

**Konzeption und Entwicklung einer an YouTube
orientierten Lösung für Video-Streaming auf Basis
von Cloud Computing-Diensten**

Sufian Abu-Rab

Diplomarbeit
Studiengang Informatik

Fakultät für Informatik
Hochschule Mannheim

6 April 2011

Betreuer: M.Sc. Christian Baun
Zweitkorrektor: Prof. Dr. Georg Winterstein

Inhaltsverzeichnis

Eidesstattliche Erklärung	i
Danksagung	iii
Kurzbeschreibung	iv
Abstract	v
1 Einleitung	2
1.1 Allgemeines	2
1.2 Aufbau der Arbeit	3
2 Grundlagen	4
2.1 Cloud Computing	4
2.2 Definition Cloud Computing	5
2.3 Cloud-Architekturen	6
2.4 Kategorien von Cloud-Diensten	7
2.4.1 Softwaredienste	7
2.4.2 Infrastrukturdienste	8
3 YouTube	12
3.1 Geschichte von YouTube	12
3.2 Funktionalität von YouTube	12
3.3 Erfolg von YouTube	13
3.4 Technik von YouTube	14
3.5 Einschränkungen von YouTube	14
4 Entwicklung der Infrastruktur für das Video-Streaming	17
4.1 Anforderungsanalyse	17
4.2 Evaluation der entwickelten Lösung in EC2	20
4.3 Ergebnis	28
4.4 Kostenanalyse	29

4.5	Bewertung	30
5	Entwicklung der Webanwendung für Video-Streaming auf Basis von Cloud-Diensten	31
5.1	Grundvoraussetzungen	31
5.2	Rahmenbedingungen	32
5.2.1	Betriebssystem des Webservers	32
5.2.2	Konfiguration des Webservers	32
5.2.3	Symfony-PHP-Framework	33
5.3	Anforderungsanalyse der Anwendungslogik	33
5.3.1	Video-Upload	33
5.3.2	Video-Anzeige	35
5.4	Realisierung der Grundvoraussetzungen und Rahmenbedingungen . . .	35
5.5	Realisierung und Implementierung der Anforderungen	38
5.5.1	Hochladen eines Videos	38
5.5.2	Video-Anzeige	45
5.6	Ergebnis	48
5.7	Vergleich mit der existierenden Lösungen YouTube	49
5.8	Erweiterungsmöglichkeiten	52
6	Zusammenfassung	53
6.1	Herausforderungen bei der Entwicklung	54
6.2	Ausblick	55
	Abbildungsverzeichnis	57
	Literaturverzeichnis	58

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich diese Diplomarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, alle Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, kenntlich gemacht sind und die Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.

Sufian Abu-Rab

Mannheim, den 13.03.11

Danksagung

Ich möchte mich an dieser Stelle bei M.Sc. Christian Baun für seine Unterstützung und Betreuung im Rahmen dieser Diplomarbeit bedanken. Des Weiteren danke ich Prof. Dipl.-Inform. Günther Bengel für seine Unterstützung während des gesamten Studiums.

Ein besonderer Dank gilt meinen Eltern Achmed Abu-Rab und Afifeh Abu-Rab, die mich immer unterstützt haben.

Kurzbeschreibung

Diese Diplomarbeit beschäftigt sich mit der Konzeption und Realisierung einer *Cloud*-basierten Streaming-Video-Lösung. Als kommerzielles Vorbild dient Googles Dienst YouTube. Die Infrastruktur ist über die öffentlich verfügbaren Cloud-Dienste von Amazon (Amazon Web Services) realisiert. Zusätzlich wird auf Basis dieser Cloud-Infrastruktur eine Webanwendung für Video-Streaming implementiert.

Das Thema der Arbeit besteht zunächst darin, mit Hilfe des Infrastrukturdienstes Elastic Compute Cloud (EC2) auf einem entferntliegenden physischen Cluster virtuelle Betriebssysteminstanzen auszuführen und diese zu einem virtuellen Cluster zusammenzufügen. Die virtuellen Maschinen dienen als Application-Server für die Video-Streaming-Webanwendung. Ein Server zum Lastausgleich (englisch: Loadbalancer) sorgt dafür, dass die Videoanfragen an freie Webserver weitergeleitet und ausgefallene Server erkannt werden. Bei Bedarf werden neue Instanzen gestartet bzw. überflüssige stoppt. Die Datenbank ist redundant und skalierbar ausgelegt, sodass diese ebenfalls bei Bedarf ihre Speicher- und Rechenkapazität skalieren kann. Des Weiteren werden selbständig zu bestimmaren Zeitpunkten Sicherheitskopien angelegt. Die Videodaten werden über ein Content Distribution Network (CDN) an die Nutzer verteilt.

Schließlich geht es um die Anforderungen und Implementierung einer Streaming-Video-Webanwendung. Hierbei wird auf Basis der Programmiersprache PHP und dem PHP-Framework Symfony das Hochladen von Videos und eine Video-Anzeige implementiert.

Die für diese Diplomarbeit entwickelte und implementierte Software fokussiert die Installation und Konfiguration eines virtuellen Streaming-Clusters und bietet Vorteile gegenüber den bereits existierenden und untersuchten Lösungen YouTube [52], Metacafe [36] und Megavideo [35], wie z.B. eigene Datenschutz und -sicherheitsbestimmungen, sowie Informationen aus Benutzereingaben zu statistischen Zwecken.

Abstract

This thesis deals with the design and implementation of a cloud-based streaming video solution. „YouTube“, a service of Google, is used as a commercial example. The infrastructure is realized within Amazons public cloud services (Amazon Web Services). Additionally a streaming video web-application is implemented, which is also based on this public cloud infrastructure.

The theme of this work is to use the public cloud infrastructure service Amazon EC2 to first run virtual operating system instances at distant physical clusters and to bring them together, forming a virtual cluster. The virtual machines are equipped with a Linux operating system and serve as a web or application server for streaming web application. A load balancing server is used to ensure that video requests are forwarded to free webservers and that failing servers are detected. If necessary, new instances will be started and unnecessary stopped. The database is designed redundant and scalable, so that it can scale its storage or computing capacity dynamically by need. Furthermore, backups will be created, which can be set to selectable timepoints. The video data is distributed via a Content Distribution Network (CDN) to users.

Finally, it is about the requirements and implementation of a streaming-video web-application. Here a videos display, based on the programming language PHP and PHP-Framework Symfony, is implemented.

The, for this thesis implemented, software is focused on the installation and configuration of a virtual streaming cluster and offers advantages concerning existing solutions (Youtube, Metacafe and Megavideo), e.g. own privacy and security regulations, as well as valuable information from user inputs.

Kapitel 1

Einleitung

1.1 Allgemeines

In den letzten Jahren ist die Nachfrage und ebenso das Angebot von Anwendungen für Video-Streaming kontinuierlich gestiegen. Inzwischen hat YouTube [56], laut seinen eigenen Angaben, zwei Milliarden Videozugriffe pro Tag. Das bedeutet, es werden täglich mehrere Terabyte an Daten versendet. Transaktionen in diesem Größenverhältnis sind auf einem einzelnen Rechner nicht in akzeptabler Zeit möglich. Für die Verteilung der Daten verwendet YouTube mehrere physisch existierende Rechenzentren, die global über mehrere Kontinente verstreut sind.

Beim Betrieb einer vergleichbaren Infrastruktur mit physischen Rechnern, überwiegen die Fixkosten für Anschaffung und Wartung den Nutzen eines solchen Streaming-Clusters¹. Eine in Vergleich dazu kostengünstige Möglichkeit ist die Nutzung des öffentlich verfügbaren Infrastrukturdienstes Amazon Elastic Compute Cloud (EC2).

In einigen Anwendungsszenarien ist es sinnvoll, Videodaten nur bestimmten, von den Benutzern selbst festgelegten, Personenkreisen zugänglich zu machen. Youtube bietet an, dass ein eingestelltes Video vom Nutzer als „*privat*“ markiert und somit für andere Nutzer nicht erreichbar ist. Jedoch werden laut AGB von Videoportal-Betreibern, wie z.B. Youtube [52], Clipfish [14] und MyVideo [38], den Anbietern unentgeltliche und weitreichende Nutzungs- und Vervielfältigungsrechte, sowie das Recht der Übertragung dieser Rechte an Dritte, eingeräumt [17]. Das bedeutet, dass auch wenn ein Video als privat markiert wurde, ist es dennoch möglich, dass dieses Video ohne Einverständnis des Urhebers z. B. im Fernsehen oder in einer Werbung der Portalbetreiber erscheint.

¹Unter einem Cluster versteht man in der Informationstechnik (IT) ein Zusammenschluss mehrerer Rechner zu einer logischen Einheit

Ein weiterer Nachteil von YouTube ist, dass die Videos auf der Festplatte des Nutzers temporär gespeichert (*gecached*) werden. Diese Streaming-Technik ist unter dem Namen *Pseudo-Streaming* bekannt. Ein Vorteil von Pseudo-Streaming ist, dass zuvor geladene Videos auch ohne Verbindung zum Internet angesehen werden können. Ein Nachteil ist aber, dass dadurch eine einfachere Möglichkeit zum Datendiebstahl gegeben ist. Es existieren zahlreiche Programme und Browser-Erweiterungen, die den temporären Speicher des Clients auslesen, um zwischengespeicherte Videos permanent auf der Festplatte zu speichern.

Diese Diplomarbeit beschäftigt sich auch mit der Konzeption und Entwicklung einer Cloud-basierten Lösung für Video-Streaming, bei der eine kosteneffiziente Infrastruktur und eine sichere Webanwendung im Vordergrund steht. Ein Vorteil der realisierten Lösung ist, dass auf Zwischengespeichern auf Seiten der Clients verzichtet wird.

1.2 Aufbau der Arbeit

In Kapitel 2 wird das Cloud Computing definiert. Es werden verschiedene Cloud-Architekturen und Kategorien von Cloud-Diensten beschrieben. Der Schwerpunkt liegt auf den Infrastrukturdiensten as a Service (IaaS).

Im Kapitel 3 wird der existierende Streaming-Video-Dienst YouTube und dessen Infrastruktur evaluiert, um mit Hilfe dieser Erkenntnisse die Anforderungen für die zu entwickelnde Video-Streaming-Lösung zu entwickeln.

In Kapitel 4 enthält eine Evaluation der Infrastrukturdienste in den öffentlich verfügbaren Clouds, wobei zunächst der Infrastrukturdienst von Amazon untersucht wird, um daran anschließend die Anforderungen und Rahmenbedingungen für eine Video-Streaming-Infrastruktur zu erläutern. Schwerpunkt ist die Lastverteilung, um bei der Speicherung, dem Transport und der Verarbeitung von Videos die vorgegebenen Kapazitätsgrenzen einzuhalten.

In Kapitel 5 wird die Cloud-basierte Video-Streaming-Lösung entwickelt und implementiert. Zunächst werden Anforderungen und Designüberlegungen beschrieben und anschließend die Implementierung erläutert. Die Vor- und Nachteile verschiedener Lösungsansätze werden diskutiert, sowie Erweiterungsmöglichkeiten vorgestellt.

Kapitel 6 enthält die Zusammenfassung dieser Diplomarbeit. In diesem letzten Kapitel werden die unterschiedlichen Probleme und Lösungsansätze gegenübergestellt und bewertet. Abschließend folgt ein Ausblick.

Kapitel 2

Grundlagen

Inhalt dieses Kapitels ist die Definition des Begriffs Cloud Computing. Öffentlich verfügbare und private Cloud-Architekturen werden analysiert. Anschließend folgt ein Schwerpunkt zu den Infrastrukturdiensten und Softwarediensten.

2.1 Cloud Computing

Der Begriff „Cloud Computing“ erfährt aktuell in der Informationstechnik (IT) eine steigende Popularität. Laut Bundesministerium für Wirtschaft und Technologie (BMWi) [13] wird das Marktvolumen im Bereich Cloud Computing in den nächsten drei Jahren von 500 Millionen Euro um das vierfache auf über 2 Milliarden Euro ansteigen. Abbildung 2.1 zeigt das prognostizierte Marktvolumen für Cloud Computing in Deutschland in den Jahren 2010 bis 2013.

Laut [16] werden Unternehmen in Deutschland im Jahr 2010 ca. 386,5 Millionen Euro für Cloud-Dienste ausgeben haben. Bis 2012 wird mit einer jährlichen Wachstumsraten von fast 50 Prozent gerechnet.

Laut Techconsult, einem Marktforschungs- und Marketingconsultingunternehmen in Zentraleuropa, werden die Ausgaben für Cloud Computing und Outsourcing-Dienste von derzeit rund 25 Prozent auf 45 Prozent in 2011 in Deutschland wachsen [15].



Abbildung 2.1: Marktvolumen für Cloud Computing

2.2 Definition Cloud Computing

Obwohl es momentan für den Begriff „Cloud Computing“ noch keine einheitliche Definition gibt, deckt laut [12] folgende Definition die Hauptmerkmale des Cloud Computings ab:

„Unter Ausnutzung virtualisierter Rechen- und Speicherressourcen und moderner Web-Technologien stellt Cloud Computing skalierbare, netzwerk-zentrierte, abstrahierte IT-Infrastrukturen, Plattformen und Anwendungen als on-demand Dienste zur Verfügung. Die Abrechnung dieser Dienste erfolgt nutzungsabhängig.“

Eines der Haupteinsparfaktoren des Cloud Computings ist die Senkung des notwendigen Startkapitals, da durch Virtualisierung von IT-Infrastrukturen diese auf Abruf zur Verfügung stehen und nicht erst physisch beschafft werden müssen. Dadurch profitieren besonders kleine und mittlere Unternehmen, da diese nun die Möglichkeit haben, solche Technologien wirtschaftlich zu beziehen und einzusetzen [12].

Durch eine Kostenabrechnung je nach Verbrauch (Pay-as-you-go) ist es sowohl dem Dienstanbieter als auch den Dienstnutzern möglich, ihre Kosten zu minimieren.

Ein weiterer Einsparfaktor für Dienstanbieter ist die Reduzierung des Verwaltungsaufwands, da alle Ressourcen über eine zentrale Stelle konsolidiert und verwaltet werden. Somit werden die Wartungs- und Administrationskosten gegenüber dem klassischen Betrieb einer IT-Infrastruktur reduziert.

Zudem kann eine virtuelle IT-Infrastruktur im laufenden Betrieb mit weniger Aufwand erweitert, als das mit einer physischen Infrastruktur möglich wäre. Dadurch kann eine fast 100 prozentige Verfügbarkeit erreicht werden, was geringere Ausfallzeiten für den Dienstnutzer bedeutet und das Risiko, durch Ressourcenengpässe Kunden zu verlieren, reduziert.

2.3 Cloud-Architekturen

Es werden drei Arten von Cloud-Architekturen unterschieden. Es können Dienste in einer öffentlich verfügbaren Cloud (*Public Cloud*) oder einer privaten Cloud (*Private Cloud*) angeboten und erbracht werden. Die dritte Art ist die hybride Cloud (*Hybrid Cloud*), welche eine Mischform der öffentlichen und privaten Clouds darstellt.

Aus organisatorischer Sicht gehören bei öffentlich verfügbaren Angeboten Dienstanbieter und -nutzer unterschiedlichen Organisationen an. Ein Beispiel die Dienste der Amazon Web Services (AWS). Die Rechen- und Speicherressourcen werden hierbei extern von einem Dienstanbieter bereitgestellt. Für den Dienstnutzer besteht keine Notwendigkeit die Ressourcen selbst zu betreiben. Um auf die Ressourcen zugreifen zu können, muss der Dienstnutzer auf eine sichere Verbindung, wie SSH oder VPN, zurückgreifen. Der Kunde muss auch prüfen, ob die Nutzung von Ressourcen eines externen Dienstleisters im konkreten Anwendungsfall mit den Datenschutzbestimmungen des Dienstbieters vereinbar ist. Des Weiteren besteht für den Dienstnutzer ein Risiko bezüglich der Verfügbarkeit der Daten, der sogenannte Vendor-Lock-In. In diesem Fall ist eine Portierung der Daten und Dienste zu einem anderen Dienstanbieter nicht mehr ohne unverhältnismäßig hohe Kosten möglich. Es besteht auch das Risiko, z.B. bei einer Insolvenz des Dienstbieters, nicht mehr auf die eigenen Daten zugreifen zu können.

Bei privaten Clouds gehören Dienstanbieter und -nutzer ein und der selben Organisation an, was bedeutet, dass die Cloud-Infrastruktur auf den eigenen Ressourcen (In-House) betrieben wird. Hierbei werden die Cloud-Dienste von der eigenen Organisation erbracht und bleiben für gewöhnlich der breiten Öffentlichkeit verwehrt. Ein Vorteil privater Clouds ist der Sicherheitsaspekt, da kein Risiko hinsichtlich Datenschutz und Vendor-Lock-In besteht. Der Grad der Datensicherheit hängt von den physischen Ressourcen des Anbieters und deren Organisation ab. Die Transaktions- und Datenhaltungskosten fallen weg, jedoch kommen u. a. Kosten für Wartung, Strom und Stellfläche hinzu. Obwohl in einer privaten Cloud dadurch höhere Fixkosten anfallen, kann ein Unternehmen durch die Konsolidierung der Ressourcen und durch die dynamische Erweiterbarkeit der IT-Infrastruktur profitieren.

Das dritte Cloud-Architekturkonzept ist eine Mischung aus öffentlicher und privater Cloud. In diesem Fall spricht man von einer hybriden Cloud. Hierbei werden sowohl öffentliche als auch private Dienste in Anspruch genommen. Ein mögliches Szenario ist, dass die private Cloud zur Ablage sensibler Daten innerhalb des eigenen Netzwerks verwendet wird. Public Clouds können zur Abdeckung von Lastspitzen (Peaks) oder Verteilung von Sicherheitskopien wichtiger Daten eingesetzt werden. Die Voraussetzung ist aber, dass die API des privaten Infrastrukturdienstes (z.B. Opennebula [39]) mit der Schnittstelle der öffentlich verfügbaren Cloud (z.B. Amazon Web Services EC2-API) kompatibel ist. Die Kosten können je nach dem, welche öffentlichen und privaten Dienste genutzt werden variieren. Um die Sicherheit zu gewährleisten, ist es notwendig die Ressourcen der öffentlichen Cloud über ein Virtual Private Network (VPN) mit den Ressourcen der eigenen privaten Cloud zu verbinden. Risiken bezüglich Datenschutz und Vendor-Lock-Ins bestehen bei einer Hybrid Cloud nicht. Der Grad der Datensicherheit ist vom konkreten Aufbau der hybriden Cloud und der einzelnen Dienste abhängig.

2.4 Kategorien von Cloud-Diensten

Zu den Kategorien, in die Cloud-Dienste unterschieden werden können, gehört Infrastrukturdienste - Infrastructure as a Service (IaaS), Plattformdienste - Plattform as a Service (PaaS), Softwaredienste - Software as a Service (SaaS) und das sogenannte Crowdsourcing - Human as a Service (HaaS). Wie in Abbildung 2.2 zu sehen ist, sind diese Kategorien in der obigen Reihenfolge anhand der Abstraktion von der Hardware in Schichten angeordnet. Die Dienste der einzelnen Schichten sind voneinander unabhängig, können aber auch in beliebiger Kombination genutzt werden. Diese Schichten folgen dem Paradigma Everything-as-a-Service (XaaS). In den folgenden Abschnitten werden die Infrastrukturdienste und Softwaredienste beschrieben.

2.4.1 Softwaredienste

Softwaredienste sind Webanwendungen, die von Diensteanbietern betrieben und den Kunden zur Verfügung gestellt werden. Hierbei müssen sich die Kunden nicht um die Installationen, Administration, Software-Updates, etc. kümmern. Die Nutzung der Software ist entweder kostenlos oder wird nach Verbrauch abgerechnet. Die Kunden benötigen lediglich einen Webbrowser. Der Diensteanbieter ist für die Datenhaltung und die Verfügbarkeit der Dienste verantwortlich. Für die Kunden besteht das Risiko des Vendor-Lock-Ins und auch Risiken bezüglich des Datenschutzes. Beispiele für öffentlich verfügbare Softwaredienste sind z.B. Google Docs [23] und Salesforce.com [41].

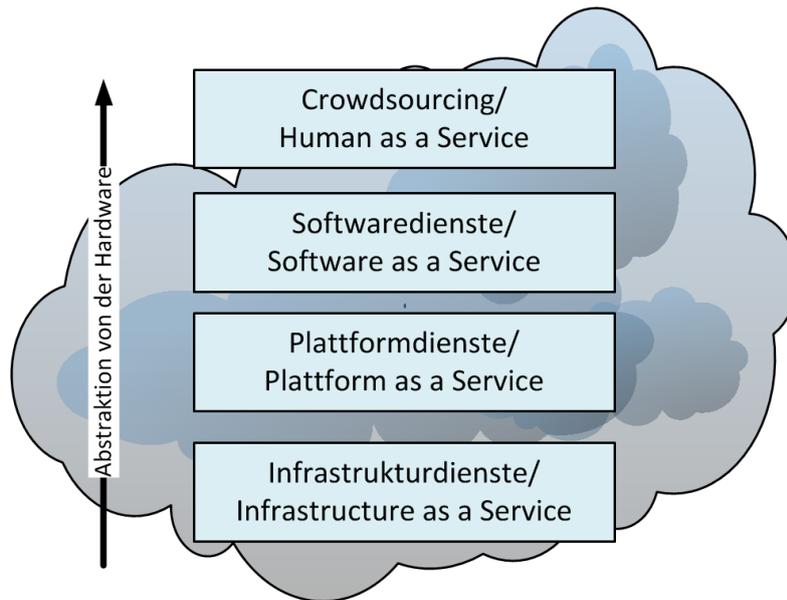


Abbildung 2.2: Schichtenmodell der Kategorien von Cloud-Diensten

2.4.2 Infrastrukturdienste

Infrastrukturdienste bieten dem Kunden eine abstrahierte Schicht auf die Hardware. Zum Angebot solcher Dienste gehören virtuelle Rechner, Massenspeicher, Netzwerke etc. Der Kunde kann über eine Benutzerschnittstelle auf die Ressourcen der Infrastruktur zugreifen. Der Kunde kann selbst entscheiden, welche Betriebssysteme er verwendet. Der Kunde startet über eine Benutzerschnittstelle oder geeignete Werkzeuge Instanzen virtueller Maschinen, die in entfernt liegenden Rechenzentren des Anbieters laufen.

Es existieren verschiedene Instanztypen, die vom Dienstanbieter vorgegeben sind. Diese unterscheiden sich unter anderem in der Größe des Arbeitsspeichers, der Anzahl der Prozessoren und Prozessorkerne, sowie in der Größe der Festplatte. Ein Kunde behält die vollständige Kontrolle über seine Instanzen und ist somit auch für die Sicherheit der Maschinen selbst verantwortlich. Es existiert eine Firewall, deren Regelwerk jeder Kunde selbstständig an seine Bedürfnisse anpassen kann. Ein Beispiel für einen öffentlich verfügbaren Infrastrukturdienst ist die Elastic Compute Cloud (EC2) von Amazon. Beispiele für Lösungen zum Betrieb privater Infrastrukturdienste sind z.B. Opennebula [39] und Eucalyptus [20]. Im folgenden Abschnitt wird EC2 vorgestellt.

Amazon Elastic Compute Cloud (EC2)

Ein Dienst der Amazon Web Services ist der Infrastrukturdienst EC2. EC2 ermöglicht dem Kunden den Betrieb virtueller Rechner, die ihren physischen Standort in einer der fünf *Regionen* haben. Diese sind wiederum in sogenannte Verfügbarkeitszonen (*Availability Zones*) unterteilt. Jede Verfügbarkeitszone stellt einen in sich abgeschlossenen Cluster dar. Es existieren fünf Regionen, davon zwei in den USA, eine in Irland und jeweils eine Region in Indonesien (Singapur) und Japan (Tokio). Die Preise für Ressourcen können je nach Region variieren. Jede Region enthält ein bis vier Verfügbarkeitszonen.

Ein Kunde kann über die Benutzerschnittstelle oder geeignete Werkzeuge auf die Ressourcen in EC2 zugreifen. Abbildung 2.2 zeigt die Webanwendung *Management Console* von Amazon.

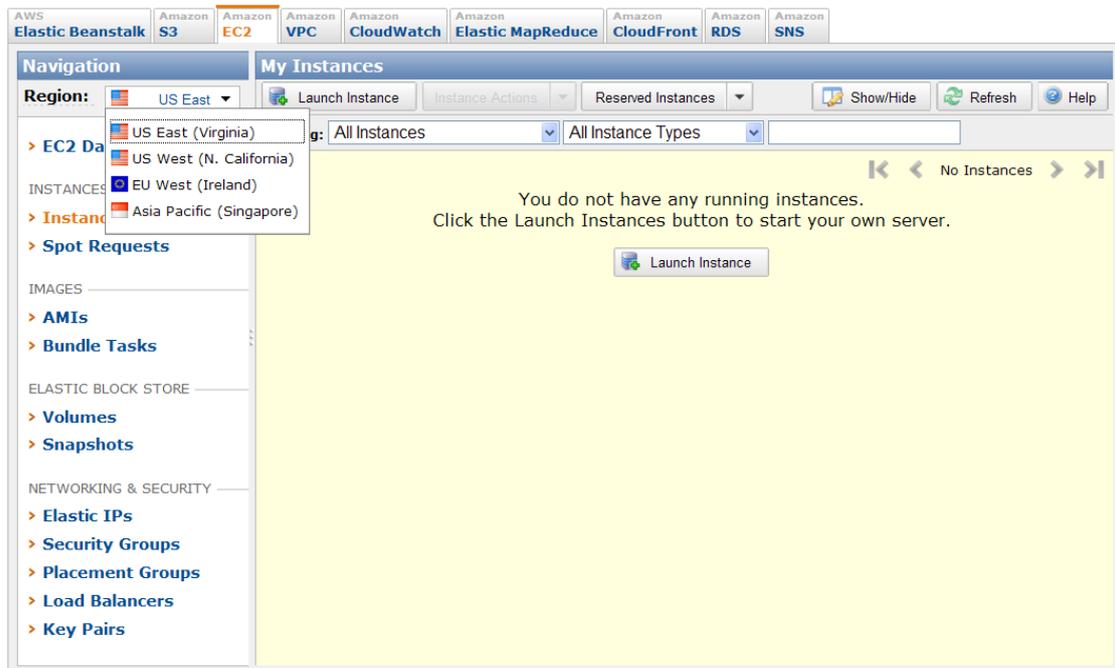


Abbildung 2.3: Amazon Management Console

Jede Instanz in EC2 besitzt einen internen Speicher, dessen Größe je nach Instanztyp variiert. Zustandsänderungen in diesem Speicher sind nicht dauerhaft, da alle Änderungen bei der Terminierung der Instanz verloren gehen. Für die persistente Datenspeicherung ist es möglich, mit dem Dienst Amazon Elastic Block Store (EBS) [4] virtuelle Datenträger zur Speicherung auf Blockebene, eigenen Instanz in EC2 zuzuweisen. Diese sogenannten EBS-Volumen werden wie externe Festplatten verwendet. Es ist zwar möglich, an eine Instanz mehrere Speichervolumen anzuhängen, jedoch ist es nicht

möglich, ein Volumen gleichzeitig mehreren Instanzen zuzuweisen. Die mögliche Speicherkapazität eines EBS-Volumens liegt zwischen 1 GB und 1 TB. Zusätzlich ist es möglich, nach Bedarf sogenannte *Snapshots* zu erstellen. Snapshots sind Momentaufnahmen von Speichervolumen und dienen u. a. als Backup, als auch als Aufhängepunkt für neue Speichervolumen.

Amazon Simple Storage Service (S3) [9] ist ein weiterer Speicherdienst der AWS. Er ermöglicht die Speicherung großer Mengen schwach strukturierter objektbasierter Daten in einem flachen Speicherraum. Mit S3 ist es möglich, hochverfügbare und nahezu unbegrenzte Speicherressourcen als Dienst zu nutzen.

Der Amazon Elastic Load Balancer (ELB) [7] ist ein weiterer Dienst der AWS. Dieser Dienst ermöglicht den Einsatz elastischer Lastverteiler. Zudem hilft der ELB, Instanzen in EC2 zu überwachen. Fällt eine Instanz aus, bemerkt der ELB dieses und leitet den eingehenden Datentransfer zu anderen noch funktionsfähigen Instanzen weiter. Sobald eine ausgefallene EC2-Instanz wieder verfügbar ist, diese wieder vom ELB bedient.

Der Dienst Amazon CloudFront ermöglicht die schnelle Übermittlung von Inhalten, vor allem von Daten in S3. Ein solcher Dienst wird auch *Content Distribution Network* (CDN) genannt [3]. Mit CloudFront können statische Inhalte und Datenströme über ein globales Netzwerk aus sogenannten *Edge-Standorten* bereitgestellt werden. Download- oder Streaming-Anfragen werden automatisch an den nächsten Edge-Standort durchgereicht, sodass die Bereitstellung der Inhalte mit einer der hohen Leistung erfolgen kann.

Amazon SimpleDB [5] ist ein Dienst für ein einfach verteiltes Datenbankmanagementsystem von Amazon. Der Dienst ist nicht für komplexe Datenbankschemata oder transaktionale Eigenschaften gedacht. Vielmehr dient er als schneller Einstieg in eine einfach strukturierte, aber zuverlässige Datenhaltung. Für komplexe (relationale) Datenbankschemata wird auf die Möglichkeit verwiesen eine solche Datenbank entweder auf ein EBS-Speichervolumen zu installieren oder auf den Dienst Amazon Relational Database Service (RDS) [8] zurückzugreifen.

RDS [8] ist ein Dienst der AWS zum Einrichten, Betreiben und Skalieren relationaler Datenbanken. Das automatische Anlegen von Sicherheitskopien der Datenbanken geschieht in RDS transparent. Die Aufbewahrungsdauer der Sicherheitskopien können die Kunden selbst bestimmen. RDS bietet vollen Zugriff auf den Funktionsumfang einer MySQL-Datenbank [37]. Das bedeutet, dass der Dienstnutzer seine bestehenden MySQL-Datenbanken, nahtlos in RDS integrieren kann. Ein weiterer Vorteil ist die Skalierbarkeit der Instanzen, die die relationalen MySQL-Datenbanken enthalten.

Der Amazon Simple Queue Service (SQS) [10] ist ein Dienst zur Realisierung von Nachrichtenwarteschlangen, um Daten zwischen Instanzen in EC2 austauschen zu können.

Mit SQS können Daten zwischen den verteilten Komponenten einer Anwendung verschoben werden. Dadurch lassen sich z.B. kritische Komponenten auf mehreren EC2-Instanzen parallel betreiben, wodurch sich Engpässe auch schon zur Laufzeit besser behoben werden können.

Bei allen vorgestellten Diensten der AWS fallen keine Vorabkosten an. Die Kunden zahlen nur für die Ressourcen, die sie tatsächlich nutzen.

Kapitel 3

YouTube

Dieses Kapitel enthält eine Untersuchung von Googles Dienst YouTube [52]. Zunächst wird auf die Entstehungsgeschichte sowie die angebotenen Leistungen analysiert. Anschließend wird der aktuelle Stand des Dienstes und die von YouTube verwendete Technik beschrieben. Anschließend werden die Vor- und Nachteile von YouTube diskutiert.

3.1 Geschichte von YouTube

YouTube ist ein Internetportal, das im Februar 2005 von den drei ehemaligen Paypal-Mitarbeitern Chad Hurley, Steve Chen und Jawed Karim gegründet wurde. Auf der Website können Nutzern Videos ansehen und hochladen. Am 9. Oktober 2006 gab Google die Übernahme von YouTube für ca. 1,31 Milliarden Euro in Aktien bekannt [50].

3.2 Funktionalität von YouTube

YouTube ist ein Internetportal, über das die Benutzer Video ansehen und hochladen können. Zudem bietet YouTube kostenlos Film- und Fernsehausschnitte, Musikvideos sowie selbstgedrehte Filme an. Die Videos können als sogenannte *Video-Feeds* in Blogs gepostet oder in Webseiten über eine Programmierschnittstelle eingebunden werden. Von Benutzern als anstößig gemeldete Videos werden von den Mitarbeitern von YouTube zunächst überprüft und anschließend eventuell entfernt. Laut [50] werden solche Videos spätestens nach Eingang einer Abmahnung bezüglich des betreffenden Videos gelöscht. Abbildung 3.1 zeigt die Startseite von YouTube.

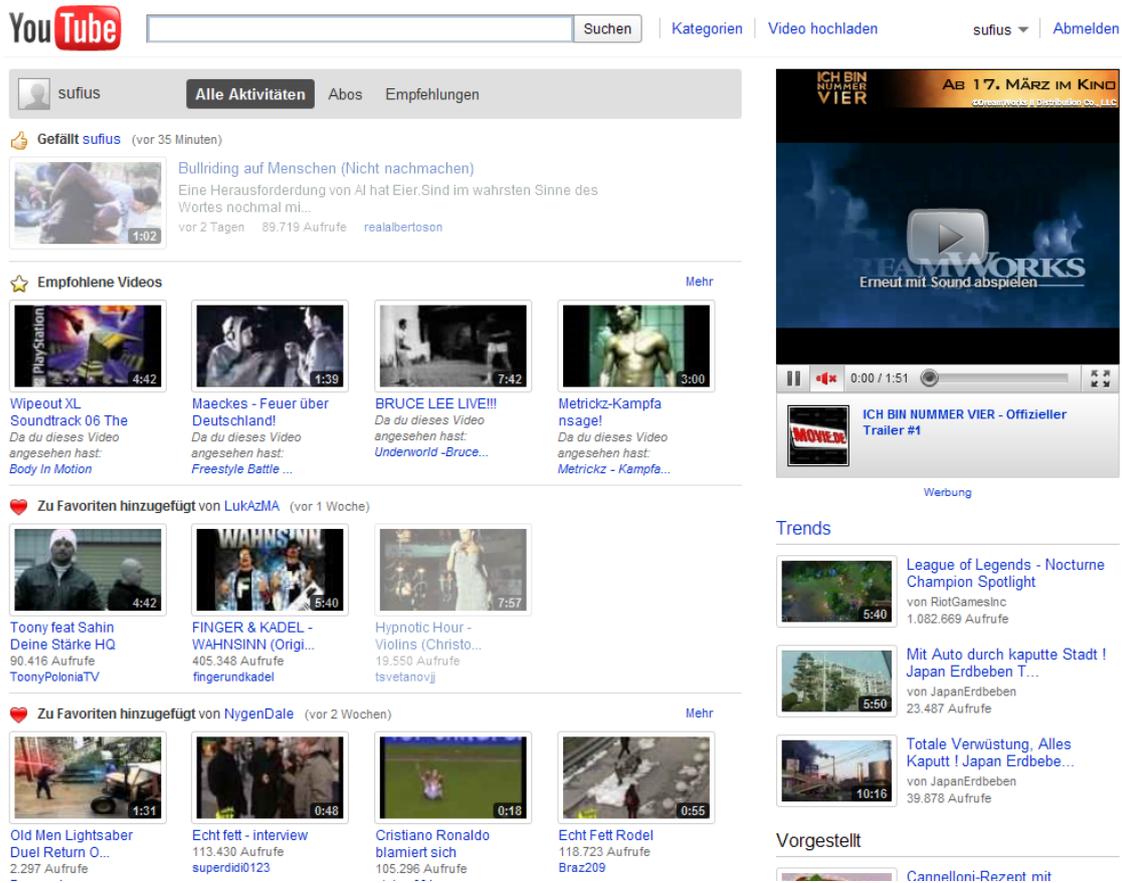


Abbildung 3.1: Startseite von YouTube.com

3.3 Erfolg von YouTube

Der Erfolg von YouTube lässt sich anhand der ständig wachsenden Anzahl an Benutzern, die Videos hochladen, bewerten und kommentieren, ablesen. YouTube hat sich seit seiner Gründung im Jahr 2005 zum führenden Videoportal im Internet etabliert. So wurden laut [26] im Jahr 2006, nur 1,5 Jahre nach der Gründung bereits ca. 65.000 neue Videos täglich hochgeladen und 100 Millionen Clips jeden Tag angesehen. Im Oktober 2009 wurden laut [54] bereits über eine Milliarde Videoabrufe pro Tag verzeichnet. Im Mai 2010 wurden laut [55] bereits 2 Milliarden Videos pro Tag angesehen. Im November 2010 wurden pro Minute über 35 Stunden Videomaterial auf die Plattform geladen [26].

Laut einer Studie des Marktforschungsinstituts Hitwise [28] von März 2008 werden YouTube 73 Prozent aller Besuche von US-Videoportalen zugeordnet [26]. Laut [29] geht man derzeit¹ von einem Marktanteil von etwa 43 Prozent aus, wodurch YouTube zum populärsten Dienst dieser Art zählt.

¹Juni 2010

Laut den Berechnungen von Ellacoya Networks [19] ist YouTube für 10 Prozent des gesamten Internet-Datenverkehrs und 20 Prozent des HTTP-Aufkommens [57] verantwortlich.

3.4 Technik von YouTube

Die Video-Dateien werden im Dateiformat Flash-Video (*.FLV) an die Benutzer verteilt. Die Benutzer können die Videos online als Stream im Webbrowser mit Hilfe des Flash-Browser-Plugins von Adobe anschauen. YouTube unterstützt für das Hochladen von Videos alle gängigen Formate, wie z.B. MP4, H.264, MPEG, AVI, WMV, Quicktime und FLV selbst. Die maximale Video-Dateigröße beträgt 2 Gigabyte und die maximale Wiedergabezeit pro Video beträgt 15 Minuten. Vor der Weiterverbreitung von YouTube werden die hochgeladenen Video-Dateien in das Flash-Video-Dateiformat konvertiert.

Als Webserver verwendet YouTube für Bilder, Videos und andere statische Inhalte eine modifizierte Version von Lighttpd mit verbesserter Lastverteilung [27]. Für die dynamischen Inhalte, wie z.B. Kommentare und Bewertungen, verwendet YouTube einen Apache-Webserver.

Zusätzlich werden inzwischen laut [50] die hochgeladenen Videos auch in das Dateiformat H.264 konvertiert, um diese auch auf dem Apple iPhone wiedergeben zu können [4].

Die Oberfläche von YouTube ist in der Programmiersprache Python implementiert. Als Datenbanksystem wird YouTube MySQL [27] verwendet.

YouTube verlagert die Videos, die häufig abgerufen werden, in ein CDN [27].

3.5 Einschränkungen von YouTube

Rechte, die YouTube eingeräumt wird

Laut den AGB von YouTube [51] räumt man mit dem Hochladen eines Videos ...

„YouTube eine weltweite, nicht-exklusive und gebührenfreie Lizenz ein (mit dem Recht der Unterlizenzierung) bezüglich der Nutzung, der Reproduktion, dem Vertrieb, der Herstellung derivativer Werke, der Ausstellung und der Aufführung der Nutzerübermittlung im Zusammenhang mit dem Zur-Verfügung-Stellen der Dienste und anderweitig im Zusammenhang mit dem Zur-Verfügung-Stellen der Webseite und YouTubes Geschäften, einschließlich, aber ohne Beschränkung auf Werbung

für und den Weitervertrieb der ganzen oder von Teilen der Webseite (und auf ihr basierender derivativer Werke) in gleich welchem Medienformat und gleich über welche Verbreitungswege. ...“

Das heißt, dass YouTube das Recht hat, hochgeladene Videos der Benutzer selbst zu verwenden (z.B. im öffentlichen Fernsehen), zu vervielfältigen und diese Rechte an Dritte weiterzugeben. Diese Rechte werden YouTube unbefristet und unwiderruflich auch für alle Nutzerkommentare eingeräumt.

Videos mit fragwürdigem Inhalt

Laut den Nutzungsbedingungen von YouTube ist es nicht erlaubt, Videos mit rassistischem und/oder ethnisch diskriminierendem Inhalt hochzuladen. Dennoch tauchen immer wieder solche Videos bei YouTube auf. Obwohl diese von Zuschauern als unangebracht deklariert wurden, werden sie nicht gelöscht, sondern lediglich nur noch für registrierte Nutzer zugänglich gemacht. Ein weiterer Kritikpunkt ist die nicht ausreichende Altersverifizierung. Hierbei stößt YouTube vor allem bei Jugendschützern und den deutschen Medien auf Kritik [32].

Urheberrechtsverletzungen

Beim Hochladen neuer Videos kommt es vielfach zu Urheberrechtsverletzungen. Laut [50] gehen einige Experten davon aus, dass dies bis zu 90 Prozent des hochgeladenen Materials betrifft. Nach der geltenden amerikanischen Rechtsprechung muss YouTube solche Inhalte erst nach einer Abmahnung durch die Rechteinhaber löschen (Opt-Out-Verfahren). Laut [50] verbot das Landgericht Hamburg in einem Urteil vom 3. September 2010 die Verbreitung urheberrechtswidriger Inhalte über YouTube [33]. Parallel dazu konnte sich laut [50] YouTube vor einem spanischen Gericht in Madrid bei einer ähnlichen Klage gegen den Fernsehsender Telecinco durchsetzen. Google feierte dies indes als „Sieg für das Internet“ [25].

Authentizität der Inhalte

Ähnlich wie bei anderen Internet-Diensten mit Social-Networking-Charakter, z.B. studivz.net [45] oder facebook.com [21], wird YouTube zunehmend als Plattform für Guerilla-Marketing genutzt. Daher ist die Authentizität von Inhalten häufig nur schwer zu beurteilen. In einigen Fällen wurden Videos gezielt von Medienagenturen produziert, jedoch mit dem Anschein verbreitet, von Privatpersonen hergestellt und veröffentlicht

worden zu sein. Ein Beispiel dafür ist laut [50] die Verbreitung eines politischen Videos, das sich kritisch mit dem ehemaligen US-amerikanischen Präsidentschaftskandidaten Al Gore und seinem Engagement für eine Reduzierung des Ausstoßes von Treibhausgasen auseinandersetzte.

Kapitel 4

Entwicklung der Infrastruktur für das Video-Streaming

In diesem Kapitel erfolgt eine Analyse der Anforderungen an eine Video-Streaming-Infrastruktur. Anschließend werden diese Anforderungen mit den Infrastrukturdiensten in der öffentlich verfügbaren Cloud der Amazon Web Services umgesetzt.

4.1 Anforderungsanalyse

Architektur- und Designüberlegung

Die Architektur ist nach dem Prinzip der Drei-Schichten-Architektur zu strukturieren. Abbildung 4.1 zeigt den Aufbau einer typischen Drei-Schichten-Architektur. Ein Vorteil dieses Architekturtypes ist die Entkoppelung der Logik von der Datenhaltung, wodurch eine bessere Skalierbarkeit der Anwendung möglich ist.

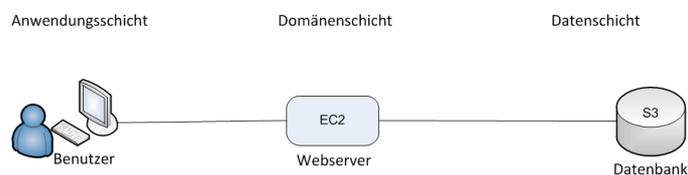


Abbildung 4.1: Aufbau einer typischen Drei-Schichten-Architektur

Die Anwendungsschicht stellt die grafische Benutzeroberfläche und die Repräsentation der Daten, in diesem Falle das Abspielen der Videos, dar. Zudem ist die Anwendungsschicht für die Benutzereingaben verantwortlich und dient als Schnittstelle zur

Domänen- und Datenhaltungsschicht. In Kapitel 5 erfolgt eine Analyse der Anwendungsschicht.

Die Domänenschicht ist die Schnittstelle für den Zugriff auf die Datenhaltungsschicht. Die Realisierung der Domänenschicht erfolgt mit EC2. Mit diesem Dienst wird der nötige Webserver realisiert. Die Daten der Webanwendung selbst liegen in einem persistenten EBS-Speichervolumen.

Die Datenhaltungsschicht basiert auf den beiden Diensten S3 und CloudFront. Das heißt, dass der Streaming-Server und die Datenbank in der Datenhaltungsschicht zusammengefasst sind. Während S3 ist für die Speicherung der Videos verantwortlich ist, ermöglicht CloudFront den Download und das Streaming der Videos in kurzer Zeit. Zudem nutzt die Datenhaltungsschicht RDS. Dieser Datenbankdienst dient nicht der Speicherung der Videos, sondern von deren Metadaten. Diese werden als Datensätze in einer MySQL-Datenbank abgelegt. Abbildung 4.2 zeigt den Aufbau der Infrastruktur und den Einsatz der AWS in der Drei-Schichten-Architektur.

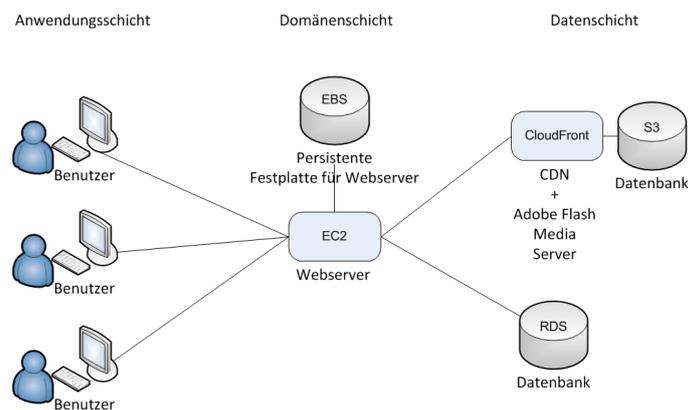


Abbildung 4.2: Nutzung der AWS im Rahmen einer Drei-Schichten-Architektur

Anforderung 1: Skalierbarer Webserver

Zunächst ist eine grundlegende Architektur zu erstellen. Der integrale Bestandteil dieser Architektur ist der Webserver. Die Ressourcenausstattung des virtuellen Servers ist so zu wählen, dass die Video-Streaming-Funktionalität erbracht werden kann. Da es sich bei Streaming, um eine Anwendung, die auch von einem einzelnen Rechner übernommen werden kann, muss sich der Aufbau am Prinzip eines High-Throughput-Clusters orientieren. Das bedeutet, dass vor allem der Arbeitsspeicher, die Durchsatzrate des verwendeten Netzwerks zwischen den Ressourcen und eine leichtgewichtige Webserver-Software wichtige Kriterien für die Datenübertragung darstellen.

Anforderung 2: Sicherheit

Ein Anforderungspunkt ist der Sicherheitsaspekt. Alle gängigen Sicherheitsmaßnahmen, wie z.B. das Einrichten von Firewalls und die Möglichkeit des Zugriffs auf Instanzen ausschließlich über sichere SSH-Verbindungen.

Anforderung 3: Skalierbare Datenbank

Des Weiteren ist ein dedizierter Datenbank-Server nötig. Dieser speichert die Metadaten der Videos, wie z.B. Dateiname, Erstellungsdatum, Anzahl der Aufrufe, Bewertungspunkte, etc. Ebenso wie der Webserver ist der Datenbank-Server so einzurichten, dass dieser bei Bedarf seine Speicher- und Rechenkapazität skalieren kann. Zusätzlich soll der Datenbank-Server redundant ausgelegt sein, um eine hohe Verfügbarkeit zu gewährleisten. Die automatische Generierung von Sicherheitskopien ist eine weitere geforderte Fähigkeit des Datenbank-Servers.

Anforderung 4: Praktisch unbegrenzter und hochverfügbarer Datenspeicher

Der Ablageort für die Dateien soll Engpässe oder Datenverlust prinzipiell ausschließen. Datenobjekte von S3 sind auf mehreren physischen Speicherpunkten an verschiedenen Standorten einer Region von S3 redundant abzulegen.

Anforderung 5: Streaming und Verteilung

Eine Wiedergabe der Videos soll, anders als beim gewöhnlichen Download von Dateien, schon während der Übertragung möglich sein. Um Audio-, Video- und sonstige Daten über das Internet von einem Server zu einem Client mit Adobe Flash zu übertragen, ist statt des Netzwerkprotokolls HTTP ein Streaming-Protokoll, wie z. B. das proprietäre Real Time Messaging Protocol (RTMP) von Adobe oder das Real Time Streaming Protocol (RTSP), zu verwenden. Der Server kodiert die Videodateien überträgt die Datenpakete. Gängige Lösungen zur Kodierung von Videos sind beispielsweise der Adobe Media Encoder und der Adobe Live Media Encoder für das Einbinden von Live-Streams [1]. Die kodierten Videos werden über das Flash-Plugin im Browser des Benutzers angezeigt. Es ist wichtig, dass die eingesetzte Client-Software eines der gängigen Streaming-Protokolle, z.B. das RTMP-Protokoll, unterstützt. Ein entscheidendes Kriterium beim Video-Streaming ist die Netzwerkbandbreite zwischen Anbieter und Benutzer. Ebenso ist die Entfernung zu den Rechenzentren zu berücksichtigen, da die Übertragungszeit (Latenzzeit) mit wachsender Entfernung zunimmt.

Anforderung 6: Redundante Lastverteilung

Ein weiterer Anforderungspunkt ist das Einrichten eines Servers zur Lastverteilung (Loadbalancer), der den eingehenden Datenverkehr weiter verteilt und die ständige Verfügbarkeit der Server kontrolliert. Fallen Server-Instanzen aus, müssen diese identifiziert und der eingehende Datentransfer auf die noch verfügbaren Server umgeleitet werden. Ausgefallene Server sollen nach Möglichkeit schnellstmöglich zu reaktiviert werden können. Zudem muss der Lastverteiler selbst überwacht werden, da dieser für die Weiterleitung des gesamten eingehenden Datenverkehrs verantwortlich ist. Beim Ausfall dieser Komponente sind alle dahinter liegenden Server nicht mehr erreichbar. Dadurch stellt der Lastverteiler einen Single Point of Failure (SPOF) für das gesamte System dar. Zur Überwachung der Lastverteiler soll das etablierte Heatbeat-Verfahren in einer Aktiv/Passiv-Konfiguration eingesetzt werden.

4.2 Evaluation der entwickelten Lösung in EC2

Im folgenden Abschnitt wird anhand von Amazons öffentlich verfügbaren Infrastruktur EC2 eine Video-Streaming-Infrastruktur aufgebaut. Zunächst wird für jede Anforderung der jeweils passende Dienst evaluiert. Anschließend werden die Komponenten verknüpft. Anschließend findet eine Kalkulation der laufenden Kosten statt. In die Auswertung fließen ebenso Faktoren, wie Bandbreite, Datenschutz und Datensicherheit, sowie der administrative Aufwand.

Vorbereitung

Im ersten Schritt muss sich der Dienstanutzer bei den Amazon Web Services registrieren. Voraussetzung für die Nutzung der Dienste ist der Besitz einer gültigen Kreditkarte. Jeder gewünschte Dienst muss einzeln freigeschaltet werden. Elastische Lastverteiler und elastische IP-Adressen sind genau so ein Teil von EC2 wie die Schlüsselpaare, Sicherheitsgruppen und Snapshots.

Als nächsten Schritt muss ein Benutzer sich für eine der fünf Regionen in EC2 entscheiden. Die Regionen unterscheiden sich unter anderem in den anfallenden Kosten. Die preisgünstigste Region ist die Region US-East in Virginia. Während Virginia die günstigere Alternative wäre, liegen andere Regionen wie z.B. EU-West in Irland geographisch näher an Deutschland, was zu kürzeren Latenzzeiten führt. Die Preise für Instanzen pro Stunde variieren je nach Region um bis zu 10 Prozent.

Skalierbarer Webserver

Für die Erfüllung von Anforderung 1 aus Kapitel 4 wird EC2 verwendet. In EC2 kann der Dienstnutzer virtuelle Rechner Instanzen starten. Die Instanzen laufen in den Rechenzentren des Anbieters. Der Dienstanbieter stellt vorgefertigte Images, sogenannte *Amazon Machine Images* (AMI), bereit. Diese AMIs unterscheiden sich in Bezug auf das Betriebssystem und die installierte Software. Amazon bietet AMIs für verschiedene Unix-Derivate und Windows-Betriebssysteme.

Ein einfacher Weg für den Dienstnutzer, eine virtuelle Maschinen Instanzen zu starten, ist über die *Amazon Management Console*. Abbildung 4.3 zeigt den EC2-Menüpunkt der Amazon Management Console und die Möglichkeit, die Region auszuwählen.

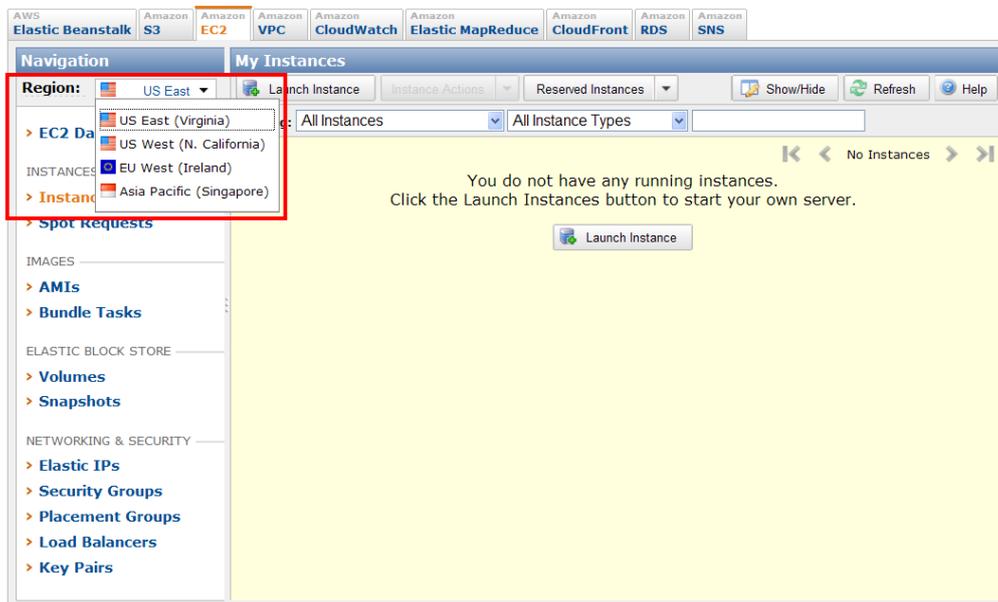


Abbildung 4.3: Startpunkt für neue Instanzen

Es existieren unterschiedliche Instanztypen, die der Anbieter Amazon vorgibt. Diese unterscheiden sich unter anderem in der Größe des Arbeitsspeichers, der Anzahl der Prozessoren und in der Festplattenkapazität. In Abbildung 4.4 sind die Instanztypen für 64-Bit-Instanzen in EC2 zu sehen.

Sicherheit

Aus Sicherheitsgründen sind neue Instanzen standardmäßig nicht über das Internet erreichbar. Um den Zugriff via SSH das vorherige Anpassen einer existierenden oder Anlegen einer neuen Sicherheitsgruppe nötig. Abbildung 4.5 zeigt die freigegebenen Portnummern einer Sicherheitsgruppe.

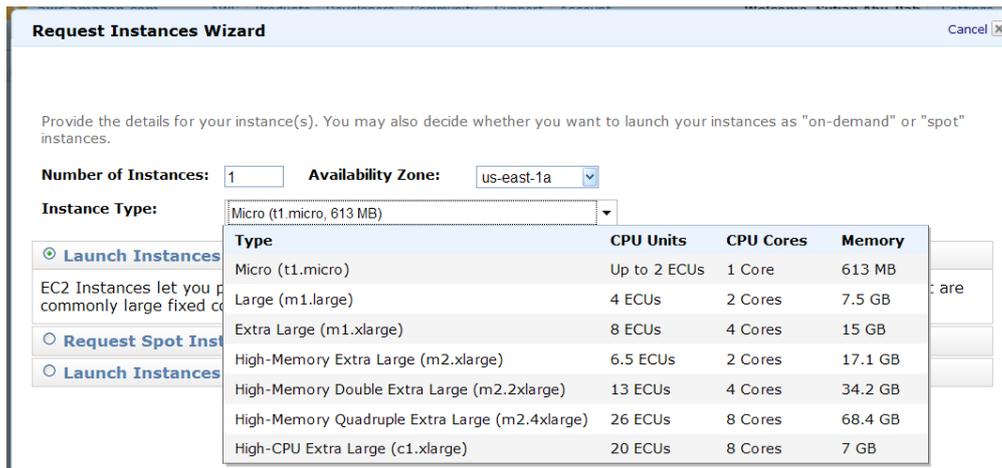


Abbildung 4.4: Instanztypen in EC2

Mit Hilfe eines Schlüsselpaares können die Benutzer sich ohne Passwort bei ihren Instanzen anmelden. Amazon bietet einen Generator für die Erstellung neuer Schlüsselpaare an. Bei Anlegen einer Instanz kann ein neues Schlüsselpaar erstellt oder ein existierendes ausgewählt werden. Abbildung 4.6 zeigt die Auswahlmöglichkeit der Schlüsselpaare.

Anschließend wählt der Dienstonutzer beim Anlegen einer neuen Instanz die entsprechende Sicherheitsgruppe aus. Sobald die Instanz läuft, ist es möglich, sich via SSH bei dieser anzumelden und beliebige Software-Pakete zu installieren und Kommandos auszuführen. Nach den nötigen Änderungen ist aus der laufenden Instanz ein neues AMI zu erstellen. Dieses AMI dient als Vorlage für weitere Instanzen, die zu Skalierungszwecken gestartet werden können.

Skalierbare Datenbank

Der Dienst Amazon Relational Database Services (RDS) bietet den Funktionsumfang einer MySQL-Datenbank. Das bedeutet, dass der Dienstonutzer eine bestehende MySQL-Datenbank, nahtlos in RDS integrieren kann. Abbildung 4.7 zeigt das Fenster in der Amazon Management Console.

Die für die Video-Infrastruktur nötige RDS-Datenbank muss in der gleichen Region in EC2 liegen, wie die zuvor erstellte Instanz.

Ein Vorteil von RDS ist die flexible Skalierung der Ressourcen für die Instanz mit der relationalen Datenbank. Hierfür stellt RDS dem Dienstonutzer eine API zur Verfügung. Der folgende Befehl dient zur sofortigen Skalierung der Speicherkapazität der Datenbankinstanz „*mydbinstance*“ auf 1024 Gigabyte.

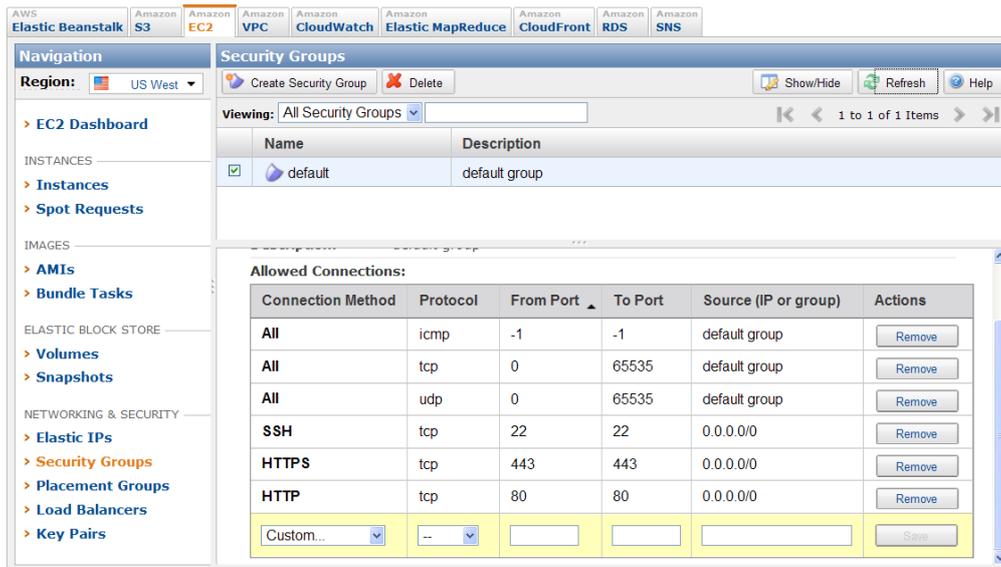


Abbildung 4.5: Freigegebene Portnummern einer Sicherheitsgruppe

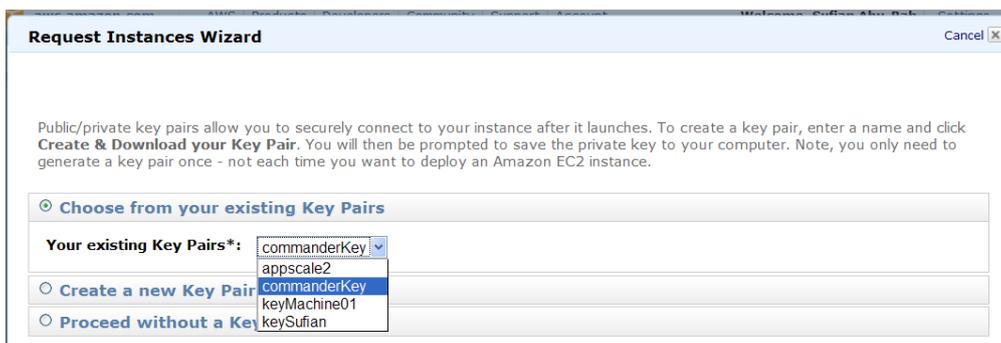


Abbildung 4.6: Auswahlmöglichkeit der zuvor erstellten Schlüsselpaare

rds-modify-db-instance mydbinstance -s 1024

RDS bietet zusätzlich mit der Option „Multi-AZ“ die Möglichkeit, der aktiven Datenbankinstanz eine weitere passive Datenbankinstanz hinzuzufügen. Diese wird permanent mit der aktiven Datenbankinstanz synchronisiert. Der Name Multi-AZ steht für die Verwendung unterschiedlicher Verfügbarkeitszonen für beide Datenbankinstanzen. Diese passive Datenbankinstanz führt bei Ausfall der aktiven Datenbankinstanz deren Arbeit fort, wodurch die Anforderung einer redundanten Datenbank erfüllt ist. Der Nachteil der Multi-AZ-Instanz sind doppelt so hohe laufende Kosten für den Dienstnutzer. Abbildung 4.8 zeigt die allgemeinen Einstellungsmöglichkeiten für die Datenbankinstanz in der Amazon Management Console.

Ein wichtiger Faktor bei der Erstellung der Datenbankinstanz ist die geographische Nähe zu den Anwendungsservern. Falls die Datenbankinstanz in einer anderen Region als

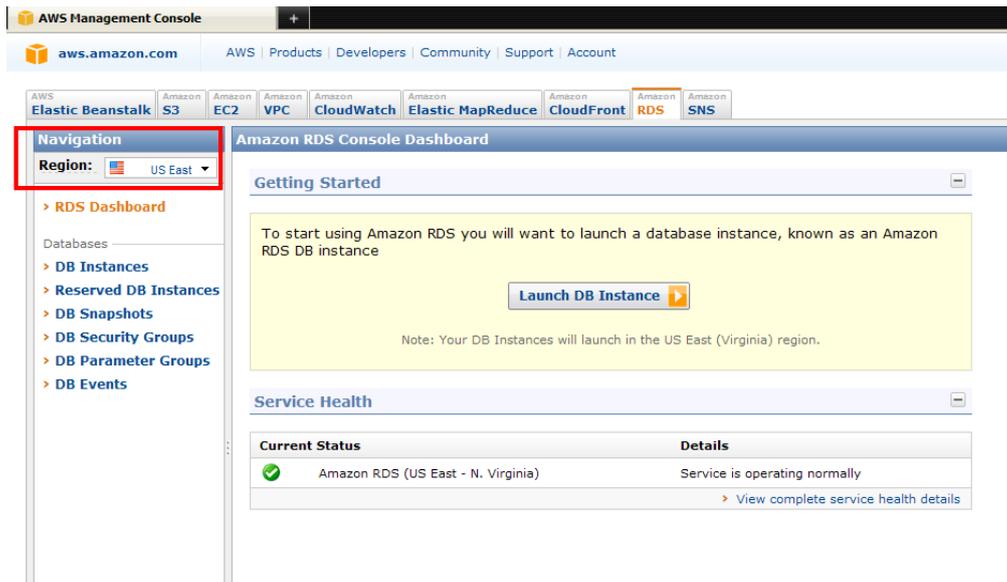


Abbildung 4.7: RDS-Fenster in der Amazon Management Console

die zuerst gestartete Instanz aufgesetzt wird, fallen höhere Kosten an, da laut [8] eine überregionale Datenübertragung auf beiden Seiten, einmal als eingehenden und einmal als ausgehender Datentransfer, berechnet wird. Zusätzlich erledigt RDS verschiedene Datenbank-Verwaltungsaufgaben für den Dienstnutzer, wie transparent das automatische Einspielen von Patches der Datenbank-Software, sowie das automatische Anlegen von Sicherheitskopien der Datenbank. Die Aufbewahrungsdauer dieser Sicherheitskopien kann der Kunde selbst bestimmen. Abbildung 4.9 zeigt die Einstellungsmöglichkeiten für das automatische Anlegen von Sicherheitskopien.

Es ist darauf zu achten, dass die RDS-Sicherheitsgruppe der RDS-Instanz an die Sicherheitsgruppe der zuvor gestarteten Instanz angepasst ist. Es werden nur die für den Datenbankzugriff nötigen Ports freigegeben. Hierdurch wird laut [24] die Sicherheit der Datenbank erhöht, da ein Einbrecher zuerst in die EC2-Instanz eindringen muss, um anschließend in die Datenbank eindringen zu können. Abbildung 4.10 zeigt die Einstellungen der Sicherheitsgruppe der RDS-Instanz.

Unbegrenzte und hochverfügbare Datenhaltung

Die Anforderung aus Kapitel 4.1.4 kann mit Amazons Speicherdienst Simple Storage Service (S3) erfüllt werden. Hierfür ist es zunächst nötig ein sogenanntes *Bucket* anzulegen. Neben dem Namen des Buckets, sind dessen Zugriffsrechte festzulegen. Abbildung 4.11 zeigt das S3-Fenster in der Amazon Management Console.

Launch DB Instance Wizard Cancel X

DB INSTANCE DETAILS ADDITIONAL CONFIGURATION MANAGEMENT OPTIONS REVIEW

To get started, choose a DB Instance engine and class below

Engine: MySQL

DB Instance Class: db.m1.small

DB Engine Version: 5.5.8

Auto Minor Version Upgrade: Yes No

Multi-AZ Deployment: Yes No

Provide the details for your RDS Database Instance.

Allocated Storage:* 20 GB (Minimum: 5 GB, Maximum 1024 GB)

DB Instance Identifier:* dbglobalvideo (e.g. mydbinstance)

Master User Name:* root (e.g. awsuser)

Master User Password:* ●●●●●●●● (e.g. mypassword)

Continue ▶

Abbildung 4.8: Einstellungsmöglichkeiten für die Datenbankinstanz

Zusätzlich sorgt S3 für redundante Datenhaltung, in dem die Objekte in den S3-Standorten redundant gespeichert sind. Amazon garantiert eine Verfügbarkeit von 99,999999999 Prozent oder alternativ 99,99 Prozent bei geringeren laufenden Kosten für den Dienstnutzer pro Jahr [9].

Streaming

Die Streaming-Funktionalität lässt sich mit dem Content Distribution Network (CDN) des Dienstes Amazon CloudFront realisieren. CloudFront ist ein Dienst zur schnellen Übermittlung von Daten und arbeitet eng mit S3 zusammen, das die Daten persistent speichert. Die Edge-Standorte befinden sich global verteilt in den USA, Europa und in Asien. In Europa befinden sich die vier Edge-Standorte Amsterdam, Dublin, Frankfurt und London. Das heißt, dass die meisten Nutzer aus Deutschland durch die Edge-Standorte Frankfurt oder Amsterdam bedient werden. CloudFront reagiert auch automatisch auf ansteigende oder eine abnehmende Nachfrage. Das heißt, um eine Nachfrage zu bedienen, kann CloudFront automatisch weitere Webserver-Instanzen starten.

CloudFront unterstützt drei Netzwerkprotokolle. Diese sind HTTP- und HTTPS-Protokoll zum Herunterladen von Dateien und das Real Time Messaging Protocol (RTMP) zum

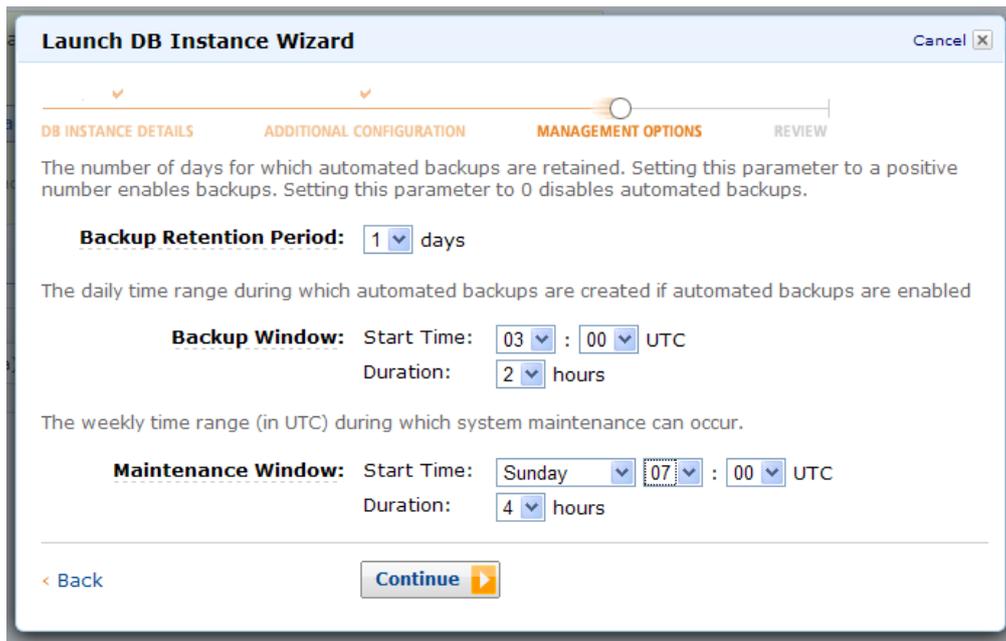


Abbildung 4.9: Einstellungsmöglichkeiten für das automatische Anlegen von Sicherheitskopien in RDS

Streamen von Videos. Zusätzlich verwendet CloudFront den Flash Media Server von Adobe [2].

CloudFront bietet eine Web-basierte Schnittstelle, mit der für Buckets eine Download- oder Streaming-Unterstützungen festgelegt werden kann. Dabei wird vom Dienstanutzer der Ort der Originalversion der Objekte angegeben.

Beide Verteilungsarten, Download und Streaming, unterstützen standardmäßig Datenübertragungsgeschwindigkeiten von maximal 1.000 Megabit pro Sekunde und maximale 1.000 Anfragen pro Sekunde. Zu den unterstützten Dateiformaten für Streaming gehören u.a. die FLV und MP4, sowie die VP6 und H.264.

Um CloudFront für die im Rahmen dieser Arbeit realisierte Infrastruktur zu nutzen, muss zunächst in der Amazon Management Console die Schaltfläche „Create Distribution“ betätigt werden. Anschließend öffnet sich ein Dialogfenster, in dem die Auslieferungsart (Streaming oder Download) festgelegt wird. Zusätzlich wird ein Bucket in S3 angegeben. Die Inhalte dieses Buckets, werden beim ersten Aufruf zu dem Edge-Standort, der dem Nutzer geographisch am nächsten ist, kopiert. Wird diese Datei erneut aufgerufen, wird sie direkt vom Edge-Standort aus an den anfordernden Benutzer gesendet.

Informationen zum Datenverkehr erhält ein Kunde des Dienstes man durch die Zugriffsprotokollierung. Bei den Zugriffsprotokollen handelt es sich um Aktivitätsaufzeichnungen mit detaillierten Informationen aller Anfragen zu einzelnen Inhalten. Für das

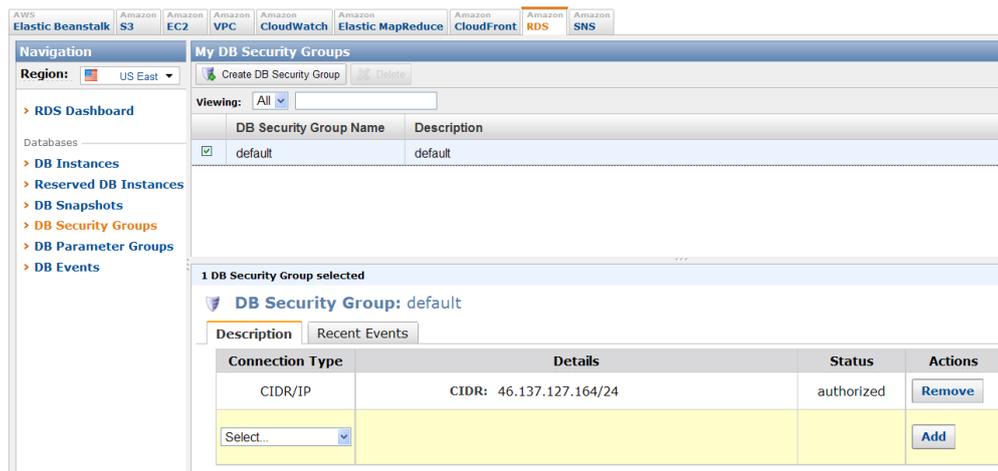


Abbildung 4.10: Einstellungen der Sicherheitsgruppe der RDS-Instanz

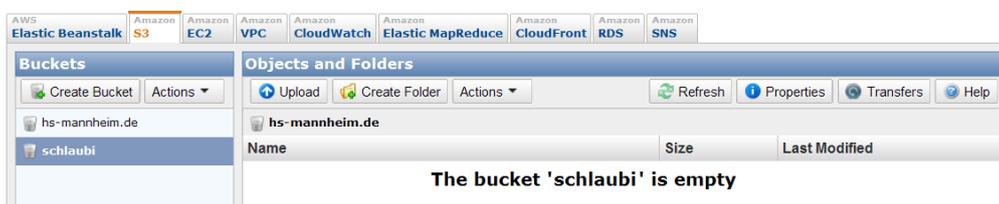


Abbildung 4.11: S3-Fenster in der Amazon Management Console

Schreiben, Speichern und Abrufen der Zugriffsprotokolle entstehen Kosten wegen der Verwendung von S3. Abbildung 4.12 zeigt das Dialogfenster zum Erstellen einer Streaming-Unterstützung für ein Bucket.

Nach dem Erstellen einer Streaming-Unterstützung für ein Bucket, wird ein eindeutiger CloudFront Domänenname zurückgegeben. Mit diesem Domännennamen kann auf die Objekte im Netzwerk verwiesen werden.

Zusätzlich besteht die Möglichkeit, die Streaming-Unterstützung eines Buckets für das dynamische Bit-Raten-Streaming zu konfigurieren. Dadurch werden verschiedene Kopien eines Videos gespeichert und jede Kopie für eine unterschiedliche Qualitätsstufe verschlüsselt. Die Qualität des Videos wird somit automatisch an die Geschwindigkeit der Internetverbindung des Benutzers angepasst [3].

Redundanter Server zur Lastverteilung

Die Anforderung aus Kapitel 4.1.6 erfüllt der Dienst ELB. Dieser verteilt den eingehenden Verkehr und garantiert, dass ständig Server-Instanzen verfügbar sind. Fallen einzelne Server-Instanzen aus, werden diese identifiziert und der eingehende Daten-

Create Distribution Cancel

Delivery Method: Download Streaming

Origin*: schlaubi

Logging: On Off

CNAMEs:

Comments:

Distribution Status: Enabled Disabled

*Required Field

Abbildung 4.12: Dialogfenster zum Erstellen einer Streaming-Unterstützung für ein Bucket

transfer wird auf die noch verfügbaren Instanzen umgeleitet. Abbildung 4.13 zeigt die Integration des Dienstes zur Lastverteilung in die bisherige Streaming-Architektur.

4.3 Ergebnis

Die mit Amazons öffentlichen Cloud-Diensten erstellte Infrastruktur besteht zunächst aus mindestens einem Webserver, der in EC2 läuft. Der Webserver ist über EBS mit einem persistenten Speichervolumen gekoppelt, der als Speicherort für die Anwendung dient. Für die MySQL-Datenbank steht über den Datenbankdienst RDS ein skalierbarer Datenbank-Server zur Verfügung. Die Videos sind im Speicherdienst S3 redundant gespeichert. Die Streaming-Funktionalität wird durch den Dienst CloudFront realisiert. Der Dienst ELB stellt den Lastverteiler bereit. Diese Komponenten bilden zusammen die Grundinfrastruktur für die Realisierung der in Kapitel 5 vorgestellten Streaming-Webanwendung.

Aufgrund der zeitlichen Vorgaben konnte nur eine Skalierfähigkeit der Speicherressourcen eines einzelnen Datenbank-Servers realisiert werden. Für eine Skalierung der Server selbst ist die lokale Installation der RDS-Werkzeuge nötig. Um ausgefallene Webserver neu zu starten, muss der ELB mit dem Skalierungs-Dienst Autoscale verbunden werden. Dieses erfordert aber die Installation der Autoscale-, CloudWatch-

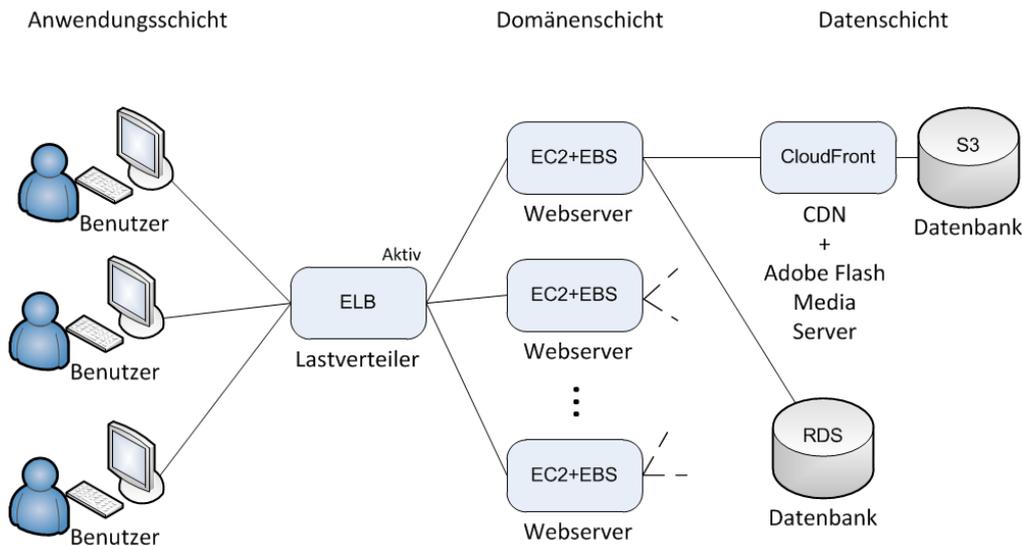


Abbildung 4.13: Integration des Dienstes ELB in die Streaming-Architektur

und Loadbalancer-Werkzeuge. Dieser Anforderungspunkt konnte in der vorgegebenen Zeit nicht realisiert werden. Ebenso konnte der Lastverteiler als Single Point of Failure (SPOF) für das gesamte System nicht ausgeschlossen werden.

4.4 Kostenanalyse

Die monatlichen Kosten für die Laufzeit einer Instanz in EC2 betragen mindestens \$7,44. Die monatlichen Kosten für eine RDS-Instanz betragen mindestens \$34,22. Im Falle einer redundant ausgelegten RDS-Mult-AZ-Instanz betragen die monatlichen Kosten für die Laufzeit \$68,44. Die monatlichen Kosten, um über ELB die Verfügbarkeit der Instanzen zu überwachen, betragen pro Instanz \$18,6. Um eine automatische Skalierung mit Hilfe von CloudWatch zu gewährleisten, ist eine detaillierte Überwachung der Auslastung jeder Instanz in EC2 bzw. RDS nötig. Hierfür berechnet Amazon monatlich pro Instanz \$11,16.

Zusätzlich fallen für den eingehenden Datentransfer ca. \$0,11 pro Gigabyte an. Der ausgehende Datentransfer wird mit \$0,15 pro Gigabyte berechnet. Für die Datenhaltung fallen für S3 monatlich mindestens \$0,093 pro Gigabyte an. Ebenso berechnet Amazon für den Datenbankspeicher in RDS monatlich \$0,10 pro Gigabyte. Zusätzlich sind mit Kosten in Höhe von \$0,0175 pro 10.000 GET-Anfragen, sowie \$0,01 für 1.000 PUT-Anfragen zu rechnen. Für das EBS-Speichervolumen, als auch für eine RDS-Instanz fallen \$0,10 pro 1 Million Eingabe/Ausgabe-Operationen an. Somit ergeben sich minimale Kosten für den Betrieb der Infrastruktur von \$71,42.

Zusätzlich kommen die Kosten für den eingehenden und ausgehenden Datentransfer hinzu. Bei einem eingehenden Datentransfer von 100 Gigabyte, fallen monatliche Kosten von \$11 an. Bei einem ausgehenden Datentransfer von 1000 Gigabyte, entstehen Kosten von \$150. Die Speicherung von 100 Gigabyte Daten in S3 wird mit monatlich \$9,3 berechnet. Das bedeutet, dass jeden Monat mit eine Kostenzuwachs von \$9,3 zum Vormonat zu rechnen ist.

100 Gigabyte eingehende Daten würden bei einer maximalen Dateigröße von ca. 100 Megabyte 1000 Videodateien, also 1000 Datenbankeinträge ergeben. Bei einer Größe von 1 Megabyte pro Datensatz ergibt das einen Datenbankzuwachs von 1 Gigabyte pro Monat. Dadurch ergibt sich ein monatlicher Kostenzuwachs von \$0,10. Die Kosten für die PUT- und GET-Anfragen, sowie für Ein-/Ausgabe-Operationen spielen im Verhältnis zu den Gesamtkosten keine Rolle. Alle Preise sind exklusive Mehrwertsteuer. Somit ergeben sich im ersten Monat minimale Kosten von ca. \$284,22 zuzüglich Mehrwertsteuer. Abbildung 4.14 zeigt eine Darstellung der Kosten für die Infrastruktur.

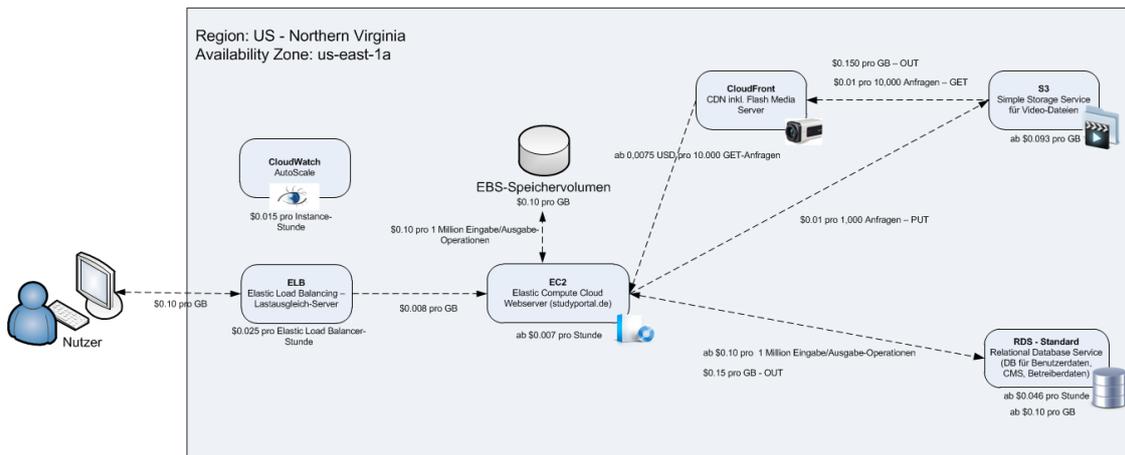


Abbildung 4.14: Kosten für die Infrastruktur

4.5 Bewertung

Für den im Vergleich zu traditionellen Webhosting-Angeboten hohen Preis, erhält der Kunde eine flexible Infrastruktur mit einer hohen Verfügbarkeit. Risiken bezüglich der Datensicherheit kann mit einer Verschlüsselung der Daten begegnet werden. Bezüglich des Risikos des unzureichenden Datenschutzes und der Gefahr eines Vendor-Lock-In bietet Amazon über den Dienst AWS Import/Export die Möglichkeit großvolumige Datensätze zu transferieren [6].

Kapitel 5

Entwicklung der Webanwendung für Video-Streaming auf Basis von Cloud-Diensten

Dieses Kapitel enthält die Grundvoraussetzungen und Rahmenbedingungen einer Webanwendung für Video-Streaming. Die Anforderungen werden definiert und anschließend implementiert. Anschließend findet ein Vergleich mit YouTube statt und Erweiterungsmöglichkeiten werden diskutiert.

5.1 Grundvoraussetzungen

Grundvoraussetzung für eine Streaming-Plattform ist eine Infrastruktur, bestehend aus einem Webserver, sowie ein Datenbank-Server. Das Datenbank-Schema besteht aus mindestens einer Datenbanktabelle. Diese Tabelle enthält die Datei-Informationen der hochgeladenen Videos und heißt „*media*“. Die Realisierung der Wiedergabeliste kann auf zwei Arten erfolgen. Zum einen können die Videos auf Video-Datensätze der selben Tabelle referenzieren. Zum anderen kann für die Wiedergabeliste eine eigene Tabelle erstellt werden. Abbildung 5.1 zeigt die einfachere Variante der beiden Ansätze mit nur einer einzigen Tabelle.

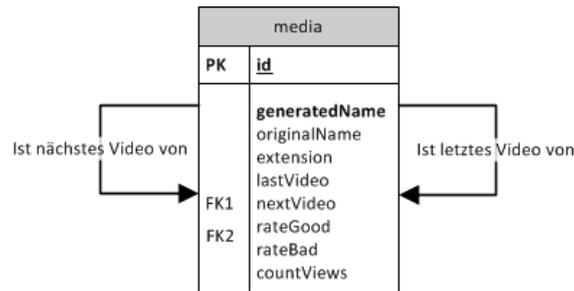


Abbildung 5.1: Datenbankschema der Video-Datenbank

5.2 Rahmenbedingungen

Der folgende Abschnitt enthält die Rahmenbedingungen für das verwendete Betriebssystem und den Webserver. Zusätzlich wird die Konfiguration des Webserver festgelegt. Anschließend wird eine weitere Rahmenbedingung für den Einsatz des PHP-Frameworks Symfony festgelegt.

5.2.1 Betriebssystem des Webservers

Ein flexibles Linux-Betriebssystem mit langer Herstellerunterstützung soll eingesetzt werden. Das Betriebssystem muss die 64-Bit-Architektur unterstützen, da die Streaming-Webanwendung große Datenmengen in den Arbeitsspeicher laden wird.

5.2.2 Konfiguration des Webservers

Eine Software zum Betrieb eines Webserver ist nötig. Es existieren zwei populäre Alternativen. Diese sind Apache2-Webserver [11] und der von YouTube eingesetzte, leichtgewichtige lighttpd-Webserver [34].

Der Webserver wird so konfiguriert, dass die Dateigröße für den Upload von Videos auf maximal 120 Megabyte begrenzt und für eine Uploadgeschwindigkeit von 5 Kilobyte pro Sekunde ausgelegt ist. Diese Geschwindigkeit ist bei einer GPRS-Verbindung der Regelfall. Somit ist die maximale Zeit zum Hochladen auf 6,66 Stunden (120 MB / (0,005 MB/S)) begrenzt.

Die Webanwendung wird in der Skriptsprache PHP implementiert. Darum muss auf den verwendeten Webservern das PHP-Modul zu installiert sein.

5.2.3 Symfony-PHP-Framework

Eine weitere Rahmenbedingung ist, dass die Webanwendung mit Hilfe des PHP-Frameworks Symfony entwickelt wird. Symfony ermöglicht es, Webanwendungen auf Basis eines strikten Model-View-Controller-Prinzips (MVC-Prinzip) aufzubauen. Zudem kümmert sich Symfony um die Ordnerstruktur der Webanwendung. Ein weiterer Vorteil von Symfony besteht in der Vereinfachung bestimmter Vorgänge, wie das Erzeugen von Standard-Eingabemasken.

5.3 Anforderungsanalyse der Anwendungslogik

Die Webanwendung wird so realisiert, dass ein Nutzer über eine grafische Web-Oberfläche Video-Dateien hochladen und anschauen kann. Abbildung 5.2 zeigt die Hauptanwendungsfälle der Nutzer und Administratoren als große und alle weiteren als kleine Kreise.

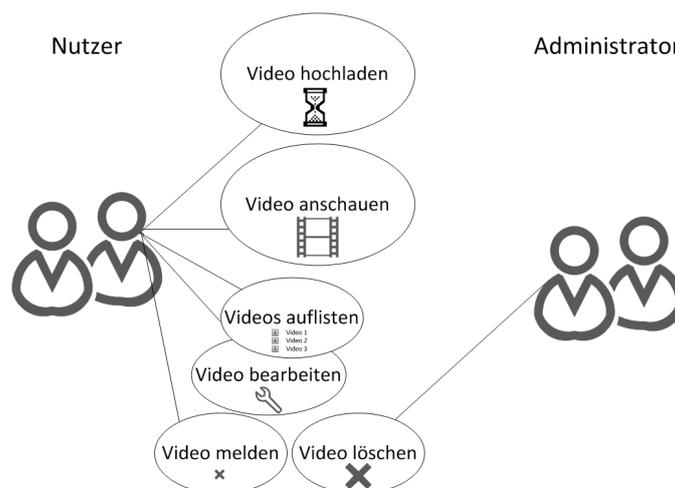


Abbildung 5.2: Hauptanwendungsfälle der Benutzer und Administratoren

5.3.1 Video-Upload

Die grafische Oberfläche für das Hochladen von Videos besteht aus dem HTML-Eingabefeld. Zusätzlich zu dem Eingabefeld existiert eine Information bezüglich der unterstützten Formate, maximalen Dateigröße, sowie ein Datenschutzhinweis (Disclaimer).

Die Metadaten des hochgeladenen Videos, wie z.B. der originale Dateiname und der neu generierte Dateiname, werden in einer MySQL-Datenbank abgelegt. Das Video selbst ist in einem Bucket in S3 gespeichert.

Unterstützung der gängigen Videoformate

Ein weiterer Anforderungspunkt ist, dass die Webanwendung das Hochladen der gängigen Videoformate unterstützt. Es sind zusätzlich zu den serverseitigen Eingabeprüfungen auf der Seite des Clients Eingabeprüfungen nötig.

Integration einer Schnittstelle zu S3

Eine Schnittstelle zu S3 ist in der Webanwendung nötig. Zu deren Hauptanforderungen gehört das Hochladen von Dateien. Hierbei ist zu berücksichtigen, dass die hochgeladenen Dateien einen eindeutigen neuen Dateinamen erhalten. Dieser eindeutige Name besteht aus mindestens 32 zufällig generierten alphanumerischen Zeichen. Anschließend wird der generierte Dateiname in eine MySQL-Datenbank abgelegt. Auch der ursprüngliche Dateiname wird gespeichert und ist in der Video-Anzeige als Titel verwendet.

Implementierung der Wiedergabeliste

Jeder Eintrag in der MySQL-Datenbank enthält Informationen zu einem Video. Über eine doppelt-verkettete Liste ist jeder Eintrag mit mindestens einem weiteren Video-Eintrag verbunden. Hierbei spielen der Zeitpunkt des Hochladens und die Bewertung der Benutzer eine wesentliche Rolle für die Reihenfolge der Videos. Dadurch wird die Wiedergabeliste realisiert. Das zuletzt hochgeladene Video wird solange auf der Startseite angezeigt, bis es durch ein neu hochgeladenes Video ersetzt wird.

Implementierung eines Fortschrittsbalkens

Beim Hochladen von Videos soll während des Hochladens ein Fortschrittsbalken angezeigt werden. Das Hochladen ist mit JavaScript und Ajax-Aufrufen realisiert.

5.3.2 Video-Anzeige

Die Video-Anzeige wird über den Flash-Player realisiert. Der Benutzer hat die Möglichkeit, schon während die der Übertragung des Videos, mit der Wiedergabe des Videos zu beginnen.

Automatische Wiedergabe

Eine weitere Anforderung ist, bei Aufruf eines Videos dieses automatisch abzuspielen. Bei der Beendigung des Videos ist automatisch auf das nächste Video zu wechseln. Das nächste Video ist das zuletzt hochgeladene Video. Dadurch wird eine Art Fernsehprogramm durch die hochgeladenen Videos der Benutzer erzeugt. Zusätzlich hat der Benutzer über Bedienfelder die Möglichkeit, durch die Wiedergabeliste zu navigieren (zappen).

5.4 Realisierung der Grundvoraussetzungen und Rahmenbedingungen

Basis für eine Streaming-Lösung ist die in Kapitel 4 realisierte Cloud-basierende Infrastruktur. Zunächst wird ein Webserver gestartet und dessen Konfiguration beschrieben. Anschließend wird ein neues Symfony-Projekt angelegt. Dazu gehört das Anlegen und die Implementierung der Anwendungen, Module, Formulare, sowie das Einbinden von Erweiterungen (Plugins) und das Erweitern vorhandener Bibliotheken.

EC2-AMI

Passend zu der Rahmenbedingung aus Kapitel 5.2.1 existiert in EC2 von Ubuntu-Linux das AMI ID *ami-3d1f2b49*. Hierbei wird Ubuntu Version 10.04 (Lucid Lynx) eingesetzt. Diese steht bis 2015 unter dem Long Term Support (LTS) [48]. Das AMI (siehe Abbildung 5.3) ist nur innerhalb der Region EU-West verfügbar. Es basiert auf der 64-Bit-Architektur und wird über ein EBS-Speichervolumen bereitgestellt.

Bei Start einer Instanz in EC2 sind die Eigenschaften der neuen Instanz zu wählen. Dazu gehört der Instanztyp, das Aktivieren einer Netzwerküberwachung und das Festlegen der Sicherheitsgruppe, sowie des Schlüsselpaars.

Ubuntu 10.04 LTS (Lucid Lynx)

Select an image

Amazon EC2 Published AMIs

The images have been published to [Amazon EC2](#), and can be used in the [ec2-run-instance](#) instruction on getting started with Amazon EC2, see the [EC2 Start](#)

Region	arch	root store	ami	ec2 command
ap-southeast-1	64-bit	ebs	ami-f092eca2	ec2-run-instance \${EC2_KEYPAIR_AP
ap-southeast-1	32-bit	ebs	ami-f292eca0	ec2-run-instance \${EC2_KEYPAIR_AP
eu-west-1	64-bit	ebs	ami-3d1f2b49	ec2-run-instance
eu-west-1	32-bit	ebs	ami-311f2b45	ec2-run-instance

Abbildung 5.3: Offizielle AMIs in EC2 von Ubuntu

Installation und Konfiguration des Webservers

Obwohl laut [27] sich der `lighttpd`-Webserver besonders gut für Video-Streaming eignet, wird die Rahmenbedingung aus Kapitel 5.2.2 mit Hilfe des Webservers `Apache2` umgesetzt, da dieser eine bessere Integration in das Betriebssystem, höhere Flexibilität und gewachsenere Stabilität aufweist.

Um das Hochladen von Dateien bis 120 Megabyte pro Datei zu unterstützen, sind Anpassungen in der Datei `php.ini` nötig. Folgend Werte sind relevant.

```
file_uploads = on
upload_max_filesize = 120M
post_max_size = 125M
memory_limit = 128M
max_execution_time = 24000
max_input_time = 24000
max_file_uploads = 1
```

Die Variable `upload_max_filesize` bestimmt die maximale Größe aller Dateien, die hochgeladen werden können.

Die Variable `post_max_size` legt die maximale Größe von Daten fest, die über POST mit dem Format `'multipart/form-data'` an den Webserver geschickt werden. Das heißt, ein Skript auf dem Webserver kann maximal `<POST_MAX_SIZE>` Daten entgegennehmen. Hierbei ist zu beachten, dass `<POST_MAX_SIZE>` größer als `<UPLOAD_MAX_FILESIZE>` sein

muss, da zum Wert von `upload_max_filesize` zusätzlich noch die Größe der HTML-Formularfelder und Header-Informationen hinzugerechnet werden müssen.

Die Variable `memory_limit` bestimmt die maximale Größe, die ein Skript pro Anfrage im Arbeitsspeicher allozieren darf. Hierbei besteht die Gefahr, dass da ein zu hoher Wert dazu führt, dass der gesamte Arbeitsspeicher des Webserverns von den Prozessen von Apache2 aufgebraucht wird, was zu einem Ausfall des Webserverns führt.

Die Variable `max_execution_time` legen die maximale Lebensdauer eines Skripts fest. Ähnlich ist die Konstante `max_input_time`, welche die maximale Zeitspanne für ein Skript festlegt, um Daten zu empfangen. Aufgrund der Rahmenbedingung aus Kapitel 5.2.2 sind diese zwei Konstanten auf 6,66 Stunden festzulegen. Umgerechnet auf Sekunden ergibt das für einen Upload eine maximale Ausführungszeit von 24.000 Sekunden.

Die Variable `max_file_uploads` legt die maximale Anzahl an Dateien fest, die auf einmal hochgeladen werden dürfen. Diese Zahl wird auf 1 festgelegt, da es sich bei Video-Dateien, anders als bei Dokumenten oder Fotos, meistens um größere Dateien handelt und somit der Webserver schneller an seine Grenzen stößt.

Symfony-PHP-Framework

Die Installation [46] des PHP-Frameworks Symfony setzt die Rahmenbedingung 5.2.3 um.

Zusätzlich ist ein PHP-Beschleuniger nötig. Hierfür ist laut [47] der PHP-Beschleuniger APC geeignet. Zunächst müssen mit dem Kommando `apt-get`-Befehl die nötigen Softwarepakete installiert werden.

```
apt-get install php-pear php5-dev libpcre++-dev
```

Anschließend ist über den Paketverwaltungsmanager PECL das APC-Paket zu installieren.

```
pecl install APC
```

Schließlich sind folgende Werte in der Datei `php.ini` nötig.

```
extension = apc.so  
apc.rfc1867 = on
```

5.5 Realisierung und Implementierung der Anforderungen

Dieser Abschnitt beinhaltet die Realisierung der Anforderungen aus Kapitel 5.3.

5.5.1 Hochladen eines Videos

Für das Hochladen eines Videos ist ein Formular nötig. Hierzu wird das `mediaForm`-Formular, welches auf Basis der Media-Datebanktabelle aus dem Datenbankschema generiert wurde, um eine Upload-Logik ergänzt. Das Formular besteht aus den beiden Methoden `configure` und `processValues`. Die Methode `configure` definiert die Formular-Initialisierung. In dieser Methode werden für die Felder des Formulars die Formularfeldtypklasse und die Validierungsklasse festgelegt. In diesem Fall wurde für das Datenbank-Feld mit dem Namen `video` der Feldtyp `sfWidgetFormInputFile` festgelegt. Diese Klasse generiert zur Laufzeit den HTML-Quelltext für das Eingabeformular. Abbildung 5.4 zeigt die `configure`-Methode des Formulars.

```
39 class mediaForm extends BasemediaForm {
40     public function configure() {
41         parent::setUp();
42         unset($this['created_at'], $this['updated_at']);
43         $this->widgetSchema['video'] = new sfWidgetFormInputFile();
44         $this->validatorSchema['video'] = new sfValidatorFile(array(
45             'max_size' => 120000000,
46             'mime_types' =>
47                 array('video/x-flv', 'video/avi', 'video/mpeg',
48                     'video/x-mpeg', 'video/mpg', 'video/mp4'),
49             'path' => sfConfig::get('sf_upload_dir').DIRECTORY_SEPARATOR."media",
50             'required' => true,
51             'validated_file_class' => 'sfValidatedFileCustom'
52         ));
53     }
```

Abbildung 5.4: Initialisierungsmethode `configure` des `mediaForm`-Formulars

Für die Validierung eines Eingabefeldes zum Hochladen von Dateien stellt Symfony die Klasse `sfValidatorFile` zur Verfügung. Diese Klasse ermöglicht es Konstanten, wie z.B. `max_size` für die Maximalgröße einer Datei (in Byte), `mime_types` für die zu unterstützenden Dateiformate und `validated_file_class` für die Nachverarbeitung einer gesäuberten Datei, festzulegen. In diesem Fall ist die Maximalgröße einer Datei 120.000.000 Byte (120 Megabyte) groß, wodurch die Rahmenbedingung aus Kapitel 5.2.2 erfüllt ist.

Die Anzahl der unterstützten Dateiformate ist vom eingesetzten Media Server abhängig. CloudFront benutzt den Flash Media Server (FMS), um die Video-Dateien zu kodieren

und als Streaming-Pakete über ein Netzwerk zu verteilen. Dieser unterstützt jedoch nur die Dateiformate FLV, MP4 und H264. Für eine Unterstützung der Dateiformate 3GP und MOV ist ein Media Server von Apple notwendig. Ebenso ist für das Format WMV von Microsoft ein spezieller Kodierer Voraussetzung. Um dieses Problem zu lösen, ist es notwendig jedes hochgeladene Video vor der Ablage in S3 in ein von CloudFront unterstütztes Dateiformat zu konvertieren. Die Formate WMV, MPG, MPEG können problemlos mittels dem Audio- und Video-Kovertierungsprogramm `ffmpeg` [22] in das FLV-Format konvertiert werden. Für das Format AVI ist aufgrund nicht mehr unterstützter Konvertierungs-Bibliotheken in Ubuntu 10.04 die Konvertierung nicht ohne den Verlust der Audiospur möglich. Dadurch kann die Anforderung aus Kapitel 5.3.1 nur teilweise umgesetzt werden.

```
ffmpeg -i <temporärerName> -b 1024k -s 320x240 -r 25 -ar 22050 -ac 2  
-acodec libmp3lame <generierterName>
```

Mit der Methode `processValues` ist es möglich, die beim Hochladen empfangenen Parameter abzufangen und deren standardmäßige Verarbeitung zu überschreiben. Hierbei wird der originale Dateiname des Videos zwischengespeichert, dem Video ein neuer Name zugeordnet und anschließend der originale Name als Titel verwendet. Des Weiteren wird das zuletzt hochgeladene Video gesucht und dessen erzeugter Name als nächstes Video für das aktuell hochgeladene Video referenziert. Abbildung 5.5 zeigt die `prozessValues`-Methode.

Für das Anzeigen des Formulars zum Hochladen wird eine eigene Action-Methode mit dem Namen `executeNew` erstellt. Abbildung 5.6 zeigt den Aufbau dieser Methode.

Zunächst wird eine Instanz des zuvor erstellten Formulars, sowie eine eindeutige ID für den Fortschrittsbalken erzeugt. Diese werden an das PHP-Skript `newSuccess.php` (siehe Abbildung 5.7) übergeben.

Zusätzlich zum Datei-Upload-Eingabefeld wurde ein Textabschnitt für die Datenschutzbestimmungen und ein weiterer für die unterstützten Video-Formate eingefügt, wodurch die Anforderung aus Kapitel 5.3.1 erfüllt ist. Ebenso befindet sich innerhalb des Formulars ein Bereich für die Fehlermeldungen.

JavaScript sendet über ein *jQuery-Form-Plugin* die Formular-Anfrage an den Server [31]. Gleichzeitig wird die Funktion `checkProgress` in einem Intervall von zwei Sekunden ausgeführt. Der Fortschrittsbalken wird weiter unten im späteren Verlauf dieses Kapitels genauer beschrieben. Ebenso überdeckt ein Layer während des Prozesses die Eingabemaske, so dass ein mehrfaches Absenden des Formulars nicht möglich ist. Abbildung 5.8 zeigt den JavaScript-Teil des Formulars zum Hochladen der Videos.

```

55     public function processValues($values) {
56         // file was uploaded
57         if ($values['video'] instanceof sfValidatedFile) {
58
59             // get last video
60             $result = Doctrine::getTable('media')->createQuery('a')
61                 ->execute();
62
63             $lastVideo = $result->getLast();
64             if (!empty($lastVideo)){
65
66                 // connect new video with last video
67                 $generatedName = $lastVideo->getVideo();
68                 if (!empty($generatedName)){
69                     $values['lastvideo'] = $generatedName;
70                 }
71             }
72             $basename = $values['video']->getOriginalName();
73             $extension = $values['video']->getExtension();
74             $extensionPosition = mb_stripos ($basename, ".");
75             if ($extensionPosition !== false){
76                 $basename = strtolower (mb_substr ( $basename, 0 ,
77                     $extensionPosition));
78             }
79             // set original name as title of new video
80             $values['title'] = $basename;
81             // save extension of new video
82             $values['extension'] = $extension;
83         }
84         return parent::processValues($values);
85     }
86 }

```

Abbildung 5.5: Vorverarbeitungsmethode processValues des Formulars

```

53     public function executeNew(sfWebRequest $request) {
54         $this->form = new MediaForm();
55         $this->uniqId = uniqid();
56         return sfView::SUCCESS;
57     }

```

Abbildung 5.6: Aufbau der Methode executeNew

Für das Absenden des Formulars werden im JavaScript-Object `options` die Einstellungen der Submit-Anfrage festgelegt. Hierzu gehören Angaben, wie der Name des HTML-Elements für die Antwort des Webserver, sowie der Name der Action-Methode. Ebenso eine Callback-Funktion definiert, die bei erfolgreicher oder fehlerhafter Übermittlung aufgerufen wird. Diese Callback-Funktion dient dazu, das Intervall des Fortschrittsbalken zu deaktivieren, sowie das Formular wieder freizugeben.

```

4 <h2>Video-Upload</h2>
5
6 <form id="mediaUploadForm" action="<?php echo url_for('media/upload') ?>"
7     method="POST" enctype="multipart/form-data">
8
9     <div id='flash'>
10         <div id='divToUpdate'>Just select a video file and click on Upload!
11             - That's it
12         </div>
13     </div>
14
15     <input type="hidden" name="APC_UPLOAD_PROGRESS"
16         id="progress_key" value="<?php echo $uniqId ?>" />
17     <?php echo $form['video']->render(); ?>
18     <?php echo $form['_csrf_token']->render(); ?>
19     <input type="submit" value="Upload!" />
20
21 <p>
22     Supported formats: flv, wmv, mp4, mpg, mpeg, mpeg2, H.264 ...<br/>
23     Not yet supported formats: avi, mov, 3gp (coming soon)<br/>
24     Max-Size: 120 MB<br/>
25 </p>
26
27 <p style="font-size: 0.8em">
28     Important: Do not upload any TV shows, music videos, music concerts,
29     or commercials without permission unless they consist entirely of
30     content you created yourself. The Copyright Tips page and the Community
31     Guidelines can help you determine whether your video infringes someone
32     else's copyright. By clicking "Upload!", you are representing that this
33     video does not violate Global-video's Terms of Service and that you own
34     all copyrights in this video or have authorization to upload it.
35 </p>
36 </form>

```

Abbildung 5.7: Aufbau des Templates newSuccess zum Hochladen

Das Formular sendet die Parameter an die Action-Methode `executeUpload`, welche zunächst die Eingabeprüfung auf Seiten des Servers vornimmt und anschließend die Formulardaten verarbeitet. Abbildung 5.9 zeigt den Aufbau der Methode `executeUpload`.

Über die Methode `$form->isValid()`, werden zunächst die Formulardaten anhand der in der Validierungsklasse `sfValidatorFile` festgelegten Bedingungen geprüft. Anschließend, falls die Formulardaten erfolgreich validiert wurden, wird über die Methode `$form->save()` die benutzerdefinierte Klasse, die unter der `sfValidatorFile`-Konstante `validated_file_class` angegeben wurde, aufgerufen. Diese ist für die Nachverarbeitung der gesendeten Parameter zuständig. Abbildung 5.10 zeigt die Validierungsklasse `sfValidatedFileCustom`.

```

8 class sfValidatedFileCustom extends sfValidatedFile {
9 public function save($file = null, $fileMode = 0666, $create = true, $dirMode = 0777){
10     $tempName = parent::getTempName();
11     $generatedName = parent::generateFilename();
12     // upload file to s3 bucket defined in conf/app.yml
13     $this->uploadFileToS3($tempName, $generatedName);
14     $extensionPosition = mb_stripos ( $generatedName, ".");
15     $generatedName = strtolower (mb_substr ( $generatedName, 0
16                                     , $extensionPosition));
17
18     // set next Video For Last Video
19     $result = Doctrine::getTable('media')->createQuery('a')->execute();
20     $lastVideo = $result->getLast();
21     if (!empty($lastVideo)){
22         $lastVideo->setNextVideo($generatedName);
23         $lastVideo->save();
24     }
25     // return the saved file as normal
26     return $generatedName;
27 }

```

Abbildung 5.8: JavaScript-Teil des Formulars zum Hochladen der Videos

Integration einer Schnittstelle zu S3

Die Klasse `sfValidatedFileCustom` erbt von der Klasse `sfValidatedFile` und bietet dadurch die Möglichkeit, deren Methoden, in diesem Fall die Methode `save`, zu überschreiben. Normalerweise würde der Aufruf der `save`-Methode die hochgeladene Datei lokal auf dem Server speichern. Da dieses aber nicht erwünscht ist, werden die Daten mit Hilfe der Zend-S3-API [58] in S3 abgelegt, um Speicherengpässen zu vermeiden. Hierzu wird die im temporären Verzeichnis zwischengespeicherte Video-Datei mit `$s3->putFile()` unter dem neuen Namen im Bucket abgelegt, wodurch die Anforderung aus Kapitel 5.3.1 realisiert ist. Abbildung 5.11 zeigt die Fortsetzung der Validierungsklasse `sfValidatedFileCustom`.

Anschließend wird dem zuletzt hochgeladenen Video eine Referenz zum neuen Video hinzugefügt, wodurch die Anforderung einer doppeltverketteten Liste aus Kapitel 5.3.1 ebenfalls erfüllt ist.

Implementierung des Fortschrittsbalkens

Für den Balken, der den Fortschritt beim Hochladen anzeigt, ist es notwendig, in den Cache eines Prozesses schauen zu können. Der zuvor installierte PHP-Beschleuniger APC bietet mit der Funktion `apc_fetch()`, die Möglichkeit über einen eindeutigen ID-Namen in den Cache-Status eines Prozesses oder einer Variablen zu schauen und somit die Größe festzustellen. Die Abbildung 5.12 zeigt die Action-Methode `executeCheckProgress`, welche die zuvor erzeugte eindeutige ID verwendet, um den

```

58 public function executeUpload(sfWebRequest $request)
59 {
60     $this->form = new MediaForm();
61     $this->uniqId = $this->getRequestParameter('APC_UPLOAD_PROGRESS');
62     $msg = "";
63     if (!empty($this->uniqId)){
64         if ($this->getRequest()->getMethod() == sfRequest::POST)
65         {
66             $files = $request->getFiles();
67             if (!empty($files))
68             {
69                 $this->processForm($request, $this->form);
70             }else{
71                 $msg = "No file field empty";
72                 $this->renderText($msg);
73             }
74         }
75     }else{
76         $msg = "Something is wrong!";
77         $this->renderText($msg);
78     }
79     return sfView::NONE;
80 }
81 protected function processForm(sfWebRequest $request, sfForm $form) {
82     $form->bind(
83         $params = $request->getParameter($form->getName()),
84         $files = $request->getFiles($form->getName())
85     );
86     $msg = "";
87     if ($form->isValid()) {
88         $media = $form->save();
89         $msg = "<span class='successful' >Upload successful</span>";
90         $this->renderText($msg);
91     }else{
92         $msg = "<span class='error'>" . $form["video"]->getError() . "</span>";
93         $this->renderText($msg);
94     }
95 }

```

Abbildung 5.9: Aufbau der executeUpload-Methode

Cache-Status des dazugehörigen Skript-Prozesses zu erhalten. Dieser wird anschließend mit `$this->renderText()` an eine AJAX-Funktion zurückgegeben. Abbildung 5.12 zeigt die Action-Methode `executeProgress` mit der Logik des Fortschrittsbalkens.

Aus der Größe des Caches kann im Vergleich zur Gesamtgröße der hochzuladenden Datei der aktuelle Status berechnet werden. Abbildung 5.13 zeigt den JavaScript-Quelltext der Logik des Fortschrittsbalkens.

Der Aufruf der Action-Methode `executeCheckProgress` wird gleichzeitig mit dem Absenden des Formulars in einem zwei Sekunden-Intervall ausgeführt. Der Rückgabewert enthält die Dateigröße und die Gesamtgröße der Datei. Daraus ergibt sich die Prozentzahl des Fortschritts beim Hochladen, welche über die JavaScript-Funktion `reportprogress()` im HTML-Element mit der ID `progressbar` in Form eines Fortschritts-

```

8 class sfValidatedFileCustom extends sfValidatedFile {
9 public function save($file = null, $fileMode = 0666, $create = true, $dirMode = 0777){
10     $tempName = parent::getTempName();
11     $generatedName = parent::generateFilename();
12     // upload file to s3 bucket defined in conf/app.yml
13     $this->uploadFileToS3($tempName, $generatedName);
14     $extensionPosition = mb_stripos ( $generatedName, ".");
15     $generatedName = strtolower ( mb_substr ( $generatedName, 0
16                                     , $extensionPosition));
17
18     // set next Video For Last Video
19     $result = Doctrine::getTable('media')->createQuery('a')->execute();
20     $lastVideo = $result->getLast();
21     if (!empty($lastVideo)){
22         $lastVideo->setNextVideo($generatedName);
23         $lastVideo->save();
24     }
25     // return the saved file as normal
26     return $generatedName;
27 }

```

Abbildung 5.10: Aufbau der Validierungs-Klasse sfValidatedFileCustom

```

26 protected function uploadFileToS3($tempName, $generatedName) {
27     set_include_path(sfConfig::get('sf_plugins_dir') .
28                     '/dbAmazonS3Plugin/lib/vendor/' . PATH_SEPARATOR . get_include_path());
29
30     //initialize de amazon s3 cloud
31     $s3 = new Zend_Service_Amazon_S3(sfConfig::get('app_amazon_s3_aws_key'),
32                                     sfConfig::get('app_amazon_s3_aws_secret_key'));
33
34     $defaultS3Headers = array(Zend_Service_Amazon_S3::S3_ACL_HEADER =>
35                               Zend_Service_Amazon_S3::S3_ACL_PUBLIC_READ);
36
37     if ($this->checkOrCreateBucket($s3, sfConfig::get('app_amazon_s3_bucket'))) {
38         $s3filename = sfConfig::get('app_amazon_s3_bucket') . "/" . $generatedName;
39         if (is_file($tempName)) {
40             $s3->putFile($tempName, $s3filename, $defaultS3Headers);
41         } else {
42             throw new sfException('Error while creating file: ' . $generatedName);
43         }
44     }
45 }
46
47 protected function checkOrCreateBucket($s3, $bucket) {
48     if (!$s3->isBucketAvailable($bucket)) {
49         $s3->createBucket($bucket);
50     }
51     if ($s3->isBucketAvailable($bucket)) {
52         return true;
53     } else {
54         throw new sfException('Error while creating bucket: ' . $bucket);
55     }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }
1001 }
1002 }
1003 }
1004 }
1005 }
1006 }
1007 }
1008 }
1009 }
1010 }
1011 }
1012 }
1013 }
1014 }
1015 }
1016 }
1017 }
1018 }
1019 }
1020 }
1021 }
1022 }
1023 }
1024 }
1025 }
1026 }
1027 }
1028 }
1029 }
1030 }
1031 }
1032 }
1033 }
1034 }
1035 }
1036 }
1037 }
1038 }
1039 }
1040 }
1041 }
1042 }
1043 }
1044 }
1045 }
1046 }
1047 }
1048 }
1049 }
1050 }
1051 }
1052 }
1053 }
1054 }
1055 }
1056 }
1057 }
1058 }
1059 }
1060 }
1061 }
1062 }
1063 }
1064 }
1065 }
1066 }
1067 }
1068 }
1069 }
1070 }
1071 }
1072 }
1073 }
1074 }
1075 }
1076 }
1077 }
1078 }
1079 }
1080 }
1081 }
1082 }
1083 }
1084 }
1085 }
1086 }
1087 }
1088 }
1089 }
1090 }
1091 }
1092 }
1093 }
1094 }
1095 }
1096 }
1097 }
1098 }
1099 }
1100 }
1101 }
1102 }
1103 }
1104 }
1105 }
1106 }
1107 }
1108 }
1109 }
1110 }
1111 }
1112 }
1113 }
1114 }
1115 }
1116 }
1117 }
1118 }
1119 }
1120 }
1121 }
1122 }
1123 }
1124 }
1125 }
1126 }
1127 }
1128 }
1129 }
1130 }
1131 }
1132 }
1133 }
1134 }
1135 }
1136 }
1137 }
1138 }
1139 }
1140 }
1141 }
1142 }
1143 }
1144 }
1145 }
1146 }
1147 }
1148 }
1149 }
1150 }
1151 }
1152 }
1153 }
1154 }
1155 }
1156 }
1157 }
1158 }
1159 }
1160 }
1161 }
1162 }
1163 }
1164 }
1165 }
1166 }
1167 }
1168 }
1169 }
1170 }
1171 }
1172 }
1173 }
1174 }
1175 }
1176 }
1177 }
1178 }
1179 }
1180 }
1181 }
1182 }
1183 }
1184 }
1185 }
1186 }
1187 }
1188 }
1189 }
1190 }
1191 }
1192 }
1193 }
1194 }
1195 }
1196 }
1197 }
1198 }
1199 }
1200 }
1201 }
1202 }
1203 }
1204 }
1205 }
1206 }
1207 }
1208 }
1209 }
1210 }
1211 }
1212 }
1213 }
1214 }
1215 }
1216 }
1217 }
1218 }
1219 }
1220 }
1221 }
1222 }
1223 }
1224 }
1225 }
1226 }
1227 }
1228 }
1229 }
1230 }
1231 }
1232 }
1233 }
1234 }
1235 }
1236 }
1237 }
1238 }
1239 }
1240 }
1241 }
1242 }
1243 }
1244 }
1245 }
1246 }
1247 }
1248 }
1249 }
1250 }
1251 }
1252 }
1253 }
1254 }
1255 }
1256 }
1257 }
1258 }
1259 }
1260 }
1261 }
1262 }
1263 }
1264 }
1265 }
1266 }
1267 }
1268 }
1269 }
1270 }
1271 }
1272 }
1273 }
1274 }
1275 }
1276 }
1277 }
1278 }
1279 }
1280 }
1281 }
1282 }
1283 }
1284 }
1285 }
1286 }
1287 }
1288 }
1289 }
1290 }
1291 }
1292 }
1293 }
1294 }
1295 }
1296 }
1297 }
1298 }
1299 }
1300 }
1301 }
1302 }
1303 }
1304 }
1305 }
1306 }
1307 }
1308 }
1309 }
1310 }
1311 }
1312 }
1313 }
1314 }
1315 }
1316 }
1317 }
1318 }
1319 }
1320 }
1321 }
1322 }
1323 }
1324 }
1325 }
1326 }
1327 }
1328 }
1329 }
1330 }
1331 }
1332 }
1333 }
1334 }
1335 }
1336 }
1337 }
1338 }
1339 }
1340 }
1341 }
1342 }
1343 }
1344 }
1345 }
1346 }
1347 }
1348 }
1349 }
1350 }
1351 }
1352 }
1353 }
1354 }
1355 }
1356 }
1357 }
1358 }
1359 }
1360 }
1361 }
1362 }
1363 }
1364 }
1365 }
1366 }
1367 }
1368 }
1369 }
1370 }
1371 }
1372 }
1373 }
1374 }
1375 }
1376 }
1377 }
1378 }
1379 }
1380 }
1381 }
1382 }
1383 }
1384 }
1385 }
1386 }
1387 }
1388 }
1389 }
1390 }
1391 }
1392 }
1393 }
1394 }
1395 }
1396 }
1397 }
1398 }
1399 }
1400 }
1401 }
1402 }
1403 }
1404 }
1405 }
1406 }
1407 }
1408 }
1409 }
1410 }
1411 }
1412 }
1413 }
1414 }
1415 }
1416 }
1417 }
1418 }
1419 }
1420 }
1421 }
1422 }
1423 }
1424 }
1425 }
1426 }
1427 }
1428 }
1429 }
1430 }
1431 }
1432 }
1433 }
1434 }
1435 }
1436 }
1437 }
1438 }
1439 }
1440 }
1441 }
1442 }
1443 }
1444 }
1445 }
1446 }
1447 }
1448 }
1449 }
1450 }
1451 }
1452 }
1453 }
1454 }
1455 }
1456 }
1457 }
1458 }
1459 }
1460 }
1461 }
1462 }
1463 }
1464 }
1465 }
1466 }
1467 }
1468 }
1469 }
1470 }
1471 }
1472 }
1473 }
1474 }
1475 }
1476 }
1477 }
1478 }
1479 }
1480 }
1481 }
1482 }
1483 }
1484 }
1485 }
1486 }
1487 }
1488 }
1489 }
1490 }
1491 }
1492 }
1493 }
1494 }
1495 }
1496 }
1497 }
1498 }
1499 }
1500 }
1501 }
1502 }
1503 }
1504 }
1505 }
1506 }
1507 }
1508 }
1509 }
1510 }
1511 }
1512 }
1513 }
1514 }
1515 }
1516 }
1517 }
1518 }
1519 }
1520 }
1521 }
1522 }
1523 }
1524 }
1525 }
1526 }
1527 }
1528 }
1529 }
1530 }
1531 }
1532 }
1533 }
1534 }
1535 }
1536 }
1537 }
1538 }
1539 }
1540 }
1541 }
1542 }
1543 }
1544 }
1545 }
1546 }
1547 }
1548 }
1549 }
1550 }
1551 }
1552 }
1553 }
1554 }
1555 }
1556 }
1557 }
1558 }
1559 }
1560 }
1561 }
1562 }
1563 }
1564 }
1565 }
1566 }
1567 }
1568 }
1569 }
1570 }
1571 }
1572 }
1573 }
1574 }
1575 }
1576 }
1577 }
1578 }
1579 }
1580 }
1581 }
1582 }
1583 }
1584 }
1585 }
1586 }
1587 }
1588 }
1589 }
1590 }
1591 }
1592 }
1593 }
1594 }
1595 }
1596 }
1597 }
1598 }
1599 }
1600 }
1601 }
1602 }
1603 }
1604 }
1605 }
1606 }
1607 }
1608 }
1609 }
1610 }
1611 }
1612 }
1613 }
1614 }
1615 }
1616 }
1617 }
1618 }
1619 }
1620 }
1621 }
1622 }
1623 }
1624 }
1625 }
1626 }
1627 }
1628 }
1629 }
1630 }
1631 }
1632 }
1633 }
1634 }
1635 }
1636 }
1637 }
1638 }
1639 }
1640 }
1641 }
1642 }
1643 }
1644 }
1645 }
1646 }
1647 }
1648 }
1649 }
1650 }
1651 }
1652 }
1653 }
1654 }
1655 }
1656 }
1657 }
1658 }
1659 }
1660 }
1661 }
1662 }
1663 }
1664 }
1665 }
1666 }
1667 }
1668 }
1669 }
1670 }
1671 }
1672 }
1673 }
1674 }
1675 }
1676 }
1677 }
1678 }
1679 }
1680 }
1681 }
1682 }
1683 }
1684 }
1685 }
1686 }
1687 }
1688 }
1689 }
1690 }
1691 }
1692 }
1693 }
1694 }
1695 }
1696 }
1697 }
1698 }
1699 }
1700 }
1701 }
1702 }
1703 }
1704 }
1705 }
1706 }
1707 }
1708 }
1709 }
1710 }
1711 }
1712 }
1713 }
1714 }
1715 }
1716 }
1717 }
1718 }
1719 }
1720 }
1721 }
1722 }
1723 }
1724 }
1725 }
1726 }
1727 }
1728 }
1729 }
1730 }
1731 }
1732 }
1733 }
1734 }
1735 }
1736 }
1737 }
1738 }
1739 }
1740 }
1741 }
1742 }
1743 }
1744 }
1745 }
1746 }
1747 }
1748 }
1749 }
1750 }
1751 }
1752 }
1753 }
1754 }
1755 }
1756 }
1757 }
1758 }
1759 }
1760 }
1761 }
1762 }
1763 }
1764 }
1765 }
1766 }
1767 }
1768 }
1769 }
1770 }
1771 }
1772 }
1773 }
1774 }
1775 }
1776 }
1777 }
1778 }
1779 }
1780 }
1781 }
1782 }
1783 }
1784 }
1785 }
1786 }
1787 }
1788 }
1789 }
1790 }
1791 }
1792 }
1793 }
1794 }
1795 }
1796 }
1797 }
1798 }
1799 }
1800 }
1801 }
1802 }
1803 }
1804 }
1805 }
1806 }
1807 }
1808 }
1809 }
1810 }
1811 }
1812 }
1813 }
1814 }
1815 }
1816 }
1817 }
1818 }
1819 }
1820 }
1821 }
1822 }
1823 }
1824 }
1825 }
1826 }
1827 }
1828 }
1829 }
1830 }
1831 }
1832 }
1833 }
1834 }
1835 }
1836 }
1837 }
1838 }
1839 }
1840 }
1841 }
1842 }
1843 }
1844 }
1845 }
1846 }
1847 }
1848 }
1849 }
1850 }
1851 }
1852 }
1853 }
1854 }
1855 }
1856 }
1857 }
1858 }
1859 }
1860 }
1861 }
1862 }
1863 }
1864 }
1865 }
1866 }
1867 }
1868 }
1869 }
1870 }
1871 }
1872 }
1873 }
1874 }
1875 }
1876 }
1877 }
1878 }
1879 }
1880 }
1881 }
1882 }
1883 }
1884 }
1885 }
1886 }
1887 }
1888 }
1889 }
1890 }
1891 }
1892 }
1893 }
1894 }
1895 }
1896 }
1897 }
1898 }
1899 }
1900 }
1901 }
1902 }
1903 }
1904 }
1905 }
1906 }
1907 }
1908 }
1909 }
1910 }
1911 }
1912 }
1913 }
1914 }
1915 }
1916 }
1917 }
1918 }
1919 }
1920 }
1921 }
1922 }
1923 }
1924 }
1925 }
1926 }
1927 }
1928 }
1929 }
1930 }
1931 }
1932 }
1933 }
1934 }
1935 }
1936 }
1937 }
1938 }
1939 }
1940 }
1941 }
1942 }
1943 }
1944 }
1945 }
1946 }
1947 }
1948 }
1949 }
1950 }
1951 }
1952 }
1953 }
1954 }
1955 }
1956 }
1957 }
1958 }
1959 }
1960 }
1961 }
1962 }
1963 }
1964 }
1965 }
1966 }
1967 }
1968 }
1969 }
1970 }
1971 }
1972 }
1973 }
1974 }
1975 }
1976 }
1977 }
1978 }
1979 }
1980 }
1981 }
1982 }
1983 }
1984 }
1985 }
1986 }
1987 }
1988 }
1989 }
1990 }
1991 }
1992 }
1993 }
1994 }
1995 }
1996 }
1997 }
1998 }
1999 }
2000 }
2001 }
2002 }
2003 }
2004 }
2005 }
2006 }
2007 }
2008 }
2009 }
2010 }
2011 }
2012 }
2013 }
2014 }
2015 }
2016 }
2017 }
2018 }
2019 }
2020 }
2021 }
2022 }
2023 }
2024 }
2025 }
2026 }
2027 }
2028 }
2029 }
2030 }
2031 }
2032 }
2033 }
2034 }
2035 }
2036 }
2037 }
2038 }
2039 }
2040 }
2041 }
2042 }
2043 }
2044 }
2045 }
2046 }
2047 }
2048 }
2049 }
2050 }
2051 }
2052 }
2053 }
2054 }
2055 }
2056 }
2057 }
2058 }
2059 }
2060 }
2061 }
2062 }
2063 }
2064 }
2065 }
2066 }
2067 }
2068 }
2069 }
2070 }
2071 }
2072 }
2073 }
2074 }
2075 }
2076 }
2077 }
2078 }
2079 }
2080 }
2081 }
2082 }
2083 }
2084 }
2085 }
2086 }
2087 }
2088 }
2089 }
2090 }
2091 }
2092 }
2093 }
2094 }
2095 }
2096 }
2097 }
2098 }
2099 }
2100 }
2101 }
2102 }
2103 }
2104 }
2105 }
2106 }
2107 }
2108 }
2109 }
2110 }
2111 }
2112 }
2113 }
2114 }
2115 }
2116 }
2117 }
2118 }
2119 }
2120 }
2121 }
2122 }
2123 }
2124 }
2125 }
2126 }
2127 }
2128 }
2129 }
2130 }
2131 }
2132 }
2133 }
2134 }
2135 }
2136 }
2137 }
2138 }
2139 }
2140 }
2141 }
2142 }
2143 }
2144 }
2145 }
```

```

98 public function executeCheckProgress(sfWebRequest $request) {
99
100     $uniqId = $request->getParameter('progress_key');
101
102     if (!empty($uniqId)) {
103
104         $status = apc_fetch('upload_'. $uniqId);
105
106         return $this->renderText( json_encode($status) );
107     }
108 }

```

Abbildung 5.12: Action-Methode executeProgress mit der Logik des Fortschrittsbalkens

```

1  jQuery(document).ready(function() {
2
3      var flobu;
4
5      flobu = new flower_bubble ({
6          base_obj: $('#mediaUploadForm' ),
7          base_dir: document.location.protocol + "://" + document.location.hostname,
8          background: {
9              css: 'white',
10             opacity: 0.78
11         },
12         bubble: {
13             image: '/images/bubble3.png',
14             width: 265,
15             height: 44
16         },
17         flower: {
18             image: '/images/flower.gif',
19             width: 200,
20             height: 44
21         },
22         block_mode: 'base_obj'
23     });
24
25     var checkProgress = function (uniqId) {
26         $.getJSON(
27             documentPath + '/media/checkProgress',
28             { progress_key: uniqId },
29             function JSON(json) {
30                 $('#progressbar').show();
31                 var progress = eval(json.current/json.total*100);
32                 if ( !isNaN(progress) ) {
33                     $('#progressbar').reportprogress(progress);
34                 }
35             }
36         );
37     }

```

Abbildung 5.13: JavaScript-Quelltext der Logik des Fortschrittsbalkens

5.5.2 Video-Anzeige

Die Video-Anzeige besteht aus der Action-Methode `executeIndex`, die standardmäßig das zuletzt hochgeladene Video lädt. [Abbildung 5.14](#) zeigt den Aufbau der Methode `executeIndex`.

```

8 | * @author      Sufian Abu-Rab
9 | * @version     SVN: $Id: actions.class.php 23810 2009-11-12 11:07:44Z Kris.Walls:mith
10 | */
11 | class mediaActions extends sfActions {
12 |
13 |     public function executeIndex(sfWebRequest $request) {
14 |
15 |         $this->videos = array();
16 |         $this->generatedNameLastVideo = "";
17 |         $this->generatedNameNextVideo = "";
18 |         $this->id = $this->getRequestParameter('id');
19 |
20 |         if (empty($this->id)){
21 |             $result = Doctrine::getTable('media')->createQuery('a')->execute();
22 |             if (!empty($result)){
23 |                 $this->media = $result->getLast();
24 |             }
25 |         }else{
26 |             $query = Doctrine_Query::create()->from('media j')->
27 |                 where('j.video = ?', $this->id)->limit(1);
28 |
29 |             $result = $query->execute();
30 |             $this->media = $result->getFirst();
31 |         }
32 |
33 |         if (!empty($this->media)){
34 |
35 |             $generatedName = $this->media->getVideo();
36 |
37 |             $title = $this->media->getTitle();
38 |             if (!empty($generatedName) && !empty($title)){
39 |                 $this->videos[$generatedName] = $title;
40 |             }
41 |
42 |             $generatedNameLastVideo = $this->media->getLastVideo();
43 |             if (!empty($generatedNameLastVideo)){
44 |                 $this->generatedNameLastVideo = $generatedNameLastVideo;
45 |             }
46 |             $generatedNameNextVideo = $this->media->getNextVideo();
47 |             if (!empty($generatedNameNextVideo)){
48 |                 $this->generatedNameNextVideo = $generatedNameNextVideo;
49 |             }
50 |         }
51 |     }

```

Abbildung 5.14: Aufbau der Action-Methode executeIndex

Falls die Methode `executeIndex` einen ID-Parameter empfängt, wird anstatt des Startseiten-Videos der Datensatz mit der jeweiligen ID aus der Datenbank geladen. Aus diesem Datensatz wird, falls vorhanden, der Titel, die IDs des letzten und des nächsten Videos bezogen und an das Template `indexSuccess` übergeben. Das Template nimmt die Parameter der Methode `executeIndex` entgegen und bindet diese in den HTML-Kontext ein. Hierbei wird der erzeugte Name des Videos in einem Verweis (Link-Tag) integriert. Dieser Verweis wird mit Hilfe der JavaScript-Datei `jquery.flowplayer.js` zur

Laufzeit in ein Video-Objekt umgewandelt, das auf den generierten Namen des zuvor erstellten Verweises zeigt. Abbildung 5.15 enthält den HTML-Quelltext des Templates `indexSuccess`.

```
<div class="indexVideoItem">
  <span class="indexVideoName">
    <?php echo $title; ?>
  </span>
  <a class="rtmp" href="<?php echo $generatedName; ?>" ></a>
</div>
```

Abbildung 5.15: HTML-Quelltext des Templates `indexSuccess`

Automatische Wiedergabe

Zur Laufzeit durchsucht das Skript `jquery.flowplayer.js` alle Verweise der Klasse `rtmp` und wandelt diese in ein Video-Objekt um, wodurch die Anforderung aus Kapitel 5.3.2 umgesetzt ist. Dieser Kodierungsprozess kann über das Attribut `clip` festgelegt werden. Über Optionen des Attributs können u.a. das Streaming-Protokoll, der Pufferstart, sowie der Autostart festgelegt werden. Ebenso wird über einen Verweis zum Flowplayer und zum CloudFront-Streaming-Dienst der Flash-Player und die Quelle des Videos bestimmt. Abbildung 5.16 zeigt den JavaScript-Quelltext des Templates `indexSuccess`.

Implementierung der automatischen Wiedergabe

Die Anforderung aus Kapitel 5.3.2 setzt die Option `autoPlay:true` um. Diese sorgt dafür, dass die Videos sofort bei Aufruf abgespielt werden. Zudem können Callback-Funktionen definiert werden. Für die Realisierung einer Wiedergabeliste ist es nötig bei der Beendigung eines Videos automatisch auf das nächste Video zu wechseln. Dieses wird realisiert durch die Callback-Funktion `onFinish: function()...`. Diese Funktion enthält den Befehl `window.location.href`, der beim Aufruf die übergebene Ziel-URL im Browser lädt. Am Ende der Wiedergabe führt die Callback-Funktion `onFinish` den Befehl `window.location.href = „http://global-video.de/media/index/id/“ + nextVideo` aus. Die Variable `nextVideo` enthält die ID des letzten Videos. Obwohl der Flowplayer bereits eine Wiedergabeliste anbietet, ist von deren Verwendung abzuraten, da diese die „don't break the Back-Button“-Regel bricht [49]. Stattdessen ist in der implementierten Lösung die Navigation der Wiedergabeliste im Template `indexSuccess` integriert. Falls ein vorheriges oder nächstes Video existiert, ist dieses in Form einer Vor- und Zurück-Schaltfläche aufrufbar.

```

95 <script type="text/javascript">
96     //http://flowplayer.org/documentation/api/index.html
97     $f("a.rtmp", "http://s3-eu-west-1.amazonaws.com/schlaubi/flowplayer-3.2.5.swf", {
98         // This next code configures the player to use rtmp plugin for streaming.
99         clip: {
100             provider: 'rtmp',
101             autoPlay: true,
102             autoBuffering: true,
103             onFinish: function(){
104                 var nextVideo = "<?php echo $generatedNameLastVideo;?>";
105                 if (nextVideo != ""){
106                     window.location.href = document.location.protocol + "://"
107                         + document.location.hostname + "/media/index/id/" + nextVideo;
108                 }
109             },
110             // Here is our rtpm plugin configuration
111             plugins: {
112                 rtmp: {
113                     url: 'http://s3-eu-west-1.amazonaws.com/schlaubi/flowplayer.rtmp-3.2.3.swf',
114                     netConnectionUrl: 'rtmp://s2h4very7f7oo5.cloudfront.net/cfx/st'
115                 }
116             }
117         }
118     });
119 </script>

```

Abbildung 5.16: Logik des Flowplayers im Template indexSuccess

5.6 Ergebnis

Die in dieser Arbeit erstellte Lösung zum Video-Streaming ermöglicht sowohl das Hochladen, als auch Anzeigen von Videos. Das Hochladen von Videos ist ohne Registrierung möglich. Ein Video darf maximal 120 Megabyte groß sein. Eine Beschränkung der maximalen Laufzeit wurde nicht gesetzt. Der Server überprüft hochzuladende Videos auf ihre Größe und das Dateiformat. Die Eingabepfung findet serverseitig statt, wodurch längere Wartezeiten zustande kommen können bis das Resultat der Prüfung zurückgegeben werden kann. Der Grund dafür ist, dass nur vollständig übertragene Dateien geprüft werden können. Währenddessen zeigt ein Fortschrittsbalken den aktuellen Status beim Hochladen an. Abbildung 5.17 zeigt den Versuch ein Video mit dem Dateityp JPEG hochzuladen.

Die Fehlermeldung ist rot gekennzeichnet und weist auf einen falschen Dateityp hin. Kompatible Videos werden an den Speicherdienst S3 weitergeleitet. Abbildung 5.18 zeigt den Fortschrittsbalken während des Hochladens eines Videos.

Eine Eingabepfung auf Seiten des Clients ist nur teilweise realisierbar, da es nicht möglich ist, über JavaScript, die Binärdaten der markierten Datei auszulesen, um dadurch eine Überprüfung des Dateityps vorzunehmen. Das heißt, es ist ebenfalls nicht möglich, über eine AJAX-Anfrage ein Formarelement des HTML-Typs `file` zu versenden. Lediglich der Dateiname ist clientseitig überprüfbar. Eine Auswertung des Dateityps

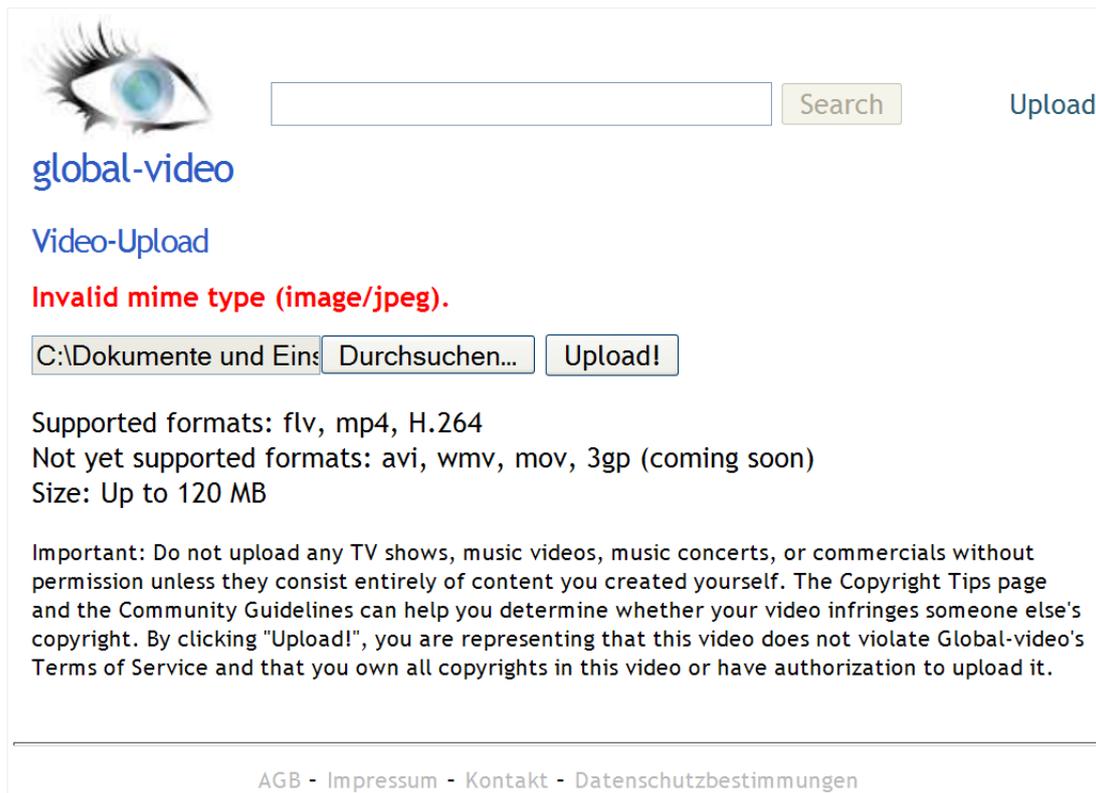


Abbildung 5.17: Fehlgeschlagener Versuch eine JPEG-Bilddatei hochzuladen

auf dem Client ist von einer vollständigen Dateibenennung des Benutzers abhängig. Das heißt, dass der Dateityp immer dann nicht ermittelt werden kann, wenn die Dateieindung falsch geschrieben oder nicht vorhanden ist.

Die Video-Anzeige besteht aus einem Bereich für die Wiedergabe des Videos, sowie einem Bereich für die Vorwärts- und Rückwärtsnavigation der Wiedergabeliste. Ebenso sind die Funktionen für Bewertung, Wiederholung und Suchen in die Webanwendung eingebettet, wobei deren Logik noch zu implementieren ist. Abbildung 5.19 zeigt die Video-Anzeige während der Wiedergabe eines Videos.

Auf der Startseite ist das zuletzt hochgeladene Video zu sehen. Am Ende der Wiedergabe eines Videos wird das zuvor hochgeladene Video aufgerufen und sofort abgespielt.

5.7 Vergleich mit der existierenden Lösungen YouTube

Über das Internetportal YouTube können die Benutzer Videos ansehen und hochladen. YouTube enthält eine umfangreiche Datenbank mit Filmen und Musikvideos. Der Erfolg lässt sich anhand der wachsenden Zahl der Benutzer, die Videos hochladen, bewerten

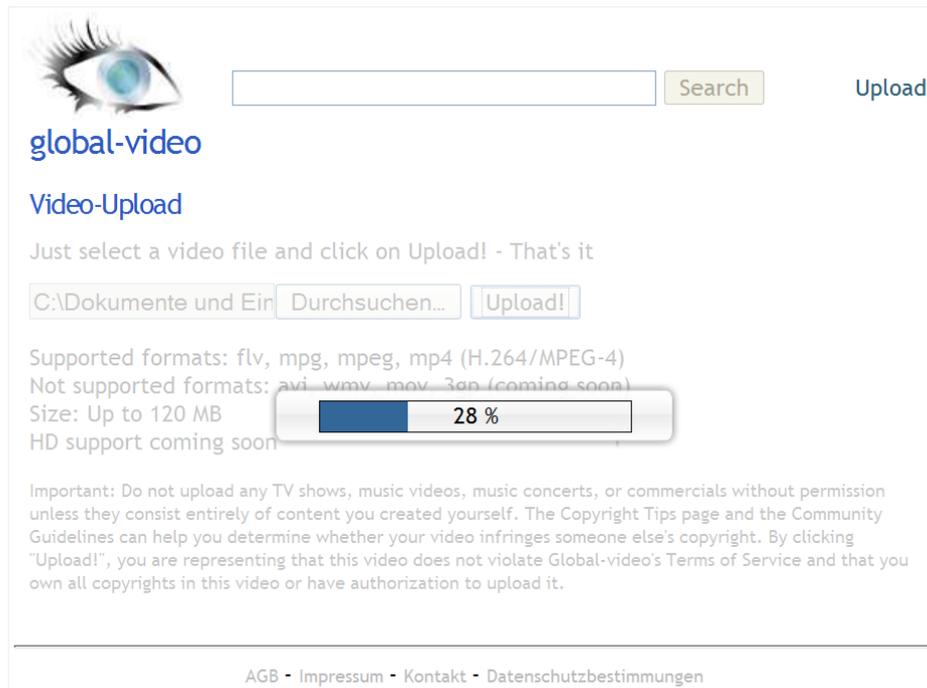


Abbildung 5.18: Fortschrittsbalken beim Hochladen eines Videos

und mit anderen teilen. Ebenso bietet YouTube den Benutzern die Möglichkeit, mit ihren eigenen Videos Geld zu verdienen, indem Werbung während der Wiedergabe geschaltet wird. Zusätzlich ist YouTube mit allen gängigen Endgeräten, wie z.B. Smartphones, kompatibel. YouTube besitzt auch ein ausgefeiltes Caching-System [27]. Ein Nachteil von YouTube ist, dass das Hochladen eines Videos nur mit einer vorherigen Registrierung möglich ist.

Im Vergleich zu YouTube enthält die in dieser Arbeit entwickelte Video-Streaming-Lösung weder eine umfangreiche Video-Datenbank, noch eine Benutzerverwaltung. Daher sind das Erstellen einer eigenen Wiedergabeliste und die Möglichkeit mit eigenen Videos Geld zu verdienen nicht möglich. Zudem ist das Abspielen von Videos mit dem Apple iPhone nicht möglich, da dieses keine den Flash-Player nicht unterstützt. Das in dieser Arbeit verwendete Caching-Verfahren hat verschiedene positive und negative Auswirkungen. Der Unterschied zu YouTube besteht darin, dass die Daten nicht auf dem Client zwischengespeichert werden, was positiv in Bezug auf die Datensicherheit, jedoch negativ bei der erneuten Wiedergabe eines bereits geladenen Videos ist. YouTube verwendet hierfür das sogenannte *Pseudostreaming*. Dabei handelt es sich um eine Mischung aus Download und Streaming. Ein Vorteil von Pseudostreaming ist, dass das Abspielen eines schon geladenen Videos auch ohne bestehende Internetverbindung möglich ist. Ein Nachteil ist, dass die Videos einfach kopiert werden können.

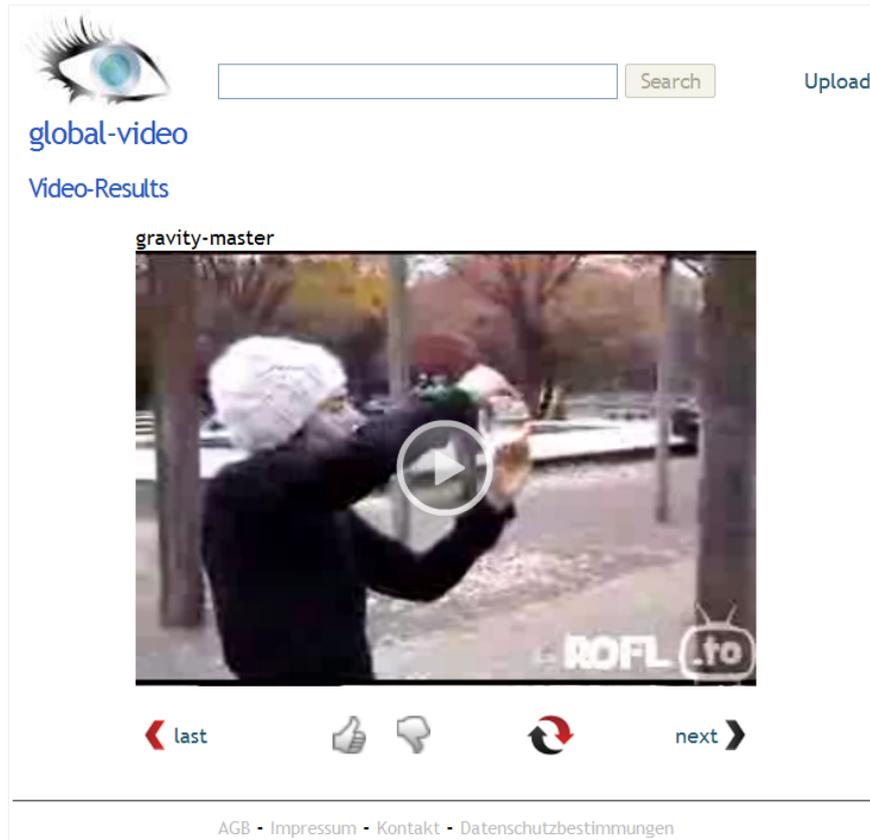


Abbildung 5.19: Wiedergabe eines Videos

Ein Vorteil der in dieser Arbeit entwickelten Streaming-Lösung ist der Verzicht auf eine Benutzerauthentifizierung. Dadurch ist für den Benutzer nicht nur das anonyme Hochladen, sondern auch die direkte Nutzung des Dienstes möglich. Eine Möglichkeit, die Serverkosten zu finanzieren ist durch Werbung. Ein weiterer Vorteil dieser Lösung ist der Einsatz von Cloud-Diensten (Rechen- und Speicherkapazität). Dadurch entfallen die Kosten für Anschaffung, Betrieb, Administration und Wartung für eigene Hardware. Durch den Dienst CloudFront können Videos global mit ähnlichen Übertragungszeiten verteilt werden. Ein weiterer wichtiger Vorteil der hier entwickelten Streaming-Lösung sind die selbst bestimmbaren Nutzungs- und Datenschutzbedingungen. Wie im Zitat auf Seite 14 beschrieben ist, werden beim Hochladen eines Videos YouTube diverse Rechte eingeräumt. Das heißt, dass YouTube das Recht hat, die hochgeladenen Videos der Nutzer zu nutzen, zu vervielfältigen und diese Rechte auch an Dritte weiterzugeben.

5.8 Erweiterungsmöglichkeiten

Die Funktionen zur Wiederholung und Bewertung könnten zunächst über AJAX-Anfragen an den Server realisiert werden. Ein weiterer Erweiterungspunkt ist die Implementierung der Sicherheitsvorkehrungen gegen Verfälschungen der Bewertungsstatistiken. Des Weiteren können die Sicherheitsvorkehrungen für das Hochladen von Videos verbessert werden. Aktuell besteht kein Schutz gegen fremde Skripte zum Hochladen.

Eine weitere sinnvolle Erweiterung ist die Suche mit Hilfe der YouTube-API [53] auf die YouTube-Datenbank umzuleiten. Dadurch könnten eigene Statistiken über die Relevanz bestimmter Suchbegriffe erstellt und gleichzeitig dem Benutzer schnelleren Zugriff auf Videos zu gewährleisten werden (siehe [40]).

Zusätzlich können Leistungsmessungen mit dem Werkzeug *jmeter* [30] durchgeführt werden, um die maximale Auslastungsfähigkeit der Infrastrukturdienste von Amazon festzustellen.

Kapitel 6

Zusammenfassung

Für die in Amazons öffentlich verfügbaren Cloud-Diensten erstellte Infrastruktur für Video-Streaming, stellt der Dienst EC2 Webserver zur Verfügung. Der Webserver ist über den Dienst EBS mit einem persistenten Speicher gekoppelt, der als Ablageort für die Anwendung dient. Mit dem Datenbankdienst RDS stellt Amazon ein skalierbares MySQL-Datenbanksystem zur Verfügung. Die Videos werden über den Speicherdienst S3 mehrfach redundant aufbewahrt. Die Streaming-Funktionalität realisiert der Dienst CloudFront. Der Dienst unterstützt das Netzwerkprotokoll RTMP, sowie die schnelle Verteilung der Daten an den Client über mehrere Edge-Standorte. Zusätzlich kümmert sich der Dienst für elastische Lastverteiler ELB um eine gleichmäßige Lastenverteilung. Schließlich realisiert der Dienst AutoScale eine automatische Anpassung der Ressourcen an den Verbrauch. Diese Komponenten zusammen bilden die grundlegende Infrastruktur der Streaming-Lösung.

Die Streaming-Webanwendung ist in der Skriptsprache PHP mit dem PHP-Framework Symfony realisiert. Die Webanwendung ermöglicht sowohl das Hochladen als auch das Anzeigen der Videos. Das Hochladen eines Videos ist ohne vorherige Benutzerregistrierung möglich. Die Eingabepfung wird anhand der Dateigröße und des Videoformates vorgenommen. Zusätzlich wird der Prozess des Hochladens über einen Fortschrittsbalken grafisch dargestellt. Ebenso wird das zuletzt hochgeladene Video automatisch an das Ende der Wiedergabeliste angefügt. Die Wiedergabeliste wird auf der Startseite angezeigt und zeigt das zuletzt hochgeladene Video, welches direkt abgespielt wird. Beim Ende der Wiedergabe startet das nächste, zuvor hochgeladene, Video. Beim Erreichen des letzten Videos in der Wiedergabeliste, wird diese deaktiviert.

6.1 Herausforderungen bei der Entwicklung

Herausforderungen gab es bei der Einschätzung voraussichtlicher Kosten und des Ressourcenverbrauchs. Sowohl das Preismodell als auch die Zusammenarbeit der Amazon Cloud-Dienste ist komplex, weshalb genaue Vorhersagen für den eigenen Ressourcenverbrauch und die damit verbundenen Kosten schwierig sind.

Nicht alle Anforderungen konnten in dieser Arbeit erfüllt werden. Eine unerfüllte Anforderung ist eine Unterstützung aller gängigen Dateiformate für Videos.

Eine weitere Herausforderung war die Implementierung des Fortschrittsbalkens, der mit Hilfe von JavaScript und AJAX-Aufrufen realisiert werden sollte. Ein Problem hierbei war, dass beim Hochladen einer Datei, diese selbst nicht über eine `http`-Anfrage an den Server geschickt werden kann. Die Lösung ist der Einsatz eines `iframe`-Fensters, welches als Quelle für den Versand des Formulars zum Hochladen diene.

Zukünftige Arbeiten sind die Bewertungsfunktion und die Schaltfläche zum Wiederholen. Eine weitere Erweiterungsmöglichkeit ist die Schaltfläche „Video melden“, mit der die Benutzer auf anstößige oder rassistische Videoinhalte aufmerksam machen können. Zusätzlich wäre eine Altersabfrage eine sinnvolle Erweiterungsmöglichkeit, die aber mit der Realisierung einer Benutzerverwaltung verbunden ist. Auch müssten die Videoinhalte in einem solchen Szenario für verschiedene Altersgruppen eingestuft werden.

Eine Benutzerverwaltung, verbunden mit einem System zur Authentifizierung und der Möglichkeit, eigene Wiedergabelisten anzulegen, ist eine weitere Erweiterungsmöglichkeit. Hierfür kann die Authentifizierungsschnittstelle von YouTube verwendet werden, um den bei YouTube registrierten Benutzern, eine erneute Registrierung zu ersparen. Andererseits wären die Benutzer dadurch gezwungen, bei YouTube Benutzerkonten anzulegen.

Für die Implementierung einer Suchfunktion existieren verschiedene Möglichkeiten. Zum einen könnte die Suche mit Hilfe der frei verfügbaren YouTube-Data-API [53] realisiert werden. Hierbei würden die Suchanfragen an YouTube weitergeleitet. Dadurch könnte von YouTube großer Videodatenbank profitiert werden. Der Nachteil ist, dass hierdurch die Nutzer mit Werbung rechnen müssen. YouTube fügt nur bei YouTube-Partnern und nicht bei Privatvideos Werbung ein. Zudem könnten so Statistiken über Suchbegriffe, Aufrufe und Bewertungen erstellt werden.

6.2 Ausblick

Laut [18] können Unternehmen mit Hilfe von Cloud Computing „produktiver, kostengünstiger und flexibler arbeiten“. Zusätzlich können sie sich mit Hilfe von Cloud Computing erstmals vom „klassischen, statischen Rechenzentrum emanzipieren“. Dies ist vor allem dann sinnvoll, „wenn das Management von Rechenzentren nicht zu den Kernkompetenzen eines Unternehmens gehört“. Ebenso ist laut [18] davon auszugehen, dass „mittelfristig vor allem in den mittelständischen Betrieben die internen Unternehmens-Rechenzentren von der IT-Landkarte verschwinden.“

Ebenso sind inzwischen laut [44] 60 Prozent aller Haushalte in Deutschland mit einer ausreichenden Bandbreite für Video-Streaming ausgestattet. Die Statistik von YouTube zeigt die wachsende Beliebtheit solcher Dienste. Laut [43] beträgt die durchschnittliche Aufenthaltszeit auf YouTube 18 Minuten, wohingegen diese laut [42] beim Fernsehen, wie z.B. das Fernsehen bei bis zu 10 Stunden pro Tag liegt. Diese Differenz zeigt wie viel Potenzial im Video-Streaming steckt.

Abbildungsverzeichnis

2.1	Marktvolumen für Cloud Computing	5
2.2	Schichtenmodell der Kategorien von Cloud-Diensten	8
2.3	Amazon Management Console	9
3.1	Startseite von YouTube.com	13
4.1	Aufbau einer typischen Drei-Schichten-Architektur	17
4.2	Nutzung der AWS im Rahmen einer Drei-Schichten-Architektur	18
4.3	Startpunkt für neue Instanzen	21
4.4	Instanztypen in EC2	22
4.5	Freigegebene Portnummern einer Sicherheitsgruppe	23
4.6	Auswahlmöglichkeit der zuvor erstellten Schlüsselpaare	23
4.7	RDS-Fenster in der Amazon Management Console	24
4.8	Einstellungsmöglichkeiten für die Datenbankinstanz	25
4.9	Einstellungsmöglichkeiten für das automatische Anlegen von Sicherheitskopien in RDS	26
4.10	Einstellungen der Sicherheitsgruppe der RDS-Instanz	27
4.11	S3-Fenster in der Amazon Management Console	27
4.12	Dialogfenster zum Erstellen einer Streaming-Unterstützung für ein Bucket	28
4.13	Integration des Dienstes ELB in die Streaming-Architektur	29
4.14	Kosten für die Infrastruktur	30
5.1	Datenbankschema der Video-Datenbank	32
5.2	Hauptanwendungsfälle der Benutzer und Administratoren	33
5.3	Offizielle AMIs in EC2 von Ubuntu	36
5.4	Initialisierungsmethode configure des mediaForm-Formulars	38
5.5	Vorverarbeitungsmethode prozessValues des Formulars	40
5.6	Aufbau der Methode executeNew	40
5.7	Aufbau des Templates newSuccess zum Hochladen	41
5.8	JavaScript-Teil des Formulars zum Hochladen der Videos	42
5.9	Aufbau der executeUpload-Methode	43

5.10 Aufbau der Validierungs-Klasse sfValidatedFileCustom	44
5.11 Fortsetzung der Validierungsklasse sfValidatedFileCustom	44
5.12 Action-Methode executeProgress mit der Logik des Fortschrittsbalkens	45
5.13 JavaScript-Quelltext der Logik des Fortschrittsbalkens	45
5.14 Aufbau der Action-Methode executeIndex	46
5.15 HTML-Quelltext des Templates indexSuccess	47
5.16 Logik des Flowplayers im Template indexSuccess	48
5.17 Fehlgeschlagener Versuch eine JPEG-Bilddatei hochzuladen	49
5.18 Fortschrittsbalken beim Hochladen eines Videos	50
5.19 Wiedergabe eines Videos	51

Literaturverzeichnis

- [1] ADOBE FLASH MEDIA SERVER: *Flash Media Encoder*. <http://www.adobe.com/products/flashmediaserver/flashmediaencoder/>. Version: April 2011
- [2] ADOBE FLASH MEDIA SERVER: *Flash Media Server Family*. <http://www.adobe.com/de/products/flashmediaserver/>. Version: April 2011
- [3] AMAZON WEB SERVICES: *CDN - Content Distribution Network - Amazon CloudFront*. <http://aws.amazon.com/de/cloudfront/>. Version: April 2011
- [4] AMAZON WEB SERVICES: *EBS-Speichervolumen*. <https://aws.amazon.com/de/ebs>. Version: April 2011
- [5] AMAZON WEB SERVICES: *Einfache Datenbank API - Amazon SimpleDB*. <http://aws.amazon.com/de/simpledb/>. Version: April 2011
- [6] AMAZON WEB SERVICES: *Import und Export von Daten nach und aus Amazon*. <http://aws.amazon.com/de/importexport/>. Version: April 2011
- [7] AMAZON WEB SERVICES: *Lastausgleichserver - Amazon ELB*. <http://aws.amazon.com/de/elasticloadbalancing/>. Version: April 2011
- [8] AMAZON WEB SERVICES: *Relationaler Datenbankdienst - Amazon RDS*. <http://aws.amazon.com/de/rds/>. Version: April 2011
- [9] AMAZON WEB SERVICES: *Speicherdienst Simple Storage Server*. <https://aws.amazon.com/de/s3/>. Version: April 2011
- [10] AMAZON WEB SERVICES: *Warteschlangendienst - Amazon SQS*. <http://aws.amazon.com/de/sqs/>. Version: April 2011
- [11] APACHE HTTP SERVER: *The Number One HTTP Server On The Internet*. <http://httpd.apache.org/>. Version: April 2011
- [12] BAUN, Christian ; KUNZE, Marcel ; NIMIS, Jens ; TAI, Stefan: *Cloud Computing: Web-basierte dynamische IT-Services*. Berlin, Heidelberg / Berlin : Springer, 2010

(Informatik im Fokus). – ISBN 978–3–642–01593–9

- [13] BMWI: *Prognostiziertes Marktvolumen für Cloud Computing in Deutschland in den Jahren 2010 bis 2013*. <http://de.statista.com/statistik/daten/studie/165458/umfrage/prognostiziertes-marktvolumen-fuer-cloud-computing-in-deutschland/>. Version:2010
- [14] CLIPFISH.DE: *Videoportal*. <http://www.clipfish.de>. Version:April 2011
- [15] CLOUDCOMPUTING21.DE: *Microsoft Deutschland-Chef gibt Ausblick 2011*. <http://www.cloud-computing-21.de/cloudcomputing-news/artikel/47173-microsoft-deutschland-chef-gibt-ausblick-2011/256/>. Version:16 Dezember 2010
- [16] CLOUDCOMPUTING21.DE: *Microsoft startet „Go Cloud“ Initiative für deutsche IT-Industrie*. <http://www.cloud-computing-21.de/related-cloudcomputing-news/artikel/46159-microsoft-startet-go-cloud-initiative-fuer-deutsche-it-industrie/256/>. Version:29 September 2010
- [17] DR. CARSTEN ULBRICHT: *Risiken für YouTube, MyVideo und Co und deren Haftung*. <http://www.rechtzweinull.de/index.php?/archives/24-Risiken-fuer-YouTube,-MyVideo-Co-und-deren-Nutzer-Haftung-bei-Videoplattformen.html>. Version:16 April 2007
- [18] DR. KLAUS MANHART: *Cloud Computing das müssen Sie wissen SaaS PaaS IaaS*. http://www.tecchannel.de/server/cloud_computing/2030180/cloud_computing_das_muessen_sie_wissen_saas_paas_iaas/index2.html. Version:13 Oktober 2010
- [19] ELLACOYA NETWORKS: *Provider of secure service control solutions for global business networks*. <http://www.ellacoya.com/news/pdf/2007/NXTcommEllacoyaMediaAlert.pdf>. Version:2007
- [20] EUCALYPTUS.COM: *Private Cloud Infrastrukturdienst*. <http://www.eucalyptus.com/>. Version:April 2011
- [21] FACEBOOK.COM: *Social Network*. <http://www.facebook.com/>. Version:April 2011
- [22] FFMPEG: *FFmpeg is a complete, cross-platform solution to record, convert and stream audio and video*. <http://www.ffmpeg.org/>. Version:April 2011
- [23] GOOGLE: *Google Docs - Google Text und Tabellen*. <http://docs.google.com>. Version:April 2011

- [24] GRANT, K.: Reese, G., (2009) .Cloud Application Architectures, O'Reilly Publisher. 208pp, price 22.99. In: *International journal of information management* (2010), S. 2010; VOL 30; NUMBER 5, 470. ISBN 978-0-596-15636-7
- [25] HEISE.DE: *YouTube setzte sich vor Gericht gegen spanischen TV-Sender durch*. <http://www.heise.de/newsticker/meldung/YouTube-setzte-sich-vor-Gericht-gegen-spanischen-TV-Sender-durch-1095151.html>. Version: 23 September 2010
- [26] HEISE.DE: *YouTube: Über 1 Milliarde Videoabrufe pro Tag*. <http://www.heise.de/newsticker/meldung/YouTube-Ueber-1-Milliarde-Videoabrufe-pro-Tag-821259.html>. Version: Oktober 2009
- [27] HIGHSCALABILITY.COM - YOUTUBE ARCHITECTURE: *Help to build scalable websites*. <http://highscalability.com/youtube-architecture>. Version: März 2008
- [28] HITWISE.COM: *Custom Data and Analytics*. <http://www.hitwise.com/us/>. Version: Mai 2006
- [29] IT-TIMES NEW YORK: *YouTube dominiert weiter den Online-Videomarkt*. <http://www.it-times.de/news/nachricht/datum/2010/06/02/youtube-dominiert-weiter-den-online-videomarkt/>. Version: 2 Juni 2010
- [30] JMETER - APACHE JAKARTA PROJECT: *Apache JMeter is open source software, a 100% pure Java desktop application designed to load test functional behavior and measure performance*. <http://jakarta.apache.org/jmeter/>. Version: April 2011
- [31] JQUERY PLUGINS: *jQuery Form Plugin*. <http://jquery.malsup.com/form/>. Version: April 2011
- [32] JUGENDSCHUTZ.NET: *Jugendschutz im Internet - Mehr Rücksicht auf Kinder*. http://www.jugendschutz.net/news/200607/news_06-07-09_18-44-28_fs.html. Version: Juli 2006
- [33] JUSTIZ.HAMBURG.DE: *Landgericht Hamburg verbietet Youtube die Verbreitung urheberrechtswidriger Inhalte*. <http://justiz.hamburg.de/2493868/pressemeldung-2010-09-03.html>. Version: 3 September 2010
- [34] LIGHTTPD WEBSERVER: *high performance webserver*. <http://www.lighttpd.net/>. Version: April 2011
- [35] MEGAVIDEO: *Videportal*. <http://www.megavideo.com/>. Version: April 2011
- [36] METACAFE: *Videportal*. <http://www.metacafe.com/>. Version: April 2011

- [37] MYSQL - SUN MICROSYSTEMS GMBH: *Datenbanksystem - MySQL ist die populärste Open-Source-Datenbank der Welt*. <http://www.mysql.de/>. Version: April 2011
- [38] MYVIDEO: *Videoportal*. <http://www.myvideo.de/>. Version: April 2011
- [39] OPENNEBULA.ORG: *Private Cloud Infrastrukturdienst*. <http://opennebula.org/>. Version: April 2011
- [40] PLAYLIST.EU youtube: *Example of YouTube-Data-API integration*. <http://www.youtube-playlist.eu>. Version: April 2011
- [41] SALESFORCE.COM: *On-Demand CRM - Geschäftsanwendung*. www.salesforce.com. Version: April 2011
- [42] SEVENONE MEDIA: *Videoportale in Deutschland: Im Spannungsfeld zwischen Fernsehen und Internet*. http://appz.sevenonemedia.de/download/publikationen/Videoportale_in_Deutschland.pdf. Version: Januar 2008
- [43] SPIEGEL.DE: *YouTube-Kritiker verstummen nach Deal mit Google*. <http://www.spiegel.de/netzwelt/web/0,1518,441734,00.html>. Version: Oktober 2006
- [44] STATISTA: *Umfragen - Marktforschung*. www.de.statista.com. Version: April 2011
- [45] STUDIVZ.NET: *Studentenverzeichnis*. www.studivz.net. Version: April 2011
- [46] SYMFONY PROJECT: *Getting Started with symfony: Symfony Installation*. http://www.symfony-project.org/getting-started/1_4/en/03-Symfony-Installation. Version: April 2011
- [47] SYMFONY PROJECT: *Trac - UploadProgress*. <http://trac.symfony-project.org/wiki/UploadProgress>. Version: April 2011
- [48] UBUNTU-LINUX: *Official Ubuntu EC2-AMI*. <http://uec-images.ubuntu.com/releases/10.04/release/>. Version: April 2011
- [49] W3C CONSORTIUM: *Use standard redirects: don't break the back button!* <http://www.w3.org/QA/Tips/reback>. Version: 02 Juni 2007
- [50] WIKIPEDIA: *YouTube*. <http://de.wikipedia.org/wiki/YouTube>. Version: April 2011
- [51] YOUTUBE: *Allgemeine Geschäftsbedingungen von YouTube*. <http://youtube.com/t/terms>. Version: April 2011
- [52] YOUTUBE: *Videoportal*. <http://www.youtube.com>. Version: April 2011

- [53] YOUTUBE - GOOGLE CODE: *Youtube API Overview Guide*. http://code.google.com/intl/de-DE/apis/youtube/getting_started.html#data_api. Version: April 2011
- [54] YOUTUBE BLOG: *Y,000,000,000uTube*. <http://youtube-global.blogspot.com/2009/10/y000000000utube.html>. Version: 11 Oktober 2009
- [55] YOUTUBE BLOG: *2 Milliarden aufrufe pro Tag*. <http://youtube-global.blogspot.com/2010/05/at-five-years-two-billion-views-per-day.html>. Version: 17 Mai 2010
- [56] YOUTUBE BLOG: *Offizieller Blog von Youtube.com*. <http://youtube-global.blogspot.com/2010/05/at-five-years-two-billion-views-per-day.html>. Version: Mai 2010
- [57] ZDNET.DE - CBS INTERACTIVE GMBH: *Premier online content network for information and entertainment*. http://www.zdnet.de/news/wirtschaft_telekommunikation_youtube_beansprucht_ein_zehntel_der_web_bandbreite_story-39001023-39155501-1.htm. Version: 21 Juni 2007
- [58] ZEND: *Zend Service Amazon S3 API*. <http://framework.zend.com/manual/en/zend.service.amazon.s3.html>. Version: April 2011