Daniel Schmidt

Analyse, Entwurf und Prototyp-Implementation zur Übertragung, Manipulation und Speicherung von Multimedia-Daten im Rahmen einer modernen Java-Webanwendung

DIPLOMARBEIT

HOCHSCHULE MITTWEIDA

UNIVERSITY OF APPLIED SCIENCES

Fachbereich Informationstechnik & Elektrotechnik

Mittweida, 2009

Daniel Schmidt

Analyse, Entwurf und Prototyp-Implementation zur Übertragung, Manipulation und Speicherung von Multimedia-Daten im Rahmen einer modernen Java-Webanwendung

eingereicht als

DIPLOMARBEIT

an der

HOCHSCHULE MITTWEIDA

UNIVERSITY OF APPLIED SCIENCES

Fachbereich Informationstechnik & Elektrotechnik

Mittweida, 2009

Erstprüfer:	Prof. Dr.Ing. Frank Zimmer
Zweitprüfer:	Dipl.Ing Roman Uhlig
Vorgelegte Arbeit wurde verteidigt am:	.....

Bibliographische Beschreibung:

Daniel Schmidt:
Analyse, Entwurf und Prototyp-Implementation zur Übertragung, Manipulation und Speicherung von Multimedia-Daten im Rahmen einer modernen Java-Webanwendung. - 2009. - 91 Seiten Hochschule Mittweida (FH), Fachbereich Informationstechnik & Elektrotechnik, Diplomarbeit, 2009

Referat:
Die Aufgaben dieser Diplomarbeit sind zum einen das Aufzeigen und Untersuchen möglicher Techniken, zur automatischen Ver- und Bearbeitung von Media-Daten in einer Java-Webanwendung, wobei die erfolgreiche Übertragung sowie die persistente Speicherung dieser Dateien ebenso zu berücksichtigen sind, und zum anderen die Umsetzung dieser Kriterien in einem Prototyp. Diesbezüglich liegt der Hauptschwerpunkt auf der Entwicklung des Prototyps und der integrierten automatischen Manipulation der Multimedia-Daten, welche von autorisierten Nutzern über ein Web Frontend auf einen Server geladen und über ein integriertes Web Backend in gewissem Maße angepasst werden können.

Inhaltsverzeichnis

Abbildungsverzeichnis

Quellcodeverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

Ajax	Asynchronous JavaScript and XML
API	Application Programming Interface
BSD	Berkeley Software Distribution
bzw.	beziehungsweise
d. h.	das heißt
DOM	Document Object Model
DVD	Digital Versatile Disc
etc.	et cetera
FTP	File Transfer Protocol
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ICC	International Color Consortium
i. S. v.	im Sinne von
JVM	Java Virtual Machine
u. a.	unter anderem / und anderes
u. ä	und ähnliche
o. ä	oder ähnliche
sog.	sogenannt
VOB	Video Objekt

1 Einleitung

1.1 Inhalt und Aufbau

Seit dem „Web 2.0 Boom“ ist die Integration von Multimedia-Inhalten in Webanwendungen spürbar angestiegen. Dabei ist die Beteiligung bzw. das Mitwirken der Nutzer, wie auch bei den Plattformen Youtube[1] und Flickr[2] zu sehen ist, erforderlich um erfolgreich im World Wide Web überleben zu können. Bis zu dem Zeitpunkt der Bereitstellung und Veröffentlichung dieser Daten im Internet sind einige Schritte, wie die erfolgreiche Übertragung, Speicherung und Verarbeitung der Media-Daten, notwendig.

In dieser Diplomarbeit werden verschiedene Herangehensweisen der Übertragung von Multimedia-Daten und der nachfolgenden Speicherung dieser aufgezeigt und analysiert. Hauptschwerpunkt der Diplomarbeit ist der Entwurf und die Implementation eines Prototyps zur Vearbeitung sowie Bearbeitung von Bild- und Videodaten auf dem Webserver. Diese Manipulation soll automatisch nach der erfolgreichen Speicherung der jeweiligen Originaldatei erfolgen.

1.2 Vorbetrachtung

Für die nachfolgenden Untersuchungen ist die Betrachtung der möglichen Programmiersprachen bzw. Techniken notwendig, wobei eine erste Einschränkung durch die Wahl des Server-Betriebssystems stattfindet. In diesem Fall wird die Nutzung eines unixartigen Systems (z.B. Linux) für die gesamten Untersuchungen und Implementierungen in dieser Diplomarbeit aufgrund von höherer Stabilität, besserer Leistung unter Spitzenbelastung, sowie insbesondere des weitverbreiteten Einsatzes in professionellen Bereichen im Vergleich zu Windows, festgelegt. Weitere wichtige Faktoren für diese Entscheidung sind das Preis-Leistungs-Verhältnis, die Kompatibilität und hohe Betriebszeiten. Insbesondere letzteres kann im Geschätfsleben über Gewinn oder Verlust entscheiden, so sind wiederkehrende und länger andauernde Ausfälle des Servers, zum Beispiel durch Update-bedingtes Neustarten des Systems, verlustbehaftet.

[1] http://www.youtube.com/
[2] http://www.flickr.com/

1.3 Betrachtungen der Programmiersprachen bzw. Techniken

In diesem Abschnitt werden auf drei verschiedene Herangehensweisen der Programmierung eingegangen, welche die Möglichkeit der Verarbeitung von Multimedia-Daten in einer Webanwendung bieten. Das ist zum einen das Common Gateway Interface, zum anderen die Programmiersprache und -plattform Java und zum Schluss die Skriptsprachen, die in den letzten Jahren zunehmend größere Beliebtheit erlangten. Zuvor soll anhand von Pflichtkriterien, welche nachfolgend erläutert werden, ein Anforderungskatalog für die spätere Entscheidung nach der jeweiligen Programmiersprache bzw. Technik aufgestellt werden.

1.3.1 Vergleichskriterien

Für die Auswahl der Programmiersprache bzw. Technik sind folgende gleichgewichtige Kriterien von entscheidender Bedeutung:

- Die Stabilität der Andwendung vor allem während der Verarbeitung von umfangreicheren Multimedia-Daten sollte stets gewährleistet sein. Dabei sind ausreichende Hilfsmittel zur Behandlung von Ausnahmen und Fehlern notwendig.
- Die Anwendung muss die Aufgaben im Bereich der Verarbeitung von Multimedia-Daten in einem angemessen Zeitfenster realisieren können. Dabei sollte der Qualitätsfaktor stets berücksichtigt werden.
- Weiterhin sollte das Backend der Applikation einfach in der Handhabung sein. Das heißt der Aufgabenbereich Wartung ist ein wesentlicher Faktor, z.B. sollten Fehler schnell gefunden werden können. Des Weiteren ist der Punkt Erweiterbarkeit ein Muss für die Anwendung, um z.B. neue Entwicklungen im Bereich der Multimedia-Daten unkompliziert in die Applikation integrieren zu können.
- Das Thema Sicherheit ist ein wichtiger Aspekt im Design von Programmiersprachen.

1.3.2 Common Gateway Interface (CGI)

Die Schnittstelle CGI (Common Gateway Interface) ist ein Standard[3], welcher den Austausch von Daten zwischen Webserver und externen Programmen bezeichnet. Die CGI-Unterstützung ist auf nahezu allen Webservern durch Plattform- und Programmiersprachenunabhängigkeit gegeben. Mit Hilfe dieser Schnittstelle kann ein Programm von dem Webserver gestartet, benutzerspezifische Daten aus dem HTML-Dokument (Hypertext Markup Language), wie z.B. Eingabedaten aus dem Formularelement, übergeben und im Skript ausgewertet und verwendet

[3] CGI Request for Comments: 3875 - http://www.ietf.org/rfc/rfc3875

werden. Die Programmausgabe wird dann nach der Ausführung an den Webbrowser übertragen. Auf diese Art und Weise war es schon in frühen Zeiten möglich, Webseiten dynamisch zu erzeugen, insbesondere durch die Interaktion (Eingaben) mit dem Benutzer, z.B. das Absenden eines Formulares.

Für die Erstellung von CGI-Skripten wird überwiegend die freie und ebenfalls plattformunabhängige Programmiersprache Perl (Practical Extraction and Report Language)[4] verwendet, wobei nahezu alle gängigen Sprachen, wie z.B. C/C++, TCL (Tool command language) und Visual Basic ebenfalls genutzt werden können.

In Bezug auf die nachfolgenden Bearbeitungsmöglichkeiten von Multimedia-Daten in dieser Diplomarbeit sind Schnittstellen in verschiedensten Programmiersprachen, die in Verbindung mit CGI genutzt werden können, vorhanden. Zum einen für die Bildbearbeitung mit der freien Grafikbibliothek GD oder über eine Schnittstelle mit dem Programm ImageMagick, welches im Abschnitt 4.1.3 genauer erläutert wird, und zum anderen im Bereich der Videoverarbeitung mit dem Tool FFmpeg, das im Kapitel 4.2.2 näher beschrieben wird.

Die Beanspruchung der Ressourcen aufgrund der eigenwilligen Abarbeitung von CGI ist ein gravierender Nachteil. Vor dem Initialisieren eines Skriptes wird dieses analysiert und auf Korrektheit überprüft. Wenn ein HTTP-Request das Abarbeiten eines Programmes verlangt, wird dieses bei jedem Request neugestartet, was wiederum sehr zeitaufwendig ist und die Ausführung verlangsamt, insbesonders bei mehreren Serveranfragen, wodurch die Performance erheblich darunter leidet. Eine Weiterentwicklung von CGI ist FastCGI[5], welche mit Common Gateway Interface vergleichbar ist, allerdings wurden deren Performance-Probleme behoben. Ein weiterer wesentlicher Aspekt ist das Thema Sicherheit. Da es sich bei „CGI-Skripten“ um eigenständige Programme handelt, die lediglich über das Common Gateway Interface aufgerufen und ausgeführt werden, liegt die sicherheitsrelevante Programmierung in den Händen des Softwareentwicklers. Somit sind mangelnde Kenntnisse über Sicherheitsprobleme und unsaubere Softwareprogrammierung Kriterien für das Entstehen von Sicherheitslöchern in der Anwendung. Diese können zu erheblichen Sicherheitsrisiken und -problemen, wie das Eindringen von unbefugten Personen oder das Löschen von Dateien auf dem Server führen.

Als weiterführende Literatur zum Thema Perl und CGI ist diese Webseite [17] empfehlenswert.

1.3.3 Java

Java, von der Firma Sun Microsystems[6], ist sowohl als objektorientierte und stabile Programmiersprache als auch Plattform in der heutigen Zeit kaum noch weg zu denken. Java Plattformen sind zum einen die Java Standard Edtition (Java SE), die alle standardmäßigen Entwicklungswerkzeuge, den Compiler und die Laufzeitumgebung beinhalten. Zum anderen die

[4] http://www.perl.org/
[5] http://www.fastcgi.com/
[6] http://www.sun.com

Java Enterprise Edition (Java EE), welche die Java SE insbesondere um den Industriestandard zum Implementieren von serviceorientierten Architekturen (SOA) erweitert und im Bereich der Webanwendungen Verwendung findet. Des Weiteren wird die Java Micro Edition (Java ME) für die Programmierung von Applikationen für mobile Endgeräte (z.B. Handy, PDA), Java Card für Java Applets auf Chipkarten und JavaFX zur Entwicklung von Rich Internet Applications[7] zur Verfügung gestellt.

Durch die Plattformunabhängigkeit kann Java auf den meisten Computern und Betriebssystemen eingesetzt werden. Dabei wird der Quellcode in den Bytecode, einer Befehlssammlung für eine virtuelle Maschine, übersetzt und von der JVM (Java Virtual Machine) ausgeführt. Ein Vorteil von Java ist die Speicherverwaltung durch die Garbage Collection (GC), die in der JVM integriert ist. Die GC verwaltet die von den Java Programmen genutzten Speicherbereiche. Werden diese nicht mehr referenziert, so werden diese Bereiche von dem Garbage Collector wieder freigegeben. Durch diese Vorgehensweise wird dem Softwareentwickler nicht nur Arbeit in Bezug auf die Zuweisung und Freigabe von Speicher für die Java Programme abgenommen, sondern es werden Fehlerquellen schon von Anfang an beseitigt. Das falsche Refenzieren von Speicherobjekten und Auftreten von sog. Memory Leaks, wie es in anderen Programmiersprachen wie C oder C++ möglich ist, kann in Java, da keine Zeiger existieren und somit darüber keine direkte Speicheradressierung realisierbar ist, vermieden werden.

Für die Ausführung der in Java geschriebenen Programme wird die Laufzeitumgebung, d.h. die JRE (Java Runtime Environment), welche aus der Java Virtual Machine und den Java Klassenbibliotheken besteht, benötigt. Die Laufzeitumgebung ist für die meisten Plattformen, wie z.B. Windows, Linux und Mac, verfügbar und führt nach dem Kompilieren der Java Quelldateien den Bytecode aus. Diese Schritte sind in der Abbildung 1.1 in Bezug auf die Plattformunabhängigkeit dargestellt.

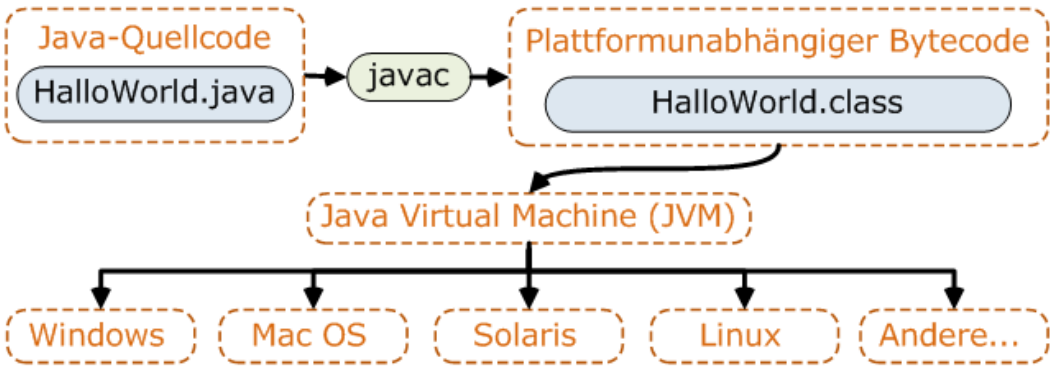

Abbildung 1.1: *Java - Plattformunabhängigkeit[5]*

Ein weiteres wesentliches Merkmal von Java ist die Multithreading Eigenschaft, die von sehr wenigen Programmiersprachen unterstützt wird. Dem Programmierer wird dadurch die Möglichkeit gegeben, Aufgaben bzw. Arbeitsvorgänge parallel oder quasi-parallel ausführen

[7] Anwendung mit intuitiver Benutzeroberfläche unter Verwendung von Internet-Techniken

zu können. Auf diese Weise können Applikationen leistungsfähiger gestaltet werden. Am Beispiel einer Internetanwendung lässt sich diese Eigenschaft gut darstellen: dabei aktualisiert ein Thread[8] die Bildschirmausgabe während ein weiterer die erhaltenen Daten im Hintergrund auf dem Server verarbeitet und speichert.

Die Programmiersprache Java findet in der Entwicklung von Webanwendungen in dem professionellen Bereich nicht zu Unrecht Verwendung. Durch den Einsatz von Java Servlets und Java Server Pages (JSP) auf der Serverseite lassen sich stabile und komplexe Webapplikationen erzeugen. Servlets sind Java Programme die durch den Servlet-Container verwaltet werden und all die Vorteile des Common Gateway Interface ohne deren Nachteile in sich vereinen. Mit Hilfe der Servlet-Engine steht im Webbereich eine Möglichkeit einer sehr schnellen Verarbeitung der Client-Anforderungen zur Verfügung, sowie die Bereitstellung und Verwendung von persistenten Datenbankverbindungen bzw. Verbindungspools und darüber hinaus die weitere Nutzung von Instanzen aufgerufener Servlets, welche für Requests sofort bereitstehen. Diese Verfahrensweise ist lediglich in der Programmiersprache Java gegeben und bietet einen deutlichen Vorteil gegenüber Skriptsprachen und dem Common Gateway Interface. Ein weiterer großer Vorteil ist die gemeinschaftliche Entwicklung neuer Java-Standards und Erweiterung bzw. Änderung existierender Java-Spezifikationen. Diese werden als Java Specification Request (JSR) über das Verfahren Java Community Process (JCP)[9] definiert und eingereicht. Die Java Servlet 2.4 Spezifikationen sind im JSR 154 und die JavaServer Pages 2.0 Spezifikationen im JSR 152 definiert.

In Bezug auf die Manipulation lässt sich mit der Programmiersprache Java sehr viel erreichen. Durch die umfangreichen Klassenbibliotheken und Java APIs (Application Programming Interface) sind komplexe Verarbeitungsmöglichkeiten für Multimedia-Daten in Java gegeben. Zum Beispiel lassen sich Bilder durch die proprietäre Java API JAI (Java Advanced Imaging)[10], Java Image I/O[11] einfach einlesen, bearbeiten und speichern. Für die Verarbeitung von Videos sind ebenso Möglichkeiten mit Hilfe der Java-Bibliothek Java Media Framework (JMF)[12] gegeben. Weitere und detailliertere Informationen zu Manipulationen bzw. Verarbeitungen von Multimedia-Daten sind ausführlich im Kapitel 4 beschrieben.

Zusammenfassend ist Java eine mächtige Programmiersprache, die sicheres und sauberes Programmieren von komplexen Aufgaben insbesondere im Bereich der Internetanwendungen ermöglicht.

[8] Thread (leichtgewichtiger Prozess): ist ein Teil eines Prozesses
[9] Java Community Process - http://www.jcp.org
[10] http://java.sun.com/javase/technologies/desktop/media/jai/
[11] http://java.sun.com/javase/6/docs/technotes/guides/imageio/
[12] http://java.sun.com/javase/technologies/desktop/media/jmf/

1.3.4 Serverseitige Skriptsprachen

Serverseitige Skriptsprachen werden, wie im Namen schon erwähnt, auf dem Webserver ausgeführt. Da diese Skripte meist in Quelltextform für eine schnelle Bearbeitung vorliegen und in der Regel nicht separat von einem Kompiler übersetzt werden, ist vor jeder Ausführung eines jeden Skriptes die Interpretation des jeweiligen erforderlich. Die Ausführungsgeschwindigkeit die ein Interpreter benötigt, um ein Skript zur Laufzeit zu „übersetzen“ und anschließend auszuführen, ist von Grund auf der Performance eines kompilierten Programmes unterlegen. Des Weiteren werden aufgrund dieses Verfahrens Fehler im Programmcode erst zur Ausführungszeit eines konkreten Teils im Skript erkannt.

Die Verwendung von serverseitigen Skriptsprachen ist aufgrund der relativ schnellen und einfachen Erstellung von kleineren dynamischen Webprojekten und Webanwendungen sehr beliebt. Programmcode und -logik wird oft an die entsprechenden Stellen im HTML-Dokument eingebunden, wodurch unkompliziert dynamisches Verhalten einer Webseite bzw. -anwendung erzeugt werden kann. Diese Tatsache birgt wiederum Einbußen in Bereichen der Wartbarkeit und Fehlerkorrektur. Ebenso kann die Erweiterbarkeit der Anwendung durch solch unstrukturiertem Programmieren erschwert werden.

Weiterhin verzichten viele Skriptsprachen auf statische Typisierung. Die dynamische Typisierung, oder auch schwache Typisierung genannt, ist ein weiteres Merkmal, welches oft in Skriptsprachen anzutreffen ist. In diesen Fällen wird den Variablen der Datentyp zur Laufzeit zugewiesen, wobei eine Änderung des Datentyps während dieser Gültigkeitsdauer ebenso möglich ist. Diese Tatsache ermöglicht im Gegensatz zu einer statisch typisierten Programmiersprache viele Fehler- und Problemstellen bei der Programmierung.

Nachfolgend werden einige ausgewählte und in der Praxis oft anzutreffende Skriptsprachen im Webanwendungsbereich genannt:

- Perl (Practical Extraction and Report Language)
- Php (Hypertext Preprocessor)
- Python
- Ruby
- Tcl (Tool Command Language)
- VBScript (Visual Basic Script, sowohl auf der Client- als auch auf der Serverseite (in ASP (Active Server Pages)) im Einsatz)

In Bezug auf die nachfolgende automatische Verarbeitung stehen wiederum einige Hilfsmittel den meisten Skriptsprachen zur Verfügung. So kann die Grafikbibliothek GD in den meisten Fällen verwendet werden. Des Weiteren existieren zahlreiche Schnittstellen für das Grafikprogramm ImageMagick, so zum Beispiel für PHP die Erweiterung „Imagick“, für Perl

„PerlMagick“ und für Python „PythonMagick“, um mal drei Möglichkeiten zu nennen. Eine vollständige Interface-Liste aller unterstützten Programmiersprachen für das Programm ImageMagick ist auf der Webseite[7] zu finden. Für die Verarbeitung von Videodaten wird in der Regel auf das externe Programm FFmpeg mittels Schnittstellen zurückgegriffen. Am Beispiel von Php ist die skriptsprachenspezifische Schnittstelle FFmpeg-phpclass[13].

1.3.5 Fazit

Zusammenfassend kann gesagt werden, dass mit allen drei Techniken zur Entwicklung und Ausführung von Webanwendungen einige gute Möglichkeiten gegeben sind, um automatische Verarbeitungen von Multimedia-Daten realisieren zu können.

Im Bereich der dynamischen Webanwendungsentwicklung war die Skriptsprache Perl in Verbindung mit dem Common Gateway Interface lange Zeit vorherrschend. Die Programmiersprache PHP hat sich seit der Version 3 deutlich entwickelt und durch den Vorteil des leichten Einstiegs in die Sprache den Einfluss im Webbereich ausgebaut. Inzwischen hat PHP die Skriptsprache Perl im Webanwendungsbreich weitgehend abgelöst, ein Grund dafür ist zum Beispiel die leicht zu erlernende Syntax und die Möglichkeit der Verwendung von PHP als Servermodul, wodurch eine performante und ressourcenschonende Arbeitsweise realisiert werden kann. Insbesondere die letzten beiden Punkte sind Kriterien, die eindeutig gegen eine Nutzung von CGI sprechen.

Die zur Zeit populärste serverseitige Skriptsprache zur Entwicklung von Webanwendungen ist PHP. Aufgrund der einfachen Syntax und der guten Webserver-Unterstützung bietet PHP sehr gute Einstiegsmöglichkeiten für die Erstellung von dynamischen Webseiten und hat sich als Skriptsprache im Serverbereich gegenüber anderen bewährt. Ein Nachteil bzw. Schwachstelle ist die schwache Modularisierung, bei der einige PHP Modulentwickler eigene Konzepte entwickelt oder Module speziell auf ein Framework abgestimmt haben. Weiterhin geht der Entwicklungsprozess sowie das Auffinden und Beheben von Sicherheitslücken zögernd voran[18]. Ab der PHP Version 5.0 wurde das Objektmodell und die Ausnahmebehandlung (Exception-Handling) hinzugefügt, wobei in Bezug auf letzteres keine vollständige Implementierung und Unterstützung gegeben ist, so werden lediglich von wenigen objektorientierten Erweiterungen Exceptions verwendet[19]. Hinsichtlich der Wartbarkeit ist eine absolute Trennung von Design und Programmlogik zu empfehlen, dies ist momentan mit der Skriptsprache PHP nicht vollends möglich.

Java ist eine gut strukturierte, leistungsfähige und objektorientierte Programmiersprache und wird vor allem in Unternehmen sowohl für kleinere als auch größere Projekte verwendet. Sie ist zukunftssicher, modern und enthält umfangreiche standardisierte Bibliotheken. Zugriffe auf Datenbanken sowie Netzwerke sind durch umfassende Unterstützungen unkompliziert möglich. Durch die Objektorientierung wird eine deutlich bessere Modularisierung gefördert, wodurch

[13] http://phpvideotoolkit.sourceforge.net/

eine gute Kombinierbarkeit erreicht wird und parallele Entwicklung möglich ist. Ein weiterer wesentlicher Vorteil ist die Erweiterbarkeit sowie die Möglichkeit der guten Strukturierung und Trennung von Desgin und Programmlogik. Diese strikte Trennung ist ein entscheidender Schritt zu einer besseren Wartbarkeit. Des Weiteren wird die Stabilität der Java-Programme durch Verwendung des einheitlichen Exception-Handlings gewährleistet.

Anhand der oben genannten Vergleichskriterien und den Ergebnissen aus den Recherchen ist in Bezug auf die für die automatische Verarbeitung von Multimedia-Daten in dieser Diplomarbeit zu verwendende Programmiersprache Java einwandfrei geeignet. Aus diesem Grund soll die Implementation des Prototyps in Java erfolgen.

Im nachfolgenden Kapitel „Übertragung der Daten auf einen Webserver“ werden die Möglichkeiten der Übertragung von Multimedia-Daten von Clients auf einen Webserver betrachtet und ebenso eine Entscheidung für eine Upload-Variante anhand von aufgestellten Pflichtkriterien gefällt.

2 Übertragung der Daten auf einen Webserver

Die Multimedia-Daten sollen von Clients über eine Webseite auf den Webserver geladen werden. Dabei ist die vollständige und somit sichere Übertragung dieser Daten zu gewährleisten. Die Übertragung der Daten über eine Webseite auf einen Webserver bietet verschiedene Möglichkeiten der Realisierung. Aus diesem Grund sollen zunächst einige zur Verfügung stehenden Upload-Varianten näher betrachtet werden.

Bevor auf die einzelnen Möglichkeiten eingegangen wird, soll das eigentliche Problem, welches bei der Übertragung von Daten von einem Webbrowser an einen Webserver auftreten kann, näher beleuchtet werden.

2.1 Timeout - Zeitüberschreitung / Zeitbeschränkung

In Bezug auf das Hochladen größerer Dateien kann es vorkommen, dass der Upload-Vorgang nach einer gewissen Zeit abgebrochen wird. Eine Ursache dafür kann ein Timeout sein, welches für eine gewisse Zeitspanne steht, die ein Prozess bzw. Vorgang in Anspruch nehmen darf, bis dieser mit einem Fehler abgebrochen wird.

Diese Zeitbeschränkung definiert eine festgelegte „Arbeitszeit“ für bestimmte Vorgänge bzw. Prozesse und ist vor allem in Bezug auf die Sicherheit und Stabilität des Systems/Servers sinnvoll. Zum einen wird dadurch die Lebensdauer von Prozessen beschränkt, um zu vermeiden, dass ein Vorgang, der auf ein bestimmtes Ereignis wartet, welches unter Umständen in näherer Zukunft nicht eintreten wird, „hängen“ bleibt. Ohne diese Methode kann die Sicherheit und Stabilität des Servers nicht mehr gewährleistet werden. Ebenso ist es auch für den Browser sinnvoll nach einer gewissen Zeitdauer des Wartens auf eine Antwort des Servers mit einer dementsprechenden Timeout-Meldung abzubrechen und den Nutzer zu informieren.

Um das Hochladen von größeren Dateien zu ermöglichen, sind Einstellungen am Server notwendig. So zum Beispiel können die Ausführungszeit der Vorgänge und die Dateigrößenbeschränkungen beim Hochladen eingestellt bzw. verändert werden, um einen reibungslosen Ablauf zu gewährleisten.

2.2 Upload über das HTML-Formular

Die einfachste Art und Weise einen Datei-Upload zu ermöglichen, ist die Nutzung des „Input"-Tags vom Typ „file" in einem HTML (Hypertext Markup Language) Formular.

```
<form method="POST" enctype="multipart/form-data">
    <input type="file" name="datei" />
    <input type="submit" name="submit" value="Absenden" />
</form>
```

Beispiel 2.1: *Upload mittels HTML Formular*

Dabei wird ein Eingabefeld im Webbrowser angezeigt und der Nutzer hat die Möglichkeit die Datei einzugeben. Die Datei wird mit Hilfe des HTTP Protokolls und der Angabe des Übertragungsformates „multipart/form-data" sowie der POST Methode, wie im Beispiel 2.1 zu sehen ist, übertragen. Die POST Methode ermöglicht das Übertragen von deutlich größeren Dateien im Vergleich zur GET Methode und ist obendrein sicherer, da die Formulardaten im Header übermittelt und nicht wie bei GET an die URL angehängt werden. Da diese Variante auf reinem Html-Code basiert, ist damit die größtmögliche Browser-Kompatibilität gewährleistet. Beim Hochladen von größeren Dateien ist der Fortschritt nur sehr schlecht bzw. gar nicht nachzuvollziehen, was wiederum die Benutzerfreundlichkeit in den Schatten stellt.

Nachfolgend soll auf die JQuery Upload-Variante eingegangen werden.

2.3 Upload mittels JavaScript Framework JQuery

JQuery ist ein sehr umfangreiches freies JavaScript Framework. Es besteht aus einer einzigen JavaScript-Datei, welche alle grundlegenden Funktionen beinhaltet. Dabei beginnt in JQuery jede Funktion mit einem Dollar Zeichen ($). Mit dieser Funktion können mit Hilfe von Selektoren (im Beispiel 2.2 über den Elemente-Namen „a") ein oder eine Reihe von DOM (document object model) Elementen ausgewählt werden. Mit diesem JavaScript Framework lassen sich in eleganter und simpler Weise DOM Elemente manipulieren, das heißt deren Eigenschaften(z.B. CSS Attribute) verändern, Elemente hinzufügen oder entfernen. Des Weiteren ist die Möglichkeit gegeben verschiedene Ereignisse an einzelne oder sogar einer ausgewählten Reihe von DOM Elementen zu binden.

```
$(document).ready(
    function(){
        $("a").click(function() {
            alert("Hallo Welt");
    });
```

```
});
```

Beispiel 2.2: *JQuery - Ereigniszuweisung*

In diesem Beispiel 2.2 werden, nachdem der DOM bereit ist, sämtliche Link-Tags, durch den Selektor „a“ bestimmt, mit dem Klick-Ereignis und der darin enthaltenen Funktion versehen. Bei einem Klick auf einen beliebigen Link wird diese Funktion ausgeführt und man erhält einen Alert mit der Meldung „Hallo Welt“.

Des Weiteren bietet JQuery Möglichkeiten die Ajax(Asynchronous JavaScript and XML) - Technologie zu nutzen. Zur Verfügung stehen zahlreiche Funktionen, wie z.B. „load(url, data, callback)“ zum Laden und Integrieren von HTML aus einer entfernten sich auf dem Server befindenden Datei und „$.post(url, [data], [callback], [type])“ zum Senden einer POST Anfrage an den Server. Darüberhinaus können diese Requests mit Hilfe von Ereignis-Funktionen überwacht und gegebenenfalls eine Statusmeldung ausgegeben werden, z.B wird „ajaxComplete(callback)“ beim Abschluss einer Anfrage ausgeführt. Eine ausführlichere Variante mit umfangreichen Einstellungsmöglichkeiten einen HTTP Request abzuschicken wird durch die $.ajax(options) Funktion ermöglicht.

```
$.ajax({
   type: "POST",
   url: "webseite.jsp",
   async: true,
   cache: false,
   data: "name=Mustermann&vorname=Max",
   dataType: "html",
   success: function(html){
     $("#ergebnis").append(html);
   }
});
```

Beispiel 2.3: *JQuery - Asynchroner HTTP-Request*

Dieses Beispiel 2.3 zeigt eine asynchrone Ajax-HTTP POST Anfrage, bei der die aktuelle Version der Datei „webseite.jsp“ angefordert wird. In dem Feld „data“ sind die Parameter (Schlüssel-Wert-Paare) „name“ und „vorname“ mit den jeweiligen Werten zu finden. Die „dataType“-Spezifikation gibt den Datentyp an, der als Antwort vom Server erwartet wird. Das vollständige Ergebnis wird nach erfolgreicher Anfrage an das DOM-Element mit der ID „ergebnis“ angehängt.

Zusätzlich zu diesem umfangreichen JavaScript Framework existieren zahlreiche Plug-ins, welche das Handling bzw. die Interaktionen mit dem Benutzer noch komfortabler gestalten können.

Als Grundlage für diese Upload-Variante wird das einfache Html-Formular (siehe „Upload über das Html-Formular“) verwendet und mit den Funktionen von JQuery erweitert. Dadurch ist die Möglichkeit gegeben den Upload mithilfe der Ajax-Technologie asynchron durchzuführen und den Benutzer durch einfache Statusanzeigen bezogen auf „Start“, „Abgeschlossen“ und „Abbruch“ zu informieren. Im Vergleich zur herkömmlichen Html Upload-Variante gibt es bezüglich der technischen Realisierung des Hochladens der Datei keinen Unterschied, das heißt der Upload wird ebenfalls mit einer HTTP POST Anfrage durchgeführt.

Mit dieser Upload-Möglichkeit ist ebenfalls eine hohe Browser-Kompatibilität gegeben. Allerdings ist zu beachten, dass JavaScript eine clientbasierte Scriptsprache ist und dadurch die Funktionalität nicht zu Hundert Prozent gewährleistet werden kann, da dem Nutzer die Option gegeben ist, die JavaScript-Funktion im Webbrowser zu deaktivieren. Für weitere Informationen zu diesem JavaScript Framework ist JQuery Webseite[1] sehr zu empfehlen.

2.4 Upload mit Hilfe von Adobe Flash

Die Autorensoftware Adobe Flash wird in der heutigen Zeit insbesondere für die Erstellung von digitalen Animationen und interaktiven Webseiten verwendet. Die Realisierung der Webseiten findet häufig in einem komplexeren Rahmen als Webapplikation mit besonderem Augenmerk auf die Nutzung von Grafiken, Musik und Videos statt. Allerdings werden auch kleinere Flash-Animationen auf Webseiten zum Beispiel als Werbebanner oder Steuerelement für Menüs zur Navigation bervorzugt verwendet.

Bei der Erstellung von Flash-Animationen mit Adobe Flash liegen die Quelldateien im FLA-Format vor und können jederzeit vom Autor bearbeitet werden. Zur Veröffentlichung dieser Anwendungen werden die FLA-Dateien in das SWF (Shockwave Flash) - Format kompiliert. Für die Präsentation der SWF-Dateien in einem Webbrowser wird jedoch ein Programm zur Darstellung dieser Dateien benötigt, z.B. der Adobe Flash Player[2] der in Form eines Webbrowser-Plugins eingebunden werden kann, dabei ist der Einsatz in vielen Webbrowsern gegeben.

Die API von Adobe Flash macht es möglich komplexe Client-Server-Anwendungen für den Download und Upload von Dateien, unter Verwendung der integrierten objektorientierten Programmiersprache ActionScript in der Version 3, zu erstellen. Der Upload kann durch das asynchrone Client-Server-Kommunikationsmodell, auf dem der Flash-Player aufgebaut ist, ohne die Webseite neu zu laden, durchgeführt werden[2]. Die Interaktion mit dem Benutzer bleibt während des gesamten Hochladeprozesses bestehen, das heißt der Nutzer hat die Möglichkeit jederzeit diesen zu unterbrechen. Die Darstellung des Prozessverlaufes zur Information für den Benutzer geschieht mittels einer Fortschrittsanzeige.

Allerdings ist zu berücksichtigen das die Vorgehensweise durch die Nutzung der Upload-Flash-API einen Upload-Request an ein serverseitiges Skript abzuschicken, exakt dem Verfahren einer von einem Html Formular gesendeten standardmäßigen Anfrage gleicht. Das bedeutet das ebenso die Methode POST und das Übertragungsformat multipart/form-data

verwendet werden, wobei für die Download- und Upload-Abläufe keine Grenzwerte für die jeweiligen Dateigrößen festgelegt sind. Für die Übertragung von Daten können ebenfalls Sockets verwendet werden, wodurch die Interaktion mit einer Vielzahl von Internetprotokollen möglich ist.

Zu guter Letzt soll auf die Upload-Variante mit Hilfe des Java-Applets eingegangen werden.

2.5 Upload mittels Java-Applet

Ein Java-Applet ist ein in der Programmiersprache Java geschriebenes Programm, welches in den HTML-Code eingebettet und auf dem Computer des Nutzers im Webbrowser mittels der Java Virtual Machine (JVM) ausgeführt wird. Es ist plattformunabhängig und bietet ideale Möglichkeiten durch die Java-API, Dateien über das Hypertext Transfer Protocol (HTTP) oder File Transfer Protocol (FTP) auf den Server zu übertragen. Weiterhin ist es auch mit dieser Variante möglich den Upload-Fortschritt darzustellen und den Benutzer zu informieren.

Ein großer Vorteil bei der Verwendung eines Java Applets ist die Möglichkeit auf die umfangreiche Java API zurückgreifen zu können, wodurch die Erstellung von sehr komplexen Anwendungen möglich ist.

Die Ausführung eines Java-Applets findet in der Regel in einem Webbrowser statt. Damit der Nutzer ein solches Java-Applet ausführen kann, wird die JRE (Java Runtime Environment) benötigt. Das Applet wird auf den Client-Computer heruntergeladen und in der sogenannten Sandbox, einer Laufzeitumgebung mit besonderen Sicherheitskriterien, gestartet. Zugriffe auf Dateien die sich auf dem Computer des Benutzers befinden sind nur durch das Signieren durch den Entwickler und der Zustimmung der Ausführung durch den Benutzer vor dem Laden des Java Applets möglich.

Nachdem nun die jeweiligen Upload-Varianten beschrieben wurden, kann abschließend eine Entscheidung für eine dieser Varianten, welche in dem Prototyp implementiert werden soll, gefällt werden.

2.6 Entscheidung

Für die Entscheidung der Upload-Variante sind Benutzerfreundlichkeit, Interaktion und Aktualität wichtige Auswahl-Kriterien. Ebenso ist die sichere Datenübertragung der Multimedia-Daten von entscheidender Bedeutung, das heißt die Sicherstellung der erfolgreichen Übertragung der Daten.

Bei der Benutzerfreundlichkeit spielt unter anderem die Ladezeit und Initialisierung des Upload-Elementes eine wichtige Rolle. Dabei können die JavaScript und Flash Varianten bei schlechter Programmierung unangenehm lange Ladezeiten verursachen, wohingegen die Zeit für das reine HTML-Upload-Element zu vernachlässigen ist. Im Gegensatz dazu wird bei der

Verwendung des Java-Applets vergleichsweise relativ lange Zeit zum Herunterladen und Initialisieren benötigt, vor allem wenn die Java Virtual Machine selbst noch nicht initialisiert wurde[3].

Das Kriterium Interaktion ist in der heutigen Zeit kaum weg zu denken. Der Nutzer sollte die Möglichkeiten bekommen, den Ablauf des Hochladens einer Datei verfolgen oder ihn gegebenenfalls abbrechen zu können. Das Verfolgen des Vorgangs lässt sich mit Hilfe einer Fortschrittsanzeige sehr gut darstellen. Die Option eine solche Anzeige zu nutzen, bieten nur die Varianten Flash und das Java Applet.

Ein weiterer Punkt ist die Abhängigkeit von zusätzlicher Software. In diesem Fall wird, um das Abspielen von Flash Anwendungen zu ermöglichen, ein Flash-Player benötigt, welcher als Plugin für den Webbrowser zur Verfügung steht, das kann zum Beispiel der Flash Player von Adobe sein. Für das erfolgreiche Laden und Abspielen von Java Anwendungen als Java Applet in einem Webbrowser ist es erforderlich die Java Laufzeitumgebung (JRE) auf dem System installiert zu haben.

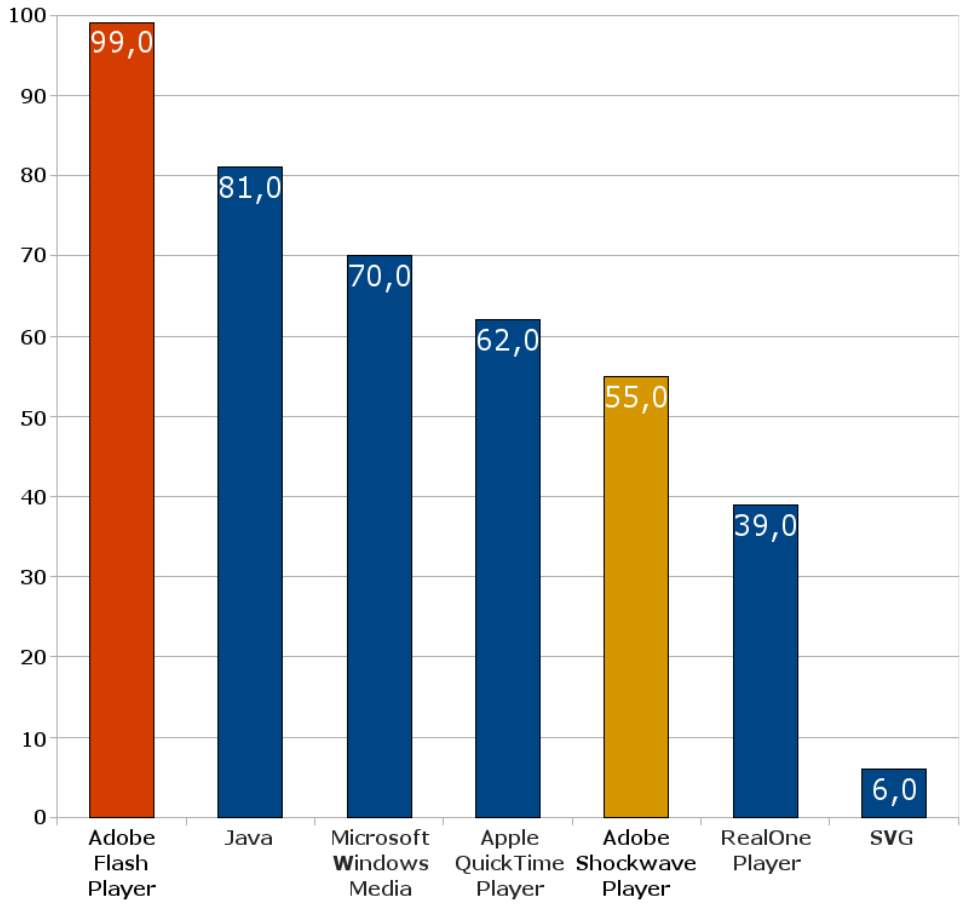

Abbildung 2.1: *Vergleich der genutzten Plug-In Technologien [15]*

Betreffend der Aktualität ist bei einer Aktualisierung Adobe Flash mit dem Flash Player (Version 10.0.32.18, Dateigröße: 1.8MB) durch seine geringe Größe dem Java Runtime Paket (Version 6 Update 16, Dateigröße: 10 MB) weit überlegen. Ein weiterer Vorteil ist die Abwärts-

kompatibilität, die bei der Nutzung des Flash Players gegeben ist. Bei der Verwendung von Java kann es durch aus vorkommen, dass ein Applet eine spezielle Lauzeitumgebung verlangt, das wiederum mehr Aufwand für den Nutzer bedeutet.

Nach einer Studie des Martkforschungsinstituts Millward Brown vom Dezember 2008[15] wird die Dominanz des Adobe Flash Players gegenüber anderen Technologien im Internet wie z.B. Java oder dem Windows Media Player von Microsoft deutlich. Die Abbildung 2.1 zeigt einen Vergleich dieser Technologien. Der Einsatz von Adobe Flash ist unter Berücksichtigung der Studie des Millward Brown Instituts aufgrund der weitverbreiteten Nutzung des Flash Players und der oben genannten Erkenntnisse sowie der aufgestellten Kriterien die beste Wahl und soll bei dem Entwurf sowie der Implementation des Prototyps verwendet werden. Damit kann dieses Kapitel abgeschlossen werden.

In dem darauffolgenden Kapitel werden Möglichkeiten zur Speicherung von Multimedia-Daten analysiert und näher beschrieben, sowie eine Entscheidung über die zu verwendende Variante zur Sicherung getroffen.

3 Möglichkeiten der Speicherung von Multimedia-Daten

In diesem Kapitel werden Möglichkeiten zur persistenten Speicherung von Multimedia-Daten, Bild- und Videodaten, aufgezeigt, analysiert und bewertet. Dabei werden zwei Varianten, die Speicherung von Binärdaten in einem für BLOBs (binary large object) spezifischen Feld in einer Datenbank oder aber als Datei in dem Dateisystem des Webservers, in Betracht gezogen. Die allgemein überwiegend genutzte Lösung ist die Archivierung der Binärdaten im Dateisystem mit einem entsprechendem Eintrag als Zeiger, meist der Pfad zu der jeweiligen Datei, in der Datenbank.

3.1 Auswahlkriterien

Bei der Auswahl der Methode zur Speicherung der Multimedia-Dateien ist ein gutes Performance-Verwaltungs-Verhältnis von entscheidender Bedeutung. Eine nahezu konstante Ausführungsgeschwindigkeit sollte sowohl bei kleineren, als auch bei größeren Dateien gewährleistet sein. Dabei sollte der Zugriff auf die Media-Daten weitestgehend einfach gestaltet sein, um eine performante nachfolgende Bearbeitung dieser Dateien zu ermöglichen. Dazu soll im folgenden Abschnitt auf die Speicherung der Multimedia-Dateien in einer Datenbank näher eingegangen werden.

3.2 Speicherung der Daten in einer Datenbank

Durch die Archivierung der Daten und Dateien in einem BLOB-Feld in einer Datenbank ergeben sich Möglichkeiten einer vorteilhafteren Kontrolle über die zu sichernden Daten. Dabei ist ein wesentlicher Vorteil die Unterstützung von Transaktionen, mit deren Hilfe die Bearbeitungen, dass heißt das Hinzufügen, Ändern oder Entfernen von Daten, überwacht werden können. Während einer solchen Transaktion ist die Unterbrechung des Vorganges möglich, sowie das vollständige Zurücksetzen in den vorhergehenden Zustand. Ein weiterer wesentlicher Vorteil ist die Sicherung der Daten, wodurch mittels Backup-Funktion der Datenbank der vollständige Inhalt gesichert werden kann. Auf diese Weise ist eine konsistente Datensicherung ebenso wie die Wiederherstellung der Daten realisierbar.

Die Speicherung von Dateien in einer Datenbank kann allerdings auch einige schwerwiegende Nachteile mit sich führen. So können große Datenmengen dazu führen, dass die Datenbankgröße schnell und verstärk anwächst und in Folge dessen die Ausführungsgeschwindigkeit der Datenbank selbst erheblich verringert wird. Ebenso sind wenige Datenbanken fähig Binärdaten als Stream (Datenstrom) kontinuierlich zu lesen, vielmehr wird die gesamte Datei in den Speicher geladen, was insbesondere in Bezug auf Dateien von beachtlicher Größe weniger effizient und aus diesem Grund zu vermeiden ist. Des Weiteren sind Datenbankverbindungen sehr ressourcenlastig, vor allem beim Laden größerer Datenmengen, wodurch ebenso die Verbindungen über einen längeren Zeitraum bestehen.

3.3 Speicherung der Daten im Dateisystem

Die Variante, die Dateien außerhalb der Datenbank in dem Dateisystem zu speichern, ist im Vergleich zur Datenbank-Option selbst um einiges performanter. Lediglich der Pfad zu der jeweiligen Datei wird in der Datenbank abgelegt. Auf diese Weise kann der Zugriff auf die Bild- und Videodaten wesentlich einfacher und effizienter gestaltet werden. Des Weiteren ist somit ein direktes Bearbeiten der Dateien, ohne diese vorher aus der Datenbank laden zu müssen, möglich.

Ein Nachteil dieser Methode im Vergleich zur Datenbank-Variante ist die Notwendigkeit der selbstständigen Transaktionskontrolle, das heißt die kritischen Bereiche, in denen Bearbeitungen bzw. Änderungen an den Dateien sowie den Daten in der Datenbank vorgenommen werden, befinden sich in eigener Verantwortung und müssen selbst überwacht werden. Gleichermaßen trifft dies beim Erstellen und Wiederherstellen von Datensicherungen zu.

3.4 Entscheidung

Für die Entscheidung der für diese Diplomarbeit geeigneten Variante zur Speicherung von MultimediaDateien sind die Vor- und Nachteile beider Möglichkeiten nachfolgend in der Tabelle 3.1 noch einmal zusammengefasst.

	Datenbank		Dateisystem
+	die Datenbank beinhaltet alle Daten	+	Performance (insbesondere bei guter bis hoher Frequentierung der Daten)
-	Ausführungsgeschwindigkeit meist langsamer	-	konsistente Datensicherung gestaltet sich schwieriger
+	Zugriff auf alle Daten einheitlich	-	größerer Verwaltungsaufwand (Transaktionskontrolle,Replikation)
-	ineffiziente Arbeistweise bei Speicherung kleiner BLOBs mit vergleichsweise großen Binärobjekten möglich	+	simpler und effizienter Zugriff auf alle Daten

Tabelle 3.1: *Vergleich der Speicherung in einer Datenbank / einem Dateisystem*

Allgemein ist die Speicherung großer binärer Objekte in einer Datenbank sehr umstritten. Da es sich hierbei um Multimedia-Dateien handelt, deren Größe recht unterschiedlich ausfallen und durchaus mehrere 100 Megabyte (MB) betragen können, ist die zweite Variante, die Dateien in dem Dateisystem zu speichern und lediglich Information zur Datei selbst, wie der Pfad, die Größe oder sonstige spezifische Eigenschaften, in der Datenbank zu sichern, die bessere Wahl. Auf diese Weise sind, wie in der Tabelle 3.1 dargestellt, deutlich schnellere Zugriffe auf die Bild- und Videodateien möglich, insbesondere bei guter bis hoher Frequentierung dieser Media-Daten.

Der nächste Schritt ist die automatische Verarbeitung der Multimedia-Dateien. Die verschiedenen Möglichkeiten werden im nachfolgenden Kapitel genauer untersucht und beschrieben.

4 Automatische Manipulation der Multimedia-Daten

Dieses Kapitel trägt im Wesentlichen zu dem Entwurf und der Implementation des Prototyps bei und beschreibt Mittel und Wege der automatischen Verarbeitung von Multimedia-Daten. Dabei wird eine erste Differenzierung zwischen Bild- und Videoinhalten vorgenommen und die dafür in Frage kommenden Anwendungen zur Manipulation zusammengetragen und analysiert.

4.1 Verarbeitung von Bilddaten

In diesem Abschnitt werden einige verschiedene Möglichkeiten der Bildverarbeitung mit Hilfe von Java- und externen Softwarepaketen beschrieben.

4.1.1 Pflichtkriterien

Bei der Manipulation der Bilder sind die standardmäßigen Bearbeitungstechniken, wie z.B. Spiegeln, Strecken und Skalieren ein wichtiges Kriterium. Dabei ist das Einlesen des Bildes und das Speichern nicht zu vernachlässigen. Diese beiden Aspekte zielen auf eine möglichst umfangreiche Formatunterstützung hin, die ein wesentlicher Faktor für die Verarbeitung der Daten ist. Des Weiteren ist es erforderlich zwischen den Farbmodellen wechseln zu können.

Kriterien im Überblick:

- Nutzung üblicher Techniken (z.B. Zuschneiden, Skalieren, Spiegeln)
- umfangreiche Formatunterstützung
- Farbmodellwechsel (CMYK -> RGB)
- Manipulation auf Javabasis bzw. aus Java heraus mit Hilfe eines externen Programms (lauffähig unter Linux)

4.1.2 Überlegung eines geeigneten Zielformats

Vor gar nicht allzu langer Zeit waren es zum größten Teil die beschränkten Übertragungskapazitäten, die im World-Wide-Web zur Verfügung standen, welche die Überlegungen nach

einem geeigneten Zielformat für Bilder im Web notwendig machten. In der heutigen Zeit ist dieses Kriterium durch den Ausbau von breitbandigen Internet-Anschlüssen mehr oder minder hinfällig geworden. Vielmehr steht nun die vollständige Unterstützung durch den Webbrowser im Vordergrund, um die Bilder ohne Einschränkungen darstellen zu können. Diese Voraussetzung macht die Überlegung nach einem geeigneten Zielformat für die Darstellung der Bilder notwendig. Die Anzeige der Bilddateien in einem Webbrowser verlangt das Vorliegen der Bilder in einem webtauglichen Bildformat. Diese Webformate sind GIF[1], JPEG[2] und PNG[3], da sie von den meisten Webbrowsern unterstützt und angezeigt werden. Allerdings bestimmt der Anwendungszweck aufgrund der unterschiedlichen Eigenschaften der Formate das zu verwendende Bildformat. Im Nachfolgenden werden diese einzeln näher beschrieben.[16]

GIF

Das Graphics Interchange Format ist ein Bildformat für gute verlustfreie Komprimierung von Grafiken oder Bildern, wobei nur eine Darstellung von bis zu maximal 256 veschiedenen Farben (8 Bit Farbtiefe) möglich ist. Weiterhin können mit diesem Format einfache Animationen erzeugt, Transparenz im Bild eingebunden und angezeigt werden. Durch die Wahl einer Farbe kann bestimmt werden, welche Flächen transparent dargestellt werden sollen. Dieses Format ist besonders für die Speicherung von Grafiken und Bildern mit wenigen Farben bzw. Bildern die über umfangreiche gleichfarbige Farbflächen verfügen, geeignet. Aus diesem Grund wird das GIF Format überwiegend für Logos, Schaltflächen, kleinere Bilder und Banner mit und ohne Transparenz verwendet.

JPEG

Das JPEG-Format (Joint Photographic Experts Group) unterstützt die Darstellung von 16,7 Millionen Farben (24 Bit Farbtiefe) und basiert auf einem verlustbehafteten Kompressionsverfahren, d.h. je höher ein Bild komprimiert wird, umso schlechter ist die Qualität des Bildes. Da JPEG bei einer Komprimierung von größeren Fotos kleine Dateigrößen mit einer relativ guten Bildqualität erzeugt, hat sich dieses Format im Laufe der Zeit als das Bildformat für Fotos im Web durchgesetzt. Das JPEG-Format ist aufgrund der umfangreichen Farbdarstellung besonders für fotorealistische Bilder und Grafiken mit zahlreichen unterschiedlichen Farben mit Farbverläufen geeignet. Allerdings ist JPEG in der Realisierung für eine pixelgenaue Wiedergabe, sowie die Anzeige von Transparenz, eher nicht prädestiniert, hierbei ist die Verwendung anderer Formate wie GIF oder PNG notwendig. Die Formate JPEG und GIF werden von allen gängigen Webbrowsern vollständig unterstützt.

[1] GIF - Graphics Interchange Format
[2] JPEG - Joint Photographic Experts Group
[3] PNG - Portable Network Graphics

PNG

Mit dem Bildformat Portable Network Graphics ist ebenfalls eine verlustfreie Kompression möglich. Im Vergleich zum Format GIF ist die PNG-8 Variante ebenso auf 256 Farben pro Einzelbild beschränkt und eignet sich für die Darstellung von flächigen Grafiken. Im Gegensatz dazu sind mit PNG-24 fließende Übergänge (harte Kanten bei PNG-8 und GIF) und die Darstellung von Transparenz in mehreren Stufen sowie die Abbildung von 16,7 Millionen Farbtönen möglich. Das Bildformat PNG kombiniert die Vorteile der beiden Vorgänger GIF und JPEG, hat allerdings auch den Nachteil, dass die Dateigröße in den meisten Fällen um einiges größer ausfällt. Das PNG Bildformat sollte GIF im Word-Wide-Web ablösen, allerdings erschwert dies die nicht vollständige Browser-Unterstützung. Probleme gibt es mit dem Alpha Kanal, der vom Internet Explorer von Microsoft bis zur Version 6 nicht dargestellt wird.

Die Internet-Browser in den neuen Versionen unterstützten das Format vollständig.[4]

Zielformat - Entscheidung

Das Ziel ist den Nutzern die Möglichkeit zu bieten, Bilder weitestgehend unabhängig von Format und Qualität hochzuladen. Diese werden auf dem Server gespeichert. Für die Darstellung im Webbrowser werden aus den originalen Dateien seperate Bilddateien in einem einheitlichen webtauglichen Format erstellt. In der nachfolgenden Übersicht sind die drei Bildformate noch einmal vergleichend zusammengefasst.

Verwendung	GIF	PNG	JPEG
Grafik ohne Farbverlauf	■	■	-
Grafik mit Farbverlauf	-	■	■
Fotografien	-	■	■
Grafik mit Transparenz	▫ [5]	■	-
Browser-unterstützung	■	▫ [6]	■
Legende	■ Verwendungsmöglichkeit vollständig gegeben ▫ Verwendungsmöglichkeit nur teilweise gegeben - Format für Verwendung ungeeignet		

Tabelle 4.1: *Vergleich der drei Bildformate für das Word-Wide-Web*

[4] der Internet Explorer ab der Version 7
[5] Transparenz: ja/nein
[6] vollständige Unterstützung in den neuesten Browserversionen

Die Nutzer bekommen die Möglichkeit Multimedia-Daten, in diesem Fall Bilddaten, hochzuladen. Da es sich bei diesen Bilddateien überwiegend um Präsentationsdaten handeln wird, das heißt Fotografien bzw. Bilddaten mit vielen Farben und/oder Verläufen (im Vergleich dazu: flickr[7]), ist das JPEG-Format am besten geeignet. Wie in der Tabelle 4.1 zu sehen ist, ist das Format für diese Verwendungsmöglichkeiten vollständig predistiniert. Eine weitere Möglichkeit wäre die Nutzung des PNG-Formats, allerdings fällt in diesem Fall insbesondere bei den oben genannten Verwendungsmöglichkeiten die Dateigröße durch die verlustfreie Komprimierung meist deutlich mächtiger aus.

Ein Nachteil bei der Verwendung von PNG ist die eingeschränkte Unterstützung in älteren Browsern. Der Internet Explorer (IE) in der Version 6 von Microsoft unterstützt die Darstellung der Transparenz nicht. Da der IE nun schon in der Version 8 verfügbar ist, ist die Frage nach der Beachtung des Problems durch den Internet Explorer 6 berechtigt. In der folgenden Abbildung 4.1 ist die prozentuale Verteilung der Nutzung verschiedener Browser über den Zeitraum von Oktober 2008 bis Oktober 2009 dargestellt.

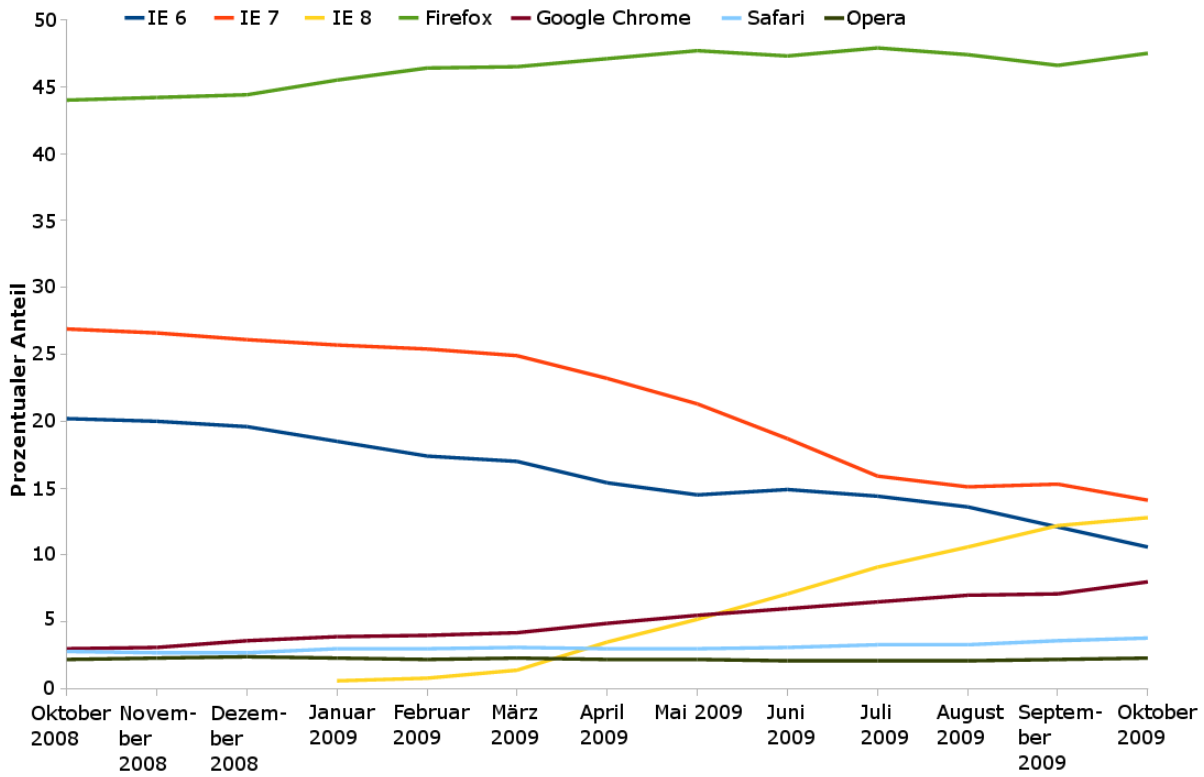

Abbildung 4.1: *Statistik über die Verwendung/Nutzung der Webbrowser [20]*

Wie in der Abbildung 4.1 zu sehen ist, wird der IE 6 (etwa 11% im Oktober 2009) noch von rund jeder neunten Person genutzt. Aus diesem Grund ist die Verwendung des GIF-Formats in Bezug auf die Transparenz zu bevorzugen. Bei der Entscheidung des im Prototyp zu verwen-

[7] http://www.flickr.com/

denden Bildformats fällt diese abschließend hinsichtlich der Verwendung auf das JPEG-Format. Ausgehend von dieser Wahl kann nun im nächsten Abschnitt auf die Möglichkeiten der Bildmanipulation eingegangen werden.

4.1.3 Möglichkeiten der Bildmanipulation

Java Advanced Imaging (JAI)

Für die Verarbeitung von Bilddaten gibt es von Sun Microsystems[4] die Erweiterung Java Advanced Imaging (JAI)[6]. Um JAI nutzen zu können, ist es erforderlich das Softwarepaket, welches derzeit in der Version 1.1.3[8] vorliegt, separat herunterzuladen[9], da es in dem Java Development Kit (JDK) nicht integriert ist.

Java Advanced Imaging wurde für eine flexiblere und anspruchsvollere Bildverarbeitung entwickelt. Die JAI API unterstützt die gängigsten Bildformate, wie Windows Bitmap (BMP), GIF, JPEG, PNG, Tagged Image File Format (TIFF) und noch einige mehr. Des Weiteren bietet diese Grafikbibliothek zahlreiche Standard-Bildmanipulationstechniken, wie z.B. Zuschneiden, Rotieren und Spiegeln. Die Möglichkeit Bilddaten in Teilen (Kacheln) darzustellen, macht JAI zu einem effizienten Werkzeug bei der Benutzung von Bilddateien mit einer sehr großen Auflösung.

Durch die objektorientierte Struktur in JAI lässt es sich hervorragend in die Java-Anwendung integrieren. Ebenfalls wird die wechselseitige Beziehung mit anderen Grafikpaketen wie dem Abstract Windowing Toolkit(AWT) und Java 2D unterstützt. Die Bildbearbeitung wird somit vollständig in Java mit Hilfe der JAI Bibliothek durchgeführt.

Java Advanced Imaging steht für komplexe und professionelle Bildverarbeitung in Java. Vor allem in den Bereichen der Medizin (z.B. die CHILI/Webanwendung für digitale Röntgenbilder) und Geophysik kommt die Grafikbibliothek zum Einsatz.

ImageJ

Die Software ImageJ[10] ist ein Bildverarbeitungs- und Bildbearbeitungsprogramm das in der Programmiersprache Java geschrieben wurde. Es wurde von einem Mitarbeiter der National Institutes of Health (NIH) in Anlehnung an das „NIH Image“ Bildverarbeitungs- und Analyseprogramm, welches lediglich für den Macintosh zur Verfügung steht, entwickelt. ImageJ ist ein Open Source - Projekt, das heißt sowohl das Programm, als auch der Quellcode darf kopiert, modifiziert und erweitert werden. Des Weiteren wurde ImageJ so entworfen, dass es als eigenständige Anwendung oder aber auch als Bibliothek für Bildverarbeitung in anderen Applikationen verwendet werden kann.

[8] die Version 1.1.4 befindet sich in der Entwicklung, JAI 2 ist geplant
[9] http://java.sun.com/javase/technologies/desktop/media/jai/
[10] http://rsb.info.nih.gov/ij/index.html

Unter Verwendung von ImageJ lassen sich 8 bit, 16 bit und 32 bit Bilder analysieren, bearbeiten und speichern. Ebenfalls sind umfangreiche Bildverarbeitungstechniken, wie Spiegeln, Rotieren oder Skalieren der Bilder integriert, mit denen die unterstützten Formate GIF, TIFF, JPEG oder Digital Imaging and Communications in Medicine (DICOM), um mal einige zu nennen, verarbeitet werden können. Ergänzend zu den eben genannten Techniken besteht die Möglichkeit der Beeinflussung von Kontrast, Schärfe und Glättung, sowie der Nutzung des Median-Filters und der Kantenerkennung[21].

Diese Analyse- und Verarbeitungsbibliothek für Bilder lässt sich aufgrund der guten Programmierstruktur unter Verwendung von Java Plugins erweitern, um somit die Funktionalität auszubauen. Des Weiteren ist sie multithread-fähig, wodurch mehrere zeitintensive Aufgaben parallel abgearbeitet werden können. Ein Anwendungsbereich von ImageJ ist die Medizin, bei der die Applikation zur Bildanalyse genutzt wird.

ImageMagick

Das Software-Paket ImageMagick[7] (IM) ist eine weitere Variante, Bilder mit zahlreichen Funktionen und Techniken zu modifizieren. Dies ist nahezu vergleichbar mit den bekannten Bildbearbeitungsprogrammen Adobe Photoshop und GIMP mit dem Unterschied, dass die Funktionen entweder über Konsolenbefehle oder über Interfaces ausgeführt werden. Es sind Schnittstellen für eine Vielzahl an Programmiersprachen verfügbar, so zum Beispiel auch für Java, C++ oder PHP. Erwähnenswert ist die umfangreiche Unterstützung der Bildformate, welche mit über 100 Formaten auf der ImageMagick-Webseite[7] angegeben ist und die mit diesem Software-Paket gelesen, modifiziert bzw. geschrieben werden können.

ImageMagick ist ein Software-Paket und besteht aus einer Vielzahl von Kommandozeilen-Tools, durch die mannigfaltige und komplexe Bildverarbeitungen und Bildbearbeitungen möglich sind. Aufgrund der Verwendung der Kommandozeile oder eines Aufrufs über Schnittstellen aus einer Programmiersprache heraus, besteht die Option, Bilder dynamisch zu generieren oder diese zu manipulieren. Diese Möglichkeit ist ein Grund für die verbreitete Nutzung im Bereich der Webanwendungen, so kann zum Beispiel ein Bild nach dem Hochladen sofort durch das Empfängerskript bearbeitet und in ein anderes Bildformat umgewandelt werden.

Für die komplexe Bildbearbeitung stehen insbesondere zwei Kommandozeilenbefehle zur Verfügung. Zum einen ist das die „convert“-Funktion und zum anderen die Funktion „mogrify“ mit denen Bilder zum Beispiel zugeschnitten, skaliert, gedreht oder gespiegelt werden können, mit dem Unterschied, dass durch den „convert“-Befehl eine neue Datei erstellt und mit dem Befehl „mogrify“ die Originaldatei überschrieben wird. Des Weiteren besteht die Möglichkeit Bilder miteinander zu vergleichen, sie zu überlagern bzw. zusammenzuführen oder diese zu einer Animation zusammenzusetzen.

Bei der Arbeit mit ImageMagick ist nicht nur die Threadsicherheit, sondern auch die Performancesteigerung durch die vollständige Nutzung von Mehrkernprozessoren gewährleistet. Ein weiterer Sicherheitsmechanismus stellt das Software-Paket durch die Ver- und Entschlüsse-

lungsfunktionen „encipher“ und „decipher“ bereit, mit deren Hilfe der Zugang zu den Bildern durch einen Geheimschlüssel eingeschränkt werden kann.

Das folgende Beispiel 4.1 soll einen Eindruck über die Vorgehensweise mit IM und dem „convert“-Befehl geben. In diesem Code-Beispiel wird das Bild mit dem Titel „landschaft“ im JPEG-Format geladen, mithilfe des Parameters „-scale“ auf die Größe 400 * 300 Pixel unter Berücksichtigung des Seitenverhältnisses angepasst. Des Weiteren wird das Bild vor dem Speichern in das PNG-Format durch den „-flop“-Parameter horizontal gespiegelt.

```
convert landschaft.jpg -scale "400x300" -flop landschaft.png
```

Beispiel 4.1: *ImageMagick: „convert“-Befehl*

Letztendlich ist ImageMagick ein absoluter Alleskönner wenn es um Bildverarbeitung und Bildbearbeitung geht. Derzeit ist das Programm in der Version 6.5.7-3 sowohl als ausführbare Installationsversion, als auch Quellcode-Version erhältlich.

GraphicsMagick

Das Software-Paket GraphicsMagick[22] (GM) ist aus dem ImageMagick-Projekt in der Version 5.5.2 heraus im Jahr 2002 entstanden und wird seit diesem Zeitpunkt unabhängig vom Eltern-Projekt weiter entwickelt. Ebenso wie die ImageMagick Anwendung unterstützt GraphicsMagick zahlreiche Bildformate, sowie ein umfangreiches Befehls- und Funktionsset. Bei der Entwicklung von GM wurde spezielles Augenmerk auf eine sichere und stabilere Benutzerschnittstelle, performantere Verarbeitung der Ressourcen und Verringerung der Programmfehler gelegt.

Derzeit ist GraphicsMagick in der Version 1.3.5 erhältlich. Ein Benchmark-Test[24] von 2008 der beiden Grafikprogramme GraphicsMagick in der Version 1.3.1 und ImageMagick in der Version 6.4.5 verdeutlicht in der Durchführung verschiedenster Operationen in den meisten Fällen markante Unterschiede. Dabei liegt die Verarbeitungszeit von GM vorherrschend um die Hälfte unter der von IM. Einige Verbesserungen in GraphicsMagick wurden im Nachhinein in ImageMagick übernommen.

Im April 2009 veröffentlichte John Allspaw von Flickr[11] eine Präsentation[23] über Verfeinerungen im Hardware- und Softwareumfeld von Flickr. Dabei werden Optimierungsansätze hinsichtlich der Software und Hardware-Aktualisierungen zur Performancesteigerung, Kostensenkung, sowie Möglichkeiten den Speicherbedarf zu verringern, beschrieben. In Bezug auf die Bildverarbeitung wurde erläutert, dass seit 2004 ImageMagick (Version 6.1.9) für die umfangreiche Verarbeitung der Bilder verwendet wird. Wenig später folgte der Wechsel zu GraphicsMagick in der Version 1.1.5, welcher zu der Zeit einen Performancegewinn von 15% erbrachte. Seit diesem Zeitpunkt nutzt Flickr größtenteils das Grafiksoftware-Paket GM, wobei ein kleiner Teil bestimmter ImageMagick-Funktionen weiterhin verwendet wird.

[11] http://www.flickr.com/

4.1.4 Entscheidung

Für die Bestimmung nach der jeweiligen Technik ist die Erfüllung der Pflichtkriterien von entscheidender Bedeutung. Alle Varianten ermöglichen eine Bearbeitung mit Java, wobei sich die Qualität der Ergebnisse stark unterscheiden, insbesondere treten bei der Größenänderung der Bilder mit Java Advanced Imaging wahrnehmbare Unterschiede auf, so DÁrcy Norman[8], welcher im Lehr- und Ausbildungszentrum der staatlichen Universität in Calgary tätig ist. Die Bildqualität ist nach der Bearbeitung mit ImageMagick wesentlich genauer und auch in Bezug auf die Performance liegt das Software-Paket deutlich vor JAI, vor allem bei der Verarbeitung von sehr großen Bilddaten. Ein weiterer wesentlicher Vorteil von ImageMagick ist die Unterstützung der zahlreichen Bildformate, sowie die umfangreichen Techniken zur Manipulation.

Eine reine Java-Lösung bietet das Programm ImageJ, welches ebenso als Bibliothek in anderen Anwendungen verwendet werden kann. Ein Nachteil dieser Anwendungsbibliothek ist die begrenzte Unterstützung und Verwendbarkeit der Formate, so ist der Zugriff auf einige Bilddateien lediglich lesend oder schreibend möglich. Mit Hilfe von Plugins lassen sich Verwendbarkeit und Unterstützung von Bildformaten erweitern.

In diesem Zusammenhang sind die beiden ähnlichen Anwendungen ImageMagick und GraphicsMagick in Bezug auf die Funktionalität und insbesondere die umfangreiche Formatunterstützung den anderen beiden Java-Möglichkeiten weit überlegen. Bei der Entscheidung zwischen diesen beiden Grafikprogrammen wird die Verwendung von GraphicsMagick präferiert und damit nochmals auf den oben genannten Benchmark-Test[24], sowie die Verbesserungen in dem Programmkern verwiesen.

Die Konvertierung eines Farbraumes in einen anderen ist in allen Fällen möglich, wobei in Bezug auf die beiden Applikationen ImageMagick und GraphicsMagick die Einbeziehung von ICC-Profilen[12] notwendig ist. Diese Profile beschreiben den Farbraum von Farbwiedergabe- oder Farbeingabegeräten, z.B. Drucker, Scanner und Monitor.

Abschließend ist zu sagen, dass entweder das Software-Paket IM oder bei möglicher Nutzung die Software GM für die Manipulation von Bilddaten genutzt werden soll. Damit kann dieser Abschnitt abgeschlossen und zum Thema „Verarbeitung von Videodaten“ übergegangen werden.

4.2 Verarbeitung von Videodaten

Dieser Abschnitt befasst sich mit der automatischen Manipulation von Videodaten auf einem Webserver. Diese sollen anhand ihrer Eigenschaften erkannt und gegebenenfalls verändert bzw. angepasst werden.

[12] ICC - INTERNATIONAL COLOR CONSORTIUM (http://www.color.org)

4.2.1 Zielbeschreibung und Bedingungen

Bei der automatischen Verarbeitung der Videodaten ist es erforderlich die Eigenschaften, sowie das Format dieser Multimedia-Dateien zu prüfen und diese anhand von vordefinierten Kriterien, welche nachfolgend beschrieben werden, anzupassen. Die Quelldatei wird nach dem erfolgreichen Hochladevorgang gesichert und eine neue Datei in einem webtauglichen Zielformat erstellt, wobei die Unterstützung vieler Videoformate dabei zu einer höheren Flexibilität und besseren Benutzerfreundlichkeit führen. Bei Verwendung von externen Programmen ist nach der Ausführung ein Rückgabewert zur eindeutigen Auswertung (Erfolg/Fehler) erforderlich.

Ein weiterer essentiell wichtiger Punkt ist die Wahl eines einheitlichen Videoformats für das Abspielen in einem Webbrowser. Dabei sind die bekanntesten Formate, welche die Streaming-Technik für Audio und Video unterstützen, QuickTime, Real Media, Windows Media und Flash Video. Die größte Hürde hierbei ist die Kompatibilität der einzelnen Formate auf den unterschiedlichen Plattformen, zusätzlich sind Qualität und Dateigröße weitere wichtige Faktoren. In diesem Zusammenhang hat sich das Flash Video Containerformat von Adobe als das Web-Video Format etabliert und wird von größeren Video-Communities und -Portalen wie YouTube[13] verwendet. Eine Ursache dafür ist, wie in der Abbildung 2.1 auf Seite 23 dargestellt, die hohe Verfügbarkeit, da Flash in vielen Browsern vorinstalliert und integriert und somit ein Abspielen meist ohne zusätzliche Software möglich ist. Aus diesem Grund wird für die Verarbeitung und Bearbeitung der Videodaten das Flash Video Format (flv) verwendet. Bei der Wahl nach der entsprechenden Möglichkeit, Videodaten zu verarbeiten, sind nachfolgend eine Reihe von Pflichtkriterien aufgelistet, welche durch den Prototyp und somit durch die entsprechende Software eingehalten bzw. erfüllt werden müssen.

Kriterien im Überblick:

- Möglichkeiten zur Änderung der Eigenschaften
  - Video-Auflösung und Bildwiederholfrequenz
  - Abtastfrequenz und Anzahl der Audio-Kanäle
  - Audio- und Videobitrate
  - Audio- und Videocodec
- umfangreiche Formatunterstützung
- Manipulation auf Javabasis bzw. aus Java heraus mit Hilfe eines externen Programms (lauffähig unter Linux)

[13] http://www.youtube.com

4.2.2 Möglichkeiten der Videomanipulation

Java Media Framework (JMF)

Das Java Media Framework (JMF) ist eine von Sun entwickelte Java-Bibliothek mit deren Hilfe anhand der bereitgestellten Schnittstellen (API), Multimedia-Daten, Audio- und Videodateien, plattformunabhängig erfasst, ver- und bearbeitet sowie umgewandelt werden können. Es ist ein optionales Paket, welches das Java Software Development Kit (SDK) um die Möglichkeiten des objektorientierten Zugriffs auf Media-Daten erweitert.

Die gut dokumentierte API, sowie die Möglichkeit aufgrund des Aufbaus des JMF die Funktionalität durch Plugins zu erweitern, erleichtern den Einstieg und die technische Umsetzung eigener Projekte mit diesem Framework. Des Weiteren werden infolge der umfangreichen Bibliothek in Bezug auf die integrierten Netzwerkprotokolle (Real-Time Transport Protocol (RTP)/ Real-Time Streaming Protocol (RTCP)) die Übertragung und das Abspielen von Multimedia-Daten in Echzeit unterstützt. Auf diese Weise werden ebenso Video-Konferenzen ermöglicht.

Mit dem Java Media Framework ist eine sehr umfangreiche und komplexe Möglichkeit gegeben, Media-Daten rein auf Java-Basis zu manipulieren. Bedauerlicherweise werden in diesem Framework in der „Grundausstattung“ lediglich einige wenige Videoformate, wie AVI (Audio Video Interlace), MPEG (Moving Picture Experts Group) und QuickTime, unterstützt. Der Support von Audioformaten hingegen ist um einiges ausgereifter. Hinsichtlich dieser anspruchsvollen Verarbeitung von Audio- und Videodaten ist die Option gegeben, sogenannte „Performance Pakete“ zu nutzen. Mit Hilfe dieser Pakete ist die Verwendung von Systemcode und somit eine Leistungssteigerung möglich. Des Weiteren kann auf diese Weise die Nutzung von Systembibliotheken realisiert werden, wodurch eine umfangreichere Unterstützung von Media-Daten ermöglicht wird.

Trotz der zahlreichen und komplexen Bearbeitungsmöglichkeiten, bei denen die umfangreichen Filter- und Effektfunktionen nicht zu vernachlässigen sind, ist die Version 2.1.1e fortwährend die aktuelle stabile Veröffentlichung seit Mai 2003.

FFmpeg

FFmpeg[9] ist ein schnelles kleines und plattformübergreifendes unter LGPL[14] lizensiertes und in der Programmiersprache C geschriebenes Videobearbeitungsprogramm für die Kommandozeile. Es ist Bestandteil des gleichnamigen Projektes FFmpeg, welches aus insgesamt acht Komponenten besteht[25]:

ffmpeg Kommandozeilenprogramm für das Konvertieren, Verarbeiten und Bearbeiten von Videodaten

ffserver Streaming Server für Multimedia-Daten über HTTP

[14] LGPL - Library General Public License

ffplay Sehr einfaches Abspielprogramm basierend auf den FFmpeg- und SDL[15]-Bibliotheken

libavutil Bibliothek mit zahlreichen Funktionen zur programmiertechnischen Unterstützung, wie Zufallsgenerator, Mathematischen Funktionen usw.

libavcodec Diese Bibliothek stellt Audio- und Video-Encoder und -Decoder für ffmpeg bereit. Zusätzlich kann sie in weiteren externen Anwendungen verwendet werden.

libavformat Beinhaltet Multiplexer und Demultiplexer für Multimedia-Container Formate, zum Beispiel AVI (Audio Video Interlace) und Mov (QuickTime-Containerformat).

libavdevice Mit Hilfe dieser Bibliothek können Zugriffe (Aufnahme und Wiedergabe) auf Geräte erfolgen.

libswscale Bibliothek für leistungsfähige Bildbearbeitung (Größenänderung, Farbraumumwandlung)

Das Projekt FFmpeg unterstützt eine Vielzahl von Audio-, Video- und Bildformaten und versucht außerdem die Media-Daten so verlustfrei wie möglich zu verarbeiten. Grundsätzlich dient FFmpeg zur Konvertierung eines Formats in ein anderes, wobei die Abtastfrequenz, Ziel-Bitrate und viele weitere Eigenschaften, wie z.B. die Anzahl der Audio Kanäle, verändert werden können. Des Weiteren ermöglicht das Programm verschiedene Eingabequellen zu nutzen, wobei unter anderem die Verwendung von Dateien, Video- oder Audiogeräten und ebenso von Bildschirmen/Displays dazu zählen. Weiterhin ist es möglich VOB (Video Objekt) Dateien einer DVD zu entschlüssen und in andere Formate unter Beachtung der Parameter, z.B. der Audio-Spur, zu konvertieren oder aus einer Reihe von Bilddateien ein Video mit entsprechender Wunschmusik zu erstellen.

FFmpeg ist ein sehr leistungsstarkes und umfangreiches Programm, das durch simple Kommandobefehle gesteuert wird. Im nachfolgenden Beispiel 4.2 wird eine Videodatei im AVI-Containerformat in das Flash-Video Format flv konvertiert, wobei die Quelldatei „video.avi“ durch den Parameter „-i“ angegeben wird. Anschließend folgen Angaben zur Videobitrate „-b“ mit 300kbit/s, sowie die Festlegung der Audio-Frequenz „-ar“ mit 44100Hz und der Audio-Bitrate „-ab“ von 96kbit/s. Schließlich wird mit dem Parameter „-s“ die Auflösung festgelegt, in diesem Fall auf „320 * 240“, und abschließend die Ausgabedatei „video.flv“.

```
ffmpeg -i video.avi -b 300k -ar 44100 \
-ab 96k -s 320x240 video.flv
```

Beispiel 4.2: *FFmpeg - Videokonvertierung vom AVI-Format in das Flash-Video Format*

[15] freie betriebssystemübergreifende Multimedia-Bibliothek für Zugriff auf Hardware (Audio, Grafik). http://www.libsdl.org/

Aufgrund der Leistungsfähigkeit des Programms FFmpeg wird es in vielen Teilen und Bereichen der Softwareentwicklung eingesetzt. Für die Programmiersprache Java existieren zahlreiche Schnittstellen, die den Zugriff und die Verwendung von FFmpeg in Java ermöglichen. Im Folgenden werden fünf Wrapper kurz vorgestellt und anschließend eine vergleichende Bewertung vorgenommen.

FFMPEG-Java

FFMPEG-Java[16] definiert eine Schnittstelle zwischen der Programmiersprache Java und dem Programm FFmpeg mit Hilfe von JNA (Java Native Access), wodurch der Zugriff von einer plattformunabhängigen Java-Anwendung auf plattformspezifische dynamische Bibliotheken ermöglicht wird. FFMPEG-Java ist ein LGPL lizensiertes Teilprojekt von Freedom for Media in Java (FMJ)[17], welches ein Open-Source Projekt ist und eine Alternative zum Java Media Framework darstellt. Die aktuelle Version ist fortwährend die vom September 2007, inwiefern in nächster Zeit mit einer Aktualisierung zu rechnen ist, bleibt abzuwarten[28].

FOBS

FOBS (Ffmpeg OBjectS) ist ein in C++ geschriebener objektorientierter Wrapper für FFmpeg, wobei sich der für die Programmiersprache Java in der Entwicklung befindet. Für letzteres wird vorläufig das FOBS4JMF angeboten, welches ein Plugin für das Java Media Framework ist. Mit Hilfe des Plugins wird die Multimedia-Unterstützung des Frameworks um eine Vielzahl weiterer Codecs und Container, die FFmpeg beinhaltet, erweitert. In Folge dessen ist die Ver- und Bearbeitung zahlreicher Medien mit JMF möglich. Ein Nachteil des FOBS4JMF-Plugins ist die essentielle Verfügbarkeit des Java Media Frameworks. Die letzte Aktualisierung des Plugins erfolgte allerdings im August 2008 und ist mit der Versionsnummer 0.4.2 gekennzeichnet[27].

Ein weiteres JMF-Plugin ist der JFFmpeg[18] JNI Wrapper, wobei in diesem Fall die letzte Aktualisierung im Januar 2006 vorgenommen wurde und aus diesem Grund auf weitere Erläuterungen verzichtet wird.

Xuggler

Zu guter Letzt soll die Bibliothek Xuggler näher erläutert werden. Xuggler ist ein Open-Source Wrapper für FFmpeg und steht als Bibliothek für die Programmiersprache Java und C++ zur Verfügung. Die Kommunikation bzw. Zugriffe auf FFmpeg aus Java heraus erfolgen über die JNI-Schnittstelle. JNI (Java Native Interface) ist eine Schnittstelle der JVM die den Zugriff auf plattformspezifische Applikationen und Bibliotheken ermöglicht. Auf diese Weise ist die

[16] http://fmj-sf.net/ffmpeg-java/getting_started.php
[17] http://fmj-sf.net/
[18] http://jffmpeg.sourceforge.net/

Be- und Verarbeitung von Multimedia-Daten mit Xuggler vollständig in der Programmiersprache Java möglich. Außerdem ist mit Xuggler eine Möglichkeit gegeben über das Java Native Interface die FFmpeg Bibliothek und somit die zahlreichen Funktionen sowie die umfangreiche Formatunterstützung nutzen zu können. Dabei ist insbesondere die ständige Entwicklung und Verbesserung der Bibliothek hervorzuheben, wodurch deren aktuellste Version stets in Java verwendet werden kann. Zu dem kommt die Unterstützung von Multithreading und die Einsatzfähigkeit im Serverbereich in Webanwendungen hinzu. Xuggler selbst kann unter Windows, Linux und Mac sowohl als 32 Bit als auch 64 Bit Version verwendet werden. Die aktuelle Version ist die 3.3, Codename „Eliza“, vom Oktober 2009. Ein bedeutender Vorteil gegenüber den bisherigen Möglichkeiten ist die hervorragende Unterstützung und Dokumentation durch die Xuggler-Community[26] sowie die stetige Weiterentwicklung.

4.2.3 Entscheidung

Für diese Entscheidung sind Kriterien wie Leistung, Zeit und Qualität von entscheidender Bedeutung. In Bezug auf die vorliegenden Möglichkeiten ist die Verwendung der in der Programmiersprache C geschriebenen Anwendung FFmpeg anzustreben. Obwohl durch die Nutzung des Java Media Frameworks eine reine Java-Lösung zur Verfügung steht, ist diese Variante nicht zu bervorzugen. Die jüngste Veröffentlichung liegt unterdessen sechs Jahre zurück, inwiefern dieses Projekt von Sun Microsystems weitergeführt und entwickelt wird, bleibt abzuwarten. Die weniger ausgeprägte Formatunterstützung lässt sich mithilfe von Plugins(FOBS4JMF) unter Verwendung von FFmpeg erweitern, allerdings ist in diesem Zusammenhang die vollständige Nutzung der FFmpeg Anwendung vorteilhafter. Das FFmpeg Programm wurde speziell für Multimedia-Verarbeitung und -Bearbeitung entwickelt und ist aufgrund der Tatsache, dass es in der Programmiersprache C geschrieben ist, um einiges leistungsfähiger. Außerdem besteht demnach die Möglichkeit auf die zahlreichen Funktionen zur Bearbeitung der Media-Daten in FFmpeg zurückzugreifen.

Um diesen Zugriff auf die Anwendung FFmpeg aus Java heraus realisieren zu können, sind sogenannte Wrapper bzw. Schnittstellen notwendig, welche die Kommunikation zwischen der Programmiersprache und der Applikation ermöglichen. In diesem Zuge ist die Verwendung des Wrappers Xuggler im Vergleich zur FFMPEG-Java Schnittstelle zu favorisieren. Mit Xuggler kann der Großteil der von FFmpeg zur Verfügung stehenden Funktionen genutzt werden, außerdem wird der Wrapper fortwährend weiterentwickelt. In diesem Zusammenhang soll nochmals auf die jüngste Veröffentlichung von FFMPEG-Java im September 2007 verwiesen werden, wobei der weitere Entwicklungsverlauf nicht bekannt ist. Des Weiteren erschwert die schlecht dokumentierte API die Entwicklungsarbeiten mit diesem Wrapper.

Neben dem Ver- und Bearbeitungstool FFmpeg stehen unter Linux zwei weitere Anwendungen mit ähnlichem Funktionsumfang zur Verfügung. Das ist zum einen der Mencoder[19]

[19] http://www.mplayerhq.hu/

und zum anderen das Programm Transcode[20], welches auf der Bibliothek von FFmpeg aufbaut. Allerdings sieht es für die beiden letzten Anwendungen in Bezug auf die Kommunikation mit Java sehr rar aus, weshalb auch in diesen Fällen keine weiteren Erläuterungen zu diesen Applikationen folgen.

In dem nachfolgenden Kapitel beginnt die Implementation des Prototyps mit den bis zu diesem Punkt ausgewählten Anwendungen, Bibliotheken bzw. Schnittstellen. Die Wahl des zu nutzenden Videoverarbeitungstools sowie der zugehörigen Schnittstelle fiel auf FFmpeg in Verbindung mit der Xuggler Bibliothek, wodurch dieser Abschnitt sowie dieses Kapitel abgeschlossen werden können.

[20] http://www.transcoding.org

5 Entwurf und Implementierung

In den vorherigen Kapiteln wurden die Technologien, Mittel und Wege, die für die Prototyp-Implementation unabdingbar sind, aufgezeigt und analysiert. Anhand dieser Technologiebetrachtungen kann nun der Entwurf und die Implementation des Prototyps erfolgen.

5.1 Ziel der Implementation

Das Ziel der nachfolgenden Beschreibung der Prototyp-Implementation ist eine Anwendung, welche auf einer Auswahl der in den vorherigen Kapiteln aufgezeigten Technologien basiert. Diese Applikation zeigt die Vorgehensweise von der Auswahl der Media-Daten durch den Nutzer und deren Upload auf den Webserver, über die automatischen Verarbeitungsschritte dieser Multimedia-Dateien und deren Modifikationen bis hin zu den jeweiligen Datenbank-Einträgen.

Der Entwurf und die Implementation des Prototyps sowie die weitere Verwendung erfolgt im Knüpfer Verlag in Dresden. Die Nutzer, wie eben beschrieben wurde, sind in diesem Falle Verlagskunden, welche die Möglichkeit bekommen sollen, ihre Media-Daten nach erfolgreicher Authentifizierung auf den Verlagsserver zu laden.

5.2 Entwurf und Implementation des Prototyps

5.2.1 Funktionalität / Pflichtkriterien

Nachfolgend sind Pflichtkriterien aufgelistet, die durch den Prototyp erfüllt werden müssen.

- Upload der Multimedia-Dateien durch den Benutzer
- automatische Verarbeitung der Media-Dateien, d.h. im Falle von
  - Bilddateien:
    - Erstellung von modifizierten Varianten (Änderung an Größe (Bildauflösung), Format, gegebenenfalls Farbraum)
    - Erstellung von Vorschaubildern (Thumbnails) in quadratischer Form
    - Berücksichtigung der Seitenverhältnisse
  - Videodateien:

    - Erstellung von Flash-Videos in unterschiedlicher Größe (Auflösung)
    - Möglichkeit der Beeinflussung/Veränderung (Resampling) der Video und Audio Eigenschaften (z.B. Auflösung, Audio- und Video-Bitrate, Abtastrate, sowie der Bildwiederholfrequenz)
    - Erstellung von Vorschaubildern/Thumbnails

- Möglichkeit für den Benutzer im Falle von
  - Bilddateien einen geeigneten Bildausschnitt zu wählen
  - Videodateien ein von drei verschiedenen Vorschaubildern (Thumbnails) zu wählen
- Änderungsmöglichkeit für Benutzer an ihren Medien (Titel, Beschreibung, Schlagwörter, Gruppenzuordnung)
- Persistente Speicherung der Daten in Datenbank und Dateisystem

5.2.2 Verwendete Technologien

Anhand der in den Kapiteln 2, 3 und 4 aufgezeigten Möglichkeiten wurde in jedem dieser Abschnitte eine Entscheidung nach der zu verwendenden Technologie getroffen, die bei dieser Implementation zum Einsatz kommen soll. In der nachfolgenden Übersicht werden die für diese Prototyp-Implementation ausgewählten Technologien noch einmal zusammenfassend dargestellt.

Aufgabe	**Technologie**
Datenquelle	Relationale Datenbank
Programmiersprache/-plattform	Java
Zugriff auf Datenquelle	Hibernate
Webframework	Apache Wicket
Bildverarbeitung/-bearbeitung	ImageMagick
Videoverarbeitung/-bearbeitung	FFmpeg über Xuggler

Tabelle 5.1: *Übersicht der genutzten Technologien*

Mit dieser allgemeinen Zusammenstellung der einzelnen Technologiebereiche ist die Darstellung eines konkreten Beispiel-Modells möglich. Siehe dazu Abbildung 5.1. Als kurze Bemerkung sei hinzugefügt, dass die Kommunikation zwischen dem Webframework Apache Wicket und den beiden Verarbeitungsschnittstellen bzw. Bibliothek zu ImageMagick sowie FFmpeg über einen zusätzlichen Verwaltungsmechanismus (gestrichelte Linie in der Abbildung) stattfindet. Im Abschnitt 5.2.9 wird im Detail darauf eingegangen.

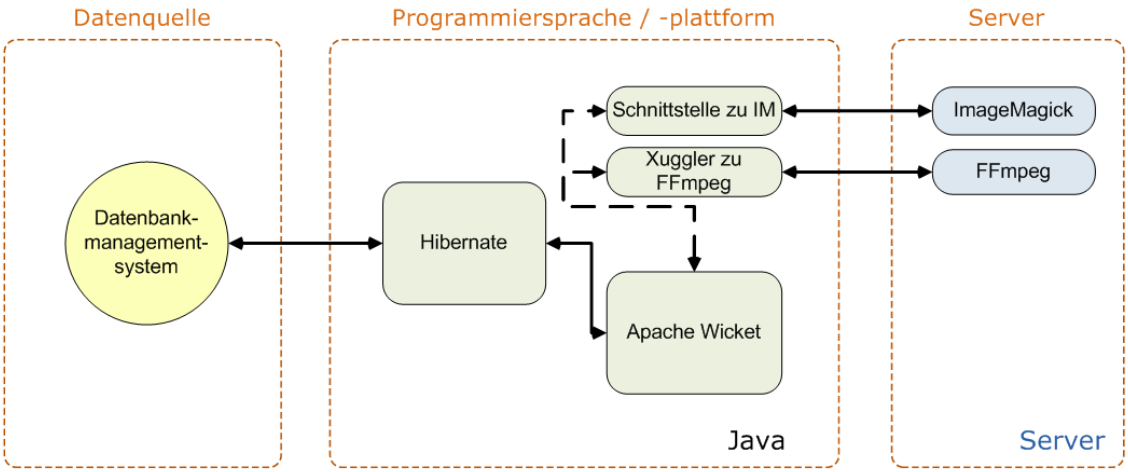

Abbildung 5.1: *Allgemeine Darstellung der implementierten Technologien*

Weiterhin ist zu sagen, dass diese Java-Anwendung die Möglichkeit bietet mit Hilfe von Hibernate über JDBC verschiedene Datenbanken nutzen zu können. Ebenso ist sie auf unterschiedlichen Serversystemen lauffähig, wobei in diesem Fall Einschränkungen aufgrund der beiden notwendigen Softwarepakete ImageMagick und FFmpeg bestehen.

5.2.3 Datenbank

Wie in dem letzten Abschnitt beschrieben, ist in dieser Applikation das Framework Hibernate für den Zugriff und die Kommunikation zwischen Datenquelle und der Anwendung verantwortlich. Dieser Zugriff erfolgt über Java Database Connectivity (JDBC), wodurch jedes beliebige Datenbankmanagementsystem, welches diese Zugriffsmethode unterstützt, verwendet werden kann. Im Falle dieser Applikation wird PostgreSQL[1] als Datenbanksystem verwendet.

Datenbankmodell

Das Datenbankmodell ist Ausgangspunkt und Voraussetzung für die weiteren Entwicklungen an dieser Anwendung, aus diesem Grund ist ein ausführlicher detaillierter Entwurf des Modells erforderlich. Die Datenbank des Prototyps besteht aus insgesamt 15 Tabellen, wobei sich die reinen Informationen der Multimedia-Daten auf zwei Tabellen beschränken. Zur klaren Trennung der Media-Daten in der Datenbank werden die Bild- und Videoinformationen separat mit eindeutiger Zuordnung zum jeweiligen Nutzer untergebracht. Die dazugehörigen Modifikationen für die spätere einheitliche Darstellung in der Webanwendung, die aus den Originaldateien erstellt werden, sind ebenfalls in eigenen Tabellen aufbewahrt und dem jeweiligen originalen Media-Datensatz zugeordnet. Weiterhin ist eine gesonderte Tabelle für die Vorschaubilder

[1] PostgreSQL - http://www.postgresql.org/

(Thumbnails) der Media-Daten notwendig. Ebenso existieren weitere Tabellen für die Gruppen- und Schlagwörter(Tags)-zuordnung. Im nachfolgenden soll der Aufbau und Zusammenhang der Tabellen durch das Entity-Relationship-Modell vereinfacht verdeutlicht werden.

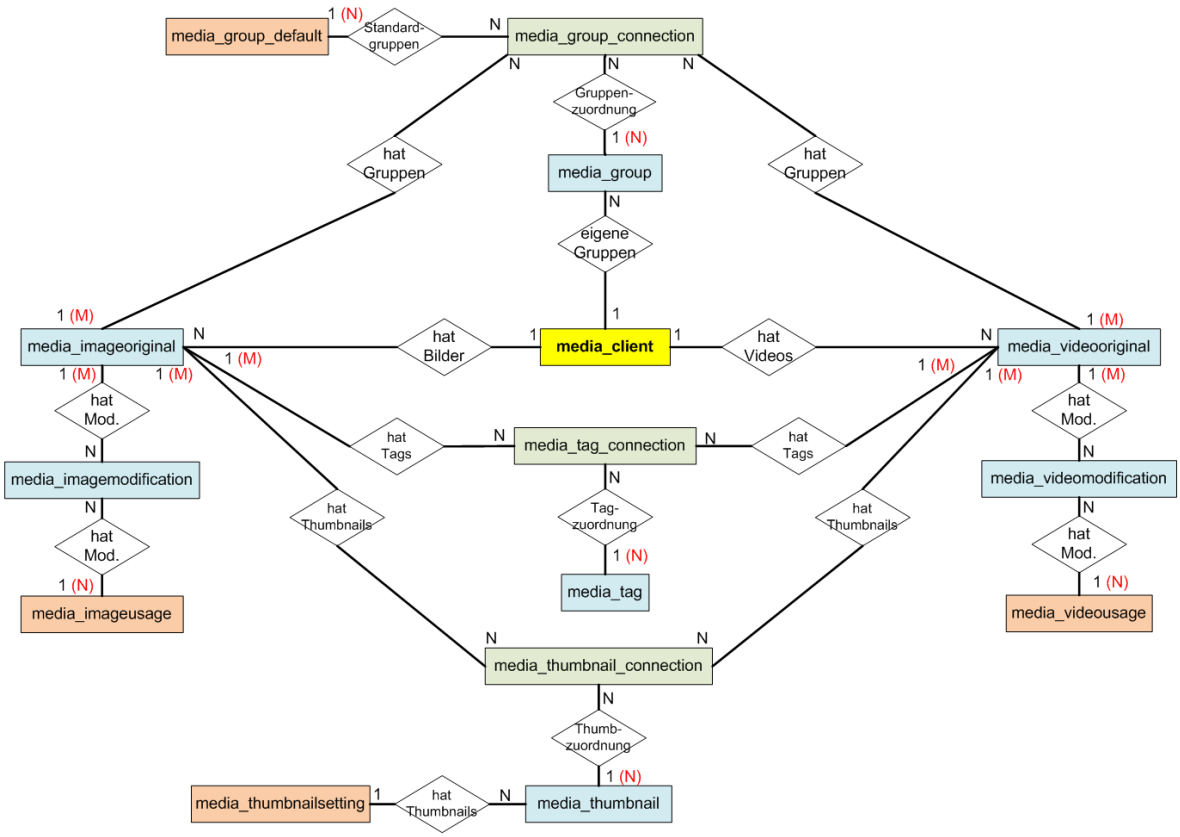

Abbildung 5.2: *Tabellenstruktur im ERM*

In der Abbildung 5.2 sind die Tabellen aus Gründen der Übersicht farbig markiert. Der Ausgangspunkt ist die „media_client“ Tabelle(gelb) in der Mitte, die den Benutzer, welcher Multimedia-Daten hinzufügen und verwalten kann, darstellt. Des Weiteren besteht die Möglichkeit für den jeweilien Nutzer eigene Gruppen zu verwalten und den eigenen Media-Daten diese zu zuordnen. Die Inhalte der in dieser Abbildung blau dargestellten Tabellen können durch Interaktionen mit dem Benutzer beeinflusst bzw. verändert werden. Im Gegensatz dazu beinhalten die Tabellen im roten Farbton Vorgabewerte, die durch den Administrator voreingestellt und im späteren Verlauf für die automatische Verarbeitung der Multimedia-Daten herangezogen werden. Darauf wird in den nachfolgenden Abschnitten tiefgründiger eingegangen. Nun zurück zum Ausgangspunkt, der „media_client“ Tabelle im gelben Farbton.

Wie aus dieser Abbildung ersichtlich ist, stehen jeweils die beiden Media-Tabellen „media_imageoriginal“ und „media_videooriginal“ sowie die Gruppentabelle „media_group“ mit der „media_client“ Tabelle in einer 1 : N Beziehung, wobei die Zuordnung über den Primärschlüssel „id“ (primary key) der Nutzertabelle stattfindet. Diese Zuordnung wird mit Hilfe

des Fremdschlüssels (foreign key) in den jeweils anderen Tabellen vervollständigt. Die eben genannte 1 : N Beziehung wird anhand der Primär- sowie den zugehörigen Fremdschlüsseln realisiert und besagt, dass genau eine Entität (ein Nutzer) der Nutzertabelle auf eine beliebige Anzahl Entitäten der jeweils anderen Tabellen verweisen kann. So zum Beispiel kann ein Nutzer beliebig viele Media-Daten vom Typ Bild verwalten, denen wiederum jeweils beliebig viele Modifikationen „media_imagemodification“ zugeordnet werden können, die jeweils einer bestimmten Bildnutzung „media_imageusage“ angehören.

Die M : N Relation ist eine weitere Beziehung im ERM, die nachfolgend am Beispiel der Schlagwörtertabelle „media_tag“ beschrieben wird. In dieser Tabelle befinden sich Schlagwörter, zu jedem Schlagwort können beliebig viele Media-Daten (Bilder, Videos) zugeordnet werden und zu jedem Media-Datensatz ebenso beliebig viele Schlagwörter. Diese Beziehung kann in einer relationalen Datenbank nur durch eine weitere Tabelle realisiert werden. In dem ERM sind diese Tabellen in einem grünlichen Farbton dargestellt und in dem jeweiligen Tabellennamen wird ebenso durch die Bezeichnung „connection“ darauf hingewiesen. Des Weiteren trifft diese Relation auf die beiden Modifikationstabellen zu, die allerdings in einem blauen Farbton erscheinen, welcher oben beschrieben wurde.

Bei dem Entwurf des Datenbankmodells wurde darauf geachtet, dass die Kriterien der Normalisierung eingehalten wurden. Aus Übersichtsgründen etwas schwer im Entity-Relationship Modell zu erkennen, liegt dieses Modell selbst in der 3. Normalform vor. Das bedeutet, dass Redundanzen vermindert wurden, jede einzelne Tabelle Daten eines Themengebietes enthält und ebenso die Nicht-Schlüsselfelder eines jeweiligen Datensatzes selbst von nur einem Schlüssel, dem Primärschlüssel, abhängig sind, und bei vorhandenem Fremdschlüsselfeld dieses auf exakt einen Datensatz verweist.

Da das Datenbankmodell in dieser Form abgeschlossen werden und zum Einsatz kommen kann, besteht der nächste Schritt aus dem Zugriff aus Java heraus auf die Datenbank selbst und wird im nachfolgenden Abschnitt erläutert.

5.2.4 Hibernate

Wie in dem vorhergehenden Abschnitt „Datenbank“ bereits erwähnt, wird für den Zugriff auf die Datenbank das Framework Hibernate[2] verwendet. Hibernate ist ein Object-Relational Mapping Framework, kurz ORM Framework, für Java und zudem Open-Source. Es unterstützt von vornherein zahlreiche Datenbankmanagementsysteme, sowie deren SQL-Dialekte, wodurch dem Entwickler weitestgehend freie Wahl der Datenbank gegeben wird. Mit Hilfe der umfangreichen und stetig wachsenden Community und aktiven Entwicklung ist stets für Unterstützung gesorgt. ORM Framework bedeutet, dass Hibernate selbst in der objektorientierten „Welt“ von Java fungiert, diese allerdings mit der „Welt“ von relationalen Datenbanksystemen verbindet und es somit ermöglicht, gewöhnliche Java-Objekte in einer relationalen Datenbank persistent

[2] Hibernate - https://www.hibernate.org/

zu verwalten, d.h. Speichern, Verändern und Entfernen dieser Objekte. Im Folgenden soll die Verbindung der beiden „Welten“ näher betrachtet werden.

Die Verbindung von Java-Klassen und Tabellen einer relationalen Datenbank erfolgt über das Mapping. Mit dem Mapping wird die Zuordnung der Tabellen und deren Spalten an die jeweiligen Java-Klassen und deren Attribute festgelegt. Dies kann auf zwei verschiedene Arten durchgeführt werden. Zum einen mit Hilfe von Annotations, die in Java-Klassen mit einem „@“ gekennzeichnet sind und somit das Mapping direkt in den entsprechenden Klassen ermöglichen und zum anderen mit Hilfe von XML-Dateien. Bei der letzteren Variante findet das gesamte Mapping in den jeweiligen zugehörigen XML-Dateien statt, d.h. zu jeder Java-Klasse, die auf eine Tabelle der Datenbank abgebildet wird, existiert eine XML-Mapping-Datei. Da die XML-Mapping Variante diejenige ist, die ausgewählt wurde, sollen nachfolgende Beispiele die Handhabung verdeutlichen.

Als erstes einen Ausschnitt aus der „MediaClient“ Java-Klasse, die ebenso im ERM Ausgangspunkt ist (Tabelle im gelben Farbton).

```
public class MediaClient implements java.io.Serializable {
    private int id;
    private String name;
    private Set<MediaImageoriginal> mediaImageoriginals =
                new HashSet<MediaImageoriginal>();
    ...
}
```

Beispiel 5.1: *Hibernate-Mapping der Java-Klasse MediaClient*

Wie in dem Beispiel 5.1 zu erkennen ist, implementiert die Java-Klasse „MediaClient“ das Interface „Serializable“ aus dem Paket „java.io“. Mit Hilfe dieses Interfaces ist es möglich Java-Objekte in den persistenten Zustand zu überführen. Weiterhin sind in diesem Quellcode-Ausschnitt drei Klassenattribute abgebildet, wobei das Attribute „id“ das Primärschlüsselfeld repräsentieren soll. Eine Besonderheit ist das Attribut „mediaImageoriginals“, welches eine bestimmte Menge an „MediaImageoriginal“ Klassen enthalten kann. Es symbolisiert die 1 : N Verbindung zwischen den Tabellen „media_client“ und „media_imageoriginal“ und wird in diesem Set abgebildet.

Nach dem Quellcode-Auszug aus der „MediaClient“ Klasse kann es nun zur eigentlichen zugehörigen Mapping-Datei kommen, wobei ebenso lediglich ein Ausschnitt dargestellt werden soll.

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
  "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
  "http://hibernate.sourceforge.net/
```

```
    hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="com.maxity.media.data.MediaClient"
        table="media_client" schema="public">
        <id name="id" type="int">
            <column name="id" />
            <generator class="native">
                <param name="sequence">
                    media_client_id_seq
                </param>
            </generator>
        </id>
    <property name="name" type="string">
        <column name="name" length="100" not-null="true" />
    </property>
    <set name="mediaImageoriginals" inverse="true"
            lazy="true"
            table="media_imageoriginal"
            fetch="select">
        <key>
            <column name="id_client" not-null="true" />
        </key>
        <one-to-many
            class="
                com.maxity.media.data.MediaImageoriginal
            " />
    </set>
    ...
    </class>
</hibernate-mapping>
```

Beispiel 5.2: *Hibernate-Mapping: XML-Datei zur MediaClient Klasse*

Das Beispiel 5.2 zeigt einen Ausschnitt der XML-Mapping-Datei für die Java-Klasse „MediaClient“. Sie beginnt mit dem Prolog, in welchem die XML-Deklaration mit Versionsangabe sowie die Dokumententypdefinition (DTD) enthalten sind. Die angegebene DTD regelt die Gültigkeit der im Wurzelelement enthaltenen Attribute und Elemente, d.h. das durch diese Definition festgelegt ist, welche Elemente und Attribute in dieser XML-Datei gültig sind. Nach dem Prolog folgt das Wurzelelement.

Nachfolgende Elemente und Attribute im Wurzelelement definieren die Abbildungsvorschrif-

ten. So wird durch die Angabe des „class“-Tags die zugehörige Java-Klasse „com.maxity.media.data.MediaClient“ angegeben, sowie die dazugehörige Tabelle in der Datenbank „media_client“. Dieser Tag beinhaltet alle weiteren Mappingvorschriften für die einzelnen Attribute in der Java-Klasse in Bezug auf die Spalten in der angebenen Tabelle bzw. den Relationen zu dieser Tabelle.

Das erste Element im „class“-Tag ist „<id>“ und bestimmt den Primärschlüssel unter Angabe des Attributes in der Java-Klasse, dem Spaltennamen in der Datenbank sowie dem Datentyp. Weiterhin wird in dem „id“-Tag durch Angabe des „generator“-Tags beschrieben, wie dieser erzeugt wird. In diesem Fall wird die entsprechende Sequenz aus der Datenbank angegeben, die für die Erzeugung eindeutiger Nummern zur Verfügung steht. Bei allen weiteren „Nichtschlüssel“-Attributen findet die Zuordnung durch die Angabe von „<property>“-Tags statt. Im obigen Beispiel wird das Java-Attribut „name“ vom Datentyp String auf die gleichnamige Spalte in der Tabelle abgebildet. Dabei können zusätzliche Angaben für die jeweilige Spalte gesetzt werden, so z.B. die maximale Zeichenlänge, die ein Feld in dieser Spalte aufnehmen kann oder aber auch ob ein Feld in dieser Spalte NULL-Werte annehmen darf. Das letzte Element in diesem Beispiel beschreibt die Beziehung („one-to-many“) zur Klasse „com.maxity.media.data.MediaImageoriginal“ und gibt zusätzlich die Tabelle mit dem entsprechendem Fremdschlüssel in der Datenbank an. Das Attribut „inverse“ besagt in diesem Fall, dass diese Klasse („MediaClient“) als Gegenstück der Beziehung einer anderen („MediaImageoriginal“) betrachtet wird und somit die Klasse „MediaImageoriginal“ für die Speicherung dieser Relation verantwortlich ist. Die beiden übrigen Attribute „lazy“ und „fetch“ bestimmen die Ladestrategie, die in diesem Beispiel besagt, dass die „MediaImageoriginal“-Objekte erst bei Bedarf nachgeladen werden. Diese Mapping-Dateien (Java-Klassen und zugehörige XML-Dateien) müssen nicht von Hand erstellt werden, sie können mit Hilfe von zusätzlichen Tools (z.B. dem Eclipse Plugin Hibernate Tools) aus einer bestehenden Datenbank heraus generiert werden.

Das Mapping bildet nun die Grundlage für die weiteren Schritte, wobei es im Nachfolgenden um das Erstellen der Hibernate-Session, sowie dem Laden und Speichern der Objekte gehen soll.

Um eine Hibernate-Session erzeugen zu können, werden einige Voreinstellungen benötigt, die in einer separaten Datei festgehalten werden. Diese Datei ist die „hibernate.cfg.xml“, in der die Hibernate Konfiguration stattfindet. Sie kann unter anderem die Datenbankzugangsdaten enthalten, sowie die Definition des Datenbanktreibers und -dialektes und vor allem die Angaben zu den jeweiligen XML-Mapping-Dateien. Ein Beispiel für eine Konfigurationsdatei ist auf der Hibernate Tutorial Webseite zu finden[29]. Nun zur eigentlichen Erstellung der Hibernate-SessionFactory.

Das nachfolgende Beispiel zeigt ebenfalls einen Ausschnitt einer Java-Klasse die auf die Hibernate-Konfigurationsdatei zu greift und mit ihrer Hilfe eine SessionFactory erzeugt.

```
public class MaxityMediaHibernate {
    private static SessionFactory sessionFactory;
    static {
        try {
            Configuration cfg = new Configuration();
            sessionFactory = cfg
            .configure(
                "com/maxity/media/data/hibernate.cfg.xml"
            ).buildSessionFactory();
        } catch (Throwable e) {
            throw new ExceptionInInitializerError(e);
        }
    }
    ...
}
```

Beispiel 5.3: *Hibernate-SessionFactory mit Hilfe der Konfigurationsdatei „hibernate.cfg.xml“*

Wie in dem Beispiel 5.3 zu sehen ist, wird die XML-Konfigurationsdatei direkt angegeben und daraus eine SessionFactory erstellt. Mit Hilfe dieser Factory werden Sessions verwaltet, welche im nächsten Schritt für das Laden und Speichern der Objekte benötigt werden. Das Laden eines Objektes kann im einfachsten Fall über die „load“-Funktion der Session erfolgen, das Beispiel 5.4 soll dies verdeutlichen.

```
Session hiberSession = sessionFactory.openSession();
MediaImageoriginal mi = (MediaImageoriginal)hiberSession
                        .load(MediaImageoriginal.class, id);
```

Beispiel 5.4: *Ein konkretes Objekt anhand einer eindeutigen ID mit „load“-Funktion laden*

Eine weitere Möglichkeit bietet die „createQuery“-Funktion, wobei in diesem Fall konkrete HQL Anweisungen anzugeben sind.

```
Session hiberSession = sessionFactory.openSession();
Query query = hiberSession.createQuery(
    "FROM MediaImagemodification as mim " +
    "WHERE mim.id IN (" +
        "SELECT mim.id " +
        "FROM MediaImagemodification as mim, " +
                "MediaImageoriginal as mi, " +
                "MediaImageusage as miu " +
        "WHERE mim.mediaImageoriginal = mi " +
```

```
            "AND mim.mediaImageusage = miu " +
            "AND mi.id = :mediaImageoriginalId " +
            "AND miu.id = :mediaImageusageId" +
        ")");
query.setInteger("mediaImageoriginalId", idImageoriginal);
query.setInteger("mediaImageusageId", idImageusage);
List<MediaImagemodification> modificationList =
    new ArrayList(query.list());
```

Beispiel 5.5: *Liste mit Objekten mit „createQuery“-Funktion laden*

Zu diesen beiden Quellcode-Beispielen 5.4 und 5.5 ist zu sagen, dass sie sich ausschließlich auf die jeweiligen Funktionen konzentrieren, ohne dabei auf das zugehörige Exception-Handling (Ausnahmebehandlung) einuzgehen. In dem ersten dieser beiden Beispiele sieht man sehr deutlich, wie unkompliziert es Hibernate ermöglicht einen bestimmten Datensatz und somit ein bestimmtes Objekt lediglich unter Angabe des Primärschlüssels zu lesen bzw. zu laden. Im zweiten Beispiel dagegen wird eine HQL (Hibernate Query Language) Anweisung benötigt. Hierbei ist zu beachten, dass HQL die Java-Klassennamen („MediaImagemodification“ und nicht wie gewohnt die Tabellennamen aus der Datenbank („media_imagemodification“) erwartet.

Ebenso unkompliziert wie die Möglichkeiten des Ladens von Objekten bietet Hibernate Funktionen für das Speichern und Entfernen von Objekten an. Weitere Informationen dazu sind auf der Hibernate Webseite[30] zu finden.

In Anbetracht der Tatsache des abgeschlossenen Datenbankmodells sowie der Datenbank selbst und nun das Ende des Abschnitts Hibernate, der Zugriffserläuterung auf die Datenbank mit Hilfe des Java Framework, erreicht wurde, kann der nächste Schritt in Angriff genommen werden. Dabei handelt es sich um das Java Web Framework Apache Wicket, welches im nachfolgenden Abschnitt näher erläutert wird. Bevor es nun zum nächsten Abschnitt geht, soll anhand des vereinfacht dargestellten UML-Klassendiagramms in Bezug auf die ERM-Abbildung 5.2 das Mapping auf die Java KLassen veranschaulicht werden.

Wie in dem UML-Klassendiagramm 5.3 zu sehen ist, sind alle Java Klassen in diesem Package von der Klasse „MaxityMediaObject“ aus dem Package „com.maxity.media.core“ abgeleitet. Dies vereinfacht den Zugriff sowie die Verarbeitung der Daten aus der Datenbank. Da diese Klassen lediglich das Mapping, also die Abbildung, der Tabellen aus der Datenbank auf Java Objekte realisieren, werden zusätzlich noch Java Klassen für die jeweiligen Datenbankaktionen zum Laden, Speichern bzw. Entfernen von Objekten benötigt. In dem nachfolgenden UML-Klassendiargramm 5.4 sind die entsprechenden Handler aufgeführt, wobei die Bezeichnungen der jeweiligen Klassen den Mapping Klassen auffällig ähnlich erscheinen. Dies wurde aus Gründen der schnelleren Übersicht sowie besseren Wartbarkeit entschieden und durchgeführt.

Nach diesen Einblicken kann nun, wie schon erwähnt, der nachfolgenden Abschnitt über das

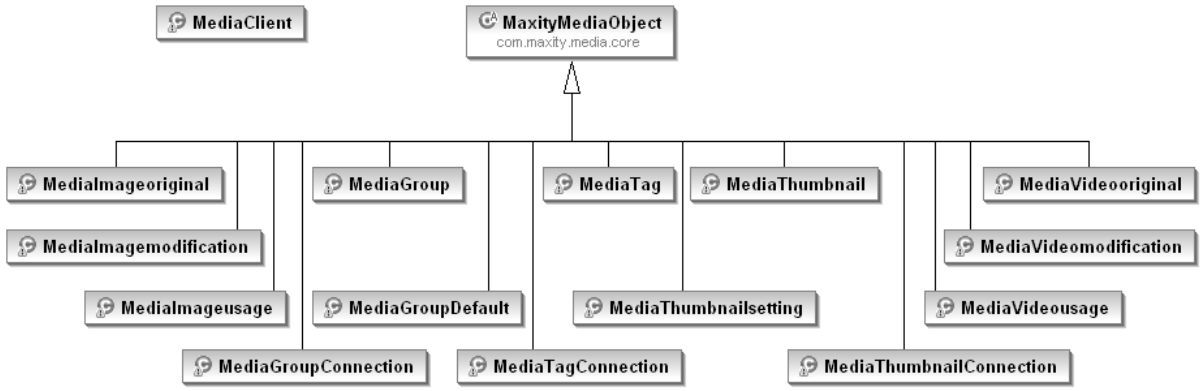

Abbildung 5.3: *Hibernate Mapping: Java Klassen im Package „com.maxity.media.data“*

Web Framework Apache Wicket beginnen.

5.2.5 Web Framework Apache Wicket

Apache Wicket[3] ist ein Java MVC Web Framework und ebenso wie Hibernate ein Open-Source-Projekt. Das Ziel, welches Wicket verfolgt, ist die Überbrückung des zustandslosen HTTP-Protokolls und der objektorientierten Programmiersprache Java. Die Programmierung bzw. Erstellung einer Webapplikation erfolgt, wie eben beschrieben, in Java und die Gestaltung der Webanwendung findet wie üblich in den Html-Dateien statt. In Wicket wird demnach die gesamte Programmlogik (Java) von dem Design bzw der Darstellung (Html) strikt getrennt, wodurch ein einheitliches Programmieren, bessere Wartbarkeit und weniger Fehlerquellen möglich sind.

Grundsätzlich besteht eine Wicket-Anwendung aus einer Java-Klasse, die Webapplication ableitet, dem Deployment Descriptor und den Webseiten, die durch Java-Klassen und den zugehörigen gleichnamigen Html-Dateien gebildet werden. Aufgrund der Komponentenarchitektur ist es in Wicket möglich die einzelnen Wicket-Komponenten dynamisch und beliebig zu kombinieren, die Seite suchmaschinenfreundlich zu gestalten und dank der umfangreichen integrierten Ajax-Bibliotheken moderne Desktop ähnliche Applikationen zu erstellen. Die Abbildung 5.5 soll den grundsätzlichen Aufbau in einer möglichen Konstellation vereinfacht darstellen.

Wie in der Abbildung 5.5 zu sehen ist, treten die einzelnen Webseiten („Index“, „Suchergebnis“ und „SeiteXyz“) jeweils als Pärchen (die Java- und Html-Datei) auf. Zusätzlich ist zu erkennen, dass diese von einer Wicket-Komponente abgeleitet sind, das in diesem Fall die Komponente „WebPage“ symbolisiert und wie der Name schon sagt, eine Webseite repräsentiert. Im Gegensatz dazu leiten „SuchPanel“ und „Weitere Komponente“ eine andere Komponente ab, wobei es sich in diesem Fall um die Wicket-Komponente „Panel“ handelt. Panel sind her-

[3] Apache Wicket - http://wicket.apache.org/

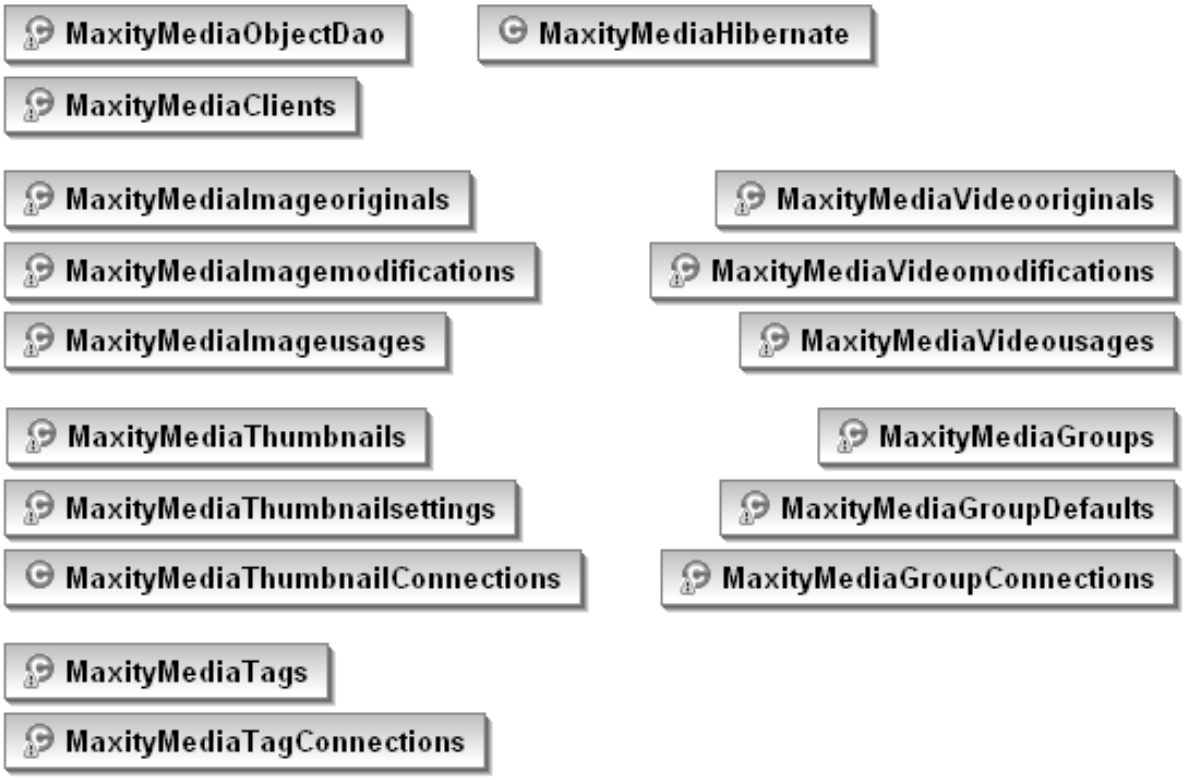

Abbildung 5.4: *Datenbankaktionen in entsprechenden Java Klassen im Package „com.maxity.media.data.handler“ gekapselt*

vorragende Komponenten, die ebenso als Java-Html-Paar auftreten und in unterschiedlichen Webseiten eingebunden werden können, wodurch wiederverwendbarer Code erstellt werden kann und somit doppelte Codesegmente vermieden werden können. In diesem Beispiel wird das „SuchPanel“ in den drei Webseiten verwendet. Der Deployment Descriptor ist für das Mapping und weitere Konfiguration der Applikation zuständig. Die Applikation an sich wird beim Start des Servers nur einmal geladen, d.h. es gibt zur gesamten Laufzeit nur eine Instanz dieser Anwendung. Nun zur eigentlich Verbindung der Java- und Html-Dateien.

Diese Verbindung von Html-Code und objektorientiertem Java-Code erfolgt über einen von Wicket eigenen bereitgestellten XHTML-Namensraum. Dabei sind lediglich Referenzen in den Html-Dateien zu vergeben, die als Bezugspunkt in den Java-Klassen verwendet werden. Diese Referenz ist das Attribut „wicket:id“, welches abhängig von der späteren Aufgabe in beliebigen XHTML-Tags eingesetzt werden kann. Zur Veranschaulichung soll das nachfolgende Minimalbeispiel dienen.

```
<html>
<body>
    <span wicket:id="text">[hier steht der Text]</span>
</body>
</html>
```

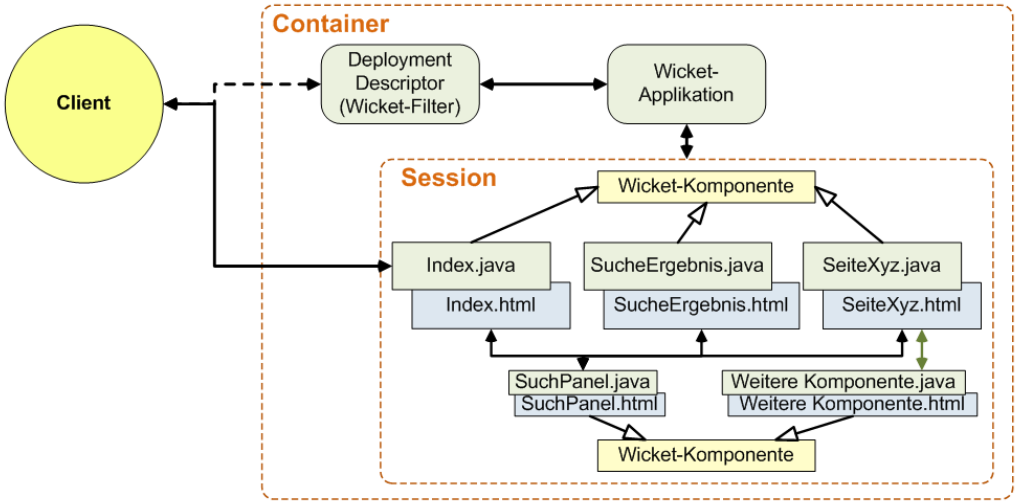

Abbildung 5.5: *Möglicher Aufbau einer Wicket-Anwendung*

Beispiel 5.6: *Index.html mit Wicket-Referenz*

In diesem Beispiel 5.6 ist das Attribut „wicket:id“ in einem Span-Tag mit dem eindeutigen Wert „text“ versehen.

```
public class Index extends WebPage {
private String newText = "das ist der neue Text";
public getNewText() {return newText;}
public setNewText(String newText) {this.newText = newText;}
public Index() {
        add(new Label(
                "text",
                new PropertyModel(this, "newText")
        ));
}
```

Beispiel 5.7: *Index.java mit Bezug auf Wicket-Referenz*

In diesem Quellcode-Beispiel 5.7 ist zu beachten, dass die „Index.java“ von der Wicket-Komponente „WebPage“, wie in der Abbildung 5.5 dargestellt, abgeleitet ist und somit eine Webseite darstellt. Anders als in Abbildung 5.5 wurde in diesem Beispiel auf das „SuchPanel“ verzichtet. Die Referenz aus „Index.html“ wird nun in dieser Webseite für die weitere Verwendung eingesetzt. In diesem Beispiel wird lediglich ein Label mit Bezug auf diese Referenz und einem neuen Wert dieser Webseite hinzugefügt. Auf diesen Wert wird mit Hilfe eines „PropertyModel“ zugegriffen.

Das eben erwähnte „PropertyModel“ gehört zu den Modellen des Wicket Frameworks, welche der Trennung der Anwendungsschicht und Präsentationsschicht dienen bzw. diese anein-

ander binden. Mit Hilfe des Modell-Prinzips wird den Wicket-Komponenten die Möglichkeit gegeben Daten zu erhalten bzw. unter entsprechenden Voraussetzungen Daten an das jeweilige Modell zu übergeben. In Wicket existieren eine Reihe von unterschiedlichen Modell-Typen, die verschiedene Vorteile mit sich bringen. In dem Beispiel 5.7 wurde das „PropertyModel“ verwendet, welches das Objekt („this“) beinhaltet und mit Hilfe des nachfolgenden Ausdrucks („newText“) auf das zugehöriges Attribut des Objektes mittels Getter und Setter zugreift. Für weitere Informationen zum Thema Wicket-Modelle ist die Wicket-Wiki-Webseite[31] sehr zu empfehlen.

Das Web Framework Apache Wicket kann im Gegensatz zu vielen anderen Java Web Frameworks durch die schnelle Einsatzfähigkeit und der unkomplizierten XML-Konfiguration glänzen. Das schlanke Framework enthält zahlreiche Komponenten für einen sofortigen Aufbau einer Webanwendung. Außerdem ist zu beachten, dass Wicket nicht wie einige andere Frameworks ein integriertes Datenbankhandling beinhaltet und der Entwickler somit auf andere Lösungen zurückgreifen muss. Dies muss kein Nachteil sein, da sich in Wicket viele Lösungen umsetzen bzw. integrieren lassen.

Abschließend lassen sich die bisherigen erläuterten Technologien sehr gut in einem Drei-Schichten-Architektur Modell einordnen, wie in der Abbildung 5.6 zu sehen ist.

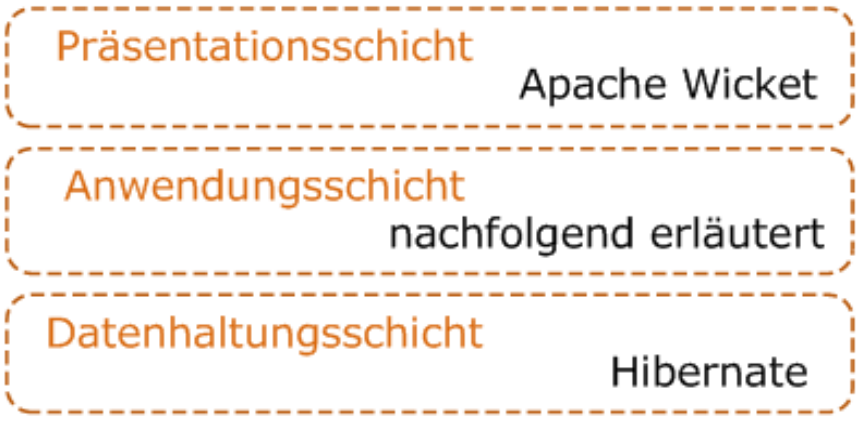

Abbildung 5.6: *Einordnung bisheriger Technologien in Drei-Schichten-Architektur Modell*

Der Anwendungsbereich ist, wie in dieser Abbildung dargestellt, noch nicht festgelegt bzw. erläutert. Dies soll in den nachfolgenden Abschnitten im Detail geschehen, angefangen mit dem Upload der Multimedia-Dateien, welchem die Bildverarbeitung mit Hilfe des ImageMagick Tools und die Verarbeitung der Videodaten unter Anwendung von Xuggler folgen.

Abschließend wird mit der Abbildung 5.7 ein Ausschnitt des UML-Klassendiagramms aus dem Wicket-Package des Prototyps dargestellt und soll einen ersten Eindruck des Aufbaus vermitteln.

Wie bereits erwähnt und mit Hilfe der Abbildung 5.7 angedeutet, wurden die Java-Klassen des Prototyps nach Funktionalität in den jeweiligen Packages untergebracht. In der nachfolgenden Grafik 5.8 sind insgesamt sechs Java-Packages dargestellt, wobei anhand der Beschrif-

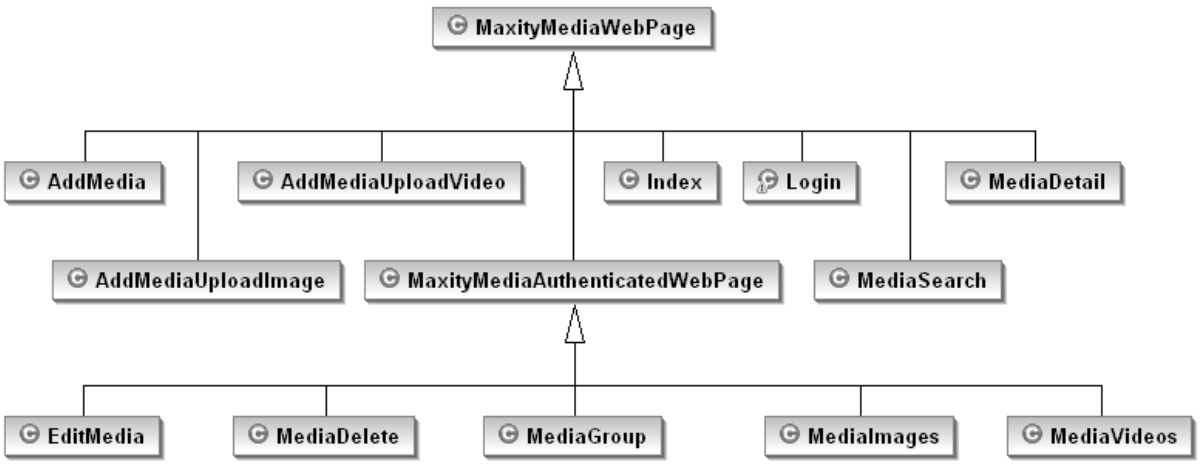

Abbildung 5.7: *UML-Klassendiagrammausschnitt vom Wicket-Package des Prototyps*

tungen auf die jeweiligen Bereiche und Funktionen geschlossen werden kann. Dabei stellt das Package „com.maxity.media.wicket“, wie in dem Drei-Schichten-Architektur Modell schon beschrieben, die Präsentationsschicht dar. Es beinhaltet alle Webseiten im Sinne von Apache Wicket, wobei die zugehörigen Dateien, wie Css-, Javascript- und Sonstige-Daten, in dem Unterpaket „resources“ untergebracht sind. Dieses Prinzip ist ebenso auf das Datenbank-Handling anzuwenden. Im Package „com.maxity.media.data“ befinden sich alle für Hibernate notwendigen Mapping-Dateien (Java-Klassen und XML-Daten) und in dem Unterpaket „handler“ sind die jeweiligen Java-Klassen zu finden, welche die entsprechenden Datenbankaktionen (Objekt(e) laden, speichern und entfernen) in sich kapseln. Das Package „com.maxity.media.core“ repräsentiert die Anwendungsschicht und übernimmt die Kernaufgaben des Prototyps - die Verarbeitung der Multimedia-Daten. Für zusätzliche wiederkehrende Funktionen oder allgemeine Hilfsfunktionen stehen die Java-Klassen im Package „com.maxity.media.util“ zur Verfügung.

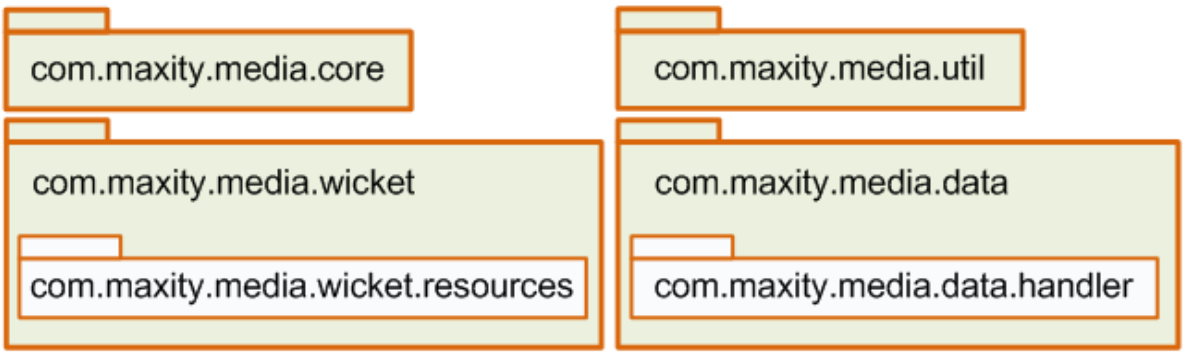

Abbildung 5.8: *Prototyp Packages*

Nachdem nun ebenso die Aufteilung der Java-Klassen erläutert wurde, kann nachfolgend auf den Upload der Multimedia-Dateien eingegangen werden.

5.2.6 Upload der Multimedia-Dateien

Der Upload von Multimedia-Dateien soll, wie im Kapitel 2 erläutert, mit Hilfe von Adobe Flash realisiert werden. Für diese Art des Uploads existieren zahlreiche dieser Upload Tools. Bei der Erstellung des Prototyps kommt die Flash- und JavaScript-Bibliothek SWFUpload[4] zum Einsatz, dabei ist sie zu anderen JavaScript Frameworks, wie zum Beispiel JQuery, kompatibel. Es erfüllt alle notwendigen, im Kapitel 2 beschriebenen, Kriterien. Da die Javascript Bibliothek JQuery bei der Entwicklung des Prototyps ohnehin eingesetzt wird, ist die Integration des Plugins ohne Probleme möglich. Zusätzlich sind zahlreiche Einstellungsmöglichkeiten während der Initialisierung des Tools sowie nach dieser realisierbar.

Die Java-Klasse „AddMedia“ im Wicket Package des Prototyps (siehe Abbildung 5.7), welche von der „MaxityMediaWebPage“ abgeleitet ist, stellt die Webseite für den Upload der Multimedia-Dateien dar. Sie beinhaltet zwei Panel, einen für den Bild- und den anderen für den Video-Upload. Je nachdem was der Nutzer auf den Server laden möchte (Bild oder Video), wird die JavaScript/Flash Bibliothek SWFUpload dem Mediatyp entsprechend initialisiert, d.h. abhängig von dem Typ (Bild oder Video) werden unterschiedliche Einstellungen für den Upload geladen. Der Nutzer kann aufgrund des Flash-Uploads diesen anhand des Ladebalkens verfolgen und ihn gegebenenfalls abbrechen. Die Datei an sich wird einer weiteren „MaxityMediaWebPage“ abgeleiteten Java-Klasse übergeben, wobei es sich im Falle von Bilddateien um die „AddMediaUploadImage“ Klasse und im Falle von Videodaten um die „AddMediaUploadVideo“ Klasse handelt. In diesen Java-Klassen werden notwendige Parameter für die Zuordnung sowie der zuerwartende Request überprüft. Sofern alles in Ordnung ist, wird dem Nutzer der erfolgreiche Upload benachrichtigt und die weiterführenden Verarbeitungsschritte können, abhängig vom jeweiligen Mediatyp, beginnen. Darauf soll in den nachfolgenden Abschnitten detailliert eingegangen werden. Die Grafik 5.9 soll unterstützend als Ausgangspunkt für die jeweiligen weiteren Abläufe mitwirken und die Trennung der weiteren Verfahrensweisen der jeweiligen Medientypen vereinfacht verdeutlichen.

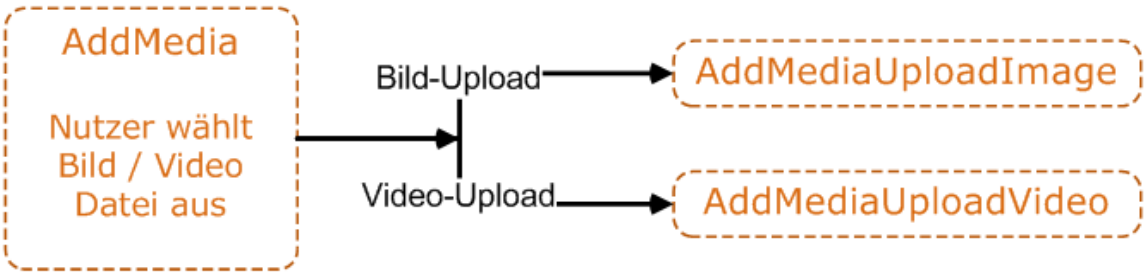

Abbildung 5.9: *Relevante Java-Komponenten beim Upload*

[4] SWFUpload - http://www.swfupload.org

5.2.7 Bildverarbeitung mittels ImageMagick API

In diesem Abschnitt wird im Detail auf die Verarbeitung von Bildern mit dem ImageMagick Software-Paket in Verbindung mit der Programmiersprache Java eingegangen. Dabei werden die Möglichkeiten der zu nutzenden Schnittstellen vorgestellt und erläutert.

ImageMagick Befehle „convert“ und „identify“ im Detail

Wie bereits in dem Kapitel 4 beschrieben, ist ImageMagick ein umfangreiches Software-Paket mit leistungsstarken Tools zur Manipulation und Verarbeitung von Bilddaten. Bei ImageMagick handelt es sich um eine Software, die über die Kommandozeile und nicht wie üblich in einem WYSIWYG[5]-Programm wie Photoshop angesprochen wird. Wie eben beschrieben besteht ImageMagick selbst aus einer Reiher von kleineren Utilities, die unterschiedlichste Funktionen im Bereich der Bildmanipulation übernehmen, so z.B. lassen sich mit dem Befehl „compare“ Bilder miteinander vergleichen und Unterschiede feststellen. Für die Implementation des Prototyps sind lediglich zwei dieser in ImageMagick enthaltenen Utitlities erforderlich. Das ist zum einen das Tool „identify“, welches, wie der Name schon verrät, Informationen aus den angegebenen Bilddateien liefert, und zum anderen das Tool „convert“. Mit dem zu letzt genannten Werkzeug lassen sich zahlreiche Manipulationen an Bildern vornehmen, wobei die jeweilige originale Bilddatei in ihrem Zustand unberührt bleibt und jeweils neue Bilddaten erstellt werden. Bezugnehmend auf die vorgegebenen Kriterien für die Manipulation und Verarbeitung von Bilddaten im Abschnitt 5.2.1 soll nachfolgend die Erfüllung dieser Pflichtkriterien mit Hilfe des reinen ImageMagick „convert“-Befehls veranschaulicht werden. Vorweg noch einmal die Liste der Kriterien, die der Prototyp für diesen Medientyp aufweisen soll.

- Erstellung modifizierter Bild-Varianten und Vorschaubildern (Thumbnails)
- Änderung bzw. Anpassung der Bildauflösung (Breite und Höhe) sowie Punktdichte (horizontale und vertikale Auflösung)
- Bildauflösung (Breite und Höhe) in quadratischer Form möglich unter Berücksichtigung des Seitenverhältnisses
- Änderung bzw. Anpassung des Formats und gegebenenfalls des Farbraums
- Auswahl des Bildausschnitts durch den Nutzer

Ausgehend von einem Bild im RGB-Farbraum soll in diesem ersten Beispiel 5.8 eine Modifikation erstellt werden, wobei das resultierende Bild an eine quadratische Auflösung in Breite und Höhe angepasst werden soll. Das Seitenverhältnis muss dabei beachtet werden. Ebenso findet eine Anpassung an der horizontalen und vertikalen Auflösung statt, d.h. die Punktdichte wird festgelegt.

[5] WYSIWYG - „What You See Is What You Get“

```
convert originalBild.png -resize 500x500 -density 72 \
    -gravity center -size 500x500 xc:white +swap \
    -composite modifiziertesBild.jpg
```

Beispiel 5.8: *ImageMagick convert-Befehl Beispiel 1*

Dieses Beispiel 5.8 erstellt aus dem Bild „originalBild.png“ eine neue Bilddatei mit dem Dateinamen „modifiziertesBild.jpg“. Dabei findet die erste Umwandlung vom Dateiformat PNG in das JPEG-Format statt, wobei dies von dem „convert“-Programm selbst anhand der Dateiendung erkannt wird. Durch den Parameter „-resize“ mit dem Wert „500x500“ wird die neue Bildauflösung (Breite und Höhe) angegeben. Die Festlegung der Punktdichte erfolgt mit dem Parameter „-density“ und dem Wert „72“, wobei die Maßeinheit in diesem Fall Punkte je Zoll (dots per inch) ist. Aufgrund der Berücksichtigung des Seitenverhältnisses kann es vorkommen, das die Zielauflösung mit der Breite und Höhe von 500 Pixeln nicht exakt eingehalten werden kann und das somit eine dieser Größen des resultierenden Bildes je nach Bildformat (Hoch- oder Querformat) einen geringeren Wert als 500 Pixel erhält. Um dennoch ein quadratisches Ergebnis zu erhalten ist es notwendig ein eigenständiges Bild in der angegebenen Größe zu erstellen. Dies geschieht mit den Parametern „-size 500x500“ und „xc:white“, wobei letzterer dafür sorgt, dass eine Ebene in weißem Farbton in der angegebenen Auflösung erzeugt wird. Mit Hilfe des „-composite“ Parameters werden die beiden Bilder, das verkleinerte Originalbild und das erzeugte weiße Bild, zusammengesetzt. Um dafür zu sorgen, dass das verkleinerte Originalbild im Vordergrund zu sehen ist und nicht einfach nur ein weißes Bild erstellt wird, ist das Setzen des Attributes „+swap“ notwendig, welches die Aufgabe hat, die beiden Bilder in der Reihenfolge zu vertauschen. Zu guter Letzt sorgt der Parameter „-gravity“ mit dem Wert „center“ dafür, dass das verkleinerte Originalbild zentriert dargestellt wird.

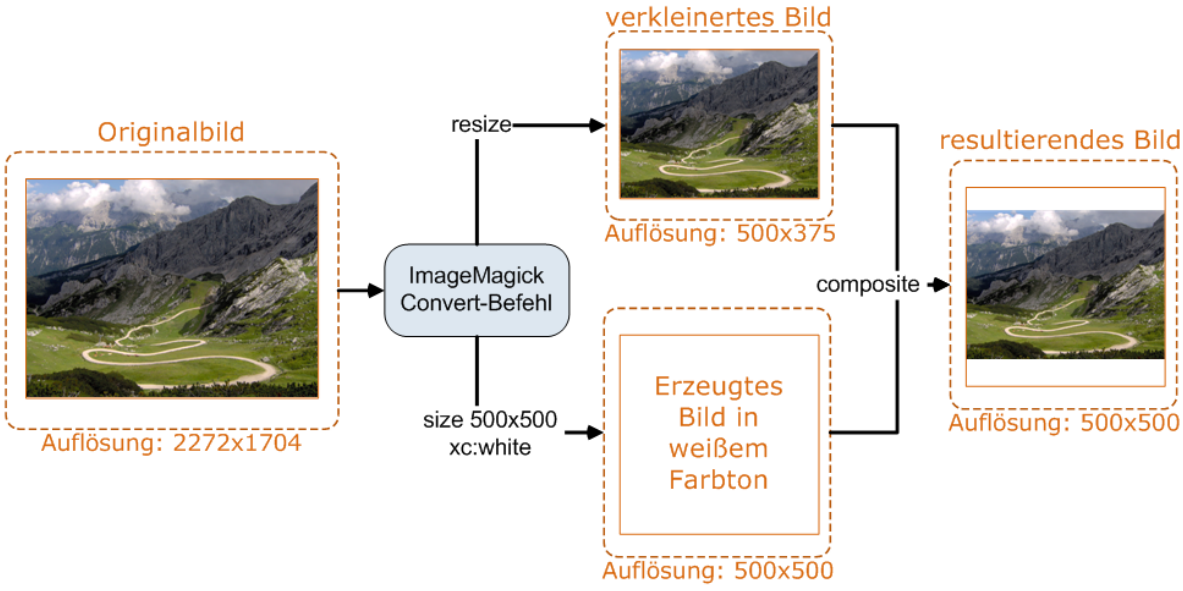

Abbildung 5.10: *Ablauf des ImageMagick „convert“-Befehls aus Beispiel 5.8*

Wie zu Beginn des Beispiels erwähnt wurde, entsteht nach Ausführung des oben angegebenen Befehls eine neue Bilddatei mit dem Namen „modifiziertesBild.jpg“. Die Abbildung 5.10 soll den Ablauf des Befehls vereinfacht veranschaulichen.

Wie in der Abbildung 5.10 zu sehen ist, führt die Konvertierung mit Hilfe der entsprechenden Parameter zu dem erwünschten Ziel und zu den ersten erfüllten Kriterien auf dieser Ebene. Für die Umwandlung eines Bildes im CMYK Farbraum in den RGB Farbraum ist es notwendig zusätzliche Parameter anzugeben. Diese zusätzlichen Parameter sind die jeweiligen Farbprofile die für die Konvertierung benutzt werden sollen.

Ein weiterer Punkt ist die zusätzliche Möglichkeit einen eigenen Bildausschnitt eines jeweiligen Bildes durch den Nutzer selbst wählen zu können. Dieses Problem kann ebenfalls mit ImageMagick und dem „convert“-Befehl gelöst werden. Dazu ist der Parameter „crop“ notwendig.

```
convert originalBild.png -crop 1000x1300+300+200 +repage \
    -resize 500x500 -density 72 \
    -gravity center -size 500x500 xc:white +swap \
    -composite modifiziertesBild.jpg
```

Beispiel 5.9: *ImageMagick convert-Befehl Beispiel 2*

In dem Quellcode-Beispiel 5.9 wird das Beispiel 5.8 um den eben erwähnten Parameter „crop“ sowie den dazugehörigen Werten erweitert. Des Weiteren wurde der Parameter „+repage“ hinzugefügt, der für die Aktualisierung der Meta-Daten verantwortlich ist. Um auf das Attribut „crop“ zurück zu kommen, ist zu sehen, wie schon erwähnt, das diesem gewisse Wert-Paare folgen. Dabei bestimmen die ersten beiden die Auflösung (die Breite mit 1000 Pixel und die Höhe mit 1300 Pixel) des neuen Ausschnitts und die beiden letzten die Verschiebung des Startpunktes (als Offset bezeichnet) auf der jeweiligen Achse, d.h. 300 Pixel in x-Richtung und 200 Pixel in y-Richtung. Ausgehend von der Beispielgrafik 5.10 unter Verwendung des „convert“-Befehls aus Beispiel 5.9 könnte das Ergebnis wie in der Abbildung 5.11 dargestellt ist, aussehen.

Mit Variationen des „convert“-Befehls von ImageMagick lassen sich alle aufgestellten Kriterien, die für die Verarbeitung von Bilddaten notwendig sind, erfüllen. Die Integration sowie die Nutzungsmöglichkeit von ImageMagick in Java wird nachfolgend erläutert. Vorerst folgt noch eine weitere kurze Einführung in den zweiten für diesen Prototypen notwendigen ImageMagick-Befehl „identify“.

Der Befehl „identify“ ist, wie der Name schon verrät, für die Bestimmung der Eigenschaften der angegebenen Bilddatei notwendig. Hierfür soll das Beispiel 5.10 diesen Befehl mit ausgewählten Bildeigenschaften verdeutlichen.Das Beispiel 5.10 bestimmt einige Eigenschaften der Bilddatei „BeispielBild.jpg“. Mit Hilfe des „format“-Parameters kann die Ausgabe der Informationen nach eigenen Bedürfnissen angepasst werden.

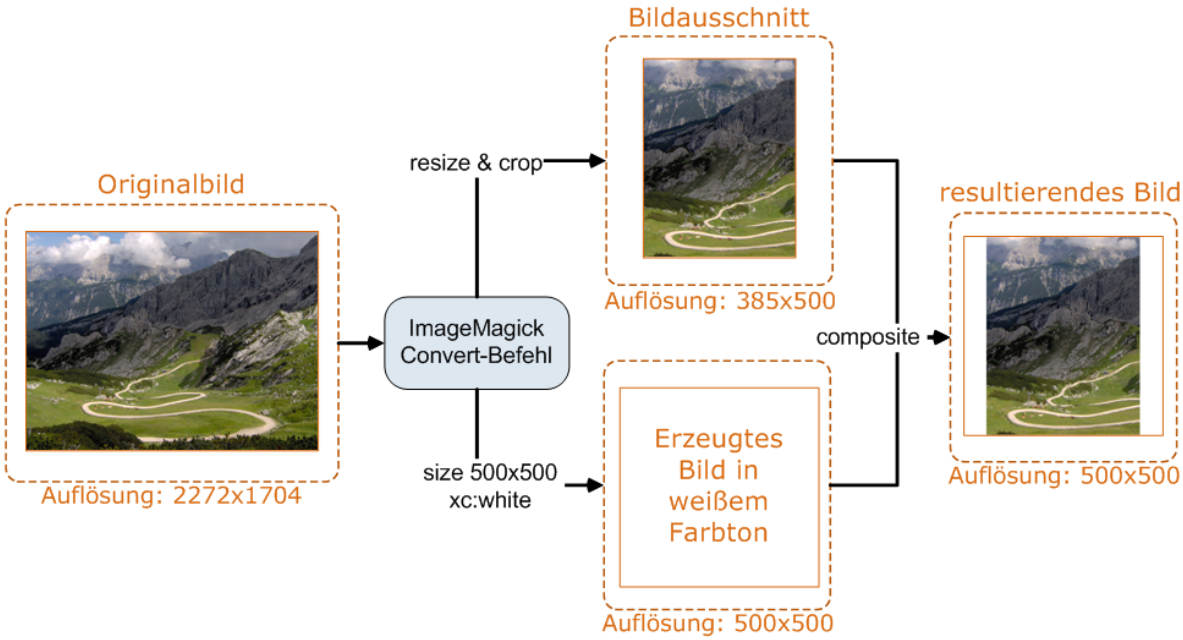

Abbildung 5.11: *ImageMagick „convert"-Befehl mit „crop"-Parameter*

```
identify -format "Dateiname=%t, Dateierweiterung=%e, \
        Dateigröße=%b, Breite=%w, Höhe=%h, \
        Farbraum=%[colorspace]" BeispielBild.jpg
```

Beispiel 5.10: *ImageMagick identify-Befehl*

In diesem Beispiel wurden die einzelnen Bildeigenschaften paarweise angeordnet und jeweils durch ein Komma getrennt. Anhand der Beschriftungen ist demnach gut zu erkennen, welches Zeichenpaar (Prozentzeichen + Buchstabe) die jeweilige Eigenschaft ausgibt, so z.B. steht „%t" für den Dateinamen ohne Dateierweiterung. Ebenso existieren eine Reihe von Bildeigenschaften die durch Angabe in Klammern ermittelt werden, wie bei der Bildeigenschaft „colorspace" zu erkennen ist. Weitere Informationen zu den beschriebenen Befehlen oder dem gesamten Software-Paket sind auf der ImageMagick-Webseite[7] zu finden.

Nachdem nun die beiden notwendigen Befehle für die weiteren Verarbeitungsschritte näher erläutert wurden, kann nun auf die möglichen Schnittstellen für ImageMagick in Java eingegangen werden.

Java-Schnittstellen für ImageMagick

Um den gesamten Funktionsumfang von ImageMagick in Java nutzen und somit komplexe Bildmanipulationen bzw. -verarbeitungen durchführen zu können, ist die Verwendung von Java-Schnittstellen zur API von ImageMagick notwendig. Dabei stehen zwei mögliche Schnittstellen, zum einen JMagick und zum anderen Im4java[6], zur Verfügung.

[6] Im4java - http://im4java.sourceforge.net/

Die Schnittstelle JMagick ist ein Interface welches über das Java Native Interface (JNI) mit der ImageMagick API interagiert. Java Native Interface ist standardisiert und stellt eine API für die Programmiersprache Java zur Verfügung, um plattformspezifischen Programmcode ausführen zu können. In diesem Fall ermöglicht JMagick als Schnittstelle aus der Programmiersprache Java heraus den Zugriff auf die in der Programmiersprache C geschriebenen Objekte und Funktionen. Allerdings wirkt sich die Verwendung von JNI für den Zugriff auf plattformspezifische Programmbibliotheken negativ auf die Plattformunabhängigkeit der jeweiligen Anwendung aus. Weiterhin ist zu JMagick zu sagen, dass lediglich ein Teil des ImageMagick Funktionsumfangs in dieses Interface integriert wurde. Weitere Informationen zu dieser Schnittstelle können auf der JMagick-Webseite[33] eingeholt werden.

Im Gegensatz dazu ist Im4java eine reine Java-Schnittstelle, die auf „Kommandozeilenebene“ mit der ImageMagick API arbeitet. Das bedeutet im Detail, das diese Schnittstelle anhand der gesetzten Operationen den vollständigen ImageMagick-Befehl erzeugt und ihn mit Hilfe des ProcessBuilders aus dem „java.lang“-Paket ausführt. Im4java beinhaltet die meisten Funktionen der ImageMagick Software, wobei außerdem die Unterstützung weiterer Tools wie GraphicsMagick[7], exiftool, dcraw und noch einige mehr gegeben ist. Weiterhin vorteilhaft sind die Ähnlichkeiten der Methodennamen der Schnittstelle in Bezug auf die Funktionen von ImageMagick, wodurch dem Entwickler der Umgang mit dem Interface erleichtert wird.

Ausgehend vom Einsatzumfeld und Verwendungszweck, wie im Abschnitt 5.1 beschrieben wurde, wird die Entscheidung auf die Java-Schnittstelle Im4java fallen. Dies soll damit begründet werden, das eine stabile und dennoch leistungsfähige Schnittstelle für die Verarbeitung von Bilddaten auf dem Verlagsserver benötigt wird. Aufgrund der umfangreichen Möglichkeiten des Interfaces ist eine höhere Kompatibilität zu unterschiedlichen Versionen der ImageMagick Software gegeben. Weiterhin wird dem Entwickler die Wahl der Grafikanwendung zum jeweiligen Zeitpunkt (ImageMagick, GraphicsMagick etc) freigestellt. Da die Entscheidung nach dem zu nutzenden Java-Interface an dieser Stelle getroffen wurde, soll nachfolgend auf die detaillierte Implementierung der Bildverarbeitung in Java mit Hilfe von Im4java eingegangen werden.

Bildverarbeitung mit Im4java und der ImageMagick API

Die Schnittstelle Im4java stellt verschiedene Klassen für die jeweiligen unterschiedlichen ImageMagick Befehle zur Verfügung. Bezogen auf die beiden im Abschnitt 5.2.7 beschriebenen Utilities, stellt Im4java die Klassen „ConvertCmd“ und „IdentifyCmd“ bereit. Wie außerdem schon im vorherigen Abschnitt angesprochen wurde, bietet dieses Interface ebenso Möglichkeiten anderweitige Software als ImageMagick nutzen zu können. Die Entscheidung des jeweiligen zu nutzenden Programms erfolgt zum Zeitpunkt der Erstellung des jeweiligen Kommando-Objektes. Diese jeweiligen Befehle benötigen zur Ausführung („run“-Methode) mindestens einen zusätzlichen Parameter, das „Operation“ Objekt. Diesem Objekt werden die im Ab-

[7] GraphicsMagick - http://www.graphicsmagick.org/

schnitt 5.2.7 erläuterten Parameter mit entsprechenden Werten hinzugefügt. Sollte der Fall eintreten, dass ein oder mehrere bestimme Parameter, die für die Bearbeitung der Bilddatei benötigt werden, nicht bzw. noch nicht in dieser Schnittstelle implementiert wurden, können mit Hilfe der Methode „addRawArgs“ diese entsprechenden Parameter mit den zugehörigen Werten an das „Operation“ Objekt übergeben und angewendet werden. Im Quellcode-Beispiel 5.11 soll die allgemeine Vorgehensweise anhand des „convert“ Befehls vom Beispiel 5.9 veranschaulicht werden.

```
String originalFile = "originalBild.png";
String modFile = "modifiziertesBild.jpg";

IMOperation imageOperation = new IMOperation();
imageOperation.addImage(originalFile);
imageOperation.crop(1000, 1300, 300, 200);
imageOperation.p_repage();
imageOperation.resize(500, 500);
imageOperation.density(72);
imageOperation.gravity("center");
imageOperation.size(500, 500);
imageOperation.addRawArgs("xc:white");
imageOperation.p_swap();
imageOperation.composite();
imageOperation.addImage(modFile);

ConvertCmd convertCommand = new ConvertCmd();
convertCommand.setAsyncMode(false);
convertCommand.run(imageOperation);
```

Beispiel 5.11: *Beispiel für Vorgehensweise zur Konvertierung mit Im4java*

Zusätzlich ist zu dem Beispiel 5.11 zu sagen, dass zur vereinfachten Darstellung auf die zugehörigen Ausnahmebehandlungen (Exception-Handling) verzichtet wurde. Wie zuvor in der Beschreibung dieser Schnittstelle erwähnt, ist nun das Merkmal der ähnlichen Methodennamen gut zu erkennen, wodurch die Umsetzung von dem reinen ImageMagick-Befehl hin zu dem Java-Code sichtlich erleichtert wird. Des Weiteren kann die Möglichkeit der asynchronen Verarbeitung („setAsyncMode“) genutzt werden. Schließlich wird dem „ConvertCmd“ Objekt das erstellte „IMOperation“ Objekt in der „run“-Methode übergeben. Anschließend folgt die Ausführung des gesamten Befehls durch den „ProcessBuilder“ und dem damit verbunden Aufruf der externen ImageMagick Software. Tritt bei der Verarbeitung ein Fehler auf, so wird automatisch eine Exception (Ausnahme) mit der entsprechenden Fehlernachricht geworfen. Soll nun statt ImageMagick die Software GraphicsMagick zum Einsatz kommen, ist es le-

diglich erforderlich beim Erzeugen des „ConvertCmd“ Objektes auf den zweiten Konstruktor zurückzugreifen und den Booleschen Wert „true“ zu übergeben. Durch diesen Parameter wird die Nutzung der Software GraphicsMagick festgelegt.

Ebenso wie für den „Convert“-Befehl soll das Beispiel 5.12 eine Vorgehensweise für die Verwendung des „Identify“-Befehls, angelehnt an das Beispiel 5.10, veranschaulichen. In diesem Fall wird ein „IdentifyCmd“ Objekt erzeugt und mit den Einstellungen des „IMOperation“ Objekts ausgeführt. Eine Besonderheit in diesem Fall ist das „ArrayListOutputConsumer“ Objekt, mit dessen Hilfe die Informationen dem „IdentifyCmd“ Objekt entlockt werden können.

```
String file = "BeispielBild.png";
IMOperation infoOperation = new IMOperation();
infoOperation.format(
        Dateiname=%t, Dateierweiterung=%e, \
        Dateigröße=%b, Breite=%w, Höhe=%h, \
        Farbraum=%[colorspace]);
infoOperation.addImage(file);
IdentifyCmd imageInfo = new IdentifyCmd();
ArrayListOutputConsumer aoc = new ArrayListOutputConsumer();
imageInfo.setOutputConsumer(aoc);
imageInfo.setAsyncMode(false);
imageInfo.run(infoOperation);
ArrayList<String> outputList = aoc.getOutput();
```

Beispiel 5.12: *Beispiel für Vorgehensweise zur Bestimmung der Informationen mit Im4java*

In ähnlicher Art und Weise, wie es in den Beispielen 5.11 und 5.12 zu sehen war, wurde die Schnittstelle Im4java in den Prototypen implementiert. Die nachfolgenden Betrachtungen beziehen sich auf den genauen Ablauf ausgehend vom Zeitpunkt des Bild-Uploads bis hin zu den fertigen Datenbankeinträgen sowie den Realisierungen der Modifikationen.

Ablauf der Bildverarbeitung im Prototyp

Dieser Abschnitt knüpft an das im Abschnitt 5.2.6 beschriebene Uploadverfahren an, d.h. die Bilddatei wurde erfolgreich hochgeladen und der Nutzer über den erfolgreichen Upload informiert. Die Java-Klasse „AddMediaUploadImage“ hat demzufolge die Daten entgegen genommen und leitet an dieser Stelle die weiteren Schritte für die Verarbeitung des Originalbildes ein. Der Ablauf der nachfolgenden Aufgaben soll mit Hilfe der Abbildung 5.12 unterstützend veranschaulicht werden.

Wie in der Abbildung 5.12 vereinfacht dargestellt wird, ist die Java-Klasse „AddMediaUploadImage“ Ausgangspunkt der Bildverarbeitung nach dem Upload. Die Ausführungen der gesamten Verarbeitung der Bilddaten finden in der „MaxityMediaImageProcessing“-Klasse

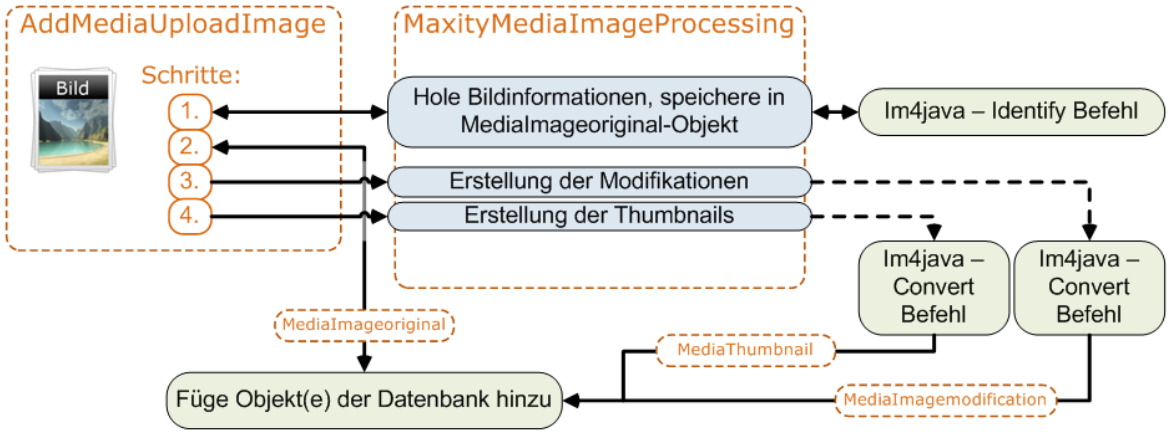

Abbildung 5.12: *Vereinfachte Darstellung des Ablaufs der Bildverarbeitung nach dem Upload*

statt, d.h. unabhängig von dem Ausgangspunkt werden in dieser Java-Klasse alle Informationen für die jeweiligen Aufgaben zusammengetragen und einzig und allein in dieser finden die jeweiligen Aufrufe der ImageMagick-Tools über die Schnittstelle Im4java statt.

Nachdem der Upload erfolgreich ausgeführt wurde, tritt der erste Schritt in Kraft. Dabei führt die Klasse „MaxityMediaImageProcessing“ die Methode „readImageInformation“ aus, welche in der Abbildung sinnbildlich mit „Hole Bildinformation, speichere in MediaImageoriginal-Objekt“ bezeichnet wurde. In dieser Methode werden die benötigten Bildeigenschaften mit Hilfe des ImageMagick Identify-Befehls zusammengetragen, ähnlich wie in dem Beispiel 5.12 veranschaulicht wurde, und in ein neues „MediaImageoriginal“-Objekt gespeichert. Anschließend wird im zweiten Schritt dieses Objekt auf Existenz geprüft und mit Hilfe der „MaxityMediaImageoriginals“ Java-Klasse, die für die Datenbankaktionen der „MediaImageoriginal“ Objekte zuständig ist, in der Datenbank gespeichert. In den Schritten drei und vier werden jeweils Modifikationen und Thumbnails von dem Originalbild erstellt. Dazu werden jeweils über die Schnittstelle Convert-Befehle, welche in ähnlicher Form, wie in dem Beispiel 5.11 veranschaulicht wurde, aufgebaut sind, mit Hilfe der ImageMagick-Software ausgeführt. Allerdings erfolgt die Ausführung dieser Befehle nicht unmittelbar nach dem Aufruf der Funktionen in der „MaxityMediaImageProcessing“-Klasse. In der Abbildung 5.12 ist die Verbindung zwischen den „Ersteller“-Funktionen und den jeweiligen Im4java-Convert-Befehlen gestrichelt dargestellt. Dies soll auf Zwischenschritte hinweisen, die im Abschnitt 5.2.9 näher erläutert werden, wobei in diesem Fall reine Verwaltungsschritte des Prototyps gemeint sind, welche keinen Einfluss hinsichtlich der Art und Weise auf die eigentliche Bildverarbeitung haben. Zurück zu den Convert-Befehlen, denen nach ihrem Abschluss die Existenzprüfungen der jeweiligen Dateien folgen. Nach erfolgreichem Ergebnis können die jeweiligen Objekte erstellt und abschließend in die Datenbank geschrieben werden.

Ausgangspunkt für jede einzelne Konvertierung ist der jeweilige zugehörige Verwendungszweck, welcher durch den Administrator in der Datenbank gepflegt wird. Diese jeweiligen Verwendungszwecke sind in den, wie in der Abbildung 5.2 zu sehen, drei unteren Tabellen im

roten Farbton („media_imageusage“, „media_videousage“ und „media_thumbnailsetting“) festgelegt. Weiterhin ist zu sagen, dass die einzelnen Konvertierungen nacheinander ablaufen und nicht parallel stattfinden.

Nachdem nun die einzelnen Verarbeitungsschritte für Bilddateien erläutert wurden, kann nun auf die der Videodaten näher eingegangen werden.

5.2.8 Videoverarbeitung mittels Xuggler API

In diesem Abschnitt wird im Detail auf die Verarbeitung von Videos mit der Software FFmpeg in Verbindung mit der Programmiersprache Java eingegangen. Dabei wurden im Kapitel 4 die Möglichkeiten der Videoverabreitung in Java aufgezeigt und beschrieben. Die Entscheidung fiel in diesem Zusammenhang auf die Bibliothek Xuggler, welche als Schnittstelle zwischen der Programmiersprache Java und dem Videobearbeitungsprogramm FFmpeg dient. Nachfolgend sind die notwendigen Kriterien, die durch den Prototyp erfüllt werden müssen, nochmals aufgeführt.

- Änderungsmöglichkeiten der Video-Eigenschaften (Breite, Höhe, Bildwiederholfrequenz)
- Änderung an der Audio-Eigenschaften Abtastfrequenz und Anzahl der Audio-Kanäle
- Einstellungsmöglichkeiten an den Audio- und Videobitraten
- Modifikation des Audio- und Videocodec

Erfüllung der Kriterien mit FFmpeg

Anhand der Zuhilfenahme der reinen FFmpeg-Software soll auf die Erfüllung der aufgestellten Pflichtkriterien eingegangen werden. Das nachfolgende Beispiel 5.13 zeigt die Verarbeitung einer Videodatei, welche mit FFmpeg unter Verwendung einiger Parameter zur Konfiguration der Audio- und Videoeinstellungen bearbeitet wird. Dabei wird die originale Videodatei „originalVideo.avi“ in die „modVideo.flv“ Datei umgewandelt, wobei die Originaldatei erhalten bleibt.

```
ffmpeg -i originalVideo.avi \
    -acodec libmp3lame -ar 44100 -ab 128000 -ac 2 \
    -vcodec flv -b 768000 -r 30 -s 480x360 \
    modVideo.flv
```

Beispiel 5.13: *Videoverarbeitung mit FFmpeg*

Der FFmpeg-Befehl aus dem Beispiel 5.13 enthält neben der durch den Parameter „i“ gekennzeichneten Eingabedatei (Inputdatei) ebenso zahlreiche weitere Parameter für die Modifikation des Videos. Angefangen mit dem Setzen der Zieleinstellungen für die Audiospur

und somit dem ersten Parameter in der zweiten Zeile „acodec“, womit der Audiocodec auf „libmp3lame“ festgelegt wird. Daraufhin folgen die einzelnen Audioeinstellungen mit der Abtastrate durch den Parameter „ar“ und dem Wert 44100 in Hz, der Audiobitrate „ab“ mit 128000 bit/s und durch den Parameter „ac“ mit dem Wert 2, der für die festgelegten Audio-Kanäle steht. Den Audio-Einstellungen folgen in der dritten Zeile die Konfigurationen für die Videospur. Ebenfalls wird in diesem Fall der Videocodec „vcodec“ mit dem Wert „flv“, welcher auf das Flash-Video Format zielt, festgelegt. Mit Hilfe des Parameters „b“ findet die Konfiguration der Videobitrate mit dem Wert 768000 bit/s statt. Weiterhin folgt mit „r“ die Bildfrequenz (Framerate) mit dem Wert 30 Hz und zu guter Letzt wird die Ausgangsauflösung (Breite und Höhe) der resultierenden Videodatei durch den Parameter „s“ mit der Breite „480“ Pixel und der Höhe „360“ Pixel festgelegt. Diese Verarbeitungsschritte sollen in einer zusätzlichen Abbildung 5.13 noch einmal vereinfacht veranschaulicht werden.

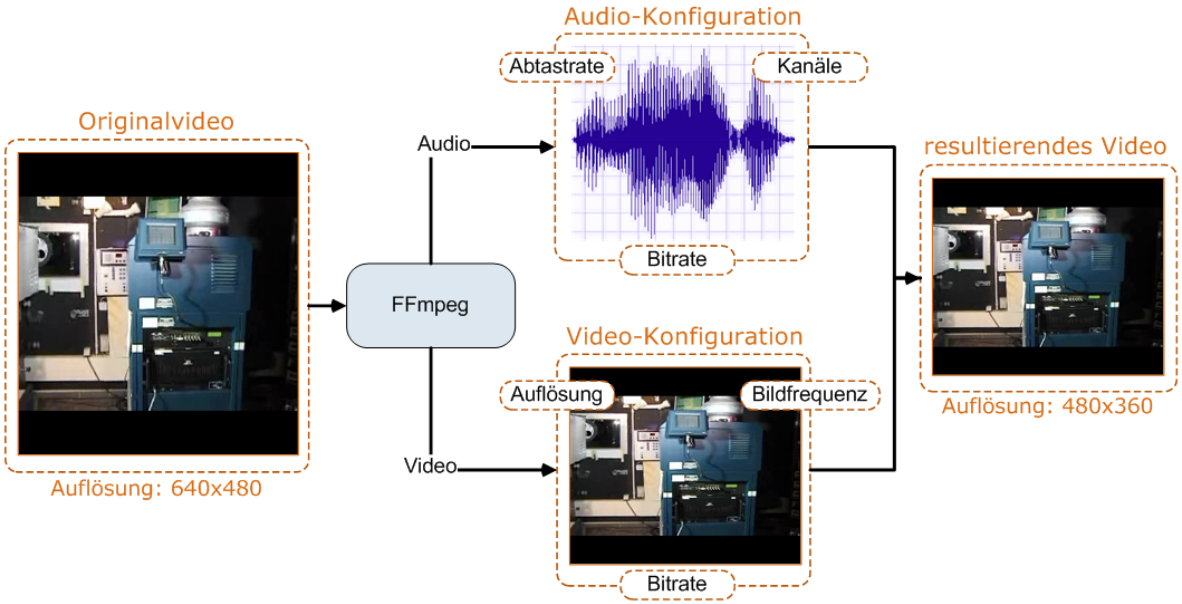

Abbildung 5.13: *Ablauf der FFmpeg-Anweisung*

In der Grafik 5.13 ist die Trennung der einzelnen Verarbeitungsschritte zwischen Audio und Video hervorgehoben. Des Weiteren sollen die im Beispiel 5.13 beschriebenen Parameter mit Hilfe der an den jeweiligen Konfigurationen angebrachten Beschriftungen diesen zugeordnet und vereinfacht dargestellt werden. Sinnbildlich betrachtet können demnach die einzelnen Befehlszeilen aus dem Beispiel 5.13 den vier großen Bereichen in dieser Abbildung, wie zu erkennen ist, zugeordnet werden. Wie zuvor schon beschrieben, wird mit der in diesem Beispiel letzten Angabe die resultierende Videodatei bestimmt. Zusätzlich ist zu sagen, dass die Angabe des Audio- und Videocodecs, wenn nicht unbedingt notwendig, nicht angegeben werden muss. Mit Hilfe der Dateierweiterung der Zieldatei „flv“, welche das resultierende Containerformat vorgibt, entscheidet FFmpeg selbst welcher Audio- bzw. Videocodec zu wählen ist und integriert den jeweiligen automatisch.

Ausgehend von dem eben beschriebenen Beispiel sowie der zugehörigen Abbildung in Bezug auf die aufgestellten Pflichtkriterien, können bis zu diesem Zeitpunkt all diese definierten Kriterien mit dem reinen FFmpeg-Befehl auf dieser Ebene der Ausführung erfüllt werden. Nachdem nun auf unterster Ebene die Durchführung unter Berücksichtigung der Kriterien abgeschlossen wurde, kann nun auf die Java Bibliothek Xuggler eingegangen werden.

Die Java Bibliothek Xuggler

Um die in diesem Abschnitt am Anfang definierten Kriterien mit Hilfe der Java Bibliothek Xuggler zu erfüllen, ist es notwendig diese ausführlicher zu beleuchten.

Wie schon erwähnt nutzt Xuggler die in der Programmiersprache C geschriebene Bibliothek FFmpeg. Der in diesem Interface integrierte Funktionsumfang in Bezug auf FFmpeg selbst wird von den Entwicklern von Xuggler auf über 90% angegeben, d.h. das fast alle Funktionen bzw. Möglichkeiten aus Java heraus genutzt werden können. Bei der Verwendung von Xuggler können Objekte wie bisher üblich in Java erstellt und verwendet werden, ebenso unterliegt das Speichermanagement der Java Virtual Machine, d.h. das sich nach Abschluss der Verwendung dieser Objekte der Garbage Collector um die Freigabe dieser Objekte aus dem Speicher bemüht.

Die Kommunikation zwischen den Java-Objekten und der FFmpeg Bibliothek erfolgt über Xuggler als Schnittstelle. Diese selbst nutzt für den Zugriff auf die native Bibliothek das Java Native Interface (JNI). Dafür verwendet Xuggler die eigenen in der Programmiersprache C++ geschriebenen Klassen, die ebenso als Wrapper bzw. Hülle um die Bibliothek FFmpeg fungieren.

Bemerkenswert und in jedem Falle erwähnenswert sind die Bemühungen hinsichtlich der Bereitstellung von Tutorials durch die Entwickler selbst, um dem Anwender die Bibliothek Xuggler näher zu bringen. In dieser Hinsicht setzen die Programmierer auf Lernvideos, wobei ebenso anderweitige Unterstützungen nicht zu kurz kommen. Auf den jeweiligen Webseiten, dem Xuggle Blog[8], dem Xuggle Wiki[9] sowie dem Forum[10] sind Hilfen, Hinweise und weitere Anmerkungen zu finden, welche die Arbeit mit diesem Interface erleichtern. Nachfolgend soll auf die eigentliche Umsetzung in Java eingegangen werden.

Xuggler bietet mehrere Möglichkeiten sich unterschiedlich der Videoverarbeitung in Java zu nähern. Im einfachsten Fall kann die „Converter“ Klasse von Xuggler für simple Konvertierungen sowie Änderungen einiger Eigenschaften genutzt werden, wobei diese ebenso als Beispielklasse für die Verwendung der Xuggler-Bibliothek dienen soll. Weiterhin beinhaltet Xuggler eine vereinfachte API mit dem Namen „MediaTool“, mit deren Hilfe es in wenigen Schritten möglich ist, Videodaten zu manipulieren und gegenenfalls benutzerdefinierte Kriterien mit einzubringen. Abschließend gibt es die Möglichkeit die „Xuggler Advanced API“ zu nutzen, welche ebenso von der „MediaTools API“ aufgerufen wird, um umfangreiche Zugrif-

[8] Xuggle Blog - http://blog.xuggle.com/
[9] Xuggle Wiki - http://wiki.xuggle.com/
[10] Forum - http://groups.google.com/group/xuggler-users

fe und Konfigurationsmöglichkeiten realisieren zu können. Zu Beginn soll die Funktionsweise anhand der „MediaTool API“ demonstriert werden, wobei dieser Beschreibung die Darstellung eines Beispiels für die „Xuggler Advanced API“, wie sie im Prototyp verwendet wird, folgt.

```
String originalFile = "OriginalVideo.avi";
String modFile = "ModifiziertesVideo.flv";
IMediaReader reader =
        ToolFactory.makeReader(originalFile);
IMediaWriter writer =
        ToolFactory.makeWriter(modFile, reader);
reader.addListener(writer);
while(reader.readPacket() == null);
```

Beispiel 5.14: *Videoverarbeitung mit Xuggler und der „MediaTool API“ Beispiel 1*

In diesem ersten „MediaTool“ Beispiel 5.14 soll die vereinfachte Möglichkeit dieser API, welche im Hintergrund auf die Advanced API zugreift, veranschaulicht werden. Mit Hilfe der „ToolFactory“ werden „IMediaReader“ und „IMediaWriter“ unter Verwendung der jeweiligen Zieldatei erstellt. Bei der „makeWriter“ Funktion kommt hinzu, dass der „IMediaReader“ (Reader) als Ausgangs- bzw. Vorgabecontainer bereitgestellt wird. Als nächstes wird dem Reader lediglich der „IMediaWriter“ (Writer) als Empfänger hinzugefügt. Nun folgt das Lesen (Dekodieren) der einzelnen Pakete aus der Videodatei, wobei diese zeitnah dem Writer übergeben und wieder codiert werden. Die „MediaTool API“ folgt dabei, wie in dem Beispiel zu sehen ist, der ereignisorientierten Programierung, wodurch es möglich ist, benutzerdefinierte Ereignis-Listener zu schreiben und diese an der jeweiligen Stelle als Listener hinzuzufügen, wodurch eine Art „Ereignis-Kette“ entsteht. Dieses Prinzip wurde in dem Prototyp in der für die Videoverarbeitung relevanten Java-Klasse „MaxityMediaVideoProcessing“ integriert. Dabei war es notwendig weitere Java-Klassen für die benuterdefinierten Konfigurationen zu erstellen, wobei diese die Klasse „MediaToolAdapter“ aus dem Package „com.xuggle.mediatool“ ableiten müssen. In dem Prototyp befinden sich zwei Klassen, die diesen „MediaToolAdapter“ ableiten. Das ist zum einen die „MaxityMediaVideoAdapter“ und zum die „MaxityMediaVideoConverter“ Klasse. Mit Hilfe dieser beiden Java-Klassen können, wie an dem nachfolgenden Beispiel zu sehen ist, die am Anfang dieses Abschnitts definierten Kriterien erfüllt werden. Die Beispiele 5.15, 5.16 und 5.17 zeigen die jeweiligen Klassen-Ausschnitte, welche für die Verarbeitung notwendig sind. Dabei deuten drei aufeinanderfolgende Punkte „...“ auf Kürzungen in der jeweiligen Situation hin.

```
public class MaxityMediaVideoProcessing {
    ...
    String originalFile = "OriginalVideo.avi";
    String modFile = "ModifiziertesVideo.flv";
```

```
    IMediaReader reader =
            ToolFactory.makeReader(originalFile);
    IMediaWriter writer =
            ToolFactory.makeWriter(modFile, reader);
    MaxityMediaVideoAdapter adapter =
        new MaxityMediaVideoAdapter(einigeParameter...);
    MaxityMediaVideoConverter converter =
        new MaxityMediaVideoConverter(einigeParameter...);

    reader.addListener(converter);
    converter.addListener(writer);
    writer.addListener(adapter);

    while(reader.readPacket() == null);
    ...
}
```

Beispiel 5.15: *Videoverarbeitung mit Xuggler und der „MediaTool API“ Beispiel 2*

Das Beispiel 5.15 baut auf den Quellcode 5.14 auf und wurde lediglich um zwei der benutzerdefinierten „MediaToolAdapter“ abgeleiteten Klassen ergänzt, die als Event-Handler dem jeweiligen Objekt hinzugefügt werden, was wiederum die Kette von Ereignissen sehr gut erkennen lässt. Die beiden Klassen „MaxityMediaVideoAdapter“ und „MaxityMediaVideoConverter“ erhalten jeweils einige Parameter, die für die weiteren Verarbeitungsschritte notwendig sind. In diesen beiden Fällen wurde aus Platzgründen auf die einzelnen Parameter verzichtet.

In der „MaxityMediaVideoAdapter“ Klasse wird der Ausgangscontainer, also der „IMediaWriter“ auf die entsprechenden Verhältnisse vorbereitet, d.h. das z.B. die Änderungen der Auflösung, Audio-Kanäle und Abtastfrequenz beim Erzeugen des Ausgangscontainers bekannt gegeben werden müssen, andernfalls würde es zum späteren Zeitpunkt beim Umwandeln der einzelnen Pakete zu Differenzen hinsichtlich dieser Größen kommen. Aus diesem Grund wird die Methode „onAddStream“ implementiert und überschrieben. Mit dem nachfolgenden Beispiel soll ein Ausschnitt der wichtigsten Funktion der „MaxityMediaVideoAdapter“ Klasse dargestellt werden.

```
@Override
public void onAddStream(IAddStreamEvent event) {
  ...
  IStreamCoder coder = event.getSource().getContainer()
    .getStream(event.getStreamIndex()).getStreamCoder();
  if(coder.getCodecType() == ICodec.Type.CODEC_TYPE_AUDIO) {
    ...
```

```
      coder.setSampleRate(audioSampleRate);
      coder.setChannels(audioChannel);
      coder.setBitRate(audioBitrate);
    }
    if(coder.getCodecType() == ICodec.Type.CODEC_TYPE_VIDEO) {
      ...
      coder.setWidth(width);
      coder.setHeight(height);
      coder.setBitRate(videoBitrate);
      coder.setFrameRate(IRational.make(frameRate));
    }
    ...
    super.onAddStream(event);
  }
```

Beispiel 5.16: *Videoverarbeitung mit Xuggler: Ausschnitt „MaxityMediaVideoAdapter“*

In dem Adapter-Beispiel 5.16 werden in der „onAddStream“ Methode die anstehenden Änderungen an den jeweiligen Streams (Audio oder Video) vorbereitet. Anhand dieser Eigenschaften entsteht ein Container für das „IMediaWriter“ Objekt, welches jeweils einen neuen Audio- und Videostream mit den aktuell gesetzten Attributen enthält. Diese Änderungen der Attribute werden lediglich an dem jeweiligen zu dem Zeitpunkt aktuellen „IStreamCoder“ Objekt durch einfaches Setzen neuer Werte vorgenommen. Damit ist auch schon das Ende der Funktionalität dieser Klasse erreicht, deren Zweck lediglich in der Erstellung und Anpassung des Containers besteht.

Im Gegensatz dazu ist Klasse „MaxityMediaVideoConverter“ für die weiteren Audio- und Video-Konfigurationen bzw. Verarbeitungen verantwortlich. Dazu werden die einzelnen Audio- und Video-Pakete mit den sogenannten jeweiligen „Resampler“ Objekten umgewandelt. Ebenso soll an dieser Stelle ein Ausschnitt der drei notwendigen Funktionen dargestellt werden.

```
  @Override
  public void onAddStream(IAddStreamEvent event) {
    IStreamCoder coder = event.getSource().getContainer()
      .getStream(event.getStreamIndex()).getStreamCoder();
    if(coder.getCodecType() == ICodec.Type.CODEC_TYPE_AUDIO) {
      ...
      coder.setSampleRate(audioSampleRate);
      coder.setChannels(audioChannel);
    }
    if(coder.getCodecType() == ICodec.Type.CODEC_TYPE_VIDEO) {
      ...
```

```
      coder.setWidth(width);
      coder.setHeight(height);
    }
    super.onAddStream(event);
  }
  @Override
  public void onAudioSamples(IAudioSamplesEvent event) {
    IAudioSamples samples = event.getAudioSamples();
    ...
    IAudioSamples out = IAudioSamples.make(
        samples.getNumSamples(),
        samples.getChannels());
    resamplerAudio.resample(
        out, samples, samples.getNumSamples());
    AudioSamplesEvent ase = new AudioSamplesEvent(
        event.getSource(),
        out, event.getStreamIndex());
    super.onAudioSamples(ase);
    ...
  }
  @Override
  public void onVideoPicture(IVideoPictureEvent event) {
    IVideoPicture video = event.getPicture();
    ...
    IVideoPicture out = IVideoPicture.make(
        resamplerVideo.getInputPixelFormat(),
        resamplerVideo.getOutputWidth(),
        resamplerVideo.getOutputHeight());
    resamplerVideo.resample(out, video);
    VideoPictureEvent vpe = new VideoPictureEvent(
        event.getSource(), out, event.getStreamIndex());
    super.onVideoPicture(vpe);
    ...
  }
```

Beispiel 5.17: *Videoverarbeitung mit Xuggler: Ausschnitt „MaxityMediaVideoConverter“*

Wie in dem Beispiel 5.17 zu sehen ist, werden drei Methoden aus der abgeleiteten Klasse implementiert und überschrieben. Diese Methoden kümmern sich um die korrekte Anpassung der Audio- und Video-Eigenschaften, also zum Beispiel um Abtastfrequenz, Anzahl der Audio-

Kanäle sowie die Auflösung (Breite und Höhe) der Videodatei. In der „onAddStream“ Methode werden diese Eigenschaften beim Hinzufügen des Videostreams gesetzt. Die darauffolgende Methode „onAudioSamples“ ist für die Audio-Verarbeitung bzw. Anpassung erforderlich, mit deren Hilfe die einzelnen Audiopakete verändert werden können, wobei diese Veränderung mittels „IAudioResampler“ Objekts, welches jedes Audio-Sample umwandelt, vorgenommen und abschließend der Oberklasse als neues „AudioSamplesEvent“ übergeben werden. Dabei ist zu sagen, dass der momentane Stand der Bibliothek Xuggler maximal zwei Audio-Kanäle und eine maximale Abtastrate von 44100 Hz zulässt. Diese Einschränkungen sind allerdings aktuell in keinster Weise ein Hindernis für die Erfüllung der Kriterien des Prototyps. Die Methode „onVideoPicture“ wird immer dann aufgerufen, wenn es sich bei „reader.readPacket()“ (siehe Beispiel 5.15) um ein Videopaket handelt. Außerdem erfolgt die Umwandlung des jeweiligen Videobildes („IVideoPicture“) mit Hilfe eines „IVideoResampler“ Objekts, wobei im Anschluss darauf ein „VideoPictureEvent“ Objekt erstellt und der Oberklasse für die weiteren Schritte übergeben wird.

Da der Ablauf der Videoumwandlung mit Hilfe der „MediaTool API“ anhand von Quell-Code Beispielen näher beschrieben wurde, ist es nun notwendig mit dem Zugriff auf die „Xuggler Advanced API“ auf die Informationen von Videodaten zuzugreifen.

Voraussetzung für die Erstellung der Modifikationen und den Thumbnails ist das erfolgreiche Speichern der Originaldatei im Dateisystem sowie als Datensatz in der Datenbank mit den erforderlichen Informationen, welche direkt nach dem Speichervorgang im Dateisystem von der originalen Videodatei gelesen werden. Bei den Informationen handelt es sich um audio- und videospezifische Kenngrößen wie Abtastrate, Bildfrequenz und Bitraten, um nur einige zu nennen. Um diese Informationen aus der Videodatei lesen zu können, wird in diesem Prototyp mit Hilfe der „MediaTool API“ auf die „Xuggler Advanced API“ zurückgegriffen. Diese jeweiligen Verarbeitungsschritte wurden in der „MaxityMediaVideoProcessing“ Klasse implementiert und sollen im nachfolgenden Beispiel 5.18 ausschnittsweise veranschaulicht werden.

```
...
String file = "OriginalVideo.avi";
IMediaReader reader = ToolFactory.makeReader(file);
if(!reader.isOpen())
  reader.open();
...
IContainer container = reader.getContainer();
container.getFileSize();
container.getContainerFormat().getInputFormatShortName();
container.getDuration();
...
int nos = container.getNumStreams();
```

```
for(int i=0; i<nos; i++) {
  IStream stream = container.getStream(i);
  IStreamCoder coder = stream.getStreamCoder();
  if(coder.getCodecType() == ICodec.Type.CODEC_TYPE_AUDIO) {
    audioBitrate = coder.getBitRate();
    audioSampleRate = coder.getSampleRate();
  }
  if(coder.getCodecType() == ICodec.Type.CODEC_TYPE_VIDEO) {
    width = coder.getWidth();
    height = coder.getHeight();
    videoBitrate = coder.getBitRate();
  }
}
...
MediaVideooriginal videoOriginal =
  new MediaVideooriginal(Parameter...);
```

Beispiel 5.18: *Videoinformationen mit Xuggler Advanced API*

In diesem Beispiel 5.18 wird ebenfalls ein „IMediaReader“ Objekt erstellt und der enthaltene Container geöffnet. Anhand des Containers können die ersten Informationen wie Dateigröße, das Format des Containers sowie die Laufzeit des Videos abgefragt werden. Anschließend werden die jeweiligen in der Videodatei enthaltenen Streams einzeln durchlaufen und deren Kenngrößen gesichert. Diese Kenngrößen sind in diesem Beispiel für den Audiostream die Bitrate und Abtastrate und für den Videostream ebenso die Bitrate und Auflösung des Videos (Breite und Höhe). Abschließend erfolgt die Erstellung eines neuen „MediaVideoriginal“ Objekts mit den gesamten zum Teil hier dargestellten Informationen. Das erstellte Objekt kann anschließend in die Datenbank geschrieben werden. In diesem Abschnitt wurden die für die Verarbeitung bzw. Manipulation von Videodateien notwendigen Kriterien sowie Realisierungen aufgezeigt und erläutert. Weiterhin konnten alle aufgestellten Pflichtkriterien mit Hilfe der Xuggler Bibliothek erfüllt werden. Aus diesem Grund kann nun im folgenden Abschnitt auf den Ablauf der Videoverarbeitung im Prototyp eingegangen werden.

Ablauf der Videoverarbeitung im Prototyp

Dieser Abschnitt knüpft ebenso an das Uploadverfahren an, welches im Abschnitt 5.2.6 beschrieben wurde. Die Java-Klasse „AddMediaUploadVideo“ hat demzufolge die Daten entgegen genommen und leitet an dieser Stelle die weiteren Schritte für die Verarbeitung des Originalvideos ein. Der Ablauf der nachfolgenden Aufgaben soll mit Hilfe der Abbildung 5.14 unterstützend veranschaulicht werden.

Wie auch die Bildverarbeitung ist ebenso die Videoverarbeitung in vier Verarbeitungsschrit-

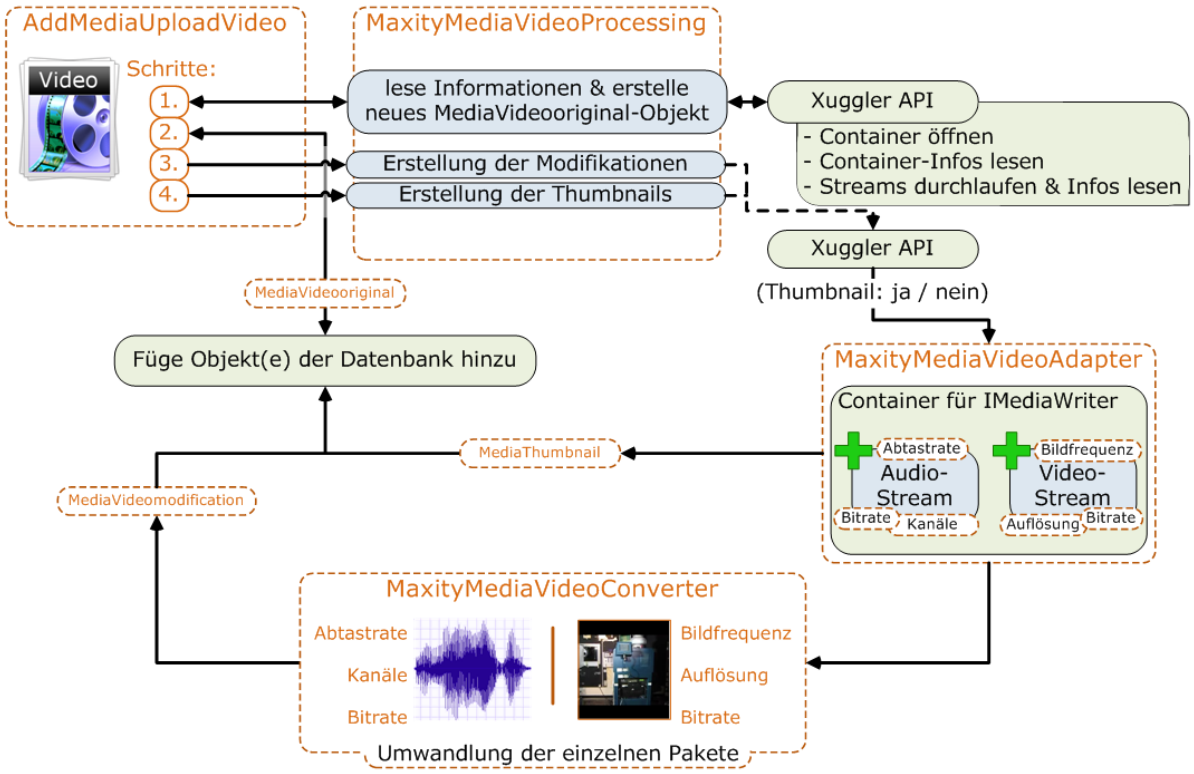

Abbildung 5.14: *Vereinfachte Darstellung des Ablaufs der Videoverarbeitung nach dem Upload*

te gegliedert. Dabei werden anfangs nach erfolgreichem Upload die Informationen des Videos mit Hilfe der Xuggler Bibliothek gelesen und in einem neuen „MediaVideooriginal“ Objekt gesichert. Vorab erfolgt die Öffnung des Videocontainers, in welchem allgemeine Informationen, wie z.B. die Anzahl der Streams, die Dauer des Videos oder formatspezifische Merkmale, enthalten sind. Anschließend liegt der Schwerpunkt auf den streamspezifischen Eigenschaften, woraufhin die einzelnen Streams durchlaufen und nach bestimmten Informationen durchsucht werden. Diese einzelnen Abläufe charakterisieren den in der Abbildung dargestellten ersten Schritt. Der zweite Schritt soll lediglich das Speichern des erstellten „MediaVideooriginal“ Objekts in die Datenbank darstellen. Nun folgen die Schritte drei und vier, wobei diese in diesem Fall, anders als in der Bildverarbeitung, parallel ablaufen. Für jeden Datensatz in der „media_videousage“ Tabelle (siehe Abbildung 5.2) muss die originale Videodatei durchlaufen und konvertiert werden. Da der Zugriff auf den Videostream während eines Durchlaufs ohnehin notwendig ist, können aus diesem Grund während dieser Zeit die jeweiligen Thumbnails sowie die zugehörigen „MediaThumbnail“ Objekte erstellt werden, wobei die jeweiligen Thumbnail-Objekte zeitnah in der Datenbank gespeichert werden. Wie in der Abbildung zu sehen ist, existiert ein Entscheider „Thumbnail: ja / nein“ mit deren Hilfe die Thumbnailerstellung aktiviert bzw. deaktiviert wird. Wie eben erwähnt wird das Originalvideo abhängig von der Anzahl der Datensätze in der „media_videousage“ Tabelle durchlaufen, um die jeweiligen speziellen Modifikationen zu erstellen. Dazu findet in der „MaxityMediaVideoAdapter“ Klasse die An-

passung der neu hinzugefügten Streams für die Erstellung des Containers statt. In der Grafik wurde diese Anpassung, das Hinzufügen und Adaptieren der jeweiligen Streams, zu Darstellungszwecken mit Hilfe eines grünen Kreuzes und den dazugehörigen Schlagwörtern abgebildet. Der nächste Schritt ist in der Klasse „MaxityMediaVideoConverter“ die Durchführung der Änderungen an den jeweiligen Streams. Dazu werden die audio- und videospezifischen Merkmale, wie in der Abbildung zu sehen ist, mit den jeweiligen originalen Eigenschaften verglichen und gegenenfalls verändert. Nach dem Abschluss eines jeden Durchlaufs werden die jeweiligen neu erstellten „MediaVideomodification“ Objekte in die Datenbank geschrieben.

Abschließend soll die Aufmerksamkeit auf die Verbindung zwischen den beiden „Ersteller“ Methoden in der „MaxityMediaVideoProcessing“ Klasse und der „Xuggler API“ gelenkt werden. Diese Verbindung ist in der Abbildung durch eine gestrichelte Linie dargestellt und soll, wie schon im Abschnitt 5.2.7 der Bildverarbeitung, auf Zwischenschritte hinweisen, welche in dem nachfolgenen Teil 5.2.9 näher erläutert werden.

In diesem Abschnitt wurde detailliert auf die einzelnen notwendigen Verarbeitungsschritte für Videodaten mit Hilfe der Java Bibliothek Xuggler eingegangen. Dabei wurde hervorgehoben, dass sowohl die Art und Weise Informationen aus Videocontainern zu lesen, als auch die Manipulation von Videoeigenschaften bzw. Videostream-Merkmalen wesentliche Kriterien für die Entwicklung des Prototyps sind. Ebenso konnten die anfangs festgelegten Pflichtkriterien, die in jedem Fall vom Prototyp erfüllt werden müssen, vollständig in dem Prototyp implementiert und realisiert werden, wie an den Beispielen und Abbildungen unterstützend dargestellt wurde. In diesem Zusammenhang kann im nachfolgenden Abschnitt auf das Aufgabenmanagement eingegangen werden.

5.2.9 Aufgabenmanagement

Dieser Abschnitt beschäftigt sich mit der Verwaltung spezieller Aufgaben, die nachfolgend ausführlicher erläutert werden. Wie schon in den Abschnitten zuvor, wurde die Integration bzw. die Beteiligung der Aufgabenverwaltung in den jeweiligen Verarbeitungsschritten erwähnt.

Bei den einzelnen Verarbeitungen bzw. Manipulationen der jeweiligen Multimedia-Dateien ist unter Umständen über einen längeren Zeitraum mit höherem Performancebedarf der Anwendung zu rechnen. Die nachfolgende Tabelle 5.2 soll den Aufwand der jeweiligen in den vorangegangenen Beispielen beschriebenen Bild- sowie Videoverarbeitungen hinsichtlich der Zeitdauer durch den entsprechenden verantwortlichen Prozess darstellen. Dabei wurden die aufgeführten Beispiele an einem PC mit folgender Systemspezifikation durchgeführt und getestet.

- Prozessor: AMD Athlon 64 X2 Dual Core Processor 6000+
- Arbeitsseicher: 2 x 1 GB DDR2

Beispiel	Dauer (in Sekunden)
Verarbeitung eines Bildes mit Dateigröße ca. 1 MB, Auflösung 3466x2355	
Bild-Modifikation ähnlich dem Beispiel 5.11	ca. 2
Bild-Thumbnailerstellung	ca. 1,4
Verarbeitung eines Videos mit Dateigröße ca. 25 MB, Auflösung 320x240, Dauer 11 Min.	
Video-Modifikation ähnlich dem Beispiel 5.15	ca. 54
reine Video-Thumbnailerstellung, d.h. kein „IMediaWriter“	ca. 17

Tabelle 5.2: *Aufwandsübersicht der beschriebenen Beispiele*

Ausgehend von den Ergebnissen in der Tabelle 5.2 in Bezug auf die späteren Belastungen, ist die Verwaltung dieser ressourcenbelastenden Verarbeitungen von Multimedia-Daten ebenso sinnvoll wie auch notwendig. Dies macht ein einfaches Beispiel deutlich, bei dem der Server ohne Verwaltungsmechanismen an die eigenen Grenzen stößt, wenn dieser mehrere Anfragen für Bild- bzw. Videoverarbeitungen entgegen nimmt und diese sowohl direkt nach dem Request, als auch parallel durchführt. In einem solchen Fall kann die korrekte Ausführung der Anwendung nicht weiter gewährleistet werden.

Im Falle des Prototyps besteht das Aufgabenmanagement aus eine Java-Klasse, welche ebenso als Cache oder Zwischenspeicher angesehen werden kann. Dabei handelt es sich um die „MaxityMediaProcessingCache“ Klasse, welche von der „MaxityMediaApplication“ Klasse, also der Hauptanwendung, beim Starten dieser geladen wird. Das Cache Objekt an sich basiert dabei auf dem Singleton Modell und wird demnach nur ein einziges Mal initialisiert, d.h. es besteht während der gesamten Laufzeit der Applikation nur diese eine Instanz. Realisiert wird dies mit Hilfe eines privaten Konstruktors und einer statischen Variable. Für die Verwaltung der Aufgaben bzw. Tätigkeiten wird eine seit Java 5 bestehende Implementierung genutzt, dabei handelt es sich um die „ThreadPoolExecutor“ Klasse aus dem Package „java.util.concurrent“, die nachfolgend näher erläutert werden soll.

ThreadPoolExecutor

Mit Hilfe des ThreadPoolExecutor[32] existiert eine fertige, gut konfigurierbare und sehr flexible Implementierung für die Verwaltung von Threads, die wiederum asynchron ausgeführt werden. Dies macht im Normalfall die Erstellung eigener Threadpools überflüssig, da diese Implementierung jede Menge Konfigurationsmöglichkeiten mit sich bringt.

Unter Anwendung des ThreadPoolExecutor wird eine vorgegebene Anzahl von Threads erzeugt und mittels dieser laufenden Threads übergebene Aufgaben (Tasks bzw. Tätigkeiten) asynchron abgearbeitet. Diese Vorgehensweise der Nutzung bestehender Threads bringt den Vorteil mit sich, dass der beim Erzeugen, Starten sowie Beenden von Threads enstehende Overhead entfällt und somit in dieser Hinsicht Performance-Einbußen entgegen gewirkt wer-

den kann. Erforderlich für die Erzeugung eines ThreadPoolExecutor Objekts sind mindestens fünf Parameter, mit deren Hilfe die Konfiguration dessen erfolgt. Dies sind Angaben zur Kerngröße des Pools und anschließend zur maximalen Poolgröße, wobei Kerngröße die Basisanzahl der zu laufenden Threads im Pool angibt. Weiterhin folgen die Parameter zur Festlegung der Lebensdauer von Threads und der zugehörigen Zeiteinheit, wobei auf diese Zeitangabe erst dann zurückgegriffen und der jeweilige Thread beendet wird, wenn mehr Threads, als durch die Pool Kerngröße festgelegt wurde, im Pool vorhanden sind. Abschließend folgt ein Parameter für die interne Warteschlange, welche die laufwilligen Threads aufnimmt, wenn kein Thread im Pool für die Abarbeitung bereit steht. Dabei ist zu beachten, dass als Warteschlange lediglich ein „BlockingQueue“ Objekt übergeben werden kann, wie zum Beispiel die „SynchronousQueue“, „LinkedBlockingQueue“ oder die „ArrayBlockingQueue“. Je nach Art dieser Warteschlange kann sich das Verhalten des ThreadPoolExecutors ändern. Dies waren die Parameter die im Normalfall ausreichen, für spezielle Anpassungen stehen zwei weitere Konstruktoren zu Verfügung, die an dieser Stelle nicht weiter erläutert werden sollen. Anhand der nachfolgenden Abbildung 5.15 soll die Funktionsweise vereinfacht dargestellt werden.

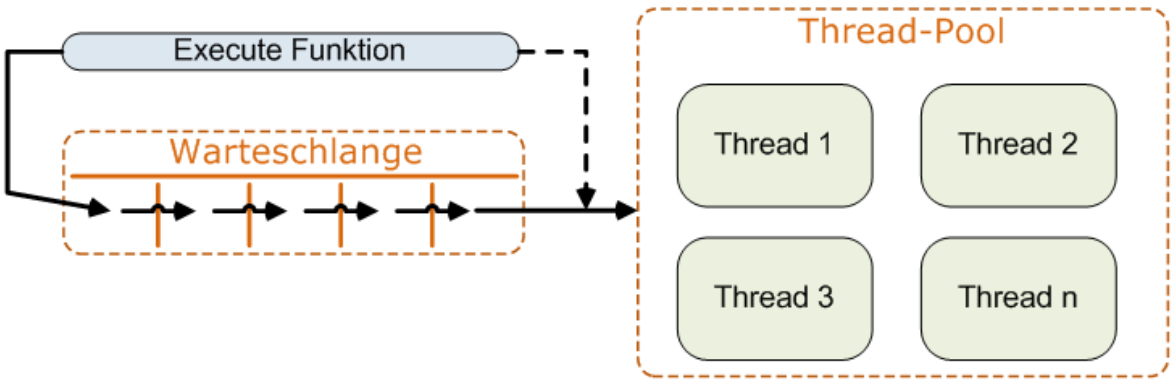

Abbildung 5.15: *Ablauf im Thread-Pool*

Wie zuvor beschrieben, wird die Anzahl der im Pool laufenden und auf Aufgaben wartenden Threads durch den ersten im Konstruktor aufgeführten Parameter bestimmt. In der Abbildung 5.15 soll die Kerngröße des Pools anhand der dargestellten Threads veranschaulicht werden. Diese Größe kann bei den jeweiligen ThreadPoolExecutor Objekten anders ausfallen, aus diesem Grund soll diese variable Größe durch den „Thread n“ gekennzeichnet werden.

Beim Aufruf der Funktion „execute“ wird unterschiedlich, abhängig von der aktuellen Situation, mit der übergebenen Aufgabe bzw. dem übergebenen Auftrag verfahren. Das bedeutet, dass so lange wie die Anzahl der bestehenden Threads im Pool kleiner ist als die Pool Kerngröße, ein neuer Thread im Pool erzeugt wird und den übergebenden Auftrag direkt und asynchron ausführt. Dieser Fall soll in der Abbildung mit der gestrichelten Linie verdeutlicht werden. Im Gegensatz dazu werden, wenn die Kerngröße des Pools erreicht und kein freier untätiger Thread verfügbar ist, die jeweiligen übergebenen Aufgaben in die Warteschlange gesteckt. Bei Warteschlangen mit begrenzter Größe kann es passieren das, wenn erneut ein

Auftrag an die „execute“ Funktion übergeben wird, sowohl kein freier untätiger Thread im Pool als auch kein Platz in der Warteschlange verfügbar ist. In diesem Fall wird ein neuer Thread im Pool erzeugt und die Aufgabe abgearbeitet, wobei mit der Erzeugung neuer Threads nur so lange weiter verfahren wird, bis die maximale Poolgröße erreicht ist. Tritt dieser Fall ein, wird der übergebene Auftrag abgelehnt und die Methode „rejectedExecution“ des Interfaces „RejectedExecutionHandler“ aufgerufen, wobei der nicht auszuführende Auftrag sowie der entsprechende ThreadPoolExecutor, welcher diesen Auftrag nicht ausführen konnte, an diese Methode übergeben wird. Im Gegensatz zu begrenzten Queues können ebenso Warteschlangen mit dynamischer Größe genutzt werden. Hierbei tritt der eben erwähnte Fall der „rejectedExecution“ niemals ein, da die Warteschlange keine feste Größe enthält und somit immer ein freier Platz verfügbar ist.

Wie man anhand dieser Erläuterung feststellen konnte, bringt Java seit der JDK Version 5 eine sehr gut konfigurierbare und flexible ThreadPool Implementierung mit sich, wodurch sich eigene Aufträge bzw. Aufgaben hervorragend verwalten lassen. Ausgehend von dieser allgemeinen Beschreibung der Funktionsweise des ThreadPoolExecutors kann nun im Folgenden detailliert auf die implementierte Variante im Prototyp eingegangen werden.

ThreadPoolExecutor-Implementierung im Prototyp

Wie schon erwähnt, übernimmt die Java-Klasse „MaxityMediaProcessingCache“ im „core“ Package des Prototyps die Aufgabe der Verwaltung der jeweiligen Multimedia-Verarbeitung. Hinsichtlich der unterschiedlichen Medientypen wurde diese Verwaltungsebene noch einmal aufgeteilt, so dass ein ThreadPoolExecutor für die Bild- und einer für die Videoverarbeitung verantwortlich ist. Diese Trennung ist insbesondere deshalb notwendig, damit sich die jeweiligen Bildaufträge und Videoaufgaben nicht gegenseitig stören bzw. blockieren, zumal die Verarbeitung der Bilddateien, natürlich abhängig von der Dateigröße, um ein Vielfaches schneller abläuft, als die Videoverarbeitung. Die weiteren Details über den Ablauf der beiden ThreadPoolExecutor sollen anhand der nachfolgenden Abbildung 5.16 unterstützend veranschaulicht und erläutert werden. Bevor allerdings mit der Beschreibung begonnen wird, soll darauf hingewiesen werden, dass sich die Aufgaben der „MaxityMediaProcessingCache“ Klasse und den beiden enthaltenen ThreadPoolExecutor in den Gesamtverarbeitungsprozess der jeweiligen Medientypen integrieren. Diese Integration wurde in den vorangegangen Abbildungen und den dazugehörigen Erläuterungen in den jeweiligen Verarbeitungsabschnitten dargestellt bzw. erwähnt. Allerdings beschränkte sich die Darstellung dieser „Zwischenschritte“, wie in den Erläuterungen angedeutet, in den Abbildungen auf eine gestrichelte Linie ausgehend von der jeweiligen „Processing“ Klasse.

Wie in der Abbildung 5.16 zu sehen ist, sind die jeweiligen Ausgangspunkte die „Processing“ Klassen. Beim Aufruf einer „Ersteller“ Methode wird ein dazugehöriges Auftragsobjekt (Task) erzeugt, was im Falle von Bilddateien das „MaxityMediaImageProcessingTask“ Objekt und bei Videodaten das „MaxityMediaVideoProcessingTask“ Objekt ist. Diese Auftragsobjekte

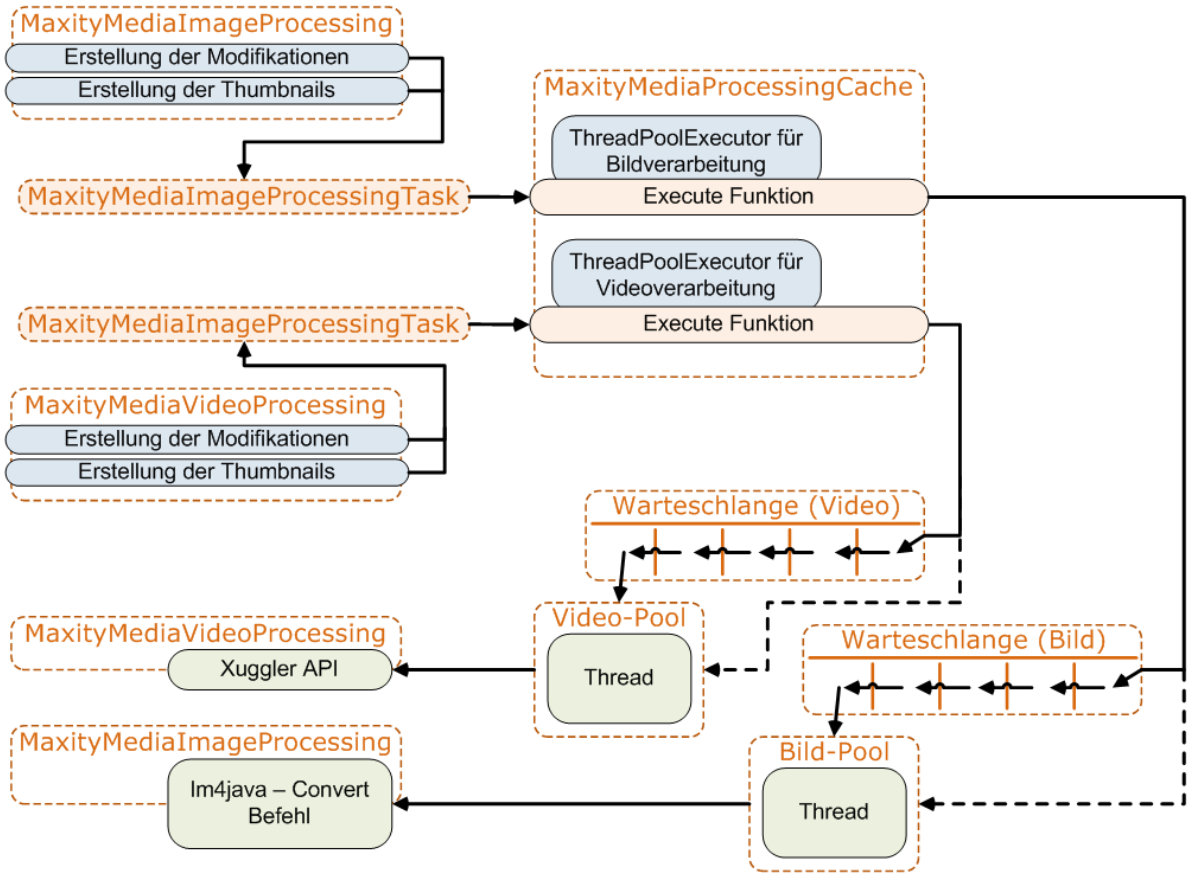

Abbildung 5.16: *Ablauf der jeweiligen Thread-Pools im Prototyp*

werden der „MaxityMediaProcessingCache“ Klasse übergeben, in welcher die „execute“ Funktion des jeweiligen ThreadPoolExecutors diese entgegen nimmt und verarbeitet. Die beiden in dieser Klasse dargestellten ThreadPoolExecutor werden beim Laden des Caches initialisiert und konfiguriert, wobei die im Prototyp vorgenommene Konfiguration wie in dem nachfolgenden Beispiel 5.19 aussieht.

```
ThreadPoolExecutor threadPoolExecutorImage =
  new ThreadPoolExecutor(1, 1, 50000L,
    TimeUnit.SECONDS, new LinkedBlockingQueue<Runnable>()
  );
ThreadPoolExecutor threadPoolExecutorVideo =
  new ThreadPoolExecutor(1, 1, 50000L,
    TimeUnit.SECONDS, new LinkedBlockingQueue<Runnable>()
  );
```

Beispiel 5.19: *Bild und Video - ThreadPoolExecutor*

Wie schon zuvor im allgemeinen Teil erwähnt, sind mindestens fünf Parameter für die Erzeugung eines ThreadPoolExecutors von Bedeutung. In diesem Beispiel 5.19 soll besonderes

Augenmerk auf die jeweiligen beiden ersten sowie dem letzten Parameter gerichtet werden. Die beiden ersten sind jeweils Ziffern mit dem Wert eins, dabei definiert der erste Parameter die Pool Kerngröße, also wieviel Threads für die angehenden Aufträge bereitstehen sollen, wohingegen der zweite Wert die Maximalgröße des Pools, also wieviel Threads dieser Pool höchstens aufnehmen darf, festlegt. Des Weiteren kann der letzte Parameter, welcher in diesem Fall die Warteschlange definiert, ebenso das Verhalten des ThreadPoolExecutors beeinflussen. Wie in der Abbildung zu sehen ist, besitzt in diesem Fall jeder ThreadPoolExecutor seine eigene Warteschlange. In diesem Zusammenhang ist es jeweils das „LinkedBlockingQueue“ Objekt, welches keine feste Größe besitzt und dynamisch wachsen kann, wodurch der im allgemeinen Teil beschriebene Fall des „RejectedExecutionHandler“ nicht eintreten kann. Nun wieder zurück zur Abbildung und den jeweiligen „execute“ Funktionen.

```
FutureTask<MaxityMediaProcessingTask> task =
  new FutureTask<MaxityMediaProcessingTask>(runnable, null);
tpeImage.execute(task);
```

Beispiel 5.20: *Bild-ThreadPoolExecutor „execute“ Funktion*

Der Funktionsaufruf „execute“ wird in dem Beispiel 5.20 anhand des ThreadPoolExecutors für die Verwaltung der Bildverarbeitung dargestellt. Dabei ist zu beachten, dass die „execute“ Funktion entweder ein „Runnable“ oder „Callable“ Objekt erwartet, welches in diesem Fall bzw. im Fall des Prototyps das „FutureTask“ Objekt ist. Die Klasse „FutureTask“ selbst implementiert die Interfaces „Runnable“ und „Future“, wobei letzteres die aktuellen Zustände der jeweiligen Tätigkeiten zur Verfügung stellt und Möglichkeiten des Zugriffs auf diese offeriert. Zustände wären zum Beispiel, ob der jeweilige Task beendet, abgebrochen oder überhaupt gestartet wurde. Die Eigenschaften der implementierten Interfaces werden in der Klasse „FutureTask“ vereinigt und geben dieser die Möglichkeit einen Auftrag asynchron auszuführen sowie die entstandenen Ereignisse oder Ausnahmen zu erfassen und aufzubewahren. Des Weiteren ist bei der Verwaltung der Tätigkeiten ebenso die vollständige Kontrolle über die jeweiligen Aufträge gegeben, zum Beispiel durch die Möglichkeit des Abbruchs einer bestimmten Tätigkeit selbst.

An die „execute“ Funktionen anknüpfend, können nun zwei Fälle unterschieden werden. Entweder ist der Thread im Pool frei und kann den übergebenen Auftrag übernehmen und ausführen (gestrichelte Linie in der Abbildung) oder der Thread ist bereits belegt, weshalb sich der aktuell übergebene Auftrag in die Warteschlange einreihen muss. Ist ein Auftrag einmal in der Queue, muss er warten, bis alle Tätigkeiten vor ihm in der Warteschlange ausgeführt wurden und der Thread frei ist. Wenn dieser Fall eintritt, kann der untätige Thread im Pool den Auftrag übernehmen und die enthaltenen einzelnen Aufgaben ausführen. Da es sich dabei um die Bild- bzw. Videoverarbeitungen handelt, werden wiederum Funktionen in den dafür vorgesehenen Klassen „MaxityMediaImageProcessing“ und „MaxityMediaVideoProcessing“ aufgerufen. Das eben erwähnte Aufgabenmanagement sowie die Verarbeitungen der

Multimedia-Daten konzentrieren sich in den Java-Klassen des „com.maxity.media.core“ Packages. Mit dem nachfolgenden UML-Klassendiagramm 5.17 sollen die erwähnten Java Klassen und deren Aufgaben abgerundet und bildhaft zusammengefasst werden.

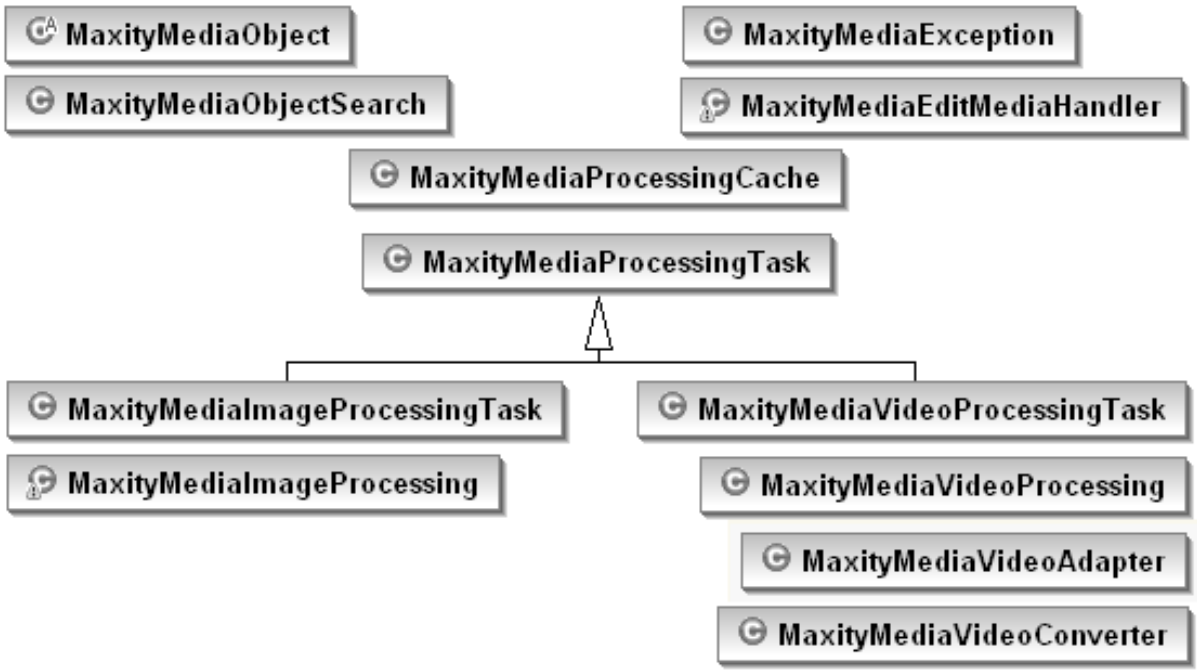

Abbildung 5.17: *Java Klassen im Package „com.maxity.media.core“ für die entsprechenden Multimedia-Verarbeitungen*

Damit schließt sich der Kreis der jeweiligen Multimediaverarbeitungen in diesem Prototyp. Abschließend ist zu sagen, dass mit Hilfe des ThreadPoolExecutor Objekts das Management der verschiedenen Aufgaben hinsichtlich der Verarbeitung und Manipulation der Bild- und Videodaten hervorragend verwalten und steuern lassen.

5.2.10 Zusammenfassung

Nachdem nun der Prototyp in seiner Drei-Schichten-Architektur im Einzelnen beschrieben und die jeweiligen Zusammenhänge näher detailliert erläutert wurden, soll in diesem Abschnitt der vollständige Ablauf eines Prozesses (vom Client bis hin zur eigentlichen Datei, dem Datenbankeintrag und den dazugehörigen Verarbeitungen) anhand eines konkreten Beispiels beschrieben werden. Dieser Prozessablauf wird anhand der Abbildung 5.18 unterstützend veranschaulicht.

Dabei ist in der Abbildung 5.18 eine Trennung zwischen Client- und Serverseite zu sehen. Der vollständige Prozessablauf, also von der Auswahl der Media-Datei durch den Nutzer, über den Upload dieser Datei bis hin zur Verarbeitung und den jeweiligen Datenbankeinträgen, soll in diesem Beispiel anhand eines Video-Uploads demonstriert werden. Hierbei handelt es sich, wie dargestellt, um eine Videodatei, welche durch den Nutzer ausgewählt und mit Hilfe des Flash-Uploads an die „AddMediaUploadVideo“ Webseite gesendet sowie in den nachfolgen-

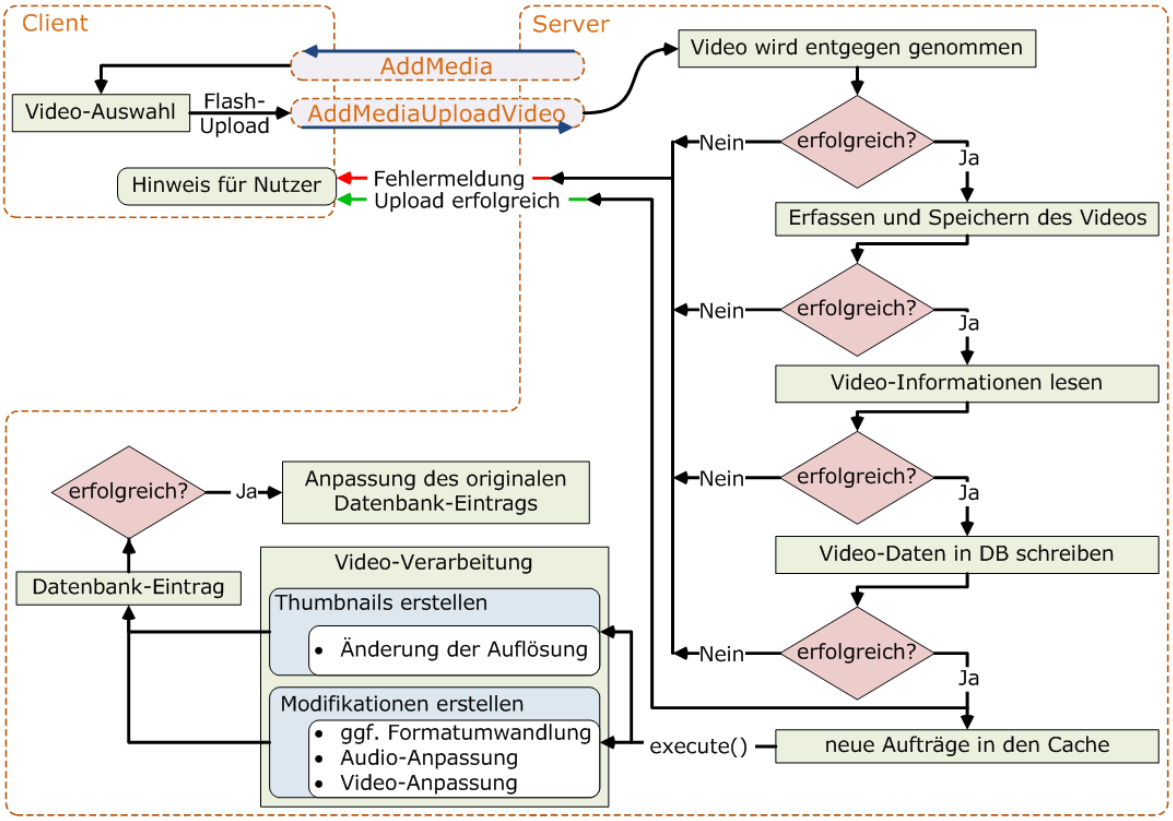

Abbildung 5.18: *Prozessablauf am Beispiel eines Video-Uploads*

den Teilschritten verarbeitet wird. Dabei ist im Einzelnen zu erkennen, dass einige Vorgänge, wie das Speichern der Videodatei, sowie das Auslesen der Informationen und Erstellen des entsprechenden Datensatzes in der Datenbank, notwendig sind, bevor die eigentlichen Verarbeitungsschritte eingeleitet werden können. Zwischen den eben genannten Teilprozessen unterliegen die jeweiligen Ergebnisse bestimmten Prüfungen, wobei bei Misserfolg ein sofortiger Status-Hinweis an den Nutzer gesendet wird. Nach diesen grundlegenden Tests können bei Erfolg neue Tätigkeiten an den Cache (wie im Abschnitt 5.2.9 beschrieben) übergeben und in die Auftragsliste eingereiht werden. Parallel dazu findet die Rückmeldung an den Nutzer statt, welche dem Client einen erfolgreichen Upload signalisieren soll. Zurück im Cache werden die jeweiligen Tätigkeiten, wenn diese an der Reihe sind, durch ihn ausgeführt („execute()“ in der Abbildung), wobei der entsprechende Auftrag Modifikationen bzw. Thumbnails erstellt. Anschließend werden abhängig von den jeweiligen Ergebnissen die Daten in der Datenbank gesichert, wobei wiederum eine Untersuchung auf Erfolg stattfindet. Tritt der der Erfolgsfall ein, werden die entsprechenden in der Datenbank vorliegenden Daten geprüft und nach erfolgreicher Prüfung der originale Videodatensatz dementsprechend geändert. Diese Änderungen sind lediglich auf sogenannte Markierungen hinsichtlich der Vollständigkeit der Modifikationen und Thumbnails bezogen.

Dieser Prozessablauf sollte einen kurzen Überblick über den Weg einer Media-Datei (in

diesem Beispiel einer Videodatei) vom Client zum Server mit den entsprechenden Verarbeitungsschritten im Prototyp geben. Weiterhin war die Absicht mit Hilfe dieser Zusammenfassung die Integration sowie das Zusammenspiel der einzelnen in den vorherigen Abschnitten beschriebenen Teilprozesse in den Gesamtprozess darzustellen und zu verdeutlichen.

5.2.11 Implementationsumgebung

Das nachfolgende Produktumfeld wurde während des gesamten Entwurfes sowie der Implementation des Prototyps verwendet.

- Datenbankmanagementsystem Postgresql 8.3
- Java Development Kit 6 Update 16 mit Java EE
- Apacha Tomcat 6.0.20 Servlet-Container
- Apach Wicket 1.4
- Hibernate 3.3.1.GA
- SWFUpload 2.2.0.1
- ImageMagick 6.5.7-0 i386
- FFmpeg 0.5
- Im4java 0.98.0
- Xuggler AGPL 3.2 - „Dalton“

5.3 Test des Prototyps

Um eine weitestgehend später fehlerfreie Funktionsweise des Prototyps zu gewährleisten, wurden zwei voneinander unterschiedliche Tests durchgeführt.

Kontrolle in einer unabhängigen Anwendung

Während der gesamten Entwicklungsphase des Prototyps wurden insbesondere die einzelnen Bereiche der Verarbeitungen separaten Tests unterzogen. Dabei dienten die in den jeweiligen Abschnitten der Verarbeitungen aufgezeigten Beispiele über die Nutzung der externen Software-Pakete (ImageMagick und FFmpeg) als Vorlage zur Erstellung der korrekten Befehle in den jeweiligen standalone Applikationen. Mit Hilfe dieser Vorgehensweise konnten die einzelnen für den Prototyp notwendigen Funktionen (siehe Pflichtkriterien 5.2.1) in den jeweiligen Verarbeitungsbereichen getrennt voneinander auf korrekte Arbeitsweise getestet werden.

In Bezug auf die Bildverarbeitung konnten unter Berücksichtigung der aufgestellten Kriterien keine Probleme oder Einschränkungen festgestellt werden, sodass alle erstellten Tests in dieser Hinsicht erfolgreich als eigenständige Applikation ausgeführt wurden. Hinsichtlich der Verarbeitung von Videodateien verliefen die ersten Tests ohne auf spezielle Kriterien einzugehen, von vornherein erfolgreich. Bei der Veränderung von bestimmten Videoeigenschaften war es dann allerdings notwendig speziellere Methoden und Objekte der Xuggler API nutzen zu müssen. Ungeachtet dessen konnten alle festgelegten Pflichtkriterien in den standalone Applikationen umgesetzt und erfüllt werden. Bei der Entwicklung und den Tests sind derzeit einige Einschränkungen und bestimmte Entwicklungsreihenfolgen gegeben. Dabei handelt es sich um Kompatibilitätsgrenzen hinsichtlich der Audioverarbeitung. Aktuell werden von der Java Bibliothek Xuggler die Verarbeitungen von maximal zwei Audio-Kanälen und einer maximalen Abtastrate von 44100 Hz unterstützt. Diese Tatsache ist allerdings momentan kein Hindernis für die Erfüllung der Pflichtkriterien.

Ausgehend von den bisherigen erfolgreichen Tests in standalone Applikationen, konnten diese einzelnen getesteten Verarbeitungsschritte, ohne das großartig Anpassungen notwendig waren, in die Webapplikation integriert werden.

Kontrolle in einer Webanwendung im Tomcat

Nach den jeweils erfolgreichen standalone Anwendungstests wurden diese Schritt für Schritt in die Webanwendung integriert und getestet. Dabei fiel im ersten Moment nichts Außergewöhnliches auf, die jeweils integrierten Abschnitte arbeiteten korrekt und lieferten richtige Ergebnisse. Die einzige Besonderheit in diesem Fall war die Integration der Xuggler Bibliothek, da sie über das Java Native Interface mit nativen Code arbeitet. Aus diesem Grund war es notwendig der Tomcat-Anwendung die spezifischen Pfadangaben bzw. die speziellen Systemvariablen mitzuteilen. Lediglich diese eine Anpassung war notwendig und die Webanwendung führte die vorher in den standalone Applikationen getesteten Aktionen erfolgreich aus. Da der Prototyp auf einem Linux System zum Einsatz kommt, wurden alle bisherigen Tests in einer Linux-Distribution realisiert und durchgeführt. Dabei wäre interessant zu wissen, ob der Einsatz auf einem Windows System ebenso unproblematisch realisiert werden kann. In diesem Sinne wurden die Test-Applikationen auf das Windows System kopiert und ausgeführt, wobei es hinsichtlich der Videoverarbeitung zu keiner Ausnahmesituation kam und alle Test-Anwendungen ebenso erfolgreich durchgeführt wurden. Ganz anders sah es diesbezüglich bei der Verarbeitung von Bilddaten aus. Bei den durchgeführten Tests konnten lediglich sehr einfache Manipulationsfunktionen der ImageMagick API genutzt werden. Kamen speziellere Methoden wie zum Beispiel „crop“ oder „composite“ zum Einsatz, wurde die Anwendung mit einer Ausnahme abgebrochen und konnte nicht erfolgreich durchgeführt werden. Woran es letztendlich gelegen hat, konnte nicht eindeutig festgestellt werden, war aber auch für die Entwicklung des Prototyps, aufgrund der Tatsache das ein Linux System verwendet wird, nicht weiter von Belang.

6 Schlussbetrachtungen und Ausblick

Schlussbetrachtung

In dem letzten Kapitel dieser Diplomarbeit soll ein kurzes Resümee aus der Prototyp-Implementation gezogen werden. Das Ziel dieser Diplomarbeit, den Entwurf und die Prototyp-Implementation zur Übertragung, Speicherung und der automatischen Verarbeitung von Multimedia-Daten sowie die Analyse dieser, konnte erfolgreich erfüllt werden. Ebenso konnten alle festgelegten Pflichtkriterien umgesetzt werden, wobei jedoch einige wenige Einschränkungen aufgezeigt wurden. Dabei wurde darauf geachtet, dass die einzelnen Aufgabenbereiche im Prototyp nicht zu sehr mit anderen verbunden wurden, um die Drei-Schichten-Architektur und somit die Modularität aufrecht zu erhalten. Mit Hilfe dieser Tatsache lassen sich die einzelnen Bereiche frei miteinander kombinieren und ebenso gegebenenfalls ersetzen. Insbesondere letzteres kann aufgrund der Konzentration der einzelnen Funktionen in den jeweiligen Verarbeitungsklassen „MaxityMediaImageProcessing“ und „MaxityMediaVideoProcessing“ unkompliziert durchgeführt werden, wodurch zum Beispiel ein Wechsel von einer integrierten Schnittstelle hin zu einer anderen erleichtert werden kann und somit lediglich eine schwache Abhängigkeit der genutzten Bibliotheken bzw. Schnittstellen im Prototyp besteht.

Ausblick

Die im Prototyp implementierten Bibliotheken und Schnittstellen genügen den derzeitigen Pflichtkriterien. Diese verwendeten Interfaces werden fortlaufend weiterentwickelt und es bleibt abzuwarten, welche Neuerungen und Verbesserungen folgen werden und sich somit gegebenenfalls bisherige Einschränkungen aufheben lassen.

Um speziell auf die Verarbeitung von Bilddaten einzugehen, kann gegebenenfalls unter Verwendung einer alternativen Schnittstelle zu ImageMagick aus Java heraus mit Performancegewinn zu rechnen sein. So können Interfaces, welche auf native Methoden zurückgreifen, die Performance, Möglichkeiten der Interaktion sowie den Funktionsumfang steigern. Dies spiegelt sich zum einen in der Art und Weise des Aufrufes wieder, wobei in diesem Fall Ressourcen eingespart werden können, da nicht für jeden Aufruf einer dieser Funktionen ein neuer Systemprozess erstellt werden muss, und zum anderen in der Möglichkeit der Nutzung umfangreicher Datentypen ohne auf den „String“ Typ eingeschränkt zu sein. Dies kann zu einer Verbesserung dieser Anwendung führen und diese gegebenfalls im Funktionsumfang erweitern.

Da insbesondere die Nutzung, Be- und Verarbeitung zur Darstellung und Anzeige von

Multimedia-Dateien im World Wide Web in der letzten Zeit stark zugenommen hat, besteht gegebenenfalls die Möglichkeit der Bekanntgabe weiterer Entwicklungen in den einzelnen Verarbeitungsbereichen hinsichtlich der Nutzung in Java. Dies wird sich zeigen.

Abschließend kann gesagt werden, dass sich der Prototyp in der nächsten Zeit in dem Unternehmen, für das diese Diplomarbeit angefertigt wurde, bewähren muss. Da der Knüpfer Verlag überwiegend tourismusspezifische Daten verwaltet und den Verlagskunden, welche eher „Normalanwender“ als Softwareentwickler sind, die Möglichkeit der Pflege dieser Daten bietet, ist eine intuitive Benutzerführung und Anwendung erforderlich. Dieser Prototyp wird sich fortan diesen Herausforderungen stellen und beweisen müssen.

Literaturverzeichnis

[1] *jQuery - JavaScript Library*. http://jquery.com, 2009

[2] *Adobe Flash-Player - Funktionen*.
http://www.adobe.com/de/products/flashplayer/productinfo/features/#model,
2009

[3] *Thomas Walter: Kompendium der Web-programmierung: Dynamische Web-sites*.
1. Aufl. - Berlin, Springer-Verlag, 2007, S. 406-414,587

[4] *Sun Microsystems*. http://www.sun.com, 2009

[5] *Teialehrbuch - Plattformunabhängigkeit*.
http://www.teialehrbuch.de/Kostenlose-Kurse/JAVA/6534-
Plattformunabhaengigkeit.html,
2009

[6] *Java Advanced Imaging (JAI)*.
http://java.sun.com/javase/technologies/desktop/media/jai/, 2009

[7] *ImageMagick*. http://www.imagemagick.org, 2009

[8] *JAI vs. ImageMagick image resizing*.
http://www.darcynorman.net/2005/03/15/jai-vs-imagemagick-image-resizing/,
2005

[9] *FFmpeg*. http://www.ffmpeg.org, 2009

[10] *FFmpeg Dokumentation*. http://www.ffmpeg.org/ffmpeg-doc.html, 2009

[11] *Transcode*. http://www.transcoding.org, 2009

[12] *Transcode - General Information*.
http://www.transcoding.org/cgi-bin/transcode?General_Information, 2009

[13] *Libavcodec*. http://de.wikipedia.org/wiki/Libavcodec, 2009

[14] *Howard Pritchett - Using ffmpeg to manipulate audio and video files*.
http://howto-pages.org/ffmpeg/, 2006

[15] *Flash Player Penetration*.
http://www.adobe.com/products/player_census/flashplayer/, 2008

[16] *Chuck Musciano; Bill Kennedy: „HTML und XHTML: Das umfassende Handbuch“*.
5. Aufl. : O'Reilly, März 2007, S. 128-134

[17] *Fortenbacher, Albrecht; Kutscher, Dirk: „Perl & CGI“*.
1. Aufl. : TEIA - Internet Akademie und Lehrbuch Verlag, März 2006

[18] *Sicherheitsspezialist Stefan Esser verlässt PHP-Security-Team*.
http://www.heise.de/newsticker/meldung/
Sicherheitsspezialist-verlaesst-resigniert-PHP-Security-Team-126467.html, 2006

[19] *PHP Ausnahmebehandlung*. http://us2.php.net/exceptions, 2009

[20] *Browser Statistik*. http://www.w3schools.com/browsers/browsers_stats.asp, 2009

[21] *ImageJ Einführung*. http://rsb.info.nih.gov/ij/docs/intro.html, 2009

[22] *GraphicsMagick*. http://www.graphicsmagick.org/, 2009

[23] *GraphicsMagick*.
http://www.kitchensoap.com/2009/04/03/
slides-from-web20-expo-2009-and-somethin-else-interestin/, 2009

[24] *Benchmark: GraphicsMagick 1.3.1 gegenüber ImageMagick 6.4.5*.
http://www.graphicsmagick.org/benchmarks.html, 2008

[25] *FFmpeg Projekt - Komponentenübersicht*.
http://ffmpeg.org/about.html, 2009

[26] *Xuggler - Bibliothek für Java und C++ Entwickler*.
http://www.xuggle.com/xuggler/, 2009

[27] *FOBS - C++ Wrapper für FFmpeg*. http://fobs.sourceforge.net/, 2008

[28] *FFMPEG-Java - Java Wrapper für FFmpeg*.
http://fmj-sf.net/ffmpeg-java/getting_started.php, 2007

[29] *Hibernate Tutorial - Konfigurationsdatei*.
http://docs.jboss.org/hibernate/stable/core/reference/en/html/
tutorial.html#tutorial-firstapp-configuration, 2004

[30] *Hibernate Tutorial - Arbeiten mit Objekten*.
http://docs.jboss.org/hibernate/stable/core/reference/en/html/objectstate.html, 2004

[31] *Apache Wicket - Working with Wicket models*.
http://cwiki.apache.org/WICKET/working-with-wicket-models.html, 2009

[32] *Java 6 - ThreadPoolExecutor*.
http://java.sun.com/javase/6/docs/api/java/util/concurrent/ThreadPoolExecutor.html, 2008

[33] *JMagick - Java interface of ImageMagick*.
http://www.jmagick.org/index.html, 2008

Selbständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Mittweida, 26. November 2009

Daniel Schmidt