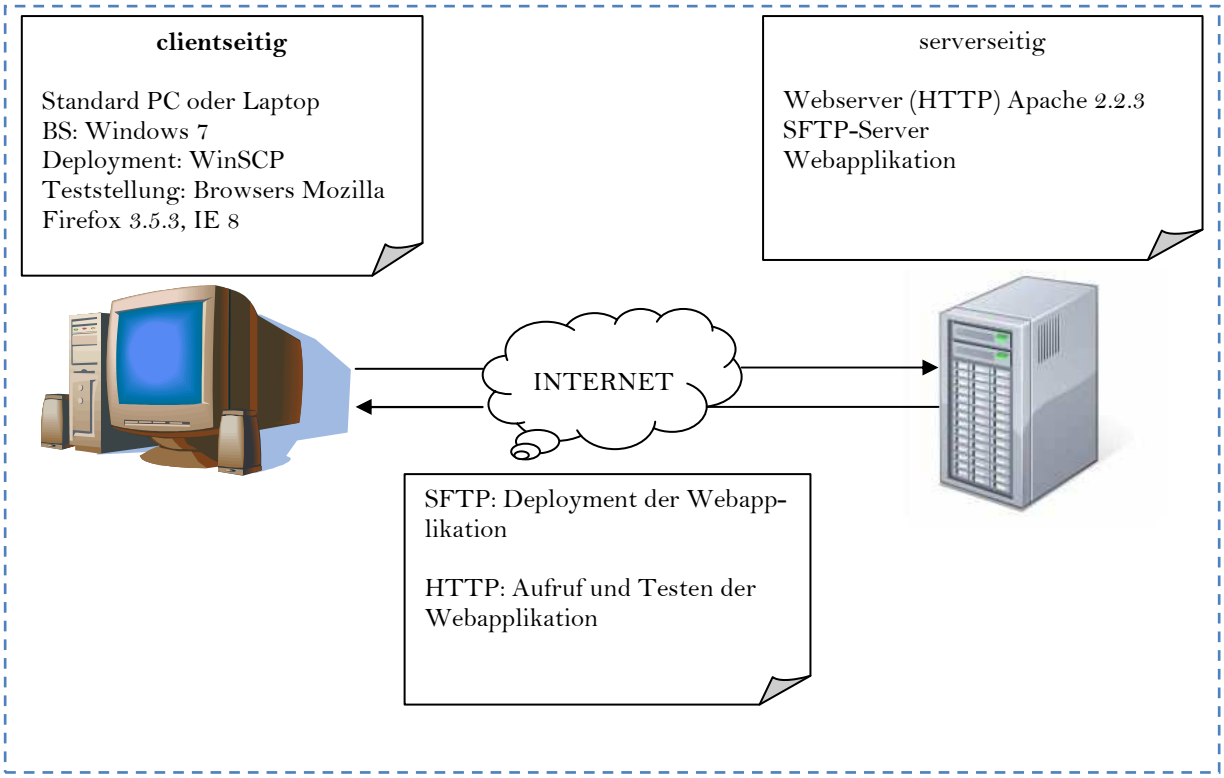

Abbildung 5.2.: Testumgebung (eigene Darstellung)

Im Webserver musste die Rich Internet Applikation, die in eine Website bzw. HTML-Seite eingebettet wurde, aus dem o.g. Studienprojekt installiert werden. Der Sicherheit halber wurden die Zugriffsrechte auf die HTML-Seite überprüft und ggf. angepasst. Zusätzlich musste in die HTML-Seite ein Java-Script-Block hinsichtlich des analytischen Web Services und Web-APIs von GOOGLE (GOOGLE ANALYTICS 2010) für das Web-Tracking eingebettet werden, damit die ANALYTIC-Komponente der Webapplikation im Zusammenhang mit den eigebetteten analytischen JS-Block funktionsfähig wird.

Für den Client-Teil wurden die Browser Firefox 3.5.3 und Internet Explorer 8 ((Firefox 2010), (IE 2010)) mit dem Flash-Plugin in der Version 10.0.32.18 (Flash Player 2010) auf dem stationären und mobilen PC installiert. Zur Verwendung kamen ein Standard- PC und Notebook mit dem Betriebssystem Windows 7 Professional (Windows 7).

Die Webanwendung (Abb. 5.3) verwendet, wie bereits bekannt ist, mehrere Web Services und Web-APIs, die bei seiner Verwendung einen Account voraussetzen. Der LOGIN-API der Webapplikation benötigt jeweils einen Account bei YAHOO und GOOGLE, der für den OpenID-Spezifikation-Zugang freigeschaltet werden sollte. Das CLOUD-API benötigt auch einen Account bei YAHOO-FLICKR (YAHOO FLICKR 2010), der über öffentliche Bilder verfügen sollte. Die analytischen Daten, die seit der Web-Verfügbarkeit gesammelt werden, sind nur für den Eigentümer der Webapplikation zugänglich bzw. unter einem bestimmten statischen Account bei GOOGLE-Analytics (GOOGLE ANALYTICS 2010) registriert und zugänglich. Eine detaillierte Beschreibung folgt einer im abschließenden Kapitel beschriebenen Empfehlung, die auf den Ergebnissen dieser Arbeit beruht (s. 6.1.4).

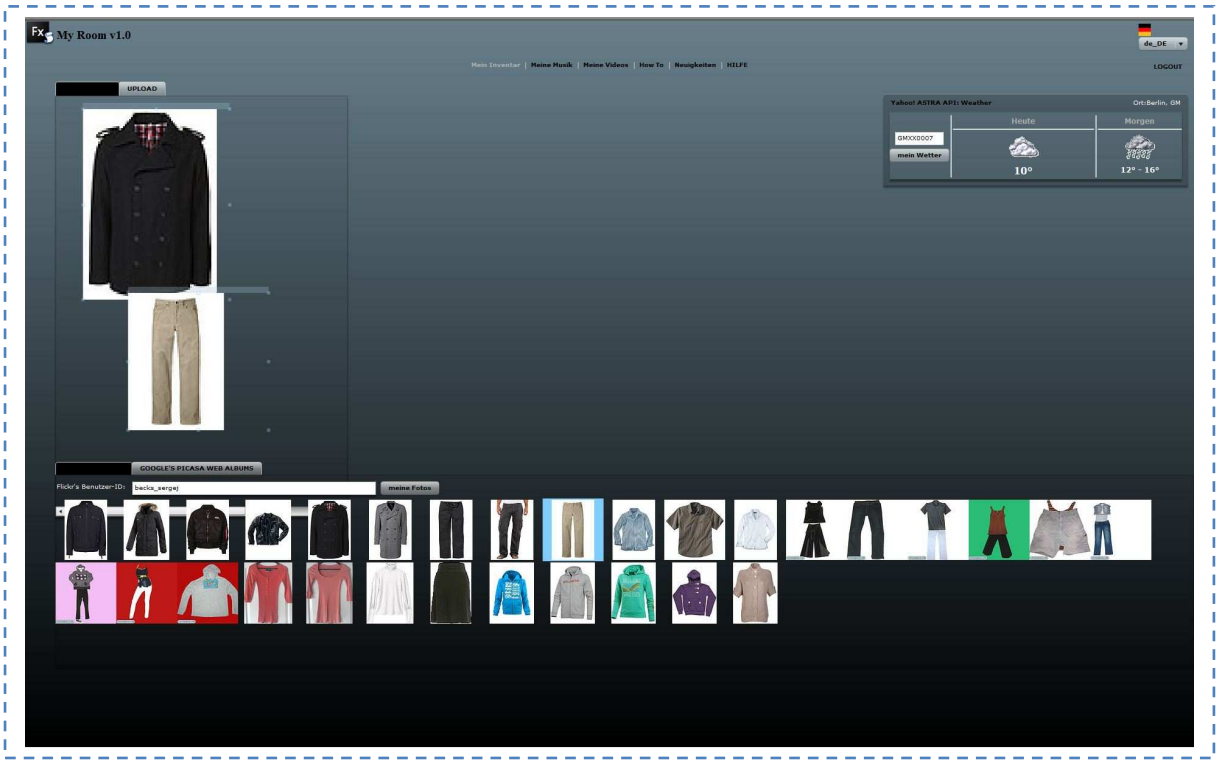

Abbildung 5.3.: UI der implementierten Webapplikation im Zustand ACTIVE (eigener Screenshot)

5.2. Realisierte Webapplikationskomponenten

Die Webapplikation wurde gemäß des Entwurfs in Kapitel 4 in Flex (Flash) implementiert. Im Folgenden werden die Fertigstellungsgrade aller Komponenten der Webapplikation aufgelistet und auf offene Punkte bzw. noch zu vervollständigende Funktionalität hingewiesen.

LOGIN-API – Implementierung Das API für Login- und Logout-Funktionen bzw. die LOGIN-Komponente ist vollständig gemäß Analyse und Entwurf implementiert. Hinsichtlich der verwendeten Web Services und Web-APIs für die Login-Dienstleistung der Spezifikation OpenID 2.0 (OPENID Specifications) ist zu bemerken, dass in der Webapplikation die „Login Web Services und Web-APIs" von GOOGLE (GOOGLE OPENID 2010) und YAHOO (YAHOO OPENID 2010) implementiert wurden.

CLOUD-API – Implementierung Das API für Cloud-Funktion (CLOUD-Komponente) wurde wie entworfen vollständig implementiert.

Es wurde bisher nur ein Web Service und Web-API der Cloud-Dienstleistung von YAHOO (YAHOO FLICKR 2010) implementiert. Weiterhin wurde das API bezüglich der Vollständigkeit des APIs vom Web Service und Web-API nur unvollständig, für die benötigte Funktionalität der Webapplikation jedoch ausreichend implementiert.

WEATHER-API – Implementierung Das API für die aktuelle Wetteranzeige bzw. die WEATHER-Komponente ist vollständig gemäß Analyse und Entwurf implementiert.

Es wurde bisher nur ein WebService und Web-API der Wetter-Dienstleistung von YAHOO (YAHOO WEATHER 2010) implementiert. Hinsichtlich der vom Web Service und Web-API zur Verfügung stehenden Wetterinformation ist zu bemerken, dass die WEATHER-Komponente nur relevante Information für den Benutzer aus der Sicht der Webapplikation bereit stellt.

ANALYTICS-API – Implementierung Das API für analytische Funktion (ANALYTIC-Komponente) wurde wie entworfen vollständig implementiert.

Hinsichtlich der verwendeten Web Services und Web-APIs für die analytische Dienstleistung ist zu bemerken, dass in der Webapplikation der analytische „Web Service und Web-API" von GOOGLE (GOOGLE ANALYTICS 2010) implementiert wurde.

Es wurde die Action-Script-Variante der *AnalyticsTracker-Klasse* des Open-Source-Projektes „Google Analytics Tracking for Flash" (GOOGLE Analytics Library) in der Implementierung verwendet. Die andere verfügbare Variante „Bridge Mode" wäre ein Ansatz für eine serverseitige Webapplikation.

CUSTOMIZATION-API – Implementierung Das API für Gestaltungsfunktionen bzw. für das Anziehen des Benutzerprofils (CUSTOMIZATION-Komponente) wurde wie entworfen vollständig implementiert.

BILDBEARBEITUNG-API – Implementierung Eine Komponente für die Bildbearbeitung bzw. ein Bildbearbeitungswerkzeug wurde aus Zeitgründen und Bildbearbeitungskomplexität nicht implementiert.

Fehler-Handling Fehler (s. 4.6) werden zwar abgefangen bzw. bearbeitet und als Text-Mitteilungen weiter an den Benutzer der Webapplikation gereicht. Die Fehler werden aber nicht genau ausgewertet und nicht alle Fehler-Textmitteilungen entsprechen den tatsächlichen Fehlern, da der mögliche Fehlerumfeld in der Netzwerkumgebung sehr umfangreich ist und die komplette Umsetzung der Fehler-Granularität für diese Arbeit zu komplex gewesen wäre.

5.3. Technische Probleme

Nennenswerte technische Probleme gab es hauptsächlich im Zusammenhang mit den verschiedenen Web Services und Web APIs bzw. mit der mangelnden Dokumentation, welche im Web für die verschiedenen Dienstleistungen verfügbar sind und mit dem umfangreichen Testen der Umsetzung der Webapplikation, da die Verwendung der Web Services und Web-APIs nur von einer registrierten Domäne möglich ist.

Zusätzlich war die Webapplikation hinsichtlich der vorgenommenen Ressourcenoptimierung so entworfen, dass sie zum größten Teil an die verschiedenen Web Services und Web-APIs delegiert wird, die fast alle aus Sicherheitsgründen einen autorisierten Zugang benötigen und damit einen sehr kleinen Overhead verursachen. Die Anwendung wurde daher so gebündelt, dass die Webapplikation nach ihrer eigenen Initialisierung einen Initialisierungsvorgang startet, welcher die zu verwendeten Web Services und Web-APIs für den autorisierten Zugang initialisiert.

5.4. Test und Testergebnisse

Die Webapplikation wurde hinsichtlich der vorgenommenen Ressourcenoptimierung (2.1.7) erfolgreich getestet. Teststellung (5.1), die beim Testen noch um ein mächtiges Analyse-Tool bzw. Plug-In Firebug (Literatur) für den Browser Mozilla Firefox erweitert wurde, ergibt, dass die Zielsetzung erreicht wurde.

Das Netzwerkanalyse-Tool von Firebug ermöglicht die Darstellung vom Netzwerkverkehr und ist somit sehr leicht analysierbar. Aus der Abb. 5.4 kann man entnehmen, dass die Ressourcenoptimierung hinsichtlich der Reduzierung vom Speicherplatz der Webapplikation erreicht wurde. Denn die Webanwendung ermöglicht Verwaltung unbegrenzter Anzahl von Benutzern und ihrer Inventarbilder und weist nur sehr geringen Bedarf von ca. 300 KB Speicherplatz auf.

Die Abb. 5.5, 5.6, 5.7 bestätigen, dass die Ressourcenoptimierung hinsichtlich der Reduzierung vom Traffic erfolgreich erreicht wurde. Nur der erste Aufruf der Webapplikation verursacht den Traffic zum Webserver, wo die Applikation deployt ist. Nur dieser einzige Traffic würde die Kosten für die Webapplikation (Domänendienstleistung) verursachen, die im Vergleich zum gesamten möglichen Traffic sehr gering sind. Die weiteren Aufrufe erfol-

gen für den Webserver der Domänendienstleistung unbemerkbar, weil die Webapplikation clientseitig läuft und mit den externen Web Services und Web-APIs kommuniziert.

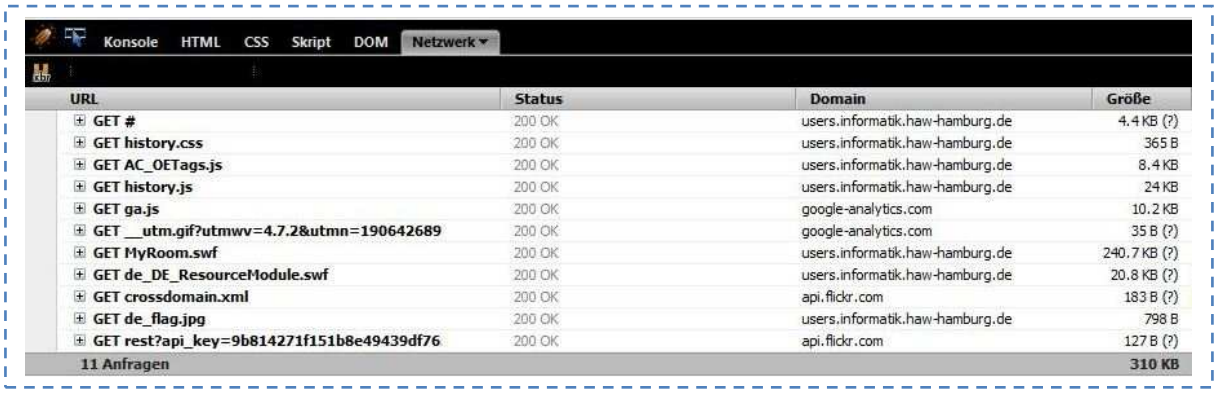

Abbildung 5.4.: Die Traffic-Darstellung nach dem ersten Aufruf der Webapplikation (eigener Screenshot)

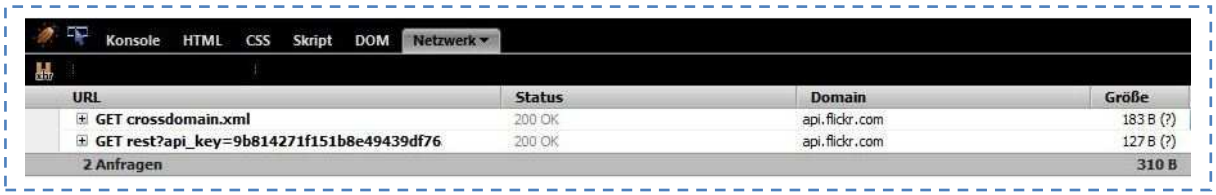

Abbildung 5.5.: Die Traffic-Darstellung nach dem Einloggen (eigener Screenshot)

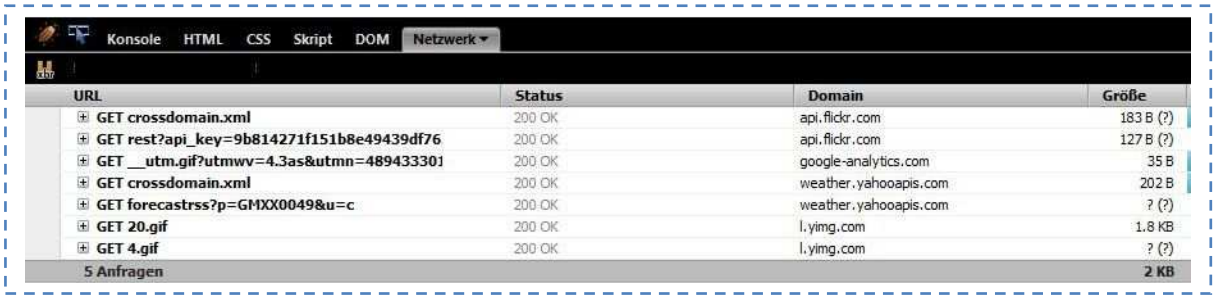

Abbildung 5.6.: Die Traffic-Darstellung nach dem Aufruf der aktuellen Wetterinformationen (eigener Screenshot)

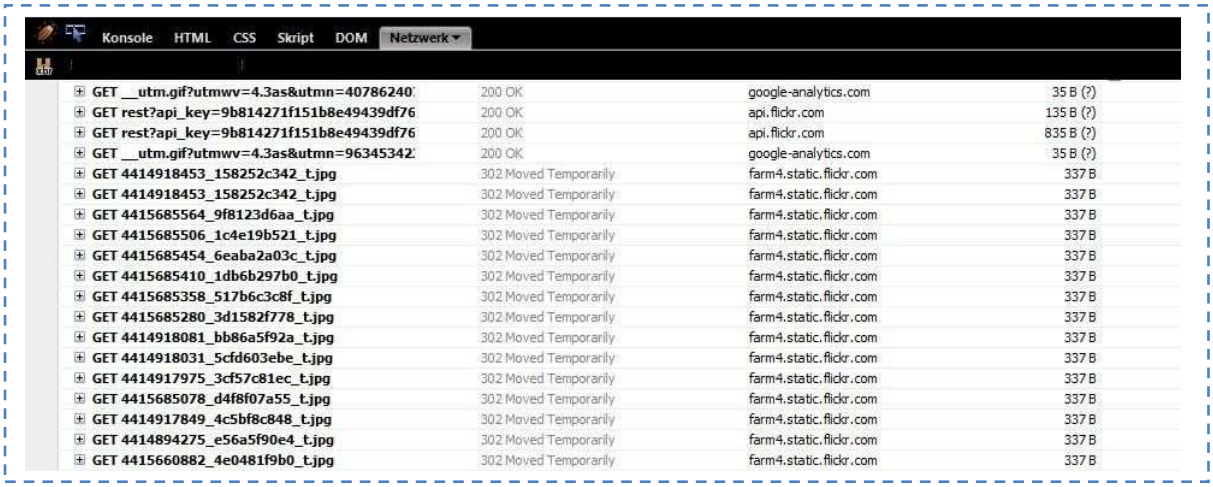

Abbildung 5.7.: Die Traffic-Darstellung nach dem Abrufen der Inventarbilder des jeweiligen Benutzers (eigener Screenshot)

Die Ressourcenoptimierung hinsichtlich der Reduzierung von wiederholenden Arbeiten wurde erfolgreich erreicht. Denn die Implementierung des eigenen Ein- und Ausloggen-Mechanismus für die Webapplikation und die Persistenz-Mechanismen für das Speichern der Benutzerinformationen und ihrer Bilder wurde komplett erspart. Der Implementierungsaufwand wurde dadurch deutlich geringer.

5.5. Fazit

Die Webapplikation wurde wie entworfen soweit implementiert, dass die delegierenden Webapplikationskomponenten gemäß den Spezifikationen funktionieren, sofern die Kommunikationsprobleme im Web (Internet) oder Dienstunerreichbarkeit nicht zum Verlust von Nachrichten führen oder Nachrichten aufgrund von Kommunikationsproblemen nicht gesendet werden können.

Da diese Ausnahmebedingungen sehr selten im Webbetrieb auftreten und die Webapplikation einen hohen Abhängigkeitsgrad der Delegation aufweist, wurden konkrete Komponenten für diese Ausnahmebedingungen vernachlässigt und im Rahmen dieser Arbeit nicht vorgesehen. Stattdessen wurden die verwendeten Web Services und Web-APIs unabhängig voneinander implementiert und dadurch eine hohe Asynchronität erreicht, die das Auftreten dieser Ausnahmebedingungen noch deutlich reduziert.

Bei einer Weiterentwicklung der Webapplikation sollten die Verbesserung von Usability der Komponenten, besonders die CUSTOMIZATION-Komponente, und Webapplikation sowie die Implementierung der Bildbearbeitungskomponente die höchste Priorität haben.

Alle Anwendungsfälle der Webapplikation (3.2.2) wurden erfolgreich mit einem stationären Standard-PC und einem mobilen PC (Notebook) getestet, wobei PCs und Web über einen DSL-Anschluss miteinander verbunden waren.

6. Fazit

Zum Abschluss dieser Arbeit wird zunächst die Erreichung der Ziele aus 1.2 anhand der wichtigsten Ergebnisse der einzelnen Kapitel dokumentiert (6.1), bevor noch ein Ausblick auf mögliche Weiterentwicklungen und Verbesserungen der Webapplikation erfolgt (0).

6.1. Ergebnisse

In 1.2 wurden die folgenden Ziele festgelegt:

- Die Vorstellung der Ressourcen von ressourcenaufwendigen Webapplikationen
- Die kritische Betrachtung mit mathematischen Berechnungen der derzeitigen Performanz, Implementierungs- und Betriebskosten von ressourcenaufwendigen Webapplikationen
- Die Erarbeitung der Ressourcenoptimierung von Webapplikationen unter Einbeziehung existierender Möglichkeiten, um Performanz, Reduzierung der Implementierungs- und Domänenkosten von ressourcenaufwendigen Webapplikationen zu realisieren
- Die Analyse einer Auswahl dieser erarbeiteten Ressourcenoptimierungsmöglichkeit mit dem Ziel am Beispiel der Implementierung einer Rich Internet Applikation (Überbegriff Webapplikation) die Machbarkeit des angestrebten Ziels zu demonstrieren und überprüfen
- Der Entwurf dieser Webapplikation anhand der Analyse, sowie
- die prototypische Umsetzung des Entwurfs zwecks Überprüfung der Realisierbarkeit

In den folgenden Abschnitten werden die Ergebnisse zusammengefasst.

6.1.1. Ressourcenoptimierung von Webapplikationen

Das Web bietet immer mehr und mehr schöne, vielfältige und umfangreiche Webapplikationen. Diese Tendenz erhöht dadurch Anforderungen an die Ressourcen und ist leider proportional zu den Entwicklungs- und Betriebskosten der Applikation. Eine wichtige Rolle spielt dabei die Ressourcenoptimierung von Webapplikationen zum Zweck der Erhöhung der Be-

nutzerfreundlichkeit, des Marketings und Ersparnisses von Entwicklungs- und Betriebskosten von Webapplikationen.
Ressourcen von Webapplikationen fassen vor allem die Speicheranforderung um, die hohe Betriebskosten (z.B. Traffic) verursacht und sorgt für den proportionalen Anstieg der wiederkehrenden Implementierungsarbeiten, wenn eine Webapplikation umfangreicher wird.

Eine Webapplikation weist viele Ressourcen auf und im Rahmen dieser Arbeit wurden nur diejenigen Ressourcen dargestellt und kritisch betrachtet, die für eine Optimierung relevant sind bzw. auf die man einen Einfluss haben kann. Für Ressourcenoptimierung von Webapplikationen existieren auch viele Optimierungsansätze, die ihre Vor- und Nachteile aufweisen. In dieser Arbeit sollte die Möglichkeit der Ressourcenoptimierung untersucht werden, die im verfügbaren Zeitraum für diese Arbeit machbar ist.

Die Auswahl der interessanten Anwendungsklassentechnologie Rich Internet Applikation im Zusammenhang mit der Verwendung der Delegation an zahlreiche Web Services und Web-APIs im Web der Ressourcenoptimierungsmöglichkeiten weist jedoch die geringste Komplexität auf und derjenige Teil der Spezifikation, der sich mit asynchronen Aufrufen von Web Services und Web-APIs befasst, wird bereits in kommerziellen und Open-Source-Produkten (Adobe LiveDocs 2009) unterstützt, weswegen dieser Ansatz für diese Arbeit ausgewählt und näher untersucht wurde.

6.1.2. Analyse von Rich Internet Applikation und Web Services und Web-APIs

Die Analyse der Auswahl der Technologien Rich Internet Applikation und Web Services und Web-APIs hinsichtlich der Umsetzbarkeit für ressourcenaufwendige Webapplikationen als eine der möglichen Ressourcenoptimierungsmöglichkeiten ergibt, dass sie für die vorgenommene Ressourcenoptimierung (2.1.7) geeignet sind. Es müssen jedoch hinsichtlich der Zielsetzung 1.2 einige Punkte besonders beachtet werden:

Clientseitige Anwendung

Die geeignete Ressourcenoptimierung, in der ein großer Wert die Traffic-Optimierung war, ist nur mit der clientseitigen Anwendung erzielbar. Eine clientseitige Anwendung läuft im Unterschied zur serverseitigen Anwendung nicht auf dem Server (z.B. einem Webcontainer), sondern auf dem Rechner des Nutzers ab (s. 2.2). Wie bei allen Systemen, gibt es auch hier seine Vor- und Nachteile:

Vorteile

- Einige Probleme (z.B. Validierung von Formulareingaben) hinsichtlich der Ressourcenoptimierung kann man ohne Beteiligung des Webservers lösen.
- Die clientseitige Anwendung ermöglicht einen Dialog mit dem Benutzer.
- Die Web-Usability der Webapplikation kann gravierend verbessert werden (z. B. wechselnde Inhalte und Darstellung in Abhängigkeit von der Mausposition).
- Direkte Reaktionen der Webapplikation auf Benutzereingaben (z.B. "Wo klickt der Benutzer hin?" oder "Wenn der Benutzer eine bestimmte Taste drückt, macht die Webapplikation folgendes") lassen sich nur mit clientseitigen Webapplikation bzw. Sprachen umsetzen, da der Server von dem clientseitigen Verhalten nichts mitbe-

kommt. Der Server kann erst reagieren wenn er wieder Daten übermittelt bekommt (z.B. durch Absenden einer Request (Server-Anfrage)).

Nachteile

- Der Quellcode bzw. alles, was sich intern befindet, ist für den Client sichtbar und zugänglich. Das kann manchmal sehr unerwünscht sein.
- Der Client bzw. der Browser muss clientseitige Techniken auch beherrschen. Wer z.B. einen alten Browser hat, der noch kein JavaScript kann bzw. nur teilweise JavaScript beherrscht, der kann das auch nicht verwenden. Genauso ist es mit der Java-Laufzeitumgebung und ggf. dem Flashplayer. Je nach Zielgruppe der Webapplikation muss sich der Web-Entwickler gut überlegen, welche Technologien er einsetzt und wie stark die Webapplikation davon abhängig ist.

Die erzielte Ressourcenoptimierung hinsichtlich aller Aspekte der Zielsetzung ist nur im Zusammenhang mit Web Services und Web-APIs bzw. Webdiensten oder einer vergleichbaren Technologie erreichbar.

Web Services und Web-APIs

Bei der Einsetzung von Web Services und Web-APIs sind folgende Vor- und Nachteile zu beachten:

Vorteile

- Kostenvorteil: es werden meistens einige Lizenzkosten vermieden, da offene Standards verwendet werden. Im Bezug auf die erzielte Ressourcenoptimierung der Webapplikation in dieser Arbeit besteht ein sehr riesiger und unvergleichbarer Vorteil in Betriebs- und Entwicklungskosten der Webapplikation.
- Vielerorts einsetzbar und sind somit universell.
- Sind auf jedes Übertragungsprotokoll aufsetzbar (üblicherweise HTTP-Protokoll).
- Sind unabhängig von den verwendeten Plattformen, Programmiersprachen und Protokollen.

Nachteile

- Die Hauptschwierigkeiten bei der Verwendung bzw. beim Ansprechen und Umsetzung von Webservices betreffen Sicherheitsaspekte. So ist beim Verwenden zu beachten, dass einige Webservices nur verschlüsselt benutzbar sind oder eine Authentifizierung und Autorisierung erforderlich sind (führt zum Overhead hinsichtlich der Ressourcenoptimierung).
- Performanz: diese wird durch XML-Parsen und Dateigröße negativ beeinflusst. Der Verwaltungsaufwand nimmt bei stark verteilten Systemen zu. Der Overhead ist teilweise erheblich wie ggf. im Vergleich zu erreichten Ergebnissen.
- Programmiersprachen (ggf. Flex), mit denen man Webservices einbinden will, brauchen spezielle Bibliotheken (Flex ist bereits damit ausgerüstet) z.B. zum schnellen

und bequemen Parsen der XML-Nachrichten bei der Kommunikation mit Web Services und Web-APIs (z.B. DOM (DOM), SAX (SAX)). Schnittstellen müssen genau definiert werden.

Asynchrone Kommunikation

Da die Webapplikation hinsichtlich der ausgewählten Technologien für die Ressourcenoptimierung sehr an die Web Services und Web-APIs angewiesen ist, ist es empfehlenswert, auf die asynchrone Kommunikation mit den Web Services und Web-APIs aufgrund der Ressourcenoptimierung zu achten. Das Framework „Flex SDK 3.2.0“ ist bereits mit Mechanismen ausgestattet, die eine asynchrone Kommunikation mit Hilfe vom Listener-Konzept anbieten.

6.1.3. Entwurf der beispielhaften Webapplikation

Der Entwurf der Webapplikation als prototypische Umsetzung des Entwurfs zwecks Überprüfung der Realisierbarkeit hat gezeigt, dass eine Umsetzung der vorgenommenen Ressourcenoptimierung (2.1.7) für ressourcenaufwendige Webapplikationen, die durch den ausgewählten Realisierungsansatz umsetzbar sind, realisierbar ist.

Durch konsequenten Einsatz von verbreiteten und möglichen für Webapplikation geeigneten Entwurfsmustern und eventuell APIs der verwendeten Web Services und Web-APIs und deren Nutzungsbedingungen (Sicherheitsaspekte) fördert der Entwurf die leichte Wartbarkeit und Erweiterbarkeit der Webapplikation. Die Verantwortlichkeiten der einzelnen Webapplikationskomponenten sowie Applikationslogik werden sauber getrennt.

Der speziellen Problematik hohes Delegationsgrades an Web Services und Web-APIs wird dadurch Rechnung getragen, dass ein flexibler Ansatz für den Aufruf von Web Services und Web-APIs konzipiert wurde, welcher dafür sorgt, dass unterschiedliche und zahlreiche Web Services und Web-APIs (Aufrufstrategien) zum Einsatz kommen können. Aufrufstrategien können z.B. darin bestehen, den Aufruf mehrerer Webdienste der gleichen Logik (z.B. LOGIN-API) auszuführen oder den Aufruf im Fehlerfall mehrfach zu wiederholen. Welche Strategien verwendet werden, sind von dem jeweiligen Benutzer entscheidbar.

6.1.4. Umsetzung und Empfehlungen

Die Webapplikation wurde gemäß den Ergebnissen von Analyse und Entwurf erfolgreich zu großen Teilen umgesetzt. Alle Anwendungsfälle unter der umgesetzten Ressourcenoptimierung konnten erfolgreich getestet werden.

Aus den verschiedenen Betrachtungen in Analyse und Entwurf sowie aus der darauf basierenden Realisierung der Webapplikation resultieren die folgenden Empfehlungen.

Webapplikationen mit hohem Abhängigkeitsgrad (ggf. Web Services und Web-APIs) sollten so entworfen werden, dass ein benutzerfreundliches und ausfallsicheres Fehler-Handling verwendet wird, da dieses der zentrale Kommunikationspartner für den Benutzer der Webapplikation ist.

Weiterhin müssen im Falle komplexer Geschäftsprozesse (ggf. CUSTOMIZING-Logik) ausreichende Unterstützungshilfen implementiert werden, dass komplexe Anwendungsfälle wie „Skalieren der Inventarbilder“ (3.2.2) genügend mit Hinweistexten begleitet werden.

Eine weitere Empfehlung besteht darin, dass für Web Services- und Web-APIs-Formulare, deren Benutzung komplex ist, Feld-Validierungen bereitgestellt werden sollten, die zur erfolgreichen Benutzung des gerade verwendbaren Web Services und Web-APIs beitragen. Ungültige Benutzereingaben würden von dem Validierungsmechanismus der jeweiligen Webapplikationskomponente solange gepuffert, bis sie ausgeliefert werden können.

6.2. Ausblick

In diesem Abschnitt wird abschließend noch auf einige Weiterentwicklungsmöglichkeiten und Anwendungen für die Webapplikation in unmittelbarer Zukunft hingewiesen. Dies ist auch als Anregung für Projekte und Abschlussarbeiten zu verstehen, die in demselben Themenfeld angesiedelt sind.

6.2.1. Fehlende Funktionalität

Eine hohe Priorität sollte zunächst die Verbesserung von Usability in dem Bereich Gestaltungsmechanismus (Customization) und die Vervollständigung der Webapplikation in dem Bereich Bildbearbeitung haben. Die in 5.2 diesbezüglich genannte fehlende Funktionalität sollte implementiert und getestet werden. Dabei ist zu beachten, dass der Bereich Bildbearbeitung komplex ist und wurde aufgrund dessen aus zeitlichen Gründen nicht vollständig implementiert.

Weiterhin wäre zu überlegen, welche Features für die „Anzieh-Anwendung“ und allgemein hinsichtlich der Sachenbearbeitung und Anzieh-Mechanismus sinnvoll sind. Mit Hilfe des Konzepts der Aufrufstrategien, dem Einsatz von Möglichkeiten der Rich Internet Applikation (ggf. Flash) und der Anregungen von unzählbaren schönen Webapplikationen im Web, ließen sich hier bestimmt Verbesserungen erzielen.

Ein Begleitungsassistent in Form der Hinweistexte für den Gestaltungsmechanismus der Webapplikation würde schließlich die Benutzbarkeit der Webanwendung erhöhen.

6.2.2. Weitere Ziele

Ein interessanter Punkt wäre die Untersuchung der Wahrnehmung der Webapplikation von Benutzern im Web hinsichtlich des hohen Abhängigkeitsgrades von Web Services und Web-APIs, die zur erzielten Ressourcenoptimierung der Webanwendung beitragen. Da die Webapplikation noch nicht von zahlreichen Benutzern auf Benutzerfreundlichkeit getestet wurde, wäre ein entsprechender Test ein nahe liegendes Projekt.

6.2.3. Erweiterungen

Die Webapplikation sollte weiterhin um Authentifizierungsmechanismus wie OAuth (s. OAuth Protocol) im Laufe der Authentifizierungsphase erweitert werden, damit die Authentifizierung beim jeweiligen Dienstleister nur einmal stattfindet und die anderen Webdienste ohne Authentifizierung benutzt werden können. Das würde der Performancesteigerung beitragen und aus der Softwareentwicklungssicht die Erweiterung von Web Services und Web-APIs des jeweiligen Anbieters erleichtern.

Abschließend soll auch der Zusammenhang zwischen den verwendeten Web Services und Web-APIs (im Sinne von Usability) noch Erwähnung finden. Die Webapplikation wäre für die Benutzer benutzerfreundlicher und intelligenter, wenn Web Services einen gewissen Abhängigkeitsgrad von den anderen hätten. Der Zusammenhang ist unabhängig von der Applikation-Anwendungslogik und sollte daher auf Ebene der Benutzer-Anwendungslogik implementiert werden.

Literaturverzeichnis

[Adobe 2010] Adobe. Home Page. 2010. – URL http://www.adobe.com/de. - (03.01.2010)

[Adobe wiki 2010] Adobe Learning Resources wiki. 2010. – URL http://learn.adobe.com. – (02.01.2010)

[Adobe LiveDocs 2009] Adobe Flex 3 LiveDocs. 2010. – URL http://livedocs.adobe.com/flex/3/. – (12.11.2009)

[Adobe Flex MXML] Adobe Flex 3 MXML LiveDocs. 2009. – URL http://learn.adobe.com/wiki/display/Flex/MXML. – (12.11.2009)

[Herrington u.a. 2008] Jack Herrington, Emily Kim: Getting Started with Flex 3, O'Reilly Media, 2008

[GOOGLE APIs 2010] Google Information Site. Web Services und Web-APIs. 2010. – URL http://code.google.com. – (01.01.2010)

[YAHOO APIs 2010] Yahoo Developer Network. Web Services und Web-APIs. 2010. – URL http://developer.yahoo.com/everything.html. - (01.01.2010)

[Hauser u.a. 2009] Hauser Tobias, Kappler Armin, Wenz Christian: Das Praxisbuch ActionScript 3, Galileo Press, 2009

[Reinhardt 2009] Reinhardt Gerald: Praxiswissen Flex 3, O'Reilly, 2009

[Scott u.a. 2009] Scott Bill, Neil Theresa: Designing web interfaces, O'Reilly, 2009

[Keefe 2008] Keefe Matthew: Flash and PHP bible, Wiley, 2008

[Shupe u.a. 2008] Shupe Rich, Rosser Zevan: Learning ActionScript 3.0: a beginner's guide, O'Reilly, 2008

[Lubkowitz 2007] Lubkowitz Mark: Webseiten programmieren und gestalten, Galileo Press, 2007

[Lott u.a. 2007] Lott Joey, Schall Darron, Peters Keith: ActionScript 3.0 cookbook, O'Reilly, 2007

[Münz 2005] Münz Stefan: Professionelle Websites, Addison-Wesley, 2005

[Wöhr 2004] Wöhr Heiko: Web-Technologien: Konzepte – Programmiermodelle - Architekturen, dpunkt-Verlag, 2004

[Kersken 2006] Kersken Sascha: Praxiswissen Flash 8: der praxisnahe Einstieg in Flash Professional 8 und Flash Basic 8; mit verständlicher Einführung in ActionScript, O'Reilly, 2006

[Kappel u.a. 2003] Kappel, Pröll, Reich, Retschitzegger: Web Engineering, dpunkt-Verlag, 2003

[W3C 2010] W3C Web Application Description Language (Member Submission 1.1). 2010. – URL http://www.w3.org/Submission/wadl. – (22.05.2010)

[Tomcat 2010] Apache Tomcat. Home Page. 2010. – URL http://tomcat.apache.org. – (22.05.2010)

[Flex-Applikationen 2010] Flex beispielhafte Webanwendungen. 2010. – URL http://flex.org/showcase. – (22.05.2010)

[Ajax-Beispiele 2010] Ajax beispielhafte Webanwendungen. 2010. – URL http://sixrevisions.com/ajax/ajax_techniques/. – (24.05.2010)

[Web-Applikationen 2010] Beispielhafte Webanwendungen. 2010. – URL http://www.pcwelt.de/_misc/galleries/detail.cfm?pk=69302&fk=90692. – (22.05.2010)

[Host4Free 2010] Beispielhafter Domänendienstleister. Home Page. 2010. – URL http://www.host4free.de. – (22.05.2010)

[Hibernate 2010] Hibernate. Home Page. 2010. – URL http://www.hibernate.org/. – (22.05.2010)

[Spring 2010] Spring. Home Page. 2010. – URL http://www.springsource.org/. – (22.05.2010)

[Struts 2010] Struts. Home Page. 2010. – URL http://struts.apache.org/. – (22.05.2010)

[Velocity 2010] Velocity. Home Page. 2010. – URL http://velocity.apache.org/. – (22.05.2010)

[Sitemesh 2010] Sitemesh. Home Page. 2010. – URL http://www.opensymphony.com/sitemesh/. – (22.05.2010)

[OSCache 2010] OSCache. Home Page. 2010. – URL http://www.opensymphony.com/oscache/. – (22.05.2010)

[Wikipedia 2010] Wikipedia. Home Page. 2010. – URL http://www.de.wikipedia.org. – (22.05.2010)

[REST 2010] Wikipedia. REST-Protokoll Page. 2010. – URL http://de.wikipedia.org/wiki/Representational_State_Transfer. – (22.05.2010)

[JavaScript 2010] SELFHTML. JavaScript Page. 2010. – URL http://de.selfhtml.org/javascript/index.htm. – (24.05.2010)

[jQuery 2010] jQuery. jQuery Page. 2010. – URL http://jquery.com. – (24.05.2010)

[J2SE 2010] J2SE. J2SE Home Page. 2010. – URL http://java.sun.com/javase. – (24.05.2010)

[IE 2010] Internet Explorer. Internet Explorer Home Page. 2010. – URL http://www.microsoft.com/germany/windows/internet-explorer. – (24.05.2010)

[HTML5 2010] HTML5. HTML5 W3C. 2010. – URL http://www.w3.org/TR/html5. – (24.05.2010)

[.NET 2010] .NET. Microsoft .NET Home Page. 2010. – URL http://www.microsoft.com/net. – (24.05.2010)

[Safari 2010] Safari. Apple, Browser Safari Home Page. 2010. – URL http://www.apple.com/de/safari. – (24.05.2010)

[Moonlight 2010] Moonlight. Plug-In Moonlight Home Page. 2010. – URL http://www.go-mono.com/moonlight. – (24.05.2010)

[WPF 2010] Windows Presentation Foundation. WPF Page. 2010. – URL http://msdn.microsoft.com/de-de/netframework/aa663326.aspx. – (24.05.2010)

[Flash Player Ubiquität 2010] Flash Player Penetration. März 2010. – URL http://www.adobe.com/products/player_census/flashplayer/version_penetration.html. – (24.05.2010)

[Firefox 2010] Browser Mozilla Firefox. Home Page. 2010. – URL http://www.mozilla-europe.org/de/firefox. – (24.05.2010)

[HTTP 2010] Hypertext Transfer Protocol. HTTP W3C. 2010. – URL http://www.w3.org/Protocols. – (24.05.2010)

[SOAP 2010] Simple Object Access Protocol. SOAP W3C. 2010. – URL http://www.w3.org/TR/soap. – (24.05.2010)

[XML 2010] Extensible Markup Language. XML W3C. 2010. – URL http://www.w3.org/XML. – (24.05.2010)

[Web Services 2010] Web Services Architektur. Web Services Architektur W3C. 2010. – URL http://www.w3.org/TR/ws-arch. – (24.05.2010)

[AMAZON Web-APIs 2010] AMAZON Web Services und Web-APIs. Home Page. 2010. – URL http://aws.amazon.com/de/products. – (24.05.2010)

[YAHOO Web-APIs 2010] YAHOO Web Services und Web-APIs. Home Page. 2010. – URL http://developer.yahoo.com/everything.html. – (24.05.2010)

[GOOGLE Web-APIs 2010] GOOGLE Web Services und Web-APIs. Home Page. 2010. – URL http://code.google.com/intl/de-DE/more. – (24.05.2010)

[UML 2010] Unified Modeling Language. Resource Page. 2010. – URL http://www.uml.org. – (24.05.2010)

[YAHOO OPENID 2010] Yahoo Developer Network. YAHOO OpenID 2.0. 2010. – URL http://developer.yahoo.com/openid. - (01.01.2010)

[YAHOO FLICKR 2010] Yahoo Developer Network. FLICKR API. 2010. – URL http://developer.yahoo.com/flickr. - (01.01.2010)

[YAHOO WEATHER 2010] Yahoo Developer Network. YAHOO! Weather RSS Feed. 2010. – URL http://developer.yahoo.com/weather. - (01.01.2010)

[GOOGLE OPENID 2010] Authentication and Authorization for Google APIs. GOOGLE OpenID 2.0. 2010. – URL http://code.google.com/intl/de-DE/apis/accounts/docs/OpenID.html. - (01.01.2010)

[GOOGLE PICASAWEB 2010] Picasa Web Albums Data API. 2010. – URL http://code.google.com/intl/de-DE/apis/picasaweb/overview.html. - (01.01.2010)

[GOOGLE ANALYTICS 2010] GOOGLE Analytics API. 2010. – URL http://code.google.com/intl/de-DE/apis/analytics. - (01.01.2010)

[Coenraets 2010] Christophe Coenraets Rich Internet Applications, Flex, AIR, Java. 2010. – URL http://coenraets.org. - (01.01.2010)

[Flex SDK 3.2.0] Flex Software Development Kit 3.2.0. – URL http://opensource.adobe.com/wiki/display/flexsdk/Download+Flex+3. - (01.01.2010)

[Flex Builder 3] Flex Builder 3. – URL https://www.adobe.com/cfusion/tdrc/index.cfm?product=flash_builder. - (01.01.2010)

[Apache 2010] Apache HTTP Server. 2010. – URL http://httpd.apache.org/. - (01.01.2010)

[SFTP 2010] Wikipedia. SSH File Transfer Protocol. 2010. – URL http://de.wikipedia.org/wiki/SSH_File_Transfer_Protocol. - (01.01.2010)

[WinSCP 2010] WinSCP. Home Page. 2010. – URL http://winscp.net/eng/index.php. - (01.01.2010)

[HAW-Webserver] Hochschule für Angewandte Wissenschaften Webserver. 2010. – URL http://users.informatik.haw-hamburg.de. - (01.01.2010)

[Sergej Becker] Webserverscope des Benutzers Sergej Becker. 2010. – URL http://users.informatik.haw-hamburg.de/~becker_s/. - (01.01.2010)

[Flash Player 2010] Adobe Flash Player. 2010. – URL http://get.adobe.com/de/flashplayer. – (24.05.2010)

[Windows 7] Microsoft. Windows 7. 2010. – URL http://www.microsoft.com/germany/windows/windows-7/. – (24.05.2010)

[OPENID Specifications] OPENID Specifications. 2010. – URL http://openid.net/developers/specs/. – (24.05.2010)

[GOOGLE Analytics Library] Google Analytics Tracking for Flash. 2010. – URL http://code.google.com/p/gaforflash/downloads/list. – (24.05.2010)

[OAuth Protocol] OAuth Protocol. 2010. – URL http://oauth.net/documentation/. – (24.05.2010)

[DOM] Wikipedia. Document Object Model. 2010. – URL http://de.wikipedia.org/wiki/Document_Object_Model. - (01.01.2010)

[SAX] Simple API for XML. 2010. – URL http://www.saxproject.org/. – (24.05.2010)

[Eclipse 2010] Eclipse Integrated Development Environment. 2010. – URL http://www.eclipse.org/. – (24.05.2010)

[Merbeth 2001] Merbeth Günter: Architekturen von Webanwendungen, 2001. – URL http://www.pst.ifi.lmu.de/lehre/WS0102/web-arch/Teil-1.ppt. - (22.02.2010)

[Customization Library] Customization Library. 2010. – URL http://www.dougmccune.com/360Flex/ResizableWrapper/srcview/index.html. – (24.05.2010)

[Cloud Library] Cloud Library. 2010. – URL http://code.google.com/p/as3flickrlib/downloads/detail?name=flickr-.87.zip. – (24.05.2010)

[Weather Library] Weather Library. 2010. – URL http://p.yimg.com/c/ycs/ydn/flash/pkg/astra-webapis-1.2.0.zip. – (24.05.2010)

[Analytics Library] Analytics Library. 2010. – URL http://code.google.com/p/gaforflash/. – (24.05.2010)

A. Inhalt der beiliegenden CD

Die beiliegende CD beinhaltet Folgendes:

- */bachelorarbeit.pdf*
- /W*ebapplikation:* Sourcecode-Ordner der Webapplikation
    - webapplikation.zip: Flex-Builder-Pro-Projekt (Version 3.2.0) der Webapplikation inkl. des Webapplikation-Sourcecodes. Diese Datei bzw. dieses Flex-Builder-Pro-Projekt wird nach der Demonstration der Webapplikation im Rahmen des Kolloquiums eingereicht.
- */Software*: benötigte Installationspakete für die Entwicklungsumgebung

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 25. Juni 2010 _____

Ort, Datum Unterschrift