Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Aleksandr Nosov

Entwurf und Realisierung eines Multi-Cloud Frameworks für Java

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Aleksandr Nosov

Entwurf und Realisierung eines Cloud Frameworks für Java

Bachelorarbeit eingereicht im Rahmen Bachelorprüfung

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

Betreuender Prüfer : Prof. Dr. Olaf Zukunft
Zweitgutachter : Prof. Dr. Philipp Jenke

Abgegeben am 14.11.2013

Aleksandr Nosov

Thema der Bachelorarbeit

Entwurf und Realisierung eines Multi-Cloud Frameworks für Java

Stichworte

Cloud, REST, OAuth 2.0

Kurzzusammenfassung

Bei der Entwicklung der Cloud-basierter Software, ist es oft notwendig mehrere Cloud Anbieter zu berücksichtigen. Die Entwicklung und Pflege der Anbindung wird dadurch erschwert. Wenn die Software mit vielen verschiedenen Cloud Anbieter kompatibel sein muss oder wenn ein Wechsel eines Cloud Anbieters durchgeführt wird, erfordert es einen großen Programmieraufwand. Das Ziel dieser Bachelorarbeit ist ein Framework für Java zu entwickeln, mit dessen Hilfe eine Anbindung an verschiedene Cloud Schnittstellen sich schnell und unkompliziert realisieren lässt.

Aleksandr Nosov

Title of the paper

Design and Implementation of a Multi-Cloud Framework for Java

Keywords

Cloud, REST, OAuth 2.0

Abstract

In the development of cloud-based software, it is often necessary to take into account multiple cloud providers. The development and maintenance of connection is difficult. It requires a large programming effort if the software has to be compatible with many different cloud providers or if a change of a cloud provider is needed. The aim of this thesis is to develop a framework for Java, which allows to implement connections to various cloud interfaces quickly and easily.

Inhaltsverzeichnis

1 Einführung

Dieses Kapitel stellt eine kurze Einführung dar. Hier wird beschrieben, was als Motivation für diese Arbeit diente und welche Ziele verfolgt diese Arbeite.

1.1 Motivation

Jedes Jahr schreitet der Cloud Markt mit rasanten Schritten voran. Jeden Tag erscheinen immer mehr Anwendungen, die die Cloud Dienste benutzen. Das Cloud-Computing wird nicht nur bei den Unternehmen beliebt, sondern auch für den privaten Nutzer interessant. Durch die Cloud Technologien können die Unternehmen bestimmte Teile Ihrer IT-Bereiche auslagern und können dadurch nicht nur IT-Kosten sparen, sondern können die Sicherheit und die Hochverfügbarkeit ihrer Anwendungen oder die Skalierung Ihrer IT-Infrastruktur sicherstellen. Auch Mobil Bereich entwickelt sich sehr schnell und trägt dazu bei. Es stehen immer mehr Softwareangebote für Thin-Clients, die die Cloud Dienste benutzen. Meistens werden Cloud-Dienste auf Smartphones und Tablets benutzt, um die Benutzerdaten auf verschiedenen Thin-Clients zu synchronisieren bzw. um die Daten zu sichern. Bei den privaten Benutzer geht es in erster Linie um den Austausch der Daten, die Sicherung der Daten, die Synchronisation mit mobilen Geräten. Ebenso der Video-Stream und Online-Spiele spielen immer größere Rolle dabei.

Die letzte Studie, die vom BITKOM[1] beauftragt und von Experton Group durchgeführt wurde, hat gezeigt, dass im Jahr 2012 in Deutschland die Umsätze mit Cloud Computing um fast 50 % gestiegen sind. Die Umsätze betrugen im Jahr 2012 voraussichtlich 5,3 Milliarden Euro und bis 2016 können die Umsätze bis auf 17,1 Milliarden Euro steigen (BITKOM, 2012). Daraus kann man entnehmen, wie schnell der Cloud Markt sich in Deutschland entwickelt. Bei solch einer schnellen Entwicklung wird Cloud-Computing immer größere Rolle in IT spielen. Es bring mit sich nicht nur neue Möglichkeiten für die Unternehmen, sondern auch neue Probleme bei der Entwicklung und Integration. Das stellt für die Softwareentwickler und –Architekten neue Herausforderungen dar.

1.2 Aufgabenstellung

Bei der Entwicklung der Cloud-basierten Software ist es oft notwendig mehrere Cloud Anbieter zu berücksichtigen. Die Entwicklung und Pflege der Anbindung wird dadurch erschwert. Wenn die Software mit vielen verschiedenen Cloud Anbietern kompatibel sein muss oder wenn ein Wechsel eines Cloud Anbieters durchgeführt wird, erfordert es einen großen Programmieraufwand. Das Ziel dieser Bachelorarbeit ist ein Framework für Java zu

[1] BITKOM - Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V.

entwickeln, mit dessen Hilfe eine Anbindung an verschiedene Cloud Schnittstellen sich schnell und unkompliziert realisieren lässt. Es gibt viele Anbieter von Cloud Diensten, die alle ihre eigenen Schnittstellen haben. Ebenso können sich Cloud Dienste unterscheiden. Es kann sich bei den Diensten z. B. um Hardware, Software oder Programmier- bzw. Laufzeitumgebung handeln. Die Abdeckung aller großen Anbieter und aller Dienste, würden den Rahmen dieser Bachelorarbeit sprengen. Aus diesem Grund werden hier nur einige Cloud Anbieter betrachtet.

Die primäre Aufgabe des Frameworks ist die Speicherung der Daten in die Cloud. Die Cloud Angebote wie AWS[2] von Amazon, Google Drive, Dropbox und Windows Azure sind bei den Entwicklern sehr verbreitet und kommen in dieser Arbeit zum Einsatz.

[2] Amazon Web Services

2 Grundlagen

In diesem Kapitel werden wichtige Grundlagen für die Arbeit beschrieben. Hier wird der Begriff Cloud Computing sowie dessen positive und negative Aspekte erklärt. Außerdem liefert dieses Kapitel das notwendige technische Hintergrundwissen und beschreibt die Unterschiede zwischen verschiedenen Cloud-Technologien.

2.1 Cloud Computing

Die Cloud Technik ist nicht revolutionär sondern evolutionär. Bereits im Jahr 1961 hatte Prof. John McCarthy eine Vision. An der MIT sagte er (Garfinkel, 2011), dass die Computer eines Tages, genauso wie Telefonsysteme, wie ein öffentliches Dienstprogramm organisiert werden und dass jeder Nutzer nur für die Kapazitäten bezahlen soll, die er tatsächlich verbraucht. Er hat aber gleichzeitig den Zugang zu allen Eigenschaften der Programmiersprachen eines sehr großen Systems. Douglas Parkhill beschrieb im Jahr 1966 in seinem Buch „The Challenge of the Computer Utility“ wie sich die Computertechnologie in Zukunft entwickeln wird. In seinem Buch sprach er schon damals über einen Eindruck von unerschöpflichen Ressourcen, den Eigenschaften wie online und elastischer Provision, und von öffentlichen, privaten, gemeinschaftlichen und behördlichen Formen der Verwendung von Computertechnologien.

Der Begriff „Cloud Computing“ wurde dann 1997 von Ramnath Chellappa ausgesprochen (Höllwarth, 2012). Die Virtualisierungstechnologien und immer leistungsfähigere Netzwerke haben dazu beigetragen, dass Cloud Computing einen Durchbruch erlebt. Die Vorväter vom Cloud Computing waren die Technologinnen wie „Distributed Computing“ und SOA[3]. Bei „Distributed Computing“ kommunizierten viele autonome Computer miteinander. SOA versuchte erst einmal verschiedene Komponente zusammen zu kapseln und als komplette Service zur Verfügung zu stellen. Einen großen Durchbruch erlebte Cloud-Computing mit Hilfe von großen Firmen wie Amazon, Google und Yahoo. Die Netzwerk bzw. Server-Kapazitäten dieser Firmen mussten auch zu Spitzenlastzeiten ausreichende Performance anbieten. Amazon hat sich schließlich im Jahr 2006 entschieden, die SOA und die Web-Dienste zu einem Produkt zu machen, das nach außen angeboten werden kann und heute als Cloud-Computing bezeichnet wird.

[3] SOA – Service Oriented Architecture

2.1.1 Definition von Cloud Computing

Cloud Computing ist ein abstrakter Begriff für alle Möglichkeiten verschiedene Ressourcen über das Netzwerk zur Verfügung zu stellen. Dabei geht es sowohl um Software als auch um Hardware Ressourcen. Genauso wie das Licht oder das Wasser von einem Anbieter bezogen wird, ohne dabei sich mit verschiedenen technischen Aspekten zu beschäftigen, werden die Server- bzw. Netzwerk-Ressourcen aus dem Internet bezogen.

Das „National Institute of Standards and Technology“ (NIST) hat 2009 eine Definition für Cloud-Computing veröffentlicht (National Institute of Standards and Technology, 2011). Diese lautet wie folgt:

> *„Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.“*

Das Bundesamt für Sicherheit in der Informationstechnik hat dafür eine andere aber ähnliche Definition (Bundesamt für Sicherheit in der Informationstechnik):

> *„Cloud Computing bezeichnet das dynamisch an den Bedarf angepasste Anbieten, Nutzen und Abrechnen von IT-Dienstleistungen über ein Netz. Angebot und Nutzung dieser Dienstleistungen erfolgen dabei ausschließlich über definierte technische Schnittstellen und Protokolle. Die Spannbreite der im Rahmen von Cloud Computing angebotenen Dienstleistungen umfasst das komplette Spektrum der Informationstechnik und beinhaltet unter anderem Infrastruktur (z. B. Rechenleistung, Speicherplatz), Plattformen und Software.“*

Das Cloud-Modell besteht aus fünf wesentlichen Charakteristika, drei Service-Modelle und vier Verteilungsmodelle.

Als wesentliche Eigenschaften vom Cloud Computing werden benannt:

- **Diensterbringung auf Anforderung** - Der Benutzer kann die Ressourcen selbst nach Bedarf anpassen, ohne dass er mit dem Service-Provider kommunizieren muss.
- **Netzwerkbasierter Zugang** - Die Ressourcen stehen über das Netzwerk zur Verfügung. Sie sind über die Standardmechanismen, die von den verschiedenen Client-Geräten (z. B. PC, Smartphones, PDA’s etc.) benutzt werden, verfügbar.

- **Ressource Pooling** - Die Ressourcen werden gebündelt und stehen nach Bedarf unterschiedlichen Nutzern zur Verfügung. Dabei gibt es ein Gefühl der Ortsunabhängigkeit und der Benutzer kennt, in vielen Fällen, die Position der Ressourcen nicht.
- **Elastizität** - Die Ressourcen können bereitgestellt und freigegeben werden. Oft passiert es automatisch. Für den Nutzer scheinen die Ressourcen grenzenlos und jederzeit verfügbar zu sein.
- **Messbare Dienstqualität** - Cloud-Systeme steuern und kontrollieren die Ressourcen automatisch. Die Nutzung der Ressourcen kann überwacht, gesteuert und berichtet werden und schafft die Transparenz der verbrauchten Dienste sowohl für Anbieter als auch für den Nutzer.

2.1.2 Cloud Service Modelle

Cloud Computing zeichnet sich als „Everything as a Service“ (XaaS) Modell ab. Die virtuelle physische Ressourcen, virtuelle Infrastruktur sowie virtuelle Middleware-Plattformen und Business-Anwendungen werden als Dienste zur Verfügung gestellt und konsumiert. Als Cloud Service Modelle gibt es „Software as a Service“ (SaaS), „Platform as a Service“ (PaaS), „Infrastructure as a Service“ (IaaS) und „Humans as a Service“ (HuaaS) (Lenk, Klems, Nimis, Tai, & Sandholm, 2009). Die NIST Definition beschreibt die wichtigsten drei Modelle wie folgt:

- **Software as a Service (SaaS)** - Der Service Provider bietet verschiedene Software, die auf der Hardware des Providers funktioniert, an. Der Nutzer kann auf die Software über die Standard Mechanismen, wie z. B. Webbrowser, zugreifen. Er kann nur die Einstellungen der Software steuern, aber nicht die Hardware Ressourcen.
- **Platform as a Service (PaaS)** - Der Service Provider bietet die Programmier- und Laufzeitumgebung an. Die Hardware Ressourcen werden automatisch gesteuert. Der Nutzer hat dabei die gesamte Kontrolle über die Software aber nicht über die Hardware.
- **Infrastructure as a Service (IaaS)** - Dem Nutzer wird die Virtualisierung der Computerhardware angeboten. Er kann z. B. die Rechner, Speicher oder Netzwerk steuern. Er muss aber selbst dafür sorgen, dass seine eigene Software funktioniert.

Bei **HuaaS** handelt es sich um die Massenhafte Aggregation und Extraktion von Informationen, die von Menschenmassen durchgeführt wird. Jeder einzelner in dieser Masse kann die Werkzeuge und Technologien benutzen, die er oder sie für richtig hält, um die Aufgabe zu lösen. Die menschliche Intelligenz kann dazu verwendet werden, um den Service zu verbessern. Die Menschenmassen erledigen das, was die Maschinen nicht können. Z. B. eine Bewertung auf Youtube.com abgeben, um die „berichtenswerte“ Videos später anzeigen zu können.

Es gibt auch weiter Service Modelle wie z. B. „Landscape as a Service“ (LaaS) oder „High Performance Computing as a Service“ (HPCaaS), die in dieser Arbeit nicht weiter erläutert werden.

Die Cloud Service Modelle lassen sich graphisch wie in Abbildung 2.1 darstellen.

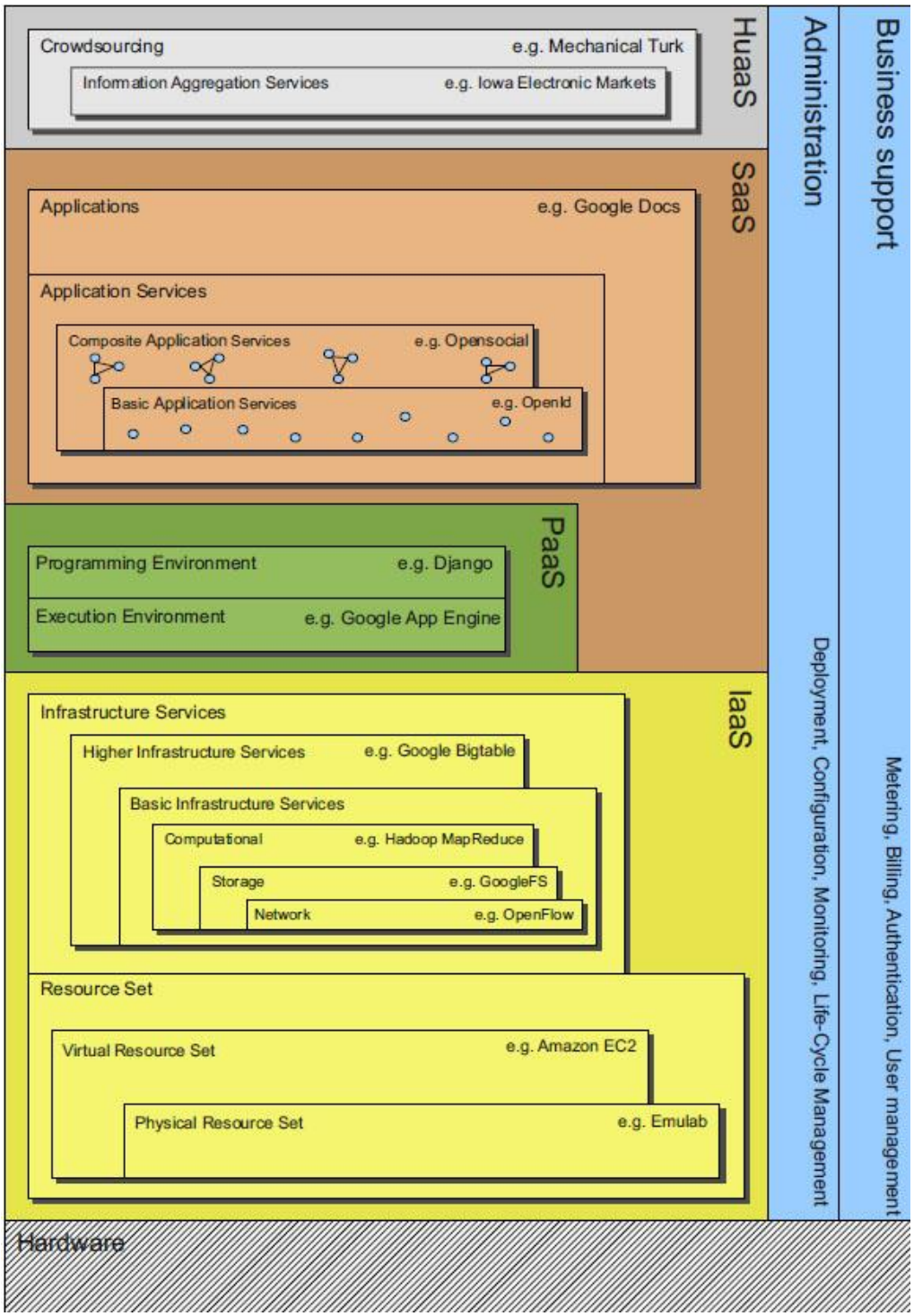

Abbildung 2.1 Cloud Service Modelle nach (Lenk, Klems, Nimis, Tai, & Sandholm, 2009)

2.1.3 Verschiedene Verteilungsmodelle

Aus unterschiedlichen Gründen, wie z. B. Sicherheit, Datenschutz oder Performance, wird nicht nur ein einziges Verteilungsmodell angewendet. Zum Verteilen von Ressourcen gibt es vier verschiedene Modelle.

Bei der **Private Cloud (auch Internal Cloud oder IntraCloud genannt)** wird der Zugang nur für Benutzer erlaubt, die derselben Organisation angehören (Baun, Kunze, Nimis, & Tai, 2011). Als Hauptgrund für Private Cloud wird meistens die Sicherheit genannt. In der Private Cloud bleibt die Kontrolle über die Dienste bei dem Benutzer bzw. Organisation. So können die privaten Daten besser geschützt werden.

Im Gegensatz zur privaten Cloud bittet die **Public Cloud** den Zugang für die Benutzer, die unterschiedlichen Organisationen angehören. Die Anbieter machen ihre Cloud öffentlich zugänglich und bieten den Benutzern ein Web-Portal an, in dem sie ihre Dienste selbst konfigurieren können.

Die Dienste in der Private Cloud werden zwar auf eigenen Ressourcen betrieben, aber es wird oft versucht die gleichen Schnittstellen, wie bei der Public Cloud, zu benutzen. Somit ist es möglich die gleichen Werkzeuge wie für Public Cloud zu benutzen und es wird die Möglichkeit offen gehalten, eigene Anwendungen später für Public Cloud zur Verfügung zu stellen.

Bei der **Hybrid Cloud** handelt es sich um eine Infrastruktur, bei der sowohl Public als auch Private Cloud Dienste zum Einsatz kommen. Mit Hybrid Cloud ist es z. B. möglich die Lastspitzen in die Public Cloud zu verlagern.

In der Abbildung 2.2 werden die Verteilungsmodelle nochmal verdeutlicht.

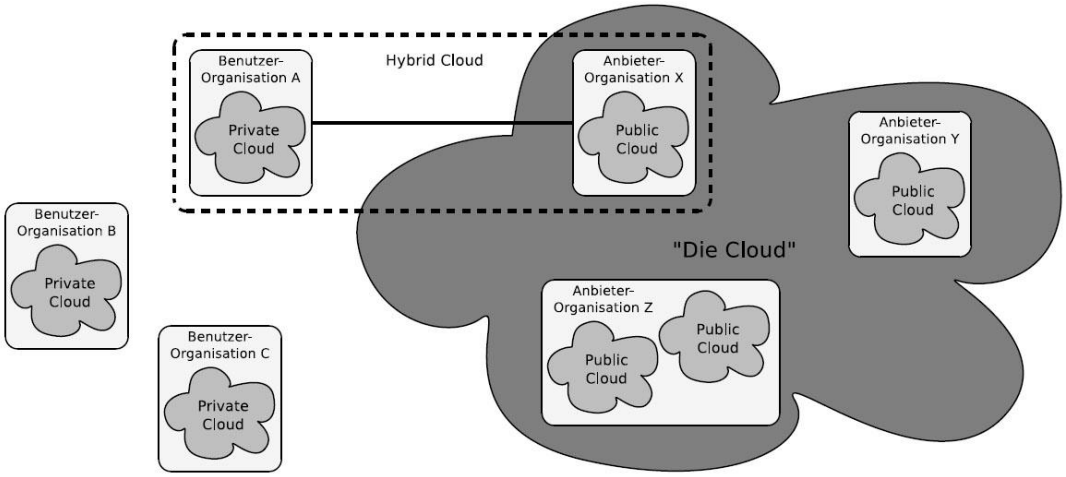

Abbildung 2.2 Public Cloud, Private Cloud und Hybrid Cloud nach (Baun, Kunze, Nimis, & Tai, 2011)

Die NIST Definition beschreibt auch ein weiteres Modell. Dabei geht es um die sogenannte **Community Cloud**. Bei dieser Infrastruktur bietet der Anbieter den Zugang für eine geschlossene Nutzergruppe (z. B. mehrere Behörden, Universitäten oder Firmen mit ähnlichen Interessen). Es ist also eine weitere Art von Private Cloud, die aber für mehrere Organisationen zur Verfügung steht und von einem oder mehreren Benutzern verwaltet wird.

2.1.4 Abgrenzung zu Grid Computing

Der Begriff „Grid Computing“ ist Mitte 90er aus dem Bereich elektrischer Netze entstanden, um die Einfachheit, Zuverlässigkeit und Verfügbarkeit hervorzuheben (Weinhardt, et al., 2009). Als die Rechenleistung einzelner Cluster in den Forschungseinrichtungen für einzelnen Aufgaben nicht mehr ausreichte, wurde die Rechenleistung in größere Domänen gebündelt. Die schnelle Vernetzung über das Internet hat es vielen wissenschaftlichen Instituten ermöglicht, geografisch verteilte Ressourcen, die zu verschiedenen Organisationen gehören, zu teilen und zu aggregieren.

Die Entwicklung von Grid-Computing erfolgte durch die Grid-Community und wurde nur selten kommerziell eingesetzt. Die größte von den Grid-Communities ist Open Grid Forum (OGF)[4]. Die Grid Computing benötigt einen Middleware. Die meisten Grid-Middlewares versuchen die Standards, die von OGF-Community festgelegt wurden, umzusetzen. Die

[4] http://www.ogf.org

Spezifikation von Open Grid Services Architecture (OGSA) ist einer dieser Standards. Diese Spezifikation enthält eine ganze Reihe von Normen.

Der wichtigste Unterschied zwischen Cloud- und Grid-Computing ist die Hardware-Virtualisierung. Die Virtualisierung durch VMWare[5] und Xen[6] ist bei dem Grid-Computing noch in der Entwicklung, wobei sie bei der Cloud ein wichtiger Bestandteil ist. Die Cloud-Computing hat eine zentralisierte und interaktive Steuerung. Bei Grid-Computing erfolgt die Steuerung durch den Batch. Der Zugang zum Cloud erfolgt nicht durch die Middleware sondern via standardisierten Web Schnittstellen.

2.1.5 Positive und negative Aspekte

Bei dem Cloud gibt es sowohl positive als auch negative Aspekte, die nachfolgend aufgestellt und beschrieben werden.

Sicherheit und Datenschutz

Besonders bei der Nutzung von Public Cloud Diensten ist es erforderlich die Sicherheitsaspekte zu beachten. Es gibt einige Sicherheitsprobleme bei der Cloud, die beachtet werden müssen (Birk & Wegener, 2010).

Verfügbarkeit

Meistens werden Netzwerke mit den Firewall-Systemen abgesichert. Zu diesen Systemen gehören Paketfilter[7], Application Level Gateway[8] und vielleicht noch die Komponenten eines Intrusion Detection System (IDS). IDS ist für die Erkennung der Angriffe am Perimeter im Netzwerk zuständig und dient meistens als Ergänzung zum Firewall. Dabei werden die Angriffe an verschiedenen Mustern erkannt und ein Alarm ausgelöst. Genau diese Komponenten fehlen meistens bei der Cloud Computing. Der Service Container, z. B. angemietete Webservice Instanz, wird zum Perimeter. Das bedeutet, dass die Service Container mit den IDS und Paketfilter ausgestattet werden müssen, was einen hohen Konfigurationsaufwand erfordert und eine mögliche Fehlerquelle darstellen kann. Angesichts der Dynamik der Cloud bleibt das noch eine technische Herausforderung.

[5] http://www.vmware.com
[6] http://www.xen.org
[7] Paketfilter – ein Teil des Firewalls, die den aus- und eingehenden Trafik im Netzwerk filtert.
[8] Application Level Gateway – eine Sicherheitskomponente, die eine dynamische Portfreigabe ermöglicht.

Integrität

Ein weiteres Problem ist die Verschlüsselung der Daten vor dem Transport zur Cloud. Der Cloud Service Provider (kurz: CSP) kann im Prinzip alle Daten, die von ihm bearbeitet werden, beliebig manipulieren und kopieren.

Eine weitere Maßnahme ist die Verschlüsselung der zu verarbeitenden Daten vor dem Transport zur Cloud. Gerade wenn es sich um geschäftsrelevante Daten handelt, die eine entsprechende Vertraulichkeit erfordern, ist eine End-zu-End-Verschlüsselung notwendig. Das ist insbesondere bei den Email Diensten wie Google Mail für den betrieblichen Einsatz zu beachten. Die anderen Verschlüsselungsarten, die keine End-zu-End-Sicherheit anbieten, bieten nur den Schutz vor der Einsichtnahme während der Übertragung an. Es muss also darauf geachtet werden, dass die Daten erst vor der Übertragung verschlüsselt werden. Andernfalls gibt es keinen Schutz vor Einsichtnahme durch den CSP. Wenn aber der CSP nicht vertrauenswürdig ist, dann nützen auch die Integritätsmechanismen wenig. Da die Integrität die ordnungsgemäß funktionierende Systeme voraussetzt und es sich schließlich keine eigenen Integritätskomponenten einbauen lassen.

Vertraulichkeit und Integrität spielen also eine sehr wichtige Rolle in Cloud-Computing. Anwender soll auch hier in der Lage sein zu bestimmen, wie und welche Daten durch den Cloud-Anbieter bearbeitet und analysiert werden. Der Cloud-Anbieter soll belegen können, wie er mit den Kunden Daten umgeht. Das kann er durch die Transparenz über Funktionsweisen und Risiken sowie mit Belegen zur Qualifikation des Personals und anerkannte Zertifizierungen oder aussagekräftige Security Service Level Agreements nachweisen (Hansen, 2012).

Vertraulichkeit

Um die Vertraulichkeit zu gewährleisten, werden, wie oben beschrieben, die Kryptoverfahren angewendet. Im idealen Fall liegen die Anwenderdaten nie unverschlüsselt auf den fremden Rechnern vor. Es wird auch intensiv an der homomorphen Verschlüsselung geforscht, die es ermöglicht einige mathematischen Operationen mit den verschlüsselten Daten zu erledigen. Diese Technologien werden aber noch nicht produktiv eingesetzt (Birk & Wegener, 2010). Die Daten lassen sich auch mit der Fragmentierung schützen, indem sie auf verschieden Clouds verteilt werden (Hansen, 2012), z. B. die sensiblen Daten in der Privat Cloud und übrigen in der Public Cloud. Hier werden mittlerweile die Lösungen mit mehreren vereinten Clouds diskutiert. Mit vereinten Clouds soll Kunde eine für ihn geeignete Cloud Kombination mit den passenden Sicherheitsgarantien bekommen.

Obwohl die Daten verschlüsselt werden, kann der Cloud-Anbieter trotzdem feststellen, wann der Benutzer aktiv war. Durch das Einsetzen von einem Dummy-Trafik oder einem Proxy ist es möglich das umzugehen. Als weiterer Aspekt muss beachtet werden, dass nicht

jeder Cloud-Anbieter gewährleistet, die Daten des Anwenders tatsächlich rückstandslos zu löscht.

Authentizität

Eine weitere Angriffsstelle sind die Schnittstellen für die Cloud-Verwaltung. In diesen Schnittstellen kann ein Angreifer den Zugriff auf ein Kundenkonto und alle seine Daten bekommen. Bei diesem Thema werden bereits einige Lösungen ausdiskutiert und auch von einigen Cloud-Anbietern implementiert. Weitere Maßnahme ist die Protokollierung aller Aktionen. Ebenso interessant ist die Verwendung von Trusted Computing, bei dem die Verwendungsregeln mit den verschlüsselten Daten zusammen gespeichert werden und beim Zugriff auf diese Daten entsprechend die Regeln für jeden Benutzer. Es wird überprüft, ob er die Aktion ausführen darf.

Je nach Land ist außerdem die gesetzliche Lage entscheidend. Trotz der Vertraulichkeit und Integrität wird einigen Behörden der Lesezugriff auf die Daten gewährt. Wenn der Anbieter die Daten verschlüsselt speichert, bedeutet es nicht, dass er sie nicht lesen kann. In einigen Fällen liegt die Kontrolle über den Schlüssel bei dem Anbieter.

Bei der Wahl eines Cloud-Anbieters muss sehr darauf geachtet werden, wie er mit den Kundendaten umgeht, durch welche Mechanismen die Daten geschützt werden und in welchem Land sich der Anbieter befindet, bzw. welche Gesetze in diesem Land gelten.

Wirtschaftliche Aspekte

Warum ist nun die Cloud so beliebt? Die genauere Betrachtung der wirtschaftlichen Aspekte beantwortet diese Frage. Ein wichtiges Argument für Cloud-Computing ist die Economies of Scal. Das heißt, dass die größeren Einheiten sich positiv auf die Kostenstruktur im Betrieb auswirken (Höllwarth, 2012). Vor ca. 10 Jahren wurde ein Rechenzentrum mit ca. tausend Servern als großes Rechenzentrum bezeichnet. Heute sind die Hardwarekosten geringer und die Verfügbarkeit der Netzwerkbandbreite knapper geworden. Gleichzeitig entsteht neue virtualisierte und automatisierte Management-Software. Mit diesen Technologien war die Industrie im Stande die Rechenzentren mit hunderttausend Servern zu betreiben. Durch die Kombination aus Hardware-, Bandbreite-, Energie- und Lohnkosten kann ein großer Cloud-Computing-Provider seine Gesamtbetriebskosten (TCO) um bis zu 80 % senken, indem er die Serverkapazitäten von 1000 auf 100.000 erhöht.

Wenn für den Bau in einer Region mit sauberer und günstigerer Energie entschieden wird, können bei dem Bau eines größeren Rechenzentrums die Energiekosten gespart werden. Dadurch ist es möglich einen günstigeren Energiepreis zu erhalten, im Vergleich zu einem normalen Verbraucher. In einem einfachen Rechenzentrum verwaltet ein Administrator ca. 100 bis 200 Server. In einem Cloud-Rechenzentrum ist ein Administrator für einige tausend

Server verantwortlich. Dadurch können die Arbeitskosten enorm gesenkt werden. Bei der Bestellung von 100.000 Servern ist ein besserer Preiseinfluss erreichbar, als bei dem Kauf von 1000 Servern. Dies ermöglicht es bei dem Bau eines Cloud-Rechenzentrums die Hardwarekosten einzusparen. Bei solchen Mengen kann auch eine speziell angepasste Hardware bestellt werden, z. B. ohne Gehäuse und mit gemeinsamen Nutzung von Stromadaptern oder die CPUs mit geringem Kühlbedarf.

Bei den Kunden können einige mehr von Cloud profitieren als die anderen. So z. B. profitieren einige Unternehmen im Einzelhandel davon, wenn ihre Umsätze in einem Monat die Umsätze in anderen Monaten weit übersteigen. Genauso die Behörde, die in einem Montan mehr zum Verwalten haben als in den anderen Monaten. Besonders lukrativ ist die Cloud für die IT Unternehmen, die ihr Angebot im Internet haben, weil sie dadurch besonders schnell wachsen können.

Es gibt bei der Cloud sowohl Stärken als auch Schwächen. Mit der Cloud-Computing lassen sich in vielen Fällen die Kosten reduzieren. Bevor zur Cloud-Computing gewechselt wird, ist es aber notwendig alle Sicherheitsfragen und rechtliche Aspekte abzuwägen.

2.2 Representational State Transfer

Die Kommunikation mit den Cloud-Server erfolgt bei den meisten CSP mit der Hilfe von Representational State Transfer (kurz: REST).

2.2.1 Definition von REST

REST ist ein Architekturstill, mit dessen Hilfe sich die Web-Services realisieren lassen. Viele CSP benutzen für ihre Web-Services REST. Aus diesem Grund ist es sinnvoll erst einmal REST genauer kennenzulernen. Als Begriff wurde REST im Jahr 2000 von Dr. Roy Thomas Fielding in seiner Dissertation eingeführt (Fielding R. T., 2000). Nach Fielding gibt es folgende Merkmale die den REST Stil kennzeichnen:

- **Client/Server** – Durch eine einheitliche Schnittstelle ist es möglich die Aufgaben zwischen Client und Server zu verteilen. Der Client ist somit nicht für die Daten Speicherung zuständig und der Server kümmert sich nicht um die Darstellung auf dem Client und die Clientzustände.
- **Zustandslosigkeit** - Jede Anfrage an den Server muss alle Informationen beinhalten, die der Server zum Bearbeiten braucht. Es wird kein Clientkontext auf dem Server zwischen den Anfragen gespeichert. Der Client ist selbst für eigene Sessions zuständig.
- **Caches** - Die Caches müssen unterstützt werden. Der Server muss eine Antwort als cache-fähig oder cache-unfähig kennzeichnen können.

- **Schichten System** – Ein Client soll nicht wissen, ob er direkt mit dem Server verbunden ist. Damit ist es möglich einen Vermittler dazwischen zu schalten und z. B. die Lastverteilung zu ermöglichen.
- **Code auf Anfrage** – Der Server hat die Möglichkeit dem Client einen neuen ausführbaren Code zu liefern. Ein Beispiel dafür ist JavaScripts oder Java Applets.
- **Einheitliche Schnittstelle** – Wie oben bereits erwähnt wurde, ist eine einheitliche Schnittstelle notwendig, um die Unabhängigkeit zwischen dem Client und Server zu ermöglichen.

2.2.2 Anwendung

In den letzten Jahren hat sich REST sehr verbreitet und verdrängt langsam SOAP oder WSDL (Rodriguez, 2008). REST ist so beliebt, da es im Vergleich zu SOAP und WSDL eine einfache Art ist, die Web-Services zu bedienen darstellt. REST erlaubt es über eine eindeutige Adresse (URI) eine Online Ressource abzurufen und zu manipulieren. Dabei werden die HTTP Methoden benutzt, um dem Server mitzuteilen, was mit der Ressource gemacht werden soll. Der Informationsaustausch findet meistens in XML- und/oder JSON Format statt. Beim Benutzen von REST sollen die HTTP Methoden konsistent zu dem Protokoll verwendet werden. Dadurch ist es möglich die CRUD[9] Operationen eins-zu-eins auf die HTTP Methoden wie folgt abzubilden.

- Um eine Datei auf dem Server zu erstellen, benutze POST.
- Um eine Datei abzurufen, benutze GET.
- Um eine Datei zu verändern, benutze PUT.
- Um eine Date zu löschen, benutze DELETE.

Das URI[10] soll am besten mit Bedacht gewählt werden, um RESTful zu bleiben. Die URL http://myserver/adduser?name=Mustermann ist nicht RESTful. In Listing 2.1 wird gezeigt, was der Browser als Quellcode an der Server schickt, wenn er diese URL interpretiert.

```
GET /adduser?name=Mustermann HTTP/1.1
```

Listing 2.1 HTTP Anfrage nicht RESTful

In diesem Beispiel identifiziert das URI nicht die Ressource, sondern die Operation, die auf die Ressource angewendet wird. Das hat einige Nebeneffekte. Der Server soll schließlich eine Mitteilung über HTTP Methoden erhalten, welche Operation auf das Objekt angewendet wird soll.

[9] CRUD – create, read, update, delete Operationen
[10] URI - Uniform Resource Identifier

Die Anfrage soll wie folg darstellt werden.

```
POST /users HTTP/1.1
Host: myserver
Content-Type: application/xml
<?xml version="1.0"?><user> <name> Mustermann </name></user>
```

Listing 2.2 POST Anfrage und REST

Als Kontext werden die Informationen, die zum Anlegen einer Ressource benötigt werden, übergeben. Die GET Anfrage sieht so aus.

```
GET /users/Mustermann HTTP/1.1
Host: myserver
Content-Type: application/xml
```

Listing 2.3 GET Anfrage und REST

Und die entsprechende Update Operation könnte so aussehen.

```
PUT / users / Mustermann HTTP/1.1
Host: myserver
Content-Type: application / xml
<? Xml version = "1.0"?><user><name> Bob </ name></ User>
```

Listing 2.4 PUT Anfrage und REST

Als allgemeines Konstrukt wird es empfohlen, anstatt von Verben die Substantive in URIs zu benutzen.

Der Client kann die MIME Types benutzen, um dem Server mitzuteilen, in welchem Format er die Daten akzeptiert. Um das mitzuteilen, wird bei der Anfrage das „Content-Type“ eingegeben. Die Benutzung von MIME Types und HTTP Accept header wird als „content negotiation“ bezeichnet und minimiert die Kopplung zwischen dem Service und der Client-Applikation.

3 Cloud Anbieter im Vergleich

Wie in Kapitel 2 bereits beschrieben und in der Aufgabenstellung erwähnt ist, gibt es viele verschiedene Funktionen die ein CSP anbieten kann. Es wird also unmöglich sein, alle möglichen Funktionen in einem Framework abzubilden. In dieser Arbeit wird also das Problem nur auf das Speichern von Daten (wie z. B. Fotos oder Text Dateien) reduziert. Dabei konzentriert sich diese Arbeit nur auf einige bekannte Anbieter. Um die Gemeinsamkeiten bei den Schnittstellen später extrahieren zu können, werden in dieser Arbeit auch die Schnittstellen einzelner Anbieter für die Datenspeicherung betrachtet. Dafür wurden Dienste ausgewählt, die für die Speicherung von großen Datenmengen verantwortlich sind.

3.1 Amazon AWS

Die Abkürzung AWS steht für Amazon Web Services. Im Jahr 2006 bot Amazon erstmals die Web Services als Infrastrukturdienstleistungen an (Varia & Mathew, Overview of Amazon Web Services, 2013). Amazon Web Services bietet eine hoch verfügbare, skalierbare und kostengünstige Rechenplattform in der Cloud, die in mehreren Hunderttausend Unternehmen in 190 Ländern weltweit eingesetzt wird. Amazon stellt für AWS die Rechenzentren in den USA, Europa, Brasilien, Singapur, Japan und Australien bereit.

AWS bietet eine breite Palette von verschiedenen Cloud-Lösungen. Dazu gehören Anwendungs-Hosting, Backup und Speicherung, die Bereitstellung von Inhalten für Endnutzer mit einer hohen Geschwindigkeit, Web-Hosting, Hosting für interne und externe IT-Anwendungen in einer sicheren Umgebung sowie eine Reihe von skalierbaren Datenbanklösungen.

Datenverarbeitung

Für die Datenverarbeitung stellt Amazon **Elastic Compute Cloud (EC2)** bereit, das skalierbare Rechenkapazität in der Cloud ermöglicht. Dazu gehören folgende Web Services: **Elastic MapReduce,** mit dem sich riesige Datenmengen verarbeiten lassen; **Auto Scaling**, das das Anpassen der EC2-Kapazitäteten nach festgelegten Bedienungen ermöglicht; **Elastic Load Balancing,** das dazu dient, den eingehenden Anwendungsverkehr automatisch über mehrere EC2-Instancen zu verteilen.

Bereitstellung von Inhalten

Amazon CloudFront ist ein Web-Service, mit dem Inhalte einfach und mit geringen Verzögerungszeiten über ein globales Netzwerk aus mehreren Edge-Standorten bereitgestellt werden können.

Datenbanken

Als Datenbanklösungen bietet Amazon mit **Relational Database Service (RDS)** eine einfache Einrichtung und Skalierung von relationalen Datenbanken in der Cloud. Die NoSQL Datenbanken lassen sich mit folgenden Services verwalten: **DynamoDB,** als leistungsstarker und vollständig verwalteter Datenbank-Service; **SimpleDB** als kleinere Datasets und **ElastiCache** als In-Memory-Cache in der Cloud.

Bereitstellung und Verwaltung

Mit **Identity and Access Management (IAM)** lassen sich die Benutzer und die Zugriffsrechte verwalten. **CloudWatch** bietet die Überwachung von AWS Cloud-Ressourcen und beginnt dabei mit Amazon EC2. **AWS Elastic Beanstalk** macht es leichter, Anwendungen in der AWS-Cloud bereitzustellen und zu verwalten. Dabei wird die Anwendung einfach hochgeladen und Elastic Beanstalk verwaltet automatisch Kapazitätsbereitstellung, Lastverteilung, automatische Skalierung und Statusüberwachung. **AWS CloudFormation** bietet die Möglichkeit an, eine Sammlung von zugehörigen AWS-Ressourcen zu erstellen und in geordneter und transparenter Form bereitzustellen. **AWS Data Pipeline** ist ein Service zur Unterstützung des zuverlässigen Verarbeitens und Verschiebens von Daten zwischen AWS-Datenverarbeitungs- und -Speicherservices sowie lokalen Datenquellen in den angegebenen Intervallen.

Anwendungsservices

Amazon bietet auch verschiedene Anwendungsservices an. So lassen sich mit **CloudSearch** die schnellen und hochskalierbaren Suchfunktionen in eine Anwendung integrieren. Die Verarbeitungsschritte einer Anwendung und die Verteilung der Ausführungszustände lassen sich mit der Hilfe von **Simple Workflow Service (SWF)** koordinieren und verwalten. Mit **Simple Queue Service (SQS)**, **Simple Notification Service (SNS)**, **Simple Email Service (SES)** ist es möglich die Nachrichtenwarteschlangen, Benachrichtigungen und Email-Services zu steuern.

Speicherung

Zum Speichern und Abrufen beliebiger Datenmengen bietet Amazon als Lösung **Simple Storage Service (S3)** an. Damit stellt Amazon eine vollständig redundante Datenspeicher-Infrastruktur bereit. Um die Daten zu archivieren bzw. zu sichern gibt es einen Service mit

dem Namen **Glacier**. Die Daten können auch auf den Datenträgern außerhalb einer EC2-Instance gehalten und zusammen mit EC2-Instancen verwendet werden. Dafür sorgt Amazon **Elastic Block Store.**

Um große Datenmengen zu tragbaren Speichergeräten zu bewegen, gibt es auch bei Amazon **AWS Import/Export** Services. Um die Softwareanwendungen mit der Cloud Speicherung zu verbinden, steht **AWS Storage Gateway** zur Verfügung. Das ermöglicht die Integration zwischen unternehmensinternen IT-Umgebung und der AWS-Speicherinfrastruktur.

Außerdem stellt Amazon die Services für Netzwerk wie DNS und VPN zur Verfügung. Es gibt auch Services für Zahlungsverkehr, einen Marketplace und Web-Datenverkehr Services. Interessant ist auch **Mechanical Turk.** Mit seiner Hilfe werden beim Bedarf tausende Arbeitskräfte weltweit verfügbar. Die Abbildung 3.1 bietet einen Überblick über die Amazon Web Services Architektur an.

Amazon bietet sehr viele Web-Services an und stellt eine sehr komplexe Infrastruktur zur Verfügung. In dieser Arbeit wird nur auf den **Amazon S3** Dienst eingegangen.

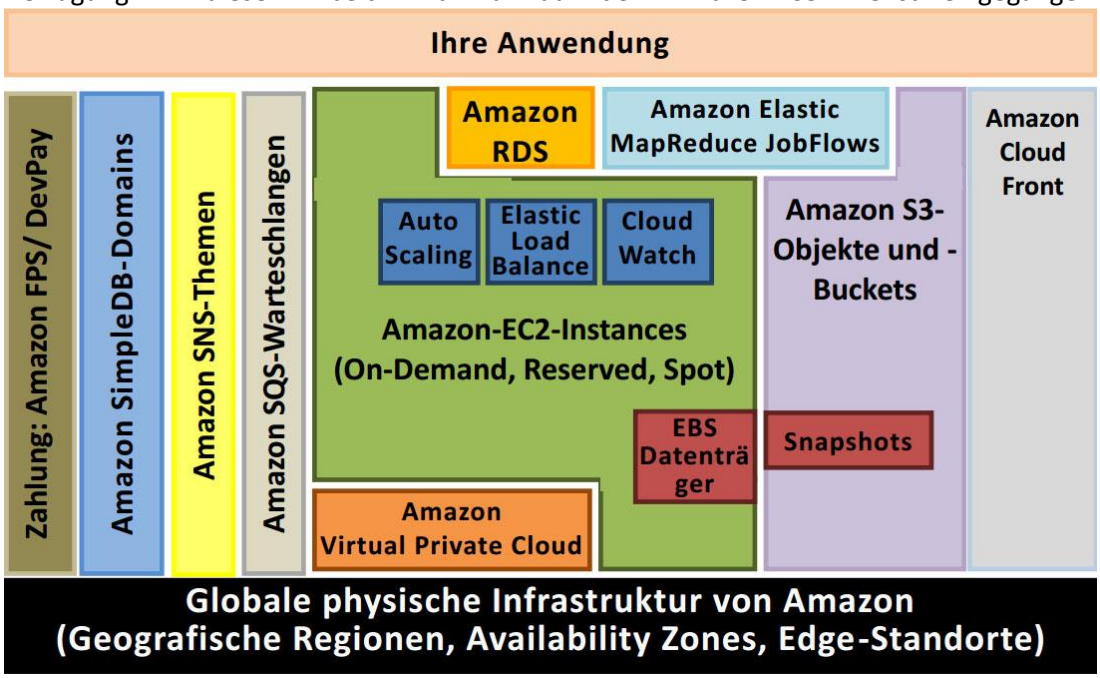

Abbildung 3.1 Amazon Web Services Architektur (Varia, Cloud-Architektur – Best Practices, 2010)

3.1.1 Dienste der Datenverwaltung

Amazon S3 ist extra mit einer minimalen Menge an Funktionalität ausgestattet (Amazon Simple Storage Service). Damit sorgt Amazon S3 für Einfachheit und Robustheit. Zu den Funktionen dieses Services gehören:

- **Create Buckets**—Buckets sind grundlegende Container zum Speichern in Amazon S3. Es wird ein Container, in welchem dann die Daten gespeichert werden, erstellt und benannt.
- **Store data in Buckets**—In einem Bucket kann beliebige Anzahl an Daten gespeichert werden. Jedes Objekt, das in einem Bucket gespeichert wird, kann bis zu 5 TB (Terrabyte) an Daten beinhalten. Um ein Objekt zu speichern und auf das zuzugreifen, benötigt jedes Objekt einen eindeutigen Entwickler Schlüssel.
- **Download data**—Die Daten lassen sich zu jeder Zeit selbst herunterladen oder auch für die anderen freigeben.
- **Permissions**—Die Rechte zum Hoch- und Runterladen von Daten können für jedes Bucket festlegt werden.
- **Standard interfaces**—Es besteht die Möglichkeit eine REST oder SOAP Schnittstelle zu nutzen.

Buckets

Wie bereits oben beschrieben, ein Bucket ist ein Container, in dem die Daten liegen. Wenn ein Objekt den Namen *photos/puppy.jpg* trägt und das Bucket *johnsmith* heißt, so wird es über die URL *http://johnsmith.s3.amazonaws.com/photos/puppy.jpg* adressiert. Buckets organisieren das Amazon S3 auf der höchsten Ebene. Sie halten die Informationen über die Zuständigkeiten, Datenverkehr, dienen als Einheit für die Nutzungsstatistik und spielen eine Rolle bei Zugriffsrechten.

Buckets können für verschiedene Regionen konfiguriert werden. Außerdem ist es möglich eine Versionskontrolle für die Daten einzurichten. Mit der Versionskotrolle bekommen alle Objekte, die zu dem Bucket hinzugefügt werden, eine Versions-ID.

Objects

Die Objekte sind die grundlegenden Einheiten, die bei Amazon S3 gespeichert werden. Sie beinhalten sowohl die Objekt Daten als auch die Metadaten. Die eigentlichen Daten sind für Amazon S3 nicht transparent. Die Metadaten sind eine Menge von Name-Value Paaren, die ein Objekt beschreiben. Zu den Metadaten gehören z. B. HTTP-Metadaten wie Content-Type oder das Datum des letzten Zugriffs. Ein Objekt wird von dem Bucket durch einen **eindeutigen Schlüssel** und die Versionsnummer identifiziert. Z. B. in URL *http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl* bezeichnet „doc“ das Bucket und „2006-03-01/AmazonS3.wsdl“ ist der Schlüssel.

Regions

Um die Kosten zu reduzieren oder um die Latenzzeiten zu verringern, gibt es die Möglichkeit eine Region auszuwählen, in der ein Bucket erstellt wird. Das können Regionen aus USA, Asien oder Südamerika sein.

Amazon S3 Data Consistency Model

Amazon erreicht eine hohe Verfügbarkeit durch die Replikation von Daten. Aus diesem Grund ist es notwendig bei einiger Operationen darauf zu achten, dass die Daten möglicherweise nicht konsistent erscheinen, bevor alle Daten komplett repliziert werden. Die Standard-US-Region ist für alle Operationen konsistent. Alle anderen Regionen sind nur für PUT Operationen konsistent.

Außerdem bietet Amazon S3 **Reduced Redundancy Storage**. Das ermöglicht das Speichern von unkritischen Daten ohne Replikationen. Das ist sinnvoll, um die Kosten zu sparen und wenn die Daten nur selten benutzt werden. Das lohnt sich z. B. bei Back-Ups. Die Zugriffskontrolle wird auf drei Wegen gewährleistet. Die Zugriffsrechte für Buckets und Objects werden bei Amazon S3 mit **IAM Policies**, **Access Control List (ACL)** oder **Bucket Policies** verwaltet. Diese können sowohl einzeln als auch kombiniert verwendet werden. Die Daten lassen sich entweder client-seitig oder server-seitig verschlüsseln. Bei der client-seitiger Verschlüsselung ist der Entwickler selbst für die Implementierung verantwortlich. Die Daten werden in diesem Fall nur als verschlüsselt markiert. Bei der server-seitigen Verschlüsselung wird alles von Amazon S3 gemanagt.

3.2 Windows Azure

Windows Azure ist eine Anwendungsplattform von Microsoft für die öffentliche Cloud (Introducing Windows Azure). Windows Azure bietet drei Möglichkeiten an, eine Anwendung in der Cloud auszuführen. Das sind IaaS, Web-Hosting und PaaS. Es ist möglich alle drei Ansätze – virtuelle Maschinen, Webhosting und Cloud Services - sowohl getrennt voneinander als auch kombiniert zu benutzen.

Datenverwaltung

Damit eine Anwendung ihre Daten irgendwo speichern kann, kann natürlich eine virtuelle Instanz angemietet werden. Dadurch wird eine komplette Kontrolle über die Datenbanken behalten. Mit Cloud Services gibt es noch drei Daten-Management-Optionen, die von Microsoft verwaltet werden. Für die rationalen Datenbanken gibt es bei Windows Azure die Lösung mit dem Namen **SQL Azure**. Es ist eine SLQ Datenbank mit dem Benutzermanagement sowie Skalierungs- bzw. Replikationsoptionen. Um die großen Daten, die keine Relationen benötigen, zu speichern, gibt es Windows Azure **Tables**. Das ist eine

Art NoSQL Ansatz, der auf key/value Speicher basiert. Zum Speichern von Dateien, wie z. B. Videos oder Backups, hat Windows Azure eine dritte Möglichkeit, die **Blobs** genannt wird. Blobs können in Verbindung mit persistenten Datenvolumen, die in das Dateisystem einer Windows Azure Instanz eingebunden werden können, verwendet werden.

Netzwerk

Windows Azure ist auf einigen Datencentren in USA, Europa und Asien verteilt. Beim Nutzen von Windows Azure ist es möglich Regionen, in denen die VMs bzw. Daten gespeichert werden, festzulegen. Es gibt auch die Möglichkeit einen VPN Netzwerk einzurichten, um sich mit den VMs zu verbinden oder sie sogar in eigene lokale Netzwerke einzubinden. Außerdem gibt es noch einen Trafik Manager, der es ermöglicht, die Last zwischen den Benutzern aus verschiedenen Weltregionen zu verteilen.

Business Analytics

Um die Daten zu analysieren bietet auch Windows Azure zwei Möglichkeiten an. Für die SQL Datenbanken hat Windows Azure einen **SQL Reporting** Dienst. Damit lassen sich die Daten aus einer SQL Datenbank analysieren und als ein Report exportieren. Für größere Datenmengen, die nicht in einer Relation zu einander stehen, kann Hadoop benutzt werden. Hadoop ist ein Open Source Projekt unter The Apache Software Foundation[11]. Mit Azure **HDInsight** ist es möglich große Datenmengen auf einem „Hadoop Distributed File System“ (HDFS™), die auf verschiedenen Servern verteilt werden, zu speichern, um dann ein Map Reduce darauf auszuführen.

Nachrichten

Um zwischen Anwendungen zu kommunizieren stehen auch zwei Dienste zur Verfügung. Die Windows Azure **Queues** bieten die einfachste Art um die Nachrichten zwischen Anwendungen auszutauschen. Dabei kann eine Anwendung eine Nachricht in die Queue speichern und eine weitere Anwendung kann diese Nachrichten aus der Queue entnehmen. Eine weitere Art ist der „publish-and-subscribe“ Mechanismus, der von Windows Azure **Service Bus** umgesetzt wird. Bei dem Service Bus melden sich die Interessenten für einen Topic an, um eine n-zu-m Beziehung zischen Anwendungen herzustellen.

Caching

Die Anwendung könnte beschleunigt werden, indem die Daten, auf die am meisten zugegriffen wird, in Cache gehalten werden. Dafür stellt Windows Azure in-memory **Cache** bereit. Mit dem in-memory Cache können die SQL-Database-, Tables- und Blobs-Dienste noch schneller gemacht werden. Dieser Cache kann sowohl auf einer als auch auf mehreren

[11] http://hadoop.apache.org/

VMs gelagert bzw. verteilt werden. Wenn große Datenmengen zum Herunterladen bereitgestellt werden, z. B. als Video-Stream, dann kann es unter Umständen vorkommen, dass eine Replication auf mehreren VMs nicht mehr ausreicht. Für solche Fälle gibt es bei Windows Azure den **Content Delivery Network (CDN**) Service. Wenn ein Benutzer aus einer Weltregion die Daten zum ersten Mal anfordert, werden die Daten aus dem Datencenter in das geographisch nähere CDN kopiert.

Windows Azure beinhaltet auch weitere Dienste. Dazu gehört auch die Benutzerverwaltung, die auf das Active Directory basiert und mit dem eignen lokalen Windows Active Directory synchronisiert werden kann. Interessant ist auch der High-Performance-Computing (HPC) Dienst. Windows Azure bietet einen HPC Scheduler an, der es ermöglicht den Code auf mehreren VMs parallel bearbeiten zu lassen. Damit ist es möglich mehre Maschinen zum Lösen eines komplexen Problems einzusetzen. Außerdem gibt es noch verschiedene Media Services zum Kodieren und Streaming von Media-Daten, die dann über CDN verteilt werden können. Um die SaaS Anwendungen zu verkaufen, verfügt Windows Azure auch über einen Marktplace. Der Poster (Microsoft, 2012) gibt eine einfache Übersicht über die Services, die Windows Azure anbietet.

3.2.1 Dienste der Datenverwaltung

Windows Azure Blob Storage ist ein Dienst zum Speichern von großen Datenmengen (How to use Blob Storage from Java). Es ist möglich von überall über HTTP oder HTTPS auf diesen Dienst zuzugreifen. Ein einzelner Blob kann hunderte von Gigabytes groß sein. Ein Storage Account ist im Stande bis zu 100 TB Daten aufzunehmen. Zu den wichtigsten Aufgaben von Blob Storage gehören:

- Die Abbildungen oder Dokumente direkt zum Browser liefern
- Speichern von Dateien für einen verteilen Zugang
- Streaming von Audio und Video
- Ausführen von Backup und Wiederherstellung
- Speichern von Daten für eine spätere Analyse

Windows Azure Blob Storage besteht aus folgenden Komponenten.

Storage Account

Alle Zugriffe auf Windows Azure Storage erfolgen über Storage Account. Das ist die höchste Ebene in dem Namensraum für Blobs. Ein Storage Account kann beliebige Anzahl an Blobs aufnehmen, soweit die gesamte Größe 100 TB nicht übersteigt.

Container

Ein Container dient zum Gruppieren von Blobs. Jedes Blob muss innerhalb eines Containers gespeichert werden.

Blob

Ein Blob kann von einem beliebigen Datentyp und einer beliebigen Größe sein. Windows Azure hat Block Blobs und Page Blobs. Am meisten werden die Block Blobs verwendet. Sie können bis zu 200 GB groß sein. Ein Block Blob besteht aus Blocks, die durch IDs identifiziert werden. Die Blocks können unterschiedliche Größe haben, aber maximal bis 4 MB. Ein Block Blob kann bis zu 50 000 Blocks beinhalten. Bei den Page Blobs handelt es sich um eine Sammlung von Pages, die ständig gelesen und geschrieben werden können. Ein Page kann dabei 512 Byte groß sein.

URL format

Die Blobs werden wie folgt adressiert:
http://<storage account>.blob.core.windows.net/<container>/<blob>

Die Abbildung 3.2 zeigt noch mal den Zusammenhang.

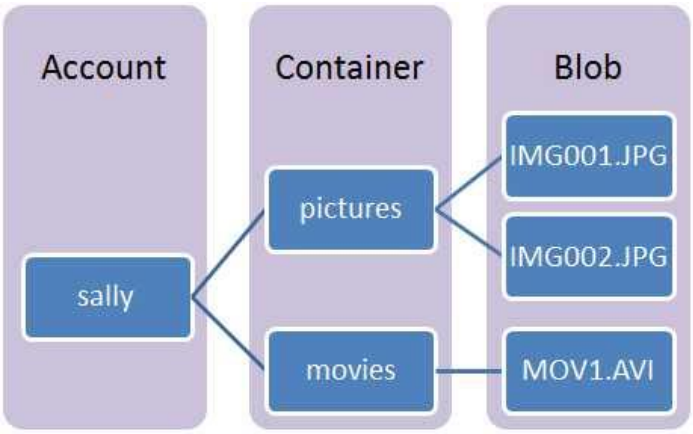

Abbildung 3.2 Blob Storage Begriffe (How to use Blob Storage from Java)

3.3 Google Cloud Platform

Google hat auch einige Cloud Dienste im Angebot (Google Cloud Platform, 2013). Einer der bekanntesten Dienste ist Google Drive. Mit Google Drive lassen sich Dokumenten nicht nur in der Cloud speichern. Zum Google Drive bietet Google auch die Cloud Software an, womit die Dokumente in der Cloud bearbeitet werden können. Mit dem **Google Cloud Platform** ist es möglich, die Ressourcen nach Bedarf anzumieten.

Google App Engine

Je mehr eine Webanwendung benutzt wird, desto mehr Ressourcen werden für diese Anwendung gebraucht. Im Internet kann es sehr schnell passieren, dass eigene Kapazitäten nicht mehr ausreichen. Google bietet dafür Google App Engine an. Mit diesem Dienst können die Kunden ihre Webanwendungen auf der Cloud-Infrastruktur von Google ausführen. Mit Google App Engine ist es möglich die Anwendungen mit jeder Programmiersprache, die Java Virtual Machine (JVM) Compiler oder Interpreter verwendet, zu entwickeln. Zu App Engine gehört auch eine dedizierte Python-Laufzeitumgebung. Grundsätzlich lassen sich die Anwendungen mit Programmiersprachen Java, JavaScript, Python, Ruby und Go entwickeln. Die Anwendungen laufen in einer Sandbox-Umgebung. Mit der Sandbox-Umgebung sind Anwendungen nur über HTTP/HTTPS verfügbar. Sie können die Funktionen des Betriebssystems nur eingeschränkt nutzen.

Google Compute Engine

Um eigene Virtuelle Maschinen (VM) ausführen zu können, bietet Google den Dienst Google Compute Engine an. Mit diesem Dienst lassen sich ganze Cluster von VMs erstellen. Es ist möglich sowohl vorkonfigurierte VMs zu starten als auch eigene Konfigurationen zu erstellen. Es gibt auch die Möglichkeit von einer VM-Instanz aus auf persistente Datenspeicher zuzugreifen. Diese Datenspeicher reichen über die Lebensdauer von einer VM-Instanz hinaus. Es lassen sich verschiede Firewall Regeln einstellen und damit auch eigene gesicherte Netzwerke bauen.

Google Cloud Storage und Google Drive

Google bietet zwei verschiedene Varianten, um die Daten in der Cloud zu speichern. Das Google Drive dient der privaten Nutzung der Daten. Es arbeitet zusammen mit Google Drive UI und dem Chrome Web Store und deckt damit auch die SaaS Ebene in der Cloud ab. Mit diesen Funktionen haben die Benutzer die Möglichkeit mit ihren Dokumenten auf verschiedenen Wegen zu Interagieren. Google Cloud Storage eignet sich mehr zum

Entwickeln von Anwendungen, die ihre eigenen Daten in Cloud speichern. Beide Dienste verfügen über ein programmatisches API und können über REST angesteuert werden.

Google Cloud SQL und Google BigQuery

Um die Daten in einer relationalen Datenbank zu verwalten, bietet Google den Service Google Cloud SQL an. Das ist eine Google MySQL Instanz. Es arbeitet eng mit anderen Diensten wie z. B. Google App Engine zusammen. Google stellt auch einen Dienst zu Verfügung, mit dem große Datenmengen schnell analysiert werden können. Dieser Dienst hat den Namen Google BigQuery. Mit Google BigQuery lassen sich die Daten mit Hilfe von CSV importieren und in einfachen Tabellen ohne Relationen abspeichern. Es ist möglich beliebige, große Datenmengen schnell zu analysieren, aber keine Daten zu verändern.

3.3.1 Dienste der Datenverwaltung

Google Cloud Storage ist sehr ähnlich aufgebaut wie Amazon S3. Es kann auch in einem Interoperablem Modus arbeiten und ist somit kompatibel zu Amazon S3 und vielen anderen API's (Google Cloud Storage Interoperability, 2013). Aus diesem Grund ist Google Cloud Storage für diese Arbeit weniger interessant. Nachfolgend wird nur auf Google Drive Service eingegangen.

Google Drive ist mehr als ein Online-Speicher. Es bietet eine Drive UI Oberfläche an. Über diese Oberfläche lassen sich die Dokumente wie Text, Tabellen, Präsentationen und Zeichnungen verfassen. Außerdem bietet Google Drive ein Realtime API, das es ermöglicht gleichzeitig in einem Collaborative Modus an den Dokumenten zu arbeiten. Dieses API basiert auf JavaScript und ist nur für Programmierung eines Web UI geeignet.
Google Drive bietet ein REST API sowie eine SDK für verschiedene Programmiersprachen. Mit der Google Drive API lassen sich die Dateien hoch- und herunterladen sowie löschen und bearbeiten. Es bietet auch die Verwaltung von Ordnern und Rechten als Access Control List an.

3.4 Dropbox

Dropbox ist ein kostenloser Service, mit dem sich die Daten auf verschieden Geräten synchronisieren lassen (Was ist Dropbox?). Gegründet wurde Dropbox im Jahr 2007 und steht als eine Plattform seit 2008 zur Verfügung. Heute hat Dropbox bereits mehr als hundert Millionen Benutzer. Am meisten wird Dropbox auf privaten Computern und mobilen Geräten verwendet. In viele Unternehmen wird ebenfalls bereits Dropbox eingesetzt, um z. B. die Daten zwischen Teammitgliedern zu verteilen bzw. zu synchronisieren. Dropbox bietet für die Unternehmen die Möglichkeit an, die Teams und Berechtigungen zu verwalten sowie eine Integration mit der Active Directory. Laut Dropbox

wird es bereits in mehr als zwei Millionen Unternehmen eingesetzt (Dropbox gibt es nun auch für Ihr Geschäft).

3.4.1 Dienste der Datenverwaltung

Dropbox bietet nur den Speicher als Dienst an. Um die Daten zu speichern, benutzt Dropbox Amazon S3 Speicher (Where does Dropbox store everyone's data?). Im Vergleich zu den anderen gehört Dropbox eindeutig zu SaaS Anbietern. Es besitzt aber die Schnittstellen, um eigene Anwendungen mit Dropbox benutzen zu können. Dropbox kann am besten mit SkyDrive[12] von Microsoft oder Google Drive verglichen werden, da sie ebenso zu SaaS Angeboten gehören. Es wäre aber interessant in dem Framework auch Dienste auf anderen Ebenen einzubinden.

Dropbox bietet drei Möglichkeiten an, sich mit ihm zu interagieren. Dropbox Chooser ist eine JavaScript Anbindung, die es ermöglicht Dropbox in die Webanwendung einzubauen. Dropbox Sync API vereinfacht die Programmierung von mobile Anwendungen, indem es die Synchronisation von lokalen Daten und Dropbox übernimmt. Dropbox Core API (Dropbox Core API Dokumentation) ist letztendlich für unser Framework interessant. Es bietet die REST Schnittstellen, über die die Daten vom und zum Dropbox geladen werden.

Über REST Schnittstelle lassen sich die Dateien hoch- bzw. herunterladen und Daten freigeben. Es können Ordner angelegt und die Metadaten abgefragt werden. Die Dateien bekommen eine Versionsnummer und können bis zu dreißig Tagen wiederhergestellt werden.

3.5 Gemeinsamkeiten

Bei allen Anbietern ist es möglich, sich bei dem Dienst über REST anzumelden. Bei Amazon S3 und Azure Blobe Storage funktioniert es über das Shared Key und die Signatur. Bei jeder Anfrage ist es nötig erst einmal eine Signatur zu erstellen. Mit einem Geheimschlüssel werden die Parameter verschlüsselt und daraus wird eine Signatur erstellt. Diese Signatur wird in die Anfrage eingebaut und verschickt. Der CSP erstellt die Signatur aus den Anfrage-Parametern und dem Schlüssel, der bei ihm gespeichert ist. Die Signaturen werden einschließend auf die Identität verglichen. Dropbox und Google Drive benutzen OAuth[13] für die Authentifikation.

Eine Datei kann hoch- bzw. heruntergeladen, gelöscht oder ersetzt werden. Teilweise funktioniert bei allen CSP‘s auch die Ordnerverwaltung. Bei Amazon S3 und Windows Azure Blob Service gibt es eine flache Hierarchie. Es kann aber eine virtuelle Hierarchie durch die

[12] http://windows.microsoft.com/de-de/skydrive
[13] http://oauth.net

Namenskonvention realisiert werden. Bei Dropbox und Google Drive sind REST Befehle für die Ordnerverwaltung vorhanden. Die Zugriffsverwaltung ist bei Amazon S3, Google Drive und Windows Azure über ACL realisierbar. Bei Dropbox lässt sich nur einstellen ob eine Datei bzw. ein Ordner für die Öffentlichkeit verfügbar ist. Sowohl Google Drive als auch Dropbox besitzen keine Versionskontrolle. Sie bieten jedoch eine Revision für die Dateien, mit welcher es möglich ist die Änderungen nachzuverfolgen.

Die Tabelle 3.1 bietet einen schnellen Überblick über die Gemeinsamkeiten der Datenverarbeitungsschnittstellen.

	Amazon S3	**Azure Blob Storage**	**Google Drive**	**Dropbox**
Container/Buckets	Ja	Ja	Nein	Nein
Blobs/Objects/ Files	Ja	Ja	Ja	Ja
Folders	Virtuell möglich	Virtuell möglich	Ja	Ja
Regions	Ja	Nur beim Account	Nein	Nein
Versionskontrolle	Ja	Ja	Nein	Nein
Zugriffskontrolle	IAM Policies /ACL / Bucket Policies	Private/Public	ACL	Private/Public
Authentifikation	Access Key + Signature	Access Key + Signature/ Shared Access Signature	OAuth 2.0	OAuth 2.0
Replikation steuerbar	Ja	Ja	Nein	Nein

Tabelle 3.1 Gemeinsamkeiten der Datenverarbeitungsschnittstellen.

4 Bereits existierende Lösungen

Vor einem Entwurf ist es sicherlich sinnvoll sich einen Überblick über die existierenden Lösungen zu verschaffen. Unter (Hogan, Liu, Sokol, & Jin, 2011) gibt NIST an, dass es unterschiedliche Szenarien bei der Nutzung von Clouds gibt. Bei den multiplen Clouds können Clouds sowohl nacheinander als auch gleichzeitig genutzt werden. Es sind folgende Szenarien möglich:

Multiple Clouds (sequentiell, nur eins auf einmal)

- Migration zwischen Clouds
- Schnittstelle zu mehreren Clouds
- Arbeit mit dem gewählten Cloud

Multiple Clouds (gleichzeitig, mehrere auf einmal)

- Agieren mit mehreren Clouds

Es wird zwischen verschiedenen Formen von multiplen Clouds unterschieden (Petcu, 2013). Einige davon sind z. B. Multi-Cloud, Cloud Federation, Inter-Cloud, Hybrid Cloud und andere. Die Unterschiede zwischen diesen Formen sind recht verschwommen. Es gibt zwei Auslieferungsmodelle in multiple Clouds: Federated Cloud und Multi-Cloud. Die Unterschiede sind festgelegt durch die Zusammenarbeit zwischen den involvierten Clouds und durch die Art der Benutzer-Interaktion mit diesen Clouds. Bei dem ersten Modell gibt es eine Vereinbarung für die Ressourcenfreigabe zwischen den Providern. Der Benutzer interagiert mit einem Cloud und weißt nicht, dass die Ressourcen von anderem Cloud mitbenutzt werden. Bei Multi-Cloud gibt es keine Vereinbarung zwischen den Providern und die Clouds werden unabhängig voneinander benutzt. Die anderen multiple Cloud Formen gehören meistens zu den beiden Modellen. Aus diesen Erkenntnissen folgt die Definition für Multi-Cloud.

Multi-Cloud Definition

Multi-Cloud ist eine sequentielle Form der Nutzung von mehreren Clouds, bei der keine Vereinbarungen zwischen den Providern existieren und die Clouds unabhängig voneinander genutzt werden.

Ein Multi-Cloud kann bibliothekbasierend oder servicebasierend sein. Für diese Bachelorarbeit sind die bibliothekbasierenden Multi-Clouds interessant. Die bekanntesten davon sind: JClouds, libcloud, δ-cloud und SimpleCloud.
Um einen Vergleich der Bibliotheken durchführen zu können, werden in der Tabelle 4.1 die Funktionalitäten vorgestellt, die bei einer optimalen Lösung erwartet werden.

	Amazon S3	Azure Blob Storage	Google Drive	Dropbox
Container/Buckets	optional	optional	Nein	Nein
Blobs/Objects/Files	Ja	Ja	Ja	Ja
Folders	Ja	Ja	Ja	Ja
Regions	optional	optional	Nein	Nein
Versionskontrolle	optional	optional	Nein	Nein
Zugriffskontrolle	Ja	Ja	Ja	Ja
Authentifikation	Ja	Ja	Ja	Ja
Replikation steuerbar	optional	optional	Nein	Nein

Tabelle 4.1 Optimale Lösung

4.1 JClouds

JClouds ist eine Open-Source-Bibliothek, die bei der Entwicklung der Cloud Anwendungen einen Entwickler unterstützt. Während diese Arbeit geschrieben wurde, wurde JClouds als ein Projekt bei Apache Software Foundation aufgenommen. Laut den Entwicklern von JClouds werden es 30 Cloud-Anbieter unterstützt (What is JClouds?, 2011). Unter diesen Anbietern sind: Amazon AWS, Microsoft Azure, Rackspace[14], HP Cloud und die anderen. Das JClouds API gibt die Möglichkeit portable Abstraktionen oder Cloud-spezifische Funktionen zu nutzen. JClouds bietet mehrere API-Abstraktionen, wie Java und Clojure Bibliotheken, an. Die ausgereiften von ihnen sind Blobstore und ComputeService.

JClouds konzentriert sich auf folgende Bereiche:

- **Einfache Schnittstelle** – Es ist einfach mit den Schnittstellen loszulegen, ohne sich mit REST-APIs oder Web Services zu beschäftigen. Statt neue Objekttypen zu erstellen, verwendet JClouds Konzepte, wie Maps, sodass die Programmierungsmodelle für einen Entwickler vertraut sind.
- **Laufzeit Portabilität** – Der JClouds Treiber ermöglicht es in eingeschränkten Umgebungen, wie Google App Engine, zu arbeiten. Es gibt nur wenige

[14] http://www.rackspace.com/

erforderlichen Abhängigkeiten, sodass eine Kollision mit eigener Anwendung unwahrscheinlich ist.

- **Deals mit Web Komplexität** - JClouds befasst sich mit den Fragen, wie vorübergehende Ausfälle und Umleitungen. Das sind die Themen, die das traditionelle netzwerkbasierte Computing einführt.
- **Unit Testbarkeit** – Um die automatische Tests ohne Netzwerkverbindungen zu ermöglichen, bietet JClouds die Stub-Verbindungen an, die eine Cloud simulieren.
- **Performance** – Es gibt die Möglichkeit die Konfiguration an die Performance-Bedürfnisse anzupassen. JClouds stellt asynchrone Befehle und einstellbare HTTP-, Datum- und Verschlüsselungsmodule bereit.
- **Standort** - JClouds bietet location-aware Abstraktionen an. Mit einem ISO-3166-Code lässt sich ein Land oder Provinz, in welchen eine Cloud läuft, bestimmen.
- **Qualität** - Vor jeder Freigabe wird jeder Anbieter mit den Live-Szenarien getestet. Falls der Anbieter den Test nicht besteht, geht die Entwicklung zurück in den Sandkasten.

Compute API

Das JClouds Compute API liefert eine grundlegende Abstraktion über Compute APIs der CSP, wie z. B. Amazon EC3 oder VMWare vCloud[15]. Das Compute API hat folgende funktionale Eigenschaften:

- **Location Aware API** - Das Compute API von JClouds erfordert keine mehrere Verbindungen zu Clouds, die mehrfach vernetzt sind. Es ist z. B. möglich, das gleiche Objekt zu nutzen, um auf Ressourcen in allen Regionen der EC2 zuzugreifen. Mit Location Aware API lassen sich Cross-WAN Ressourcen logisch zusammenbinden.
- **Node Sets** – Mit dem Compute API können mehrere Knoten, unabhängig von dem zugrunde liegenden Cloud-API, als ein Set laufen. Mit den Node-Sets ist es möglich bis zu 20 Knoten gleichzeitig zu verwalten, genauso wie einen einzelnen Knoten.
- **SSH-Schlüssel** - Hilft einen SSH-Schlüssel bei dem Start an die Knoten zu transportieren.
- **Skript ausführen** – Die runScript-Funktionalität ist eine Sammlung von Funktionen, die einfache Mittel zur Ausführung von Skripten auf den Maschinen anbieten. Mit den speziellen Ausnahme-Typen lassen sich Fehler behandeln.
- **Stub Provider** - Ein Stub-Compute-Provider hilft Bereitstellungsanweisungen zu testen, ohne eine Kreditkarte bei dem CSP verwenden zu müssen.
- **Credential Persistenz** – Mit dem Credential Persistenz ist es möglich die Anmeldeinformationen an einem einzigen Ort zu halten.

[15] http://www.vmware.com/de/products/datacenter-virtualization/vcloud-suite/overview.html

Blobstore API

Das Blobstore API ist ein tragbares Instrument zur Verwaltung der Schlüssel-Wert-Speicher-Anbieter, wie Microsoft Azure Blob-Service und Amazon S3 (jClouds BLOBSTORE GUIDE). Es bietet sowohl asynchrone als auch synchrone APIs sowie Map-basierten Zugriff auf die Daten.
Blobstore API aus mehreren APIs:

- **Location Aware API** – Location Aware API ermöglicht es einen Standort für die Cloud zu bestimmen. Z. B. Amerika oder Europa.
  Asynchronous API – Es gibt eine Möglichkeit zwischen einem synchronen oder einem asynchronen Blobstore API zu wählen. Damit lässt sich die Speicherung auf einem effizienteren Weg nutzen, ohne andere Prozesse zu blockieren.
- **Integration mit Non-Java Clients** – Die Nutzung von BlobRequestSigner ermöglicht es eine HTTP Anfrage zu erstellen, die an externe Prozesse weitergegeben werden kann.
- **Transient Provider** – Mit einem Transient Provider kann in-memory Blobstore ohne die Anmeldedaten und ohne eine Kreditkarte genutzt werden.
- **Filesystem Provider** – Ein Filesytem Provider erlaubt es, das gleiche API für Speicherung auf der Festplatte, in-memory oder remote zu benutzen.

Das JClouds Blobstore API unterstützt folgende Anbieter: AWS, CloudOne, HPCloud, Microsoft Azure, Ninefold, Rackspace und Synaptic.
Google Cloud Storage bietet einen interoperablen Zugriff an. Mit dem Interoperablen Zugriff kann eine kompatible API, wie z. B. Amazon S3, ebenso mit Google Cloud Storage arbeiten. Diese Technik benutzt auch JClouds. D.h. es ist möglich mit S3-API von JClouds ebenfalls das Google Cloud Storage zu steuern.
Einige Funktionen, wie z. B. Zugriffskontrolle, sind nicht abstrahiert und können nur über die provider-spezifische Funktionen erreicht werden. Die Tabelle 4.2 listet alle Funktionen auf, die JClouds für die ausgewählten Cloud Anbieter bereitstellt.

	Amazon S3	Azure Blob Storage	Google Drive	Dropbox
Container/Buckets	Ja	Ja	Nein	Nein
Blobs/Objects/Files	Ja	Ja	Nein	Nein
Folders	Ja	Ja	Nein	Nein
Regionen	Ja	Ja	Nein	Nein
Versionskontrolle	Ja	Nein	Nein	Nein
Zugriffskontrolle	Ja	Teilweise	Nein	Nein
Authentifikation	Ja	Ja	Nein	Nein
Replikation steuerbar	Ja	Nein	Nein	Nein

Tabelle 4.2 JClouds Blobstore API Provider Support

4.2 Libcloud

Ursprünglich wurde Libcloud von Cloudkick[16] entwickelt und ist zu einem unabhängigen freien Software-Projekt unter der Apache Lizenz(2.0) gewachsen (Apache Libcloud). Libcloud ist ein Standard Python Bibliothek, die die Differenzen zwischen den Provider APIs abstrahiert. Es besteht aus vier Komponenten die unterschiedliche Cloud Ressourcen steuern.

Compute Komponente

Die Compute API ermöglicht es, die virtuellen und Cloud-Server verschiedener Anbieter, wie Amazon, Rackspace, Linode und andere, zu steuern. Außerdem lassen sich mit Hilfe dieses API die Deployment-Scripts auf den neu erstellten Servern ausführen. Die Deployment-Scripts, die auch „bootstrap“-Scripts genannt werden, erlauben es, beliebige Shell-Befehle auszuführen. Das ist für die Konfiguration der Server sehr hilfreich und kann dazu benutzt werden, um z. B. SSH-Schlüssel auf den Servern zu installieren.

Über Compute Komponente können folgende Eigenschaften gesteuert werden:

[16] https://www.rackspace.com/cloudkick/

- **Node** – repräsentiert einen virtuellen oder Cloud-Server.
- **NodeSize** - repräsentiert Hardware-Konfiguration des Knotens. Normalerweise ist das die Menge des verfügbaren RAMs, der Bandbreite, der CPU-Geschwindigkeit und der Größe der Festplatte. Die meisten Treiber setzen auch Stundenpreis für den Knoten dieser Größe.
- **NodeImage** - stellt ein Abbild des Betriebssystems dar.
- **NodeLocation** - stellt einen physischen Standort des Servers dar.
- **NodeState** - stellt einen Knoten-Zustand dar. Standard Zustände sind: runnning, rebooting, terminated, pending und unknown.

Es stehen demzufolge die Aktionen wie Erstellen, Auflisten, Bereitstellen, Neustarten und Löschen von Servern zur Verfügung. Es ist möglich die Abbildungen der Server, dessen Größen und Standorte abzufragen.

LoadBalancer Komponente

Im Vergleich zu JClouds bietet Libcloud auch Steuerung der Load Balancer. Mit diesem API lassen sich die Load Balancer der folgenden Anbieter managen: Rackspace, GoGrid[17], Ninefold[18], CloudStack[19] und Amazon. Die Load Balancer können angelegt, aufgelistet und gelöscht werden. Bei der Erstellung lassen sich die Protokolle, wie z. B. HTTP oder FTP etc., einstellen und Algorithmen, wie z. B. Round Robin oder Random etc., festlegen. Zu den Load Balancer können selbstverständlich verschiedene IP-Adressen hinzugefügt werden.

DNS Komponente

Mit dem DNS API ist es möglich die DNS Dienste der folgenden Anbieter zu steuern: Linode[20], Zerigo[21], Rackspace, Amazon, HostVirtual[22] und Gandi.net[23].

Die wichtigsten Begriffe des DNS API sind:

- **Zone** – repräsentiert eine DNS Zone oder einen so genannten Domain.
- **Record** – repräsentiert einen DNS Eintrag. Jeder Eintrag gehört zu einer Zone und hat einen Typ und einen Datenfeld. Die Daten sind dabei von dem Typ abhängig.

[17] http://www.gogrid.com/cloud-hosting/load-balancers.php
[18] https://ninefold.com/support/display/SPT/Load+Balancing
[19] https://cloudstack.com/
[20] https://www.linode.com/wiki/index.php/Linode_DNS
[21] http://www.zerigo.com/managed-dns
[22] http://www.vr.org/
[23] http://www.gandi.net/

- **RecordType** – ein Datentyp kann sein: A, AAAA, MX, TXT etc.

Mit obengenannten Begriffen lassen sich die DNS Zonen und die entsprechende Einträge erstellen, auflisten, aktualisieren und löschen.

Storage Komponente

Mit dem Storage API können die Cloud Storage Dienste der Anbieter verwaltet werden. Es werden Amazon S3, Google Cloud Storage, Windows Azure und viele andere unterstützt. Mit dieser API lassen sich die Container bzw. Buckets erstellen, auflisten, und löschen. Die Objekte bzw. Blobs können hoch- und heruntergeladen sowie gelöscht werden. Einige Provider unterstützen das Hochladen als Stream.

Die Tabelle 4.3 bietet eine Übersicht über der unterstützten Funktionen der Datenverwaltung.

	Amazon S3	Azure Blob Storage	Google Drive	Dropbox
Container/Buckets	Ja	Ja	Nein	Nein
Blobs/Objects/Files	Ja	Ja	Nein	Nein
Folders	Ja	Ja	Nein	Nein
Regionen	Nein	Nein	Nein	Nein
Versionskontrolle	Nein	Nein	Nein	Nein
Zugriffskontrolle	Nein	Nein	Nein	Nein
Authentifikation	Ja	Ja	Nein	Nein
Replikation steuerbar	Nein	Nein	Nein	Nein

Tabelle 4.3 Libcloud Storage API Provider Support

Im Vergleich zu JClouds werden von Libcloud viel mehr Cloud Provider unterstützt. Die einzelnen Funktionen werden jedoch auf das grundlegende minimiert.

4.3 Deltacloud

Deltacloud wird ebenfalls durch Apache Software Foundation gefördert. Die Deltacloud bietet als Lösung einen API Server und die Treiber, die für die Kommunikation mit dem CPS benötigt werden (About Deltacloud).

Durch die Nutzung eines von drei unterstützten APIs, ermöglicht Deltacloud die Verwaltung von Ressourcen in unterschiedlichen Clouds. Die unterstützten APIs sind Deltacloud classic API, das DMTF[24] CIMI[25] API und das AWS EC2 API. Das API funktioniert wie ein Wrapper für viele Clouds und abstrahiert dadurch die Unterschiede zwischen diesen. Für jeden CSP existiert grundsätzlich ein natives API des Provider, das die Kommunikation mit dem Provider ermöglicht.

Deltacloud benötigt eine Ruby Installation auf dem Rechner. Um Deltacloud nutzen zu können, ist es notwendig Deltacloud Server mit Hilfe von Ruby zu installieren. Nach der Installation steht der Deltacloud Server über einen HTML Browser zur Verfügung und kann ebenso über selbstgeschriebenen Client verwendet werden. Da der Deltacloud Server eine eigene REST Schnittstelle bereitstellt, ist es möglich mit ihm über HTTP-Requests zu kommunizieren. Es werden zwei Bereiche der Clouds unterstützt. Das ist Compute und Storage Bereiche.

Die folgenden Begriffe beschreiben die Abstraktion, die bei Deltacloud unterstützt wird. Es ist zu beachten, dass diese Sammlung von Cloud zu Cloud abweichen kann.

- **Realms** – eine organisierte Einheit innerhalb der Back-End-Cloud. Diese Einheit kann z. B. ein Rechenzentrum sein. Ein Realm kann eine geografische Lage der IT-Ressourcen darstellen.
- **Instanzen** – sind realisierte virtuelle Server, die auf dem Back-End-Cloud laufen. Instanzen werden von Server-Images instanziiert.
- **Abbildungen** – sind die Vorlagen (Images virtueller Maschinen), aufgrund deren die Instanzen gebildet werden. Jedes Image bestimmt die Root-Partition und das Storage für das Instanz-Betriebssystem.
- **Keys** – repräsentieren die Anmeldeinformationen, die für die Anmeldung bei der Instanz verwendet werden. Dies können sowohl RSA Keys als auch ein Paar der Benutzername und das Password sein.
- **Storage volume** – ist ein virtuelles Speichergerät, das zu einer Instanz hinzugefügt werden kann.
- **Bucket** – kann je nach CPS ein Container oder ein Bucket sein.

[24] Distributed Management Task Force, Inc. http://dmtf.org/
[25] Cloud Infrastructure Management Interface. http://dmtf.org/standards/cloud

- **Blob** – repräsentiert je nach CPS ein Object oder ein Blob.
- **Adresse** – repräsentiert eine IP-Adresse. Das kann je nach CPS eine private oder öffentliche IP-Adresse sein.
- **Load Balancer** – ermöglicht eine Lastverteilung anhand von einer IP-Adresse zwischen mehreren Instanzen.
- **Firewalls** – ist eine Sammlung von Regeln, die den Zugang zu einer laufenden Instanz über das Internet regulieren.
- **Metrics** – ist eine Menge die nützliche Informationen, wie z. B. CPU-Auslastung oder Netzwerktrafik, über die Cloud Ressourcen liefern.

Die Compute Ressourcen des Deltacloud API beinhalten die Treiber für vierzehn Anbieter. Windows Azure wird nicht unterstützt. Mit den Compute Ressourcen lassen sich die Instanzen verwalten, die Lasten verteilen, Firewalls einstellen und die Zugriffsreche bestimmen.

Die Storage Ressourcen unterstützen sieben Cloud Anbieter. Darunter sind auch Amazon S3 und Windows Azure. Die Storage Ressourcen sind in zwei Gruppen aufgeteilt. Die virtuellen Festplatten können über Storage Volums erstellt, aufgelistet, zu den Instanzen hinzugefügt oder gelöscht werden. Mit Blob Storage können die Daten auf einem Storage Volume verwaltet werden. Es ist außerdem möglich die Buckets und Blobs zu erstellen und zu löschen, sowie die Metadaten eines Blobs zu erfragen. Die Storage Ressourcen unterstützen zudem die Storage Snapshots. Mit Storage Snapshots kann eine Sicherung eines Storage erstellt werden.

In der Tabelle 4.4 werden die Funktionen der Datenverwaltung dargestellt, die von Deltacloud unterstützt werden.

	Amazon S3	Azure Blob Storage	Google Drive	Dropbox
Container/Buckets	Ja	Ja	Nein	Nein
Blobs/Objects/Files	Ja	Ja	Nein	Nein
Folders	Ja	Ja	Nein	Nein
Regionen	Nein	Nein	Nein	Nein
Versionskontrolle	Nein	Nein	Nein	Nein
Zugriffskontrolle	Nein	Nein	Nein	Nein
Authentifikation	Ja	Ja	Nein	Nein
Replikation steuerbar	Nein	Nein	Nein	Nein

Tabelle 4.4 Deltacloud API Storage supported Provider

Der größte Unterschied von Deltacloud zu den anderen Multi-Cloud Lösungen ist die Darstellung einer server-basierten Lösung. Es stellt einen Nachteil bei einigen Anwendungen dar. Falls z. B. eine Ruby bzw. Server Installation nicht erlaubt ist, wäre eine Installation auf einem entfernten Rechner immer noch möglich. Andererseits wird dadurch der Datentrafik erhöht, da die Daten zuerst zum Deltacloud-Server transportiert werden und erst danach an den Cloud-Anbieter weitergeleitet werden.

4.4 Simple Cloud API

Simple Cloud API ist eine Lösung die in PHP realisiert ist (Simple Cloud API). Seit 2010 ist Simple Cloud ein fester Bestandteil vom Zend Framework 2.0[26]. Das API besteht aus verschiedenen Adaptern und deckt drei Bereiche der Cloud.

- **File Storage Services** - mit Adaptern für die Dienste, wie Amazon S3 und Azure Blob Storage.
- **Document Storage Services** – mit Adaptern für die Dienste, wie Amazon SimpleDB und Azure Table Storage
- **Queue Services** – mit Adaptern für die Dienste, wie Amazon SQS und Azure Queue Storage

[26] http://framework.zend.com/

- **Infrastructure** – mit Adaptern für die Steuerung von virtuellen Instanzen, wie Amazon EC2.

Infrastructure

Bei den Infrastucture Services unterstützt Simple Cloud das Erstellen, Löschen, Starten, Stoppen, Neustarten, Auflisten, Statusabfragen, IP- und DNS-Abfrage einer Instanz. Außerdem können mit Simple Cloud API die verfügbaren Instanz-Abbildungen und geografischen Zonen für die neuen Instanzen sowie die Ressource-Informationen abgefragt und die Shell-Scripts ausgeführt werden. Es werden zurzeit Amazon EC2 und Rackspace unterstützt.

Queue Services

Mit Simple Cloud lassen sich die Queues erstellen, auflisten und löschen. Über eine Queue können dann die Nachrichten verschickt, empfangen und anschließend gelöscht werden.

Document Storage Services

Die Document Storage Services erlauben es die Key-Value Datenbanken zu verwalten. Mit Simple Cloud API ist es möglich die Collections anzulegen und zu löschen, die Dokumente mit den Schlüssel-Wert Paaren zu erstellten, auszulesen, zu ersetzen, zu aktualisieren und zu löschen. Mit Queries können Collections bzw. Dokumente durchgesucht werden. Das Simple Cloud API enthält die Adapter für Amazon SimpleDB und Windows Azure Tables.

File Storage Services

Das Simple Cloud API beinhaltet die Storage Service Adapter für Amazon S3, Azure Blob Storage, Nirvanix[27], Rackspace und Dateisystem. Das Simple Cloud API unterstützt das Verwalten von Blobs bzw. Objects und dessen Metadaten.

In der Tabelle 4.5 wird der Vergleich der unterstützten Funktionen der Datenverwaltung dargestellt.

[27] http://www.nirvanix.com/

	Amazon S3	Azure Blob Storage	Google Drive	Dropbox
Container/Buckets	Nein	Nein	Nein	Nein
Blobs/Objects/Files	Ja	Ja	Nein	Nein
Folders	Ja	Ja	Nein	Nein
Regionen	Nein	Nein	Nein	Nein
Versionskontrolle	Teilweise über Metadaten	Nein	Nein	Nein
Zugriffskontrolle	Teilweise über Metadaten	Teilweise über Metadaten	Nein	Nein
Authentifikation	Ja	Ja	Nein	Nein
Replikation steuerbar	Teilweise über Metadaten	Nein	Nein	Nein

Tabelle 4.5 Simple Cloud Storage supported Provider

Um das Simple Cloud API nutzen zu können, werden eine PHP Installation und die Zend Bibliotheken benötigt. Wenn ein Rechner, wie z. B. Smartphones, keine Installation von PHP bzw. Zend erlaubt, kann das zu Problemen führen. Die Installation und Nutzung auf einem entfernten Rechner ist nur mit einem zusätzlichen Programmieraufwand möglich, da das Simple Cloud API keine eigene REST Schnittstelle anbietet. Mit dieser API ist die Nutzung meistens nur für Serverlösungen geeignet.

4.5 Bewertung der existierenden Lösungen

Aus den oben aufgeführten Tabellen wird ersichtlich, dass keine der existierenden Lösungen optimal ist. Viele Funktionen werden nur teilweise oder gar nicht unterstützt. Alle Lösungen begrenzen sich auf die Hauptfunktionen wie das Lesen, Schreiben, Aktualisieren und Löschen von Objekten bzw. Containern. Keine der existierenden Lösungen unterstützt Dropbox oder Google Drive. Die Funktionen, wie Zugriffskontrolle, Versionskontrolle, Einstellung des Standortes oder Steuerung der Replikation, werden nativ nicht unterstützt. Diese Funktionen sind nur teilweise über Metadaten der Objekte verfügbar, falls die Cloud

Anbieter dies ermöglichen. Bei einigen Lösungen ist es möglich, auf die native Treiber bzw. Adapter zuzugreifen. Solche Zugriffe zerstören die Abstraktion der Schnittstellen.

Da viele Funktionen bereits in den existierenden Lösungen enthalten sind, ist es sinnvoll auf einer der Lösungen aufzubauen. Die Tabelle 4.6 stellt ein Vergleich zwischen aller vorgestellten Lösungen und optimaler Lösung dar. Als eine Bibliothek, auf der weiter aufgebaut werden kann, passt die JClouds Bibliothek vom Funktionsumfang und der Programmiersprache am besten, wie es aus der Tabelle zu entnehmen ist.

	Amazon S3	**Azure Blob Storage**	**Google Drive**	**Dropbox**
	J | L | D | S | O	**J | L | D | S | O**	**J | L | D | S | O**	**J | L | D | S | O**
Container/Buckets	+ | + | + | - | 0	+ | + | + | - | 0	- | - | - | - | -	- | - | - | - | -
Blobs/Objects/Files	+ | + | + | + | +	+ | + | + | + | +	- | - | - | - | +	- | - | - | - | +
Folders	+ | + | + | + | +	+ | + | + | + | +	- | - | - | - | +	- | - | - | - | +
Regionen	+ | - | - | - | 0	+ | - | - | - | 0	- | - | - | - | -	- | - | - | - | -
Versionskontrolle	+ | - | - | T | 0	- | - | - | - | 0	- | - | - | - | -	- | - | - | - | -
Zugriffskontrolle	+ | - | - | T | 0	T | - | - | T | 0	- | - | - | - | 0	- | - | - | - | 0
Authentifikation	+ | + | + | + | +	+ | + | + | + | +	- | - | - | - | +	- | - | - | - | +
Replikation steuerbar	+ | - | - | T | 0	- | - | - | - | 0	- | - | - | - | -	- | - | - | - | -

Legende:

J = JClouds	S = Simple Cloud	+ = Ja	T = Teilweise
L = Libcloud		- = Nein	0 = Optional
D = Deltacloud	O = Optimal		

Tabelle 4.6 Vergleich JClouds Blobstore API und optimale Lösung

5 Lösungsansatz

Wie bereits oben erwähnt, ist es sicherlich einfacher, auf einer bereits existierenden Lösung aufzubauen. Dafür eignet sich JClouds am besten. Das Ziel dieses Kapitels ist es, zu zeigen, wie JClouds aufgebaut ist und wie es funktioniert bzw. wie es erweitert werden kann. Anschließend wird darauf eine eigene Lösung entwickelt.

Die Realisierung setzt eine Architektur und einen Entwurf voraus. Damit es auf dem JClouds Framework aufgebaut werden kann, ist es notwendig, sich die Architektur von JClouds anzuschauen. Leider gibt es keine brauchbare Entwicklerdokumentation von JClouds. In diesem Kapitel wird beschrieben, wie die Architekturen vom JClouds und XClouds Frameworks aussehen und wie sie realisiert werden. Anschließend folgt eine kurze Anleitung zu den notwendigen Installationsschritten.

5.1 JClouds Architektur

Zuerst wird eine Kontextabgrenzung des JClouds Frameworks gemacht, die in der Abbildung 5.1 dargestellt wird.

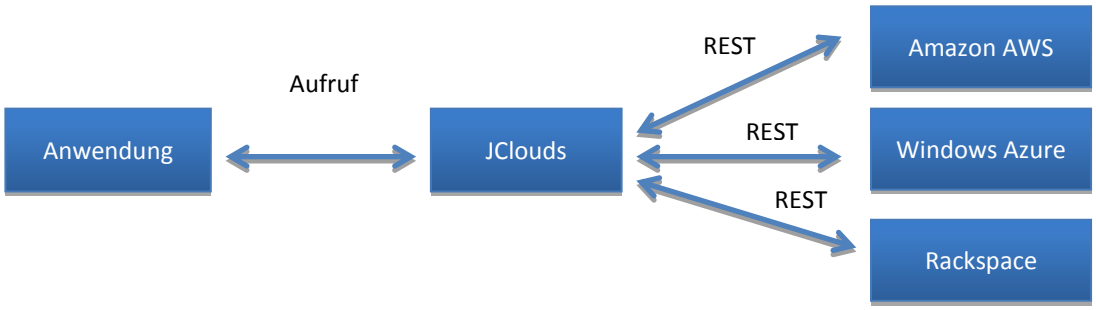

Abbildung 5.1 Die Kontextabgrenzung von JClouds

Eine Anwendung benutzt als eine Zwischenstelle JClouds Framework und kennt im Normalfall den Provider bzw. seine Schnittstellen nicht. JClouds übernimmt die Kommunikation über eine REST Schnittstelle mit den ISP's. Eine Anwendung kann in dem Fall sowohl ein Java als auch ein Clojure[28] Programm sein. Vorausgesetzt ist aber eine Java Virtual Machine.

JClouds ist Modul-basierend aufgebaut. Das gesamte JClouds Projekt aus vielen Modulen. Ein fester Bestandteil ist das JClouds-Core-Modul. Es gibt Module für JClouds API und JClouds Provider, die mit Hilfe von Dependency Injection Pattern eingebunden werden.

[28] http://clojure.org/

JClouds Core besteht aus den verschiedenen Komponenten, die Kommunikation mit einem ISP ermöglichen und viele nützliche Funktionen anbieten. Es verwaltet den gesamten Lebenszyklus des JClouds Frameworks. Alle Komponenten werden mittels Dependency Injection Pattern eingebunden.

Die Idee hinter Dependency Injection Pattern ist die lose Kopplung zwischen den Komponenten in einem System zu schaffen (Schatten, et al., 2010). Die Architektur der Software beschreibt dabei wie die Komponenten in Verbindung zueinander stehen, deren Abhängigkeiten und wie sie miteinander zur Laufzeit kommunizieren. Die Komponenten müssen wissen, welche anderen Komponenten benötigt werden, wo sie sich befinden und wie mit diesen kommuniziert wird. Dafür werden die entsprechenden Schnittstellen geschaffen. Eine Implementierung dieser Schnittstellen folgt später.

JClouds benutzt für die Injections das Google Guice Framework[29]. In Abbildung 5.2 ist die Bausteinsicht des JClouds Frameworks dargestellt. Eine Anwendung kommuniziert meistens mit JClouds Core. JClouds Core besteht aus folgenden Komponenten:

- **REST** – stellt die Annotations, Bindings, Metadata und andere Klassen, die für REST Kommunikation notwendig sind, zur Verfügung.
- **HTTP** – ist für das Absenden von Anfragen und Empfangen von HTTP Antworten zuständig.
- **Logging** – stellt die Logging Funktionen bereit.
- **Fallbacks** – übernimmt das Behandeln und weiterleiten von Ausnahmen.
- **Locations** – ermöglicht es, den Providern und API's mit verschiedenen Orten des ISP's zu arbeiten.
- **Providers** – stellt die abstrakten Klassen und die Schnittstellen für JClouds Provider bereit.
- **API's** – stellt die abstrakten Klassen und die Schnittstellen für JClouds API's bereit.

JClouds Core beinhaltet auch andere Komponenten, die hier nicht aufgelistet werden. JClouds Core bindet die JClouds API's und Providers ein, die im Normalfall für die Anwendung nicht sichtbar sind. Eine Anwendung kann aber direkt auf die API's oder Provider zugreifen, um mehr Funktionen einer Schnittstelle zu erhalten.

Die JClouds API's sind die REST Schnittstellen, die bei einigen ISP's gleich sind. Es lassen sich beispielweise mit S3 API sowohl Amazon S3 als auch Google Cloud Storage steuern. Die JClouds Provider bieten die provider-spezifischen Funktionen an, die in einer JClouds API nicht enthalten sind. So bietet AWS-S3 Provider die Funktionen an, die bei S3 API nicht enthalten sind. Es gibt dennoch Provider, die keine JClouds API benutzen, wie z. B. Windows Azure Provider.

[29] http://code.google.com/p/google-guice/

JClouds benutzt die Patterns wie Builder, um dynamisch die Provider und API's zu erstellen, und Dependency Injections, um diese in das JClouds Core einzubinden.

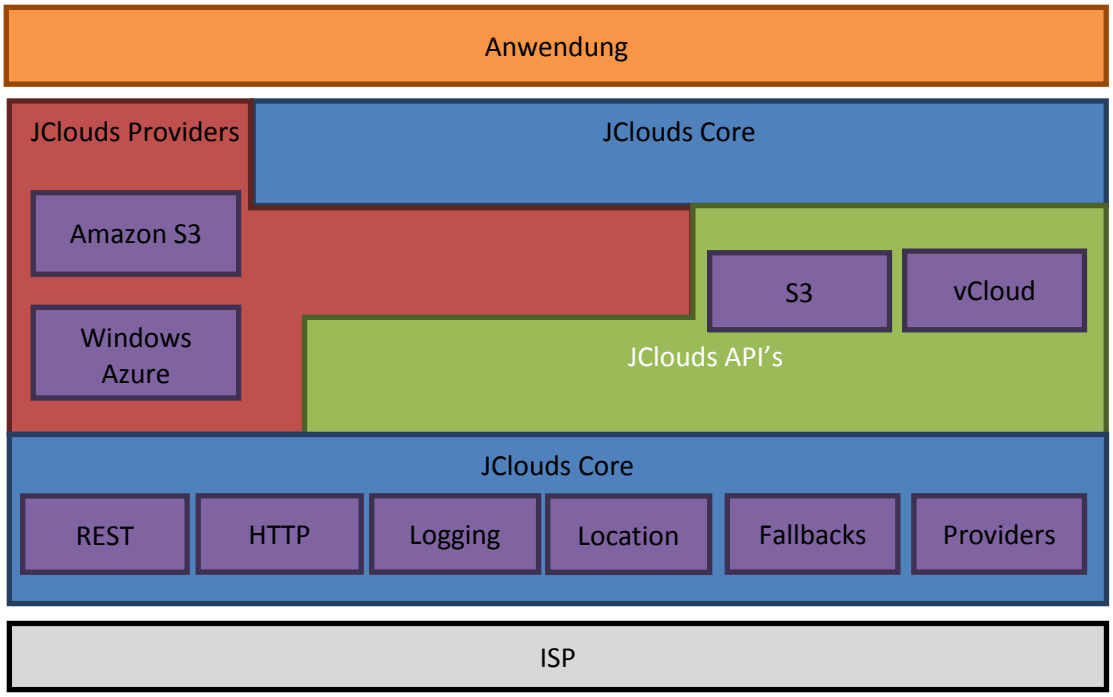

Abbildung 5.2 Die Bausteinsicht von JClouds

5.1.1 JClouds Provider

Wie in der JClouds Dokumentation für BlobStore (jClouds BLOBSTORE GUIDE) beschrieben und in Listing 5.1 dargestellt ist, wird ein ContextBuilder dafür benutzt, um über die Eingabe von einem Provider und Anmeldedaten ein BlobStoreContext Objekt zu erstellen. Dieses liefert nützliche Funktionen wie z. B. Utilities und wird zum Erstellen von einem BlobStore

```
BlobStoreContext context = ContextBuilder.newBuilder("aws-s3")
                           .credentials(identity, credential)
                           .buildView(BlobStoreContext.class);
BlobStore bs = context.getBlobStore();
Blob blob = bs.blobBuilder("mydirectory/File.png")
            .payload(new File("/path/to/file/File.png"))
            .contentDisposition("attachment; filename= File.png")
            .contentType("image/png")
            .calculateMD5().build();
```

Listing 5.1 Hochladen eines Blobs mit JClouds

Objekt sowie zum Schließen einer Verbindung benutzt. Ein BlobStore Objekt liefert alle notwendigen Funktionen, um mit einem Storage zu arbeiten. Über eine BlobBuilder, das von BlobStore Objekt geliefert wird, wird ein Blob Objekt erstellt. Dabei lassen sich die Einstellungen verwenden, die für eine REST Anfrage benötigt werden, bzw. auch die Datei zum Hochladen übergeben.

Jeder Provider oder API soll mindestens eine Klasse ApiMetadata bereitstellen. In dieser Klasse werden die Einstellungen vorgenommen, die für die Erstellung von Provider bzw. für die Kommunikation notwendig sind.

- **id** – eine eindeutige ID für einen Provider.
- **name** – ein Name für den Provider.
- **views** – es können verschiedene Views eines Providers geben.
- **endpointName** – ein Anzeigename für den Zugriffspunkt.
- **identityName** – ein Anzeigename für die Identifikation, wie z. B. „user“ oder account“.
- **credentialName** – ein Anzeige Name für einen Berechtigungsnachweis, wie z. B. „password“ oder „secret“.
- **version** – die Version der ISP‘s API, die verwendet wird.
- **buildVersion** – die Build Version des Servers, zu dem JClouds sich verbindet.
- **defaultEndpoint** – Ein Standard-Zugriffspunkt, der verwendet wird, falls er nicht vom ProviderMetadata überschrieben wird.
- **defaultIdentity** – ein Standard-Identifikationsname, der verwendet wird, falls er nicht vom ProviderMetadata überschrieben wird.
- **defaultCredential** - ein Standard- Berechtigungsnachweis, der verwendet wird, falls er nicht vom ProviderMetadata überschrieben wird.
- **defaultProperties** – verschiede Einstellungen, die für eine Verbindung relevant sein können, wie z. B. „connection timeout“.
- **documentation** – eine URL, die das ISP Service beschreibt.
- **context** – das Provider Context, wie z. B. BlobContext.
- **defaultModules** – die Module, die Dependency Injections konfigurieren.

Ein Provider kann dazu noch eine ProviderMetadata Klasse zur Verfügung stellen. Diese Klasse kann einige Einstellungen aus ApiMetadata ergänzen oder überschreiben. Um ein Cloud steuern zu können, muss ein Provider eine Sigelten Klasse, die von Blobstore Klasse abgeleitet wird, bereitstellen. Außerdem benötigt ein Provider die Klassen [Provider Name]Client und [Provider Name]AsyncClient. Die Klasse AsyncClient enthält die Annotationen, die die Schnittstelle des ISP’s beschreibt. In der Klasse Client wird die Java Schnittstelle beschrieben. Die Klasse Client korrespondiert mit der Klasse AsyncClient mittels eines Proxy, der von JClouds Core erstellt wird. Um die Headers oder den Content einer Response zu parsen, kann ein Provider verschiedene Funktionsklassen bereitstellen. Bei der Erstellung eines Request können unterschiedliche Filter eingesetzt werden, die den

Request vor dem Absenden noch einmal bearbeiten. Meistens beinhaltet ein Provider auch ein Domain Package, in dem sich die Klassen für die Arbeit mit der Domain befinden. Es können z. B. Klassen für die Metadaten oder Blobs und Objekte sein.

Damit ein Provider bzw. andere Module eingebunden werden können, werden die Klassen für die Konfiguration der Dependency Injections benötigt. Diese Klassen werden meinst [Name]Module genannt und von der AbstractModule Klasse abgeleitet.

5.2 Architektur und Entwurf von XClouds

Um das Framework der Lösung namentlich von dem JClouds Framework abzugrenzen, wird im Folgenden der Name XClouds verwendet. Das XClouds Framework soll JClouds Framework erweitern und benutzen sowie eine andere Abstraktion der Storage anbieten. Eine Anwendung wird nur über eine Fassade des XClouds Frameworks mit den ISP's kommunizieren. Wie in Abbildung 5.3 gezeigt ist, soll die Anwendung nicht wissen, dass dahinter ein JClouds Framework verwendet wird bzw. welcher der ISP genutzt wird.

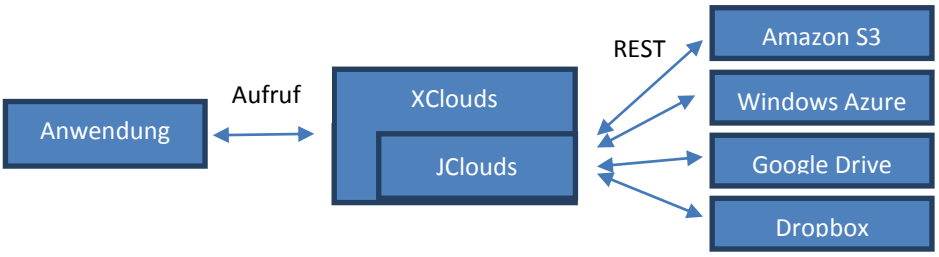

Abbildung 5.3 Die Kontextabgrenzung von XClouds

Für die Erstellung des XClouds Frameworks wird das JClouds Framework um die Provider für Dropbox und Google Drive erweitert. Anschließend wird eine Abstraktion von dem JClouds Framework geschafften, damit es eine einheitliche Schnittstelle bei allen vier Providern entsteht. Die Anwendung greift auf die XClouds Framework über die Fassade zu. Dadurch wird es möglich, die Provider in einer Anwendung auszutauschen, ohne den Quellcode zu verändern. Damit eine Kommunikation mit Dropbox und Google Drive stattfinden kann, muss XClouds noch andere Komponenten, wie z. B. OAuth für die Authentifizierung, bereitstellen. Wie die Bausteinsicht des XClouds Frameworks aussieht, ist in der Abbildung 5.4 dargestellt.

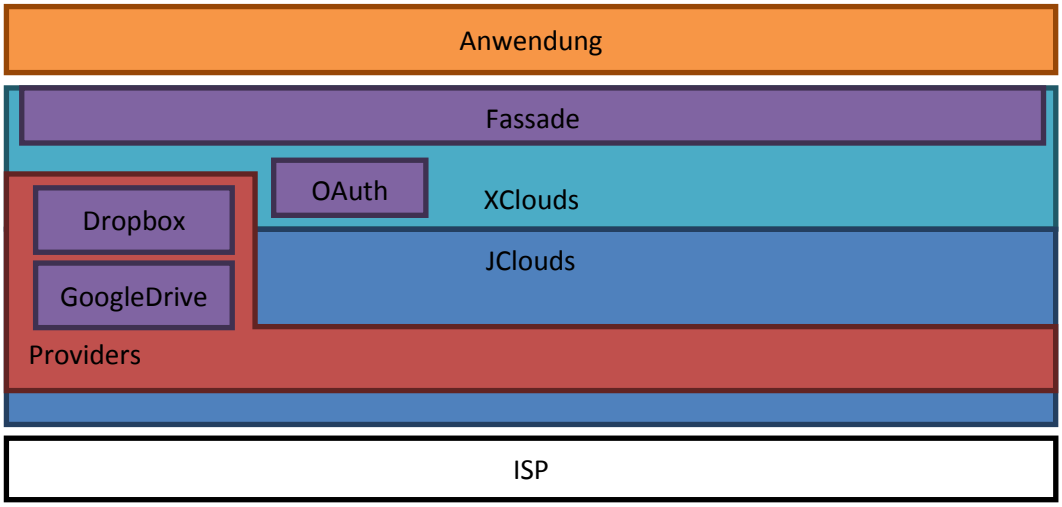

Abbildung 5.4 Die Bausteinsicht von XClouds

5.2.1 Abstraktion

Die ISP Schnittstellen der vier ISP's sind nicht zueinander kompatibel und es werden nicht alle Funktionen unterstützt. Die Verbindung von Storage-Angeboten aus den verschiedenen Ebenen führt dazu, dass viele Funktionen nicht mehr abstrahiert werden können. Dropbox und Google Drive bieten z. B. keine Replikationssteuerung, Regionen-Auswahl und Versionskontrolle, wie es aus der Tabelle 4.6 erkennbar ist. Damit es trotzdem bei den anderen Providern funktioniert, werden für die Provider optionale Komponenten entwickelt. Falls ein Provider die Funktion nicht unterstützt, wird ein Fehler ausgeworfen, sobald man versucht, einen Provider mit diesen Komponenten zu konfigurieren. Das ist notwendig, da die Aufrufe dieser Funktionen nicht einfach von einem Provider ignoriert werden können. Dies kann unter Umständen zu unerwünschten Effekten führen. Die Container oder Buckets können beim Dropbox und beim Google Drive als ein Root-Ordner angesehen werden.

Allgemein betrachtet kann jedes Cloud Storage mit den Dateien und Ordnern umgehen. Dafür kann eine Abstraktion entwickelt werden. Jedes Cloud Storage kann also folgende Funktionen anbieten:

- Dateien auflisten, lesen, erstellen, löschen.
- Ordner auflisten, lesen, erstellen, löschen.
- Die Dateien für andere Benutzer freigeben, wenn die Dateifreigabe von dem Anbieter unterstützt wird.
- Jedes ISP bietet eine Authentifizierung an. Die Authentifizierung für Amazon S3 und Windows Azure wurde bereits in JClouds umgesetzt. Es wird noch eine

Komponente für OAuth, das beim Dropbox und beim Google Drive benutzt wird, benötigt.

Die Fassade

Die Fassade dient dazu, den Zugriff auf ein komplexes Subsystem oder auf eine Menge zusammenhängender Objekte für einen Client zu vereinfachen. Der Einsatz eines Fassaden-Musters dient zur Entkopplung des Client von der Implementierung eines Subsystems. Dadurch ist es möglich sowohl das Subsystem als auch die Schnittstellen zu verändern ohne den Client anpassen zu müssen (Eilebrecht & Starke, Fassade, 2013). Die Fassade des XClouds muss Folgendes können:

- einen Provider eingeben.
- die Zugangsdaten für den Provider eingeben.
- die optionalen Komponenten zur Verfügung stellen.
- eine Datei in einem Ordner bzw. Container auflisten, erstellen, löschen und lesen.
- Einen Ordner erstellen und löschen.
- eine Datei als öffentlich oder als privat bereitstellen.

OAuth

Das OAuth Authorisation Framework ermöglicht es, einer Drittanbieter-Anwendung einen begrenzten Zugang auf die HTTP-Dienste zu erhalten. Dies kann auf zwei Wegen geschehen. Durch die Interaktion mit dem HTTP-Dienst kann der Benutzer für eine Anwendung eine Erlaubnis für den Zugriff auf die Dienste expliziert erteilen. Die Anwendung kann ebenso im Namen des Benutzers auf die Dienste zugreifen (Hardt, 2012). Diese Art von Autorisierung wird meistens in Mobilen- und Web-Anwendungen verwendet. Für die Dienste, wie Dropbox oder Google Drive, wird die OAuth2.0 Autorisierung benötigt. Diese Komponente wird von JClouds nicht bereitgestellt und wird in XClouds Framework benötigt.

Um die Funktionsweise von OAuth 2.0 zu verdeutlichen, wird am Beispiel des Google Drive die Authentifizierung mit OAuth 2.0 in der Abbildung 5.5 gezeigt.

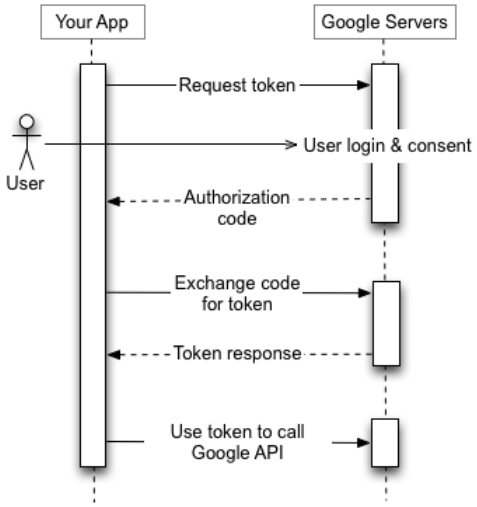

Abbildung 5.5 Google Drive Authentifizierung mit OAuth 2.0 (Google Inc.)

Bevor eine Anwendung sich über OAuth anmelden kann, muss sie in den meisten Fällen bei dem OAuth Anbieter registriert werden. Bei Dropbox dient dafür die App Console, die unter https://www.dropbox.com/developers/apps erreichbar ist. Die Google Console ist unter https://cloud.google.com/console erreichbar. In beiden Fällen muss die Applikation erst registriert werden. Für den Zugriff auf die Dienste werden die Zugriffsdaten von dem Dienstprovider bereitgestellt. Diese Zugriffsdaten werden bei der Applikation verwendet, um diese bei der OAuth Authentifikation eindeutig zu identifizieren.

Die Autorisierung wird in einem Webbrowser durchgeführt. Bei der Autorisierung meldet sich der Benutzer mit seinen Zugangsdaten bei dem Dienst an und wird weitergeleitet. Er wird gefragt, ob er eine Zugriffserlaubnis für die Anwendung erteilen möchte. Nach der Bestätigung erfolgt eine Weiterleitung an die Anwendung zurück. Die Anwendung bekommt einen Code mitgeteilt. Diesen Code verwendet die Anwendung, um eine Token-Anfrage bei dem OAuth-Server durchzuführen. Nachdem die Anwendung einen Token bekommen hat, kann dieser Token für die Autorisierung benutzt werden.

Replikationskomponente

Für den Fall, dass ein Provider die Replikation unterstützt, soll die XClouds Fassade eine Replikationskomponente zur Verfügung stellen. Dazu soll die Fassade ein Null-Objekt Pattern verwenden.

„Es wird eine Klasse definiert, die „nichts“ tut – wobei das „Nichts“ fachlich ist.“ (Eilebrecht & Starke, Null-Objekt, 2013)

Das bedeutet, dass eine Methode, anstatt ein NULL zurückzugeben, ein Null-Objekt ausgibt. Dieses Objekt soll die gleiche Schnittstelle unterstützen, wie ein Objekt, das bei der Rückgabe erwartet wird. Das Null-Objekt macht jedoch im Normalfall nichts. Alle Aufrufe an einem Null-Objekt führen zu keinem Ergebnis. Durch das Null-Objekt wird verhindert, dass das Programm bei NullPointerException abstürzt.

Es soll möglich sein, bei der Fassade nachzufragen, ob die Replikation von dem Provider unterstützt wird. Falls die Replikation von dem Provider nicht unterstützt wird und es trotzdem auf die Replikationskomponente zugegriffen wird, soll ein Null-Objekt zurückgegeben werden.

Region-Komponente

Da die Regionen auch nicht von allen Providern unterstützt werden, soll hier auch der gleiche Ansatz genutzt werden, wie auch bei Replikationskomponente.

5.2.2 REST Schnittstellen

Um die JClouds Provider zu entwickeln ist es notwendig, die REST Schnittstellen der Provider zu analysieren. Durch die Analyse soll festgestellt werden, welche der Funktionen benötigt werden.

Dropbox

Die Dropbox REST Schnittstelle bietet folgende Funktionen an (Dropbox Core API Dokumentation):

- ***OAuth 2.0***
  - **/authorize** - startet das OAuth 2.0 Ablauf und liefert den Code für die Token-Anfrage zurück.
  - **/token** - liefert den OAuth Token.
- ***Dropbox accounts***
  - **/account/info** – liefert die Account Informationen als JSON.
- ***Files and metadata***
  - **/files (GET)** – lädt die Datei herunter.
  - **/files_put** – lädt die Datei hoch.
  - **/files (POST)** – ist das gleiche, wie files_put, jedoch mit POST Methode als HTTP Request.
  - **/metadata** – gibt die Metadaten einer Datei oder eines Ordners als JSON.
  - **/delta** – gibt die Informationen über die geänderten Dateien und Ordner.

  - **/revisions** – liefert die Metainformationen zu der Vorversion einer Datei zurück.
  - **/restore** – stellt die Vorversion einer Datei wieder her.
  - **/search** – sucht nach einer Datei oder einem Ordner und liefert die Metadaten zurück.
  - **/shares** – erstellt eine URL, über die es möglich ist, eine Datei oder einen Ordner in dem Browser zu sehen.
  - **/media** – ist vergleichbar mit share. Es wird aber für Streaming benutzt.
  - **/copy_ref** – liefert eine Referenz, die benutzt werden kann, um die Datei zu einem anderen Dropbox-konto zu kopieren.
  - **/thumbnails** – liefert ein Vorschaubild für die Datei zurück.
  - **/chunked_upload** – initialisiert ein Upload für die großen Dateien.
  - **/commit_chunked_upload** – lädt ein Teil der Datei in einem Upload Prozess, das mit chunked_upload initialisiert wurde.
- ***File operations***
  - **/fileops/copy** – kopiert die Datei zu einem anderen Ort oder zu einem anderen Konto.
  - **/fileops/create_folder** – erstellt einen Ordner.
  - **/fileops/delete** – löscht eine Datei oder einen Ordner.
  - **/fileops/move** – verschiebt die Datei oder einen Ordner.

Für die Entwicklung sind zurzeit nicht alle diese Funktionen notwendig. Es werden folgende Funktionen benötigt: die Funktionen zum Erstellen und zum Löschen von Ordner; zum Hoch- und Herunterladen der Dateien; die „chunked_upload“-Funktion; die Funktionen für OAuth sowie die „metadata“ – Funktion.

Google Drive

Es gibt bei Google Drive verschiedene Ressourcentypten, zu denen es verschiedene Funktionen gibt (API Reference, 2013). Diese Ressourcen können von den Funktionen als JSON zurückgegeben oder empfangen werden. Unten sind die Ressourcen und die Funktionen, die an diese Ressourcen angewendet werden, aufgelistet.

- ***Files*** – beschreibt eine Datei Ressource.
  - **get** – gibt die Metadaten für eine Datei zurück. Benötigt eine ID der Datei.
  - **insert** – lädt die Datei hoch.
  - **patch**- aktualisiert die Metadaten einer Datei und benötigt dafür eine ID der Datei.
  - **update** – aktualisiert sowohl die Metadaten als auch den Inhalt einer Datei.
  - **copy** - macht die Kopie einer Datei.
  - **delete** – löscht die Datei permanent.
  - **list** – listet die Dateien und Ordner auf.
  - **touch** – setzt die Zeit des Zugriffes auf die aktuelle Urzeit des Servers.

  - **trash** – verschiebt die Datei in den Papierkorb.
  - **untrash** – stellt die Datei aus dem Papierkorb wieder her.
  - **watch** – startet das Überwachen für die Änderungen einer Datei.
- ***About*** – beschreibt eine Kontoinformationsressource.
  - **get** - gibt die Kontoinformationen des Benutzers zurück.
- ***Changes*** – beschreibt eine Änderungsressource.
  - **get** – liefert eine Änderungsressource zurück.
  - **list** – listet alle Änderungsressourcen auf.
  - **watch** – überwacht alle Änderungen. Es liefert eine Ressource, die die Lebenszeit der Überwachung beinhaltet, zurück.
- ***Children*** – beschreibt eine Kind-Ressource, die zu einem bestimmten Ordner gehört.
  - **delete** – löscht ein Kind in einem Ordner. Es benötigt die ID des Ordners als Parameter.
  - **get** – gibt eine Kind-Ressource zurück.
  - **insert** – speichert eine Datei in dem Ordner.
  - **list** – listet alle Kinder-Ressourcen, anhand der gegebenen Ordner-ID, auf.
- ***Parents*** – beschreibt eine Ordner- bzw. Vater-Ressource.
  - **delete** – entfernt eine Ordner-Reference von der Metadaten der Kind-Ressource.
  - **get** – liefert eine Vater-Ressource zurück.
  - **insert** – fügt eine Ordner-Reference zu der Metadaten einer Datei hinzu.
  - **list** – listet alle Vater-Ressourcen, anhand der gegebenen Kind-ID, auf.
- ***Permissions*** – beschreibt die Zugriffsrechte für eine Datei.
  - **delete** – entfernt eine Berechtigung für eine Datei.
  - **get** – liefert die Berechtigung, anhand der gegebenen ID, zurück.
  - **insert** – fügt die Berechtigung für eine Datei hinzu.
  - **list** – listet alle Zugriffsrechte für eine Datei auf.
  - **patch** – aktualisiert eine Berechtigung.
  - **update** - aktualisiert eine Berechtigung.
  - **getIdForEmail** – gibt eine Berechtigungs-ID für eine Emailadresse zurück.
- ***Revisions –*** beschreibt die Revisionen einer Datei.
- ***Apps*** – beschreibt die App's, die ein Benutzer verwendet, und die dazugehörige MIME-Typen, um die Dateien automaisch mit diesen Typen zu verbinden.
- ***Comments*** – beschreibt die Kommentare zu einer Datei.
- ***Replies*** - beschreibt die Antworten zu einem Kommentar.
- ***Properties*** – ist ein Schlüssel/Wert-Paar und kann bei der Datei zusätzlich gespeichert werden.
- ***Channels*** – repräsentiert ein Benachrichtigungskanal einer Überwachungsressource.
- ***Realtime*** – repräsentiert ein Realtime-API-Model.

Für die Entwicklung sind folgende Ressourcen interessant: Files, Children, Parents und Permissions. Die anderen Ressourcen sind für die derzeitige Entwicklung vorerst irrelevant. Die anderen Ressourcen wurden hier dennoch aufgelistet, um zu zeigen, was Google Drive anbietet und welche grundsätzlichen Unterschiede es zwischen der Google Drive und Dropbox gibt.

Nach dem Vergleich der REST Schnittstellen ist ersichtlich, dass die Schnittstellen sich sehr unterscheiden und das die Entwicklung einen gemeinsamen Providers unmöglich ist bzw. keine Vorteile liefern würde.

5.3 Realisierung von XClouds

Im Folgenden wird beschrieben, aus welchen Komponenten ein JClouds Provider im normalen Fall besteht und wie ein solcher Provider realisiert wird. Dieses Kapitel enthält auch die Unterschiede zwischen den Cloud Providern. Diese Unterschiede sind bei der Entwicklung wichtig zu beachten und sind ein entscheidender Aspekt bei der Implementierung.

5.3.1 Realisierung eines Providers

Client

Für die Arbeit mit der ISP Schnittstelle werden 2 Interfaces benötigt. Das Interface Client legt die Methoden fest, die bei dem ISP ausgeführt werden können. Das Interface AsyncClient ist das Gegenstück für das Interface Client. Es hat die gleichen Methoden, beschreibt jedoch zusätzlich diese über die Annotationen. Diese Annotationen legen fest, wie diese Methoden, die Eingangsparameter und Ausgabewerte verarbeitet werden. Anhand der Annotationen in dem AsyncClient Interface wird ein Proxy erstellt, der die REST Schnittstelle des ISP's unterstützt. Wenn eine Methode aus dem Client Interface aufgerufen wird, wird die entsprechende Methode in dem AsyncClient Interface gesucht und der Proxy führt die korrespondierende REST Anfrage aus. Anschließend wird auch über die Annotationen bestimmt, wie die Antwort von dem ISP bearbeitet wird. In der neuen Version von JClouds wird die Trennung auf Client und AsyncClient nicht mehr unterstützt. In dieser Arbeit wird es trodtzdem immer noch verwendet, da es noch unklar ist, wie das Framework in der neuen Version die Interfaces verwendet.

Wie bereits erwähnt, wird in dem AsyncClient Interface über die Annotationen das Binding festgelegt. Nachfolgend ist ein Auszug der meist verwendeten Annotationen und eine kurze Beschreibung vorgestellt:

- @RequestFilters(Filter.**class**) – Durch RequestFilter Annotation werden die Filter eingestellt, die vor dem Absenden an eine HTTP Anfrage angewendet werden.

Diese Annotation lässt sich sowohl für das gesamte Interface als auch für einzelnen Methoden einstellen.

- @Path("/pfad/{parameter}") – Die Annotation Path gibt an, welcher Pfad für die Anfrage verwendet wird. Die Path Annotation vor dem Interface bestimmt den Default-Wert, falls es bei einer Methode nicht eingestellt ist. Der Pfad wird zu dem Endpoint am Ende hinzugefügt. In geschweiften Klammern werden die Platzhalter für die Parameter eingegeben, die später mit gekennzeichneten Argumenten einer Methode ersetzt werden.
- @Headers(keys = {"key1","key2"}, values = {"value1","value2"}) – Headers bestimmen die HTTP Headers, die als Default-Werte einer HTTP Anfrage übergeben werden. Sie können sowohl für das gesamte Interface als auch für eine bestimmte Methode verwendet werden. In den Listen keys und values werden die Schlüssel/Wert-Paare der Reihe nach zugeordnet.
- @QueryParams(keys = "param1", values = "value1") – Genauso wie die Headers Annotation, wird auch die QueryParams Annotation mit den Default-Werten der Query-Parametern, die hinter einer HTTP-Anfrage an die URL angehängt werden, benutzt.
- @FormParams(keys = "param1", values = "value1") – Funktioniert ähnlich wie QueryParams. Der einzelne Unterschied ist, dass die Parameter in dem Body der Anfrage geschickt werden und nicht hinter der URL angehängt werden. Das ist dann wichtig, wenn über die HTTP POST Methode die Form Parameter erwartet werden, wie es in RFC 2612 beschrieben wird (Fielding, et al., 1999).
- @Named("MethodName") – Das ist eine Binding Annotation von Guice Framework und hilft das Binding der Injection anhand von Namen durchzuführen (Binding Annotations, 16).
- @POST/GET/PUT/DELETE/HEAD – Diese Annotationen legen den Typ einer HTTP Anfrage fest.
- @Endpoint(Endpoint.**class**) – Über Endpoint Annotation lässt sich das Endpoint bei jeder Methode einstellen. Diese Annotation ist dann hilfreich, wenn das Endpoint sich bei einigen Methoden unterscheidet. Mit dieser Annotation können z. B. einige Anfragen an ein anderes Subdomain adressiert werden.
- @Fallback(NullOnNotFoundOr404or403.**class**) – Die Fallback Annotation bestimmt, welcher Wert bei der bestimmten HTTP Ausnahmen zurück geliefert wird. Mit Fallback Annotation ist es möglich bei der 500 Not Fond Response eine Null zu erhalten, anstatt einer HTTP Exception. Ebenso lassen sich mit dieser Annotaion eigene Klassen mit eigenen Rückgabewerten definieren.
- @ResponseParser(ParseResponseFromHttpContentOrHeader.**class**) – Ein Response-Parser wird benötigt, um den Inhalt einer HTTP Response zu bearbeiten und umzuwandeln.
- @BinderParam(BindMetadataToRequest.**class**) – Bei den Argumenten einer Methode können die BinderParam-Klassen bestimmt werden, die z. B. nur bestimmte Werte eines Objektes an die Anfrage binden.

- @QueryParam("paramName") – Die QueryParam Annotation vor einem Argument der Methode bestimmt, dass dieses Argument automatisch an die Anfrage angehängt wird.

Konfiguration

Wie bereits oben beschrieben, benutzt JClouds das Dependency Injection Entwurfsmuster. Um die neuen Provider in das JClouds Framework einzubinden, wird eine Datei benötigt, die es dem JClouds sagt, wo die ProviderMetadata-Klasse zu finden ist. Je nachdem, ob es sich um den Provider oder eine API handelt, wird in dem Ressourcen Ordner unter *MATA-INF/services* die Datei *„org.jclouds.providers.ProviderMetadata“* oder -*„ApiMetadata“* angelegt. In dieser Datei sollen die vollständigen Namen der Provider- oder ApiMetadata-Klassen stehen. Über diese Klassen wird ein Provider bzw. eine API konfiguriert. Wenn es sich um Provider handelt, dann wird in der ProviderMetadata-Klasse eine Instanz der ApiMetadata-Klasse übergeben. Wie bereits im Kapitel 5.3.1 beschrieben wurde, benötigt die *„defaultModules“* Methode in der ApiMetadata-Klasse die Konfigurationsklassen. Diese sollen von AbstractModule-Klasse abgeleitet werden. In den Konfigurationsklassen, wie z. B. DropboxBlobStoreConfig, stehen die Bindings, die das Framework konfigurieren. In Listing 5.2 wird ein Beispiel für die Konfiguration eines Modules dargestellt.

```
protected void configure() {
        install(new BlobStoreMapModule());
        bind(ConsistencyModel.class).toInstance(ConsistencyModel.EVENTUAL);
        bind(BlobStore.class).to(DropboxBlobStore.class).in(Scopes.SINGLETON);
}
```

Listing 5.2 Module Konfiguration

Blobstore

Die Instanz einer Blobstore Klasse ist eine Fassade für die Clients. Eine Blobstore-Instanz stellt die wichtigsten Methoden zur Kommunikation mit einer Cloud bereit. Sie stellt über einen Client die Verbindung her und bearbeitet die Rückgabewerte. Außerdem liefert die Klasse eine BlobBuilder-Instanz, die zum Erstellen einer Blob-Instanz verwendet wird. Wie es bereits in Listing 5.2 gezeigt ist, soll eine Konkrete Blobstore-Klasse an der Stelle einer Blobstore-Schnittstelle gebunden werden.

Filters

Die Filter-Klassen werden bei den Client-Schnittstellen in den Annotationen eingetragen. Sie werden vor einer HTTP-Anfrage ausgeführt. Das ist z. B. dann notwendig, wenn die

Authentifizierungs-Header an eine Anfrage gebunden werden müssen. Die Filter-Klassen implementieren die Schnittstelle HttpRequestFilter und bieten nur eine einzige Methode „*filter*“ an, die eine Instanz von HttpRequest-Klasse erhalten und diese auch nach der Bearbeitung zurück liefern.

Functions

Der Client führt eine Anfrage aus und liefert die Werte von dem Cloud-Server zurück. Diese Werte müssen ggf. bearbeitet werden. Die Funktions-Klassen können in die Annotationen ResponseParser eingegeben werden und werden ausgeführt, wenn eine Antwort ohne Fehler Nachricht, d.h. mit dem Status Code 200, ankommt. Eine Funktions-Klasse implementiert die Schnittstelle Function<HttpResponse, T> und bietet eine Methode „*apply*“ an. Diese Methode empfängt einen HttpResponse Objekt als Parameter und liefert eine Instanz der an der Stelle T der eingegebenen Klasse zurück.

Reference

In Package Reference werden meistens die statischen Klassen abgelegt, die eine Sammlung von Konstanten bereitstellen. Diese werden meistens für die Einstellungen, wie z. B. Endpoint oder Headers, gebraucht.

Domain

Eine Anfrage an den Cloud-Server liefert meistens eine Antwort mit den Metadaten, die bestimmte Objekte im Cloud beschreiben. Das kann z. B. die Beschreibung einer Datei oder eines Containers sein. Diese Metadaten können in verschiede Objekte umgewandelt werden. Dafür sind meistens die Funktions-Klassen verantwortlich. Da sie eine Domäne des Cloud-Anbieters repräsentieren, werden die Klassen für die Metadaten-Objekte meistens in einem „*domain*“ Package abgelegt.

Eine der wichtigsten Klassen ist die Payload-Klasse. Sie repräsentiert die Datei oder der Inhalt, der an das Cloud gesendet wird. Die PayloadEnclosing-Klasse verwaltet die Payload-Instanz. Die Blob-Klasse wird von der PayloadEnclosing-Klasse abgeleitet und wird von einigen Methoden der Blobstore-Klasse als ein Parameter erwartet. Das wurde bereits im Listing 5.1 gezeigt. Das bedeutet, dass eine Instanz der Blob-Klasse den Inhalt, der zu versenden oder zu empfangen ist, aufnimmt. Das JClouds Framework erkennt, wenn die Blob- oder Payload-Klasse an die Methoden einer AsyncClient Schnittstelle übergeben werden, und bindet automatisch den Inhalt an die Anfrage an. Es ist möglich eine eigene Ableitung der Blob-Klasse zu erstellen, die z. B. noch zusätzliche Informationen aus einem HTTP-Header bereitstellen kann. Solche Klassen werden meistens in dem „*domain*“ Package abgelegt.

Options

Einige Methoden einer Blobstore-Instanz erwarten als Parameter ein Option-Objekt. Die Options-Klassen erweitern die Klasse BaseHttpRequestOptions und können die Query-, Form- und Headers-Parameter setzen. Diese Options-Objekte werden an den Client übergeben. JClouds erkennt automatisch die Eingabe der Optionen und wendet diese an die HTTP-Anfrage an.

Handlers

Wenn man mit dem Cloud arbeitet, können Fehler auf der Serverseite auftreten. Diese werden dem Client gemeldet. Um die Fehler besser zu verstehen, besitzen die HTTP Fehler einen Status Code, wie es in (Fielding, et al., 1999) beschrieben ist. Es können sowohl allgemeine HTTP Fehler, wie Server Error mit dem Status Code 5xx, als auch Client Error mit dem Status Code 4xx sein. Die letzten werden von dem Cloud Plattform generiert. Das JClouds Framework erkennt die HTTP Ausnahmen und kann eine Java Ausnahme, wie z. B. RessourceNotFountException, auswerfen. Falls jedoch eine andere Art von Ausnahmen ausgegeben werden soll, wird ein Handler benötigt, das diese Ausnahme erst erstellt. Um dies zu erreichen, wird in der Modul-Konfiguration ein neues Handler an der Stelle von HttpErrorHandler gebunden, wie in Listing 5.3 gezeigt wird. Die Klasse, die gebunden wird,

```
bind(HttpErrorHandler.class).annotatedWith(Redirection.class)
        .to(ParseErrorFromXmlContent.class);
```

Listing 5. 3 Handler Binding

soll die Schnittstelle HttpErrorHandler implementieren und somit die „*handleError*“ Methode bereitstellen. Diese Methode erhält HttpResponse und Command als Parameter und kann dieses analysieren, um die Fehlermeldungen des Cloud-Servers zu bearbeiten und entsprechende Java-Ausnahmen auszuwerfen.

Fallbacks

Ein Fallback in den Annotationen des Client Interfaces kann dazu verwendet werden, um die Fehler abzufangen und ggf., anstatt von einem Fehler, einen Rückgabewert zu liefern. Die Klasse die bei der Annotation eingegeben wird, soll eine Schnittstelle Fallback<?> implementieren. Durch diese Schnittstelle wird eine Methode „*createOrPropagate*“ bereitgestellt, die einen Objekt vom Typ Throwable als Parameter erhält und ein Objekt, das als Generic-Typ eingegeben wurde, zurückliefert.

Utils

Ein Blobstore Objekt wird mit einer Instanz BlobUtils konfiguriert. Diese lässt sich mit einer anderen abgeleiteten Klasse über die Dependency Injections Konfiguration ersetzen. Die

BlobUtils Klasse wird dazu verwendet, um das Erstellen neue Blobstore-Klassen zu vereinfachen. Diese Klasse beinhaltet folgende Methoden:

- **directoryExists** – prüft, ob ein Ordner in einem Container existiert.
- **createDirectory** – erstellt einen Ordner in einem Container.
- **countBlobs** – zählt die Dateien in einem Container oder einem Ordner.
- **clearContainer** – löscht den Inhalt eines Containers.
- **deleteDirectory** – löscht einen Ordner in einem Container.
- **blobBuilder** – gibt eine Instanz der BlobBuilder-Klasse.

Ein Blobstore delegiert Aufrufe der gleichen Methoden an die Methoden der BlobUtils-Instanz.

Strategie

Damit die Methoden der BlobUtils-Klasse der neuen Provider benutzt werden können, benutzt BlobUtils-Klasse das Strategy Pattern. Die Idee hinter dem Strategie Pattern ist es ein Algorithmus als eine Klasse zu kapseln (Eilebrecht & Starke, Strategy, 2013). Mit der Hilfe dieses Pattern ist es möglich ein Algorithmus einfach zu ersetzen. BlobUtils Klasse erwartet folgende Algorithmen, die ersetzt werden können:

- **GetDirectoryStrategy** - Ein Algorithmus, um zu prüfen, ob ein Ordner existiert.
- **MkdirStrategy** - Ein Algorithmus, um einen Ordner zu erstellen.
- **CountListStrategy** - Ein Algorithmus, um die Dateien in einem Ordner bzw. einem Container zu zählen.
- **ClearListStrategy** - Ein Algorithmus, um alles in einem Container zu löschen.
- **DeleteDirectoryStrategy** - Ein Algorithmus, um einen Ordner zu löschen.

Wenn nur die Algorithmen zum Arbeiten mit den Ordnern ausgetauscht werden müssen, reicht es aus, über die Dependency Injections die neuen Strategien zu binden.

I/O

Wenn es darum geht, eine größere Datei im Cloud zu speichern, ist es manchmal notwendig die Datei in verschiedene Teile zu zerlegen und in Stücken zu transportieren. Dafür gibt es bei JClouds die Klassen, die in *org.jclouds.io* Package liegen. Im Fall von Google Drive war es z. B. notwendig, diese Klassen zu überschreiben und eigene Klassen wie „*Part*“ und „*MultipartForm*“ zu implementieren.

5.3.2 Unterschiede bei den Providern

Nicht alle ISP's arbeiten mit den Ordnern auf gleicher Art und Weise. Aus der Benutzersicht geht Dropbox mit den Pfaden so um, dass der Pfad in dem Dateinamen als übergeordneter Ordner verstanden wird. Wenn der Ordner nicht existiert, wird dieser erstellt.

Bei der Analyse der Google Drive Schnittstelle wird sehr schnell klar, dass Google Drive keine Pfade versteht. Google Drive bietet die REST Funktion an, die die Dateien und Ordner auflistet. Wenn es jedoch alle Dateien und Ordner auf einmal ausgegeben werden, kann es viel Zeit in Anspruch nehmen, was für einen Benutzer unzumutbar ist. Diese Funktion akzeptiert jedoch Filter und Suchparameter. Das bedeutet, dass nicht alle Dateien bzw. Ordner und auch nicht alle Metadaten Felder in der Liste dabei sein müssen. Die Suchfunktion liefert also eine Liste mit Metadaten zu einer Datei. Jede Datei hat in den Metadaten einen Verweis auf den Vatersordner.

Bei dem Provider für Google Drive werden in der Blobstore-Klasse erst alle Ordner gesammelt, die in dem Pfad eines Dateinamen enthalten sind. Anschließend wird rekursiv überprüft, ob dieser Pfad tatsächlich existiert. Die Ordner, die nicht existieren, werden erstmal erstellt. Anschließend wird bei der Datei ein Verweis auf den letzten Ordner in dem Pfad gespeichert.

Amazon S3 und Windows Azure verstehen die Pfade genauso wie Dropbox. Allerdings benötigen sie immer einen Containername, in dem die Datei bzw. Ordner angelegt werden. Viele Methoden der Blobstore Schnittstelle benötigen diese Eingabe. Bei Dropbox und Google Drive muss an diesen Stellen „null" übergeben werden. Das ist nicht immer sicher, da der Anwender des Frameworks nicht immer mit dem Quellcode vertraut ist. Es gibt die Gefahr einer NullPointerException.

Außerdem sehen die Domänen der Provider unterschiedlich aus. Das heißt, dass die Metadaten sich nicht nur in Benennung, sondern auch in Funktionalität unterscheiden können. Diese Unterschiede bei den Pfaden, Metadaten und der OAuth-Authentifizierung, die bei den SaaS Providern benötigt wird, sind entscheidende Gründe für eine weitere Abstraktionsebene.

5.3.3 Realisierung von XClouds

JClouds bietet viele Klassen, die die Implementierung von Providern und Arbeit mit REST Schnittstellen vereinfachen. Um mit den Clouds aus dem SaaS Bereich zu arbeiten, ist es jedoch nicht ausreichend, diese einfach als JClouds Provider zu implementieren. Denn die SaaS Clouds unterstützen keine Container und die Art und Weise, wie Dropbox und Google Drive mit den virtuellen Ordnern umgehen, kann unterschiedlich sein. Aus diesem Grund wurde eine Abstraktion entwickelt.

XClouds Schnittstelle

Es ist sinnvoll die Provider je nach eingegebener ID zu erstellen. Damit die Provider auf gleiche Art und Weise benutzt werden können, wird eine Schnittstelle benötigt. Diese bietet folgende Methoden an:

- **uploadFile** – erhält als Parameter die Datei und den Pfad, wo diese Datei in dem Cloud gespeichert wird. Je nachdem, ob ein Provider die Container unterstützt oder nicht, wird der erste Ordnername in dem Pfad als Container Name verwendet.
- **downloadFile** – erhält als Parameter den Pfad zur Datei in dem Cloud und die Datei, in die der Inhalt heruntergeladen wird. Auf die gleiche Weise, wie es bei uploadFile funktioniert, wird der erste Ordnername in dem Pfad, abhängig von dem Provider, als Container verstanden.
- **remove** – löscht den Pfad in dem Cloud. Dabei wird es nicht unterschieden, ob es sich um die Datei, einen Container oder einen Ordner handelt.
- **list** – Listet alle Dateien unter dem angegebenen Pfad.
- **fileInfo** – gibt die Metainformationen einer Datei zurück.
- **createDirectory** – erstellt einen Ordner. Für die Container gilt, wie auch bei „uploadFile“ Methode, dass der erste Ordnername in dem Pfad als Containername gelten kann.
- **share** – macht eine Datei oder einen Ordner öffentlich zugänglich und gibt die URL zurück.
- **fileExists** – prüft, ob eine Datei existiert.
- **directoryExists** – prüft, ob ein Ordner existiert.
- **redundanceAllowed** – stellt fest ob eine Replikationskomponente unterstützt wird.
- **regionSupport** – stellt fest, ob eine Region-Komponente unterstützt wird.
- **regionComponente** – gibt eine Region-Komponente oder ein Null-Objekt.
- **redundanceComponent** – gibt die Replikationskomponente zurück oder ein Null-Objekt.
- **count** – zählt die Dateien und Ordner unter dem angegeben Pfad.

Fassade

Die Klasse XCloudsImpl dient als Fassade und setzt die Schnittstelle XClouds um. Beim Erstellen einer Instanz wird eine ID erwartet, die den JClouds Provider identifiziert. Zurzeit werden nur Dropbox, Amazon S3, Windows Azure Blobstore und Google Drive unterstützt. Als zweiten Parameter wird die Authentifikationsklasse erwartet, die die Zugangsdaten liefert. Die Fassade benutzt einen Adapter Pattern, um mit verschiedenen Clouds zu arbeiten. Sie erstellt lediglich eine Instanz eines Adapters und leitet alle Aufrufe an den Adapter weiter.

Authentifizierung

Bei der Autorisierung einer Anfrage wird ein HTTP-Header „*Authorization*“ gesetzt. JClouds benutzt dafür Credential und Identity Parameter bei der Instanziierung eines BlobstoreContext’s. Dropbox und Google Drive benötigen dennoch zunächst die OAuth Authentifizierung. Die Vorgehenswese dabei wurde bereits im Kapitel 5.2.1 erklärt. Damit die Anmeldung einheitlich funktioniert, wird eine Schnittstelle „*XCloudsAuth*“ benutzt, die zwei Methoden enthält. Diese Methoden geben Credential und Identity für Provider zurück. Um die OAuth Authentifizierung durchzuführen wurde die OAuth2 Bibliothek von Google benutzt. Es gibt zwei Klassen für die Authentifizierung. Die „Auth“ Klasse dient als ein Platzhalter für Credential und Identity, und wird für die Provider Amazon und Windows Azure benutzt. Die zweite Klasse ist „OAuth2“, die Google Bibliotheken benutzt, um OAuth Authentifizierung durchzuführen und anschließend ein OAuth Token als Credential zurückgibt. Als Identity wird dabei immer „Bearer“ ausgegeben. Das ist ein Standard-Benutzername bei der OAuth Authentifikation.

Adapter

Für jeden Provider wird ein Adapter erstellt, damit die Abstraktion funktioniert, die Arbeit mit den Pfaden einfacher wird, und die Unterscheidung zwischen einem Container und einem Ordner aufgelöst wird. Ein Adapter leitet die Schnittstelle XClouds ab. Einige diese Adapter müssen prüfen, ob der Pfad einen Ordner oder einen Container besitzt und dem entsprechend die Anfrage erstellt.

XCloudsObject

Da die Metadaten bei den Providern sich unterscheiden können, wird eine Schnittstelle XCloudsObject geschafft. Für jeden Provider gibt es eine eigene Implementierung dieser Schnittstellen. Im Grunde kapselt ein XCloudsObject die Metadaten. Es werden folgende Methoden bereitgestellt:

- **getPath** – gibt eine URL oder Public-URL für ein Objekt aus.
- **isDir** – gibt zurück, ob es sich um einen Ordner handelt. Das ist notwendig, da es keine Methode in Metadaten der JClouds Klassen dafür vorgesehen war.
- **getName** – einige Provider geben als Name einer Datei den relativen Pfad zurück. Um die Einheitlichkeit zu schaffen, muss zuerst der Name aus dem Pfad extrahiert werden. Die Methode **getName** muss sicherstellen, dass es nur ein Name und kein Pfad zurückgegeben wird.
- **getRawObject** - gibt die Metadaten des Providers zurück.

5.4 Test

Um die Funktionsweise eines Programms nachzuweisen, ist es wichtig bei der Entwicklung die Tests durchzuführen. Dabei gibt es verschiedene Verfahren, wie ein Programm getestet werden kann. Es ist wichtig bereits in den früheren Fasen der Entwicklung mit dem Testen anzufangen.

> *„Durch frühzeitiges Prüfen werden Fehler frühzeitig erkannt.“ (Spillner & Linz, Allgemeine Prinzipien des Softwaretestens, 2012)*

5.4.1 Statische Analyse

Die statische Analyse hilft dabei, die Fehler in den früheren Fasen zu erkennen. Wenn es nach dem V-Modell entwickelt wird, kann der Test mit den Reviews beginnen. Während der Reviews können die Entwurfs- und Anforderungsdokumente auf Fehler überprüft werden. Außerdem bei der Implementierung kann statische Codeanalyse helfen, die potenziellen Fehler zu entdecken und zu vermeiden (Spillner & Linz, Statische Analyse, 2012). Es gibt verschiedene Ansätze der Durchführung einer statischen Codeanalyse. Das erst Werkzeug ist der Compiler einer Programmiersprache. Er entdeckt bereits die meisten Syntaxfehler.

Durch die Datenflussanalyse wird festgestellt, welche Anomalien in dem Quellcode vorhanden sind. Es können z. B. unbenutzte Variablen oder unerreichbarer Code sein. Die Kontrollflussanalyse dient dazu die Komplexität des Quellcodes durch bestimmte Metrikern zu bestimmen, um einen Hinweis darauf zu geben, wo der Code noch vereinfacht werden soll. Z. B. an der Stellen, wo zu viele IF-Abfragen in dem Code stehen, kann es darauf deuten, dass es in diesen Codeabschnitten mehr Fehler produziert werden.

Bei der Entwicklung von XClouds kamen alle diese Verfahren zum Einsatz und es wurden die Werkzeuge, wie PMD[30] und FindBugs[31] als Plug-Ins für Eclipse IDE, eingesetzt.

5.4.2 Dynamische Tests

Es existieren verschiedene dynamische Tests. In dieser Arbeit wird jedoch auf die Black- und White-Box Testverfahren eingegangen.

Black-Box

Damit die Komponenten eines Systems gegen den Entwurf getestet werden können, wird ein Black-Box Verfahren eingesetzt. Bei einem Black-Box-Test wird davon ausgegangen,

[30] http://pmd.sourceforge.net/
[31] http://findbugs.sourceforge.net/

dass der Tester den Quellcode nicht kennt. Für einen Entwickler ist es immer schwierig ein Black-Box Verfahren anzuwenden, da er sich mit dem Quellcode normalerweise auskennt. Aus diesem Grund gibt es keine Unabhängigkeit zwischen dem Entwickler und dem Code. Demzufolge wird es empfohlen, dass ein unabhängiger Tester die Black-Box Tests entwickelt. Destotrotz können viele Anforderungen bereits bei der Entwickler mit den Unit-Tests, wenn sie wohldefiniert sind, getestet werden. Bei der Entwicklung von XClouds kam das JUnit Framework zum Einsatz und es wurde gegen die Anforderungen, die im Kapitel 5.2.1 geschrieben wurden, getestet. Nach jeder abgeschlossener Änderung im Framework wurde der Regressionstest durchgeführt. Das bedeutet, dass alle JUnit-Tests komplett ausgeführt wurden, um festzustellen, dass das Programm durch Änderungen nicht beschädigt wurde.

Bei der Entwicklung der Provider wurden je Provider folgende Tests geschrieben:

- Die Tests für das Arbeiten mit den Ordnern und Pfaden.
- Die Tests für das Arbeiten mit der Dateien.
- Einige Tests für die Utility Funktionen.

Durch diese Tests wurde festgestellt, dass die grundlegenden Funktionen der Provider gegeben sind.

Bei der Entwicklung der XClouds Abstraktion wurde eine abstrakte JUnit-Test-Datei geschrieben, die die Funktionalität irgendeinen Provider über XClouds feststellt. Sie enthält die Tests für Ordner, Pfade und Dateien. Zusätzlich wurden die konkreten JUnit-Test-Dateien, jeweils für einen Provider, geschrieben. Sie leiten die abstrakte JUnit-Test-Datei ab und enthalten nur die Einstellungen, die für einen Provider benötigt werden. Dadurch wird festgestellt, dass es nur eine Konfigurationsänderung benötigt wird, um einen Provider zu wechseln.

White-Box Verfahren

Das White-Box Verfahren stellt fest, zum welchen Grad die Überdeckung im Code gegeben ist. Das heißt, ob es Stellen im Code gibt, die nie ausgeführt werden. Die Werkzeuge wie z. B. das Plug-In Eclemma[32] für Eclipse bauen die Zählvariablen vor der Ausführung der Unit-Tests ein, um fest zu stellen, welche Stellen im Code erreicht wurden. Dieses Verfahren kam bei der Entwicklung nicht zum Einsatz, weil es sehr gute Unit-Tests erfordert.

[32] http://www.eclemma.org/

5.4.3 Ein Vorführungsprogramm

Für die Testzwecke wurde außerdem im Rahmen dieser Arbeit ein Vorführprogramm entwickelt. Das zeigt in der ersten Linie, was mit dem XClouds Framework möglich ist. In Abbildung 5.6 ist die Oberfläche des Programms dargestellt.

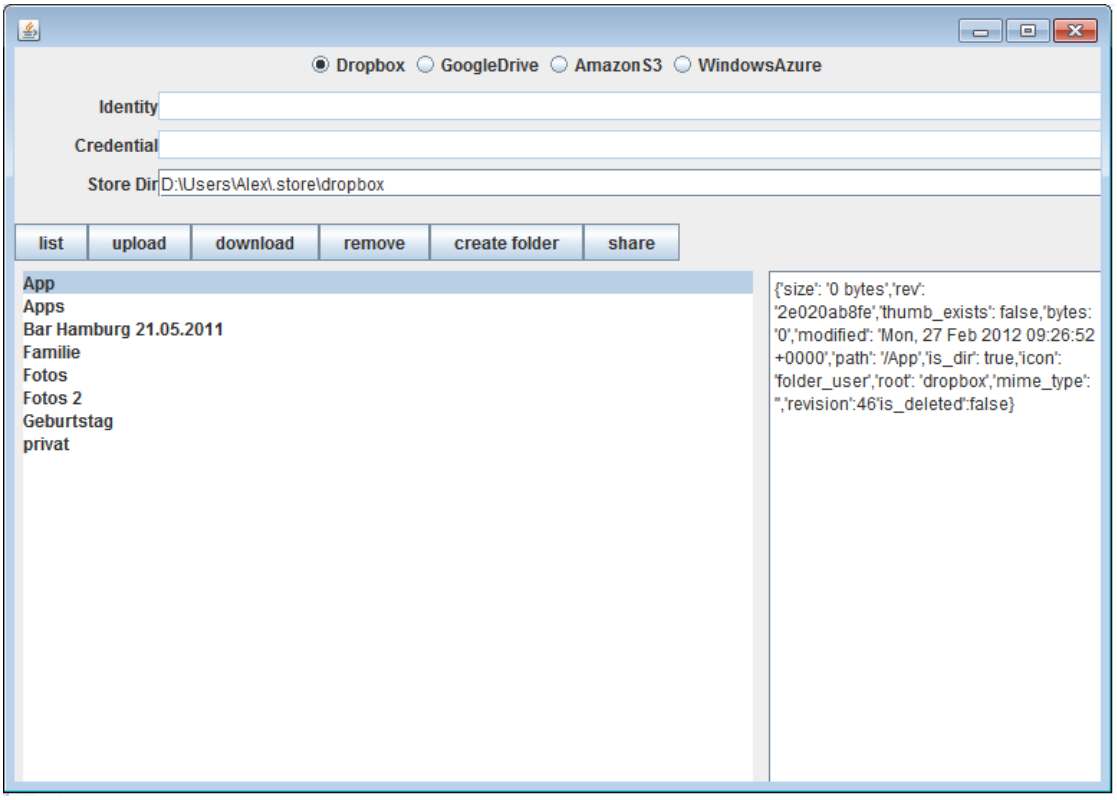

Abbildung 5.6 Vorführungsprogramm

Oben ist die Auswahl der unterstützten Clouds dargestellt. Es ist möglich gleichzeitig eins von vier möglichen Clouds auszuwählen. In den Feldern Identity und Credential werden die Zugangsdaten des Benutzers eingetragen falls es sich um Amazon S3 oder Windows Azure handelt. Bei Dropbox und Google Drive ist es möglich im Feld „Store Dir“ einen Ordner zu hinterlegen, in dem die Datei mit den OAuth Tokens gespeichert werden. Diese Datei kann später verwendet werden, um den Token, falls dieser noch nicht abgelaufen ist, wieder zu holen. Unten sind die Knöpfe angebracht, die mögliche Operationen darstellen. Damit die Liste des ersten Verzeichnisses bzw. Containers angezeigt wird, soll die „list“ Aktion ausgeführt werden. Danach ist es möglich mit der Maus durch Verzeichnisse zu navigieren und andere Operationen auf den Dateien und Ordner auszuführen.

6 Fazit

In dieser Arbeit wurde ein XCldous Framework für Java entwickelt. Es ist ein Multi-Cloud Framework, das sich auf JClouds Framework basiert und die Arbeit mit verschiedenen Cloud Anbietern, sowohl aus dem IaaS und PaaS als auch aus dem SaaS Bereich, ermöglich. Dafür wurden unterschiedliche Cloud Anbieter und deren Schnittstellen analysiert. Es wurden die bereits existierende Multi-Cloud Lösungen betrachtet und anschließend eine der Lösungen als Basis ausgewählt. Bei der Realisierung wurde zuerst die Architektur von JClouds analysiert. Anschließend wurden die Architektur und die Abstraktion für XClouds entworfen. Basierend darauf wurde dann die neuen Provider für JClouds und eine neue Abstraktionseben entwickelt. Bei der Realisierung wurde das Framework mit verschiedenen Testverfahren getestet, um die Qualität der Software sicherzustellen. Zum Vorführen des XClouds Frameworks wurde ein Programm erstellt, das zeigt, wie einfach das Wechseln der verschieden Cloud Anbieter mit Hilfe von XClouds ist.

6.1 Ausblick

Zurzeit sind es nur zwei Anbieter in dem Bereich SaaS, nämlich Dropbox und Google Drive. Es ist aber möglich viele anderen SaaS Cloud Anbieter einzubinden, welche die Speicher-Dienste anbieten.

In dieser Arbeit wurden nicht alle Funktionen der Anbieter umgesetzt. Es ist also denkbar andere Funktionen der Anbieter einzufügen. Z. B. ist es als nächstes notwendig, die „move“ bzw. „rename“-Funktion einzufügen, damit es möglich ist die Dateien und Ordner zu verschieben bzw. umzubenennen ohne diese herunterladen zu müssen. Bei Google Drive ist nicht die gesamte Funktionalität der Upload Funktion gegeben. Derzeit ist das Herunterladen der Dateien ohne die Kontrolle über den Abbruch realisiert. Es ist aber möglich nach einem Abbruch das Upload vorzusetzen. Diese Funktion ist in XClouds auch denkbar.

Die Fehlerbehandlung ist auf die grundlegenden Serverfehler begrenzt. Die Cloud Anbieter können aber eigene Fehlermeldungen produzieren. Diese Fehlermeldungen sind nur in wenigen Fällen in XClouds integriert. Es ist also notwendig in dem Framework die Fehler, die in REST Schnittstellen definiert wurden, abzufangen und eine saubere Fehlerbehandlung zu implementieren. Das erfordert mehr JUnit-Tests. Die Fehler sollen für die Tests produziert werden. In manchen Fällen ist es notwendig diese Fehler zu simulieren. Das erfordert einige Simulationsdateien bzw. eine Simulationsumgebung.

In Fragen Sicherheit lässt sich das Framework auch erweitern. Oft wird eine Client-Seitige Verschlüsselung der Daten vor der Übertragung verlangt. Bis jetzt wurde es von allen Frameworks vernachlässigt. Diese Funktionalität würde sehr gut das Framework ergänzen.

Anhang A

JClouds Installation-Voraussetzungen

Um JClouds als Maven Projekt zu installieren, wird die Java Version 7 benötigt. Bei der Einstellung der Maven Dependencies wurde in dieser Arbeit ein JClouds Release 1.6.1 benutzt, da es die letzte stabile Version zu diesem Zeitpunkt war.

Für Eclipse muss ein Maven Plug-In installiert werden. Bei dieser Arbeit wurde ein „Maven integration for Eclipse“ Plug-In aus dem Eclipse Marktplace installiert und dabei die Eclipse Juno Version benutzt. Eine genauere Installationsbeschreibung befindet sich unter (jclouds Developer Setup instructions for Eclipse).

Es gibt unterschiedliche Möglichkeiten JClouds zu installieren. Eine Möglichkeit es zu installieren ist, JClouds als Maven Projekt unter Eclipse aufzusetzen. Bei dieser Installation stehen nicht nur die JAR-Dateien, sondern auch der Quellcode, zu Verfügung.

Maven ist ein Werkzeug, das zum Erstellen und Verwalten von Java-basierten Projekten benutzt werden kann (What is Maven?). Das Ziel von Maven ist es, den Entwicklern zu ermöglichen, den gesamten Projektzustand in kürzester Zeit zu begreifen. Dabei konzentriert sich Maven auf folgende Punkte:

- Eine einfache Erstellung des Build-Prozesses
- Die Bereitstellung eines einheitlichen Build-Systems
- Die Bereitstellung hochwertiger Informationen zum Projekt
- Die Bereitstellung von Richtlinien für Best Practices Entwicklung
- Das Zulassen der transparenten Migration auf neue Features

Durch Maven ist es einfach, den gesamten Lebenszyklus eines Projekts zu kontrollieren.

Anhang B

Inhalte der beigefügten CD

- Quellcode des XClouds
- Vorführprogramm
- Bachelorarbeit im PDF Format

Abbildungsverzeichnis

Glossar

TCO - Total Cost of Ownership. Es ist ein Abrechnungsverfahren, das Verbrauchern und Unternehmen helfen soll, alle anfallenden Kosten von Investitionsgütern (wie beispielsweise Software und Hardware in der IT) abzuschätzen.

CRUD - ist ein Akronym für die grundlegende Datenbank Orationen (Create, Read, Update, Delete)

URI - ist ein eindeutiger Identifier einer Ressource (Uniform Resource Identifier) und wird meisten in WWW verwendet.

API - (engl. application programming interface). Eine Programmierschnittstelle.

XML - eine Sprache zur Darstellung hierarchisch strukturierten Daten (engl. Extensible Markup Language).

JSON - ist ein Akronym von JavaScript Object Notation. Es ist ein textbasiertes Datenformat, das sowohl für Maschinen als auch für Menschen gut lesbar ist. Jedes JSON-Dokument kann mit Hilfe von eval() Funktion wider in ein Objekt umgewandelt werden.

CSP - Cloud Service Provider

JVM - Java Virtual Machine ist ein Teil der Java-Laufzeitumgebung, der für die Ausführung von Java-Bytecode verantwortlich ist.

IAM - Identity and Access Management. Eine Benutzer- und Rechteverwaltung bei AWS.

Literaturverzeichnis

About Deltacloud. (kein Datum). Abgerufen am 05. Mai 2013 von http://deltacloud.apache.org/about.html

Amazon Simple Storage Service. (kein Datum). (Amazon) Abgerufen am 04. Mai 2013 von http://docs.aws.amazon.com/AmazonS3/latest/API/Welcome.html

Apache Libcloud. (kein Datum). Abgerufen am 05. Mai 2013 von http://libcloud.apache.org/about.html

API Reference. (9. September 2013). (Google Inc.) Abgerufen am 20. Oktober 2013 von Google Drive SDK: https://developers.google.com/drive/v2/reference/

Baun, C., Kunze, M., Nimis, J., & Tai, S. (2011). *Cloud Computing : Web-basierte dynamische IT-Services.* Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg.

Binding Annotations. (2011. Oktober 16). Abgerufen am 2013. Oktober 4 von http://code.google.com/p/google-guice/wiki/BindingAnnotations

Birk, D., & Wegener, C. (01. 09 2010). Über den Wolken, Cloud Computing im Überblick. *Datenschutz und Datensicherheit - DuD*, S. 643,644.

Bundesamt für Sicherheit in der Informationstechnik. (kein Datum). Abgerufen am 08. Mai 2013 von https://www.bsi.bund.de/DE/Themen/CloudComputing/Grundlagen/Grundlagen_node.html

Dropbox Core API Dokumentation . (kein Datum). Abgerufen am 04. Mai 2013 von https://www.dropbox.com/developers/core/docs

Dropbox gibt es nun auch für Ihr Geschäft. (kein Datum). Abgerufen am 17. Mai 2013 von https://www.dropbox.com/business

Eilebrecht, K., & Starke, G. (2013). Fassade. In *Patterns kompakt. Entwurfsmuster für effektive Software-Entwicklung* (S. 87-89). Heidelberg, Berlin, Deutschland: Springer-Verlag.

Eilebrecht, K., & Starke, G. (2013). Null-Objekt. In *Patterns kompakt. Entwurfsmuster für effektive Software-Entwicklung* (S. 186-188). Heidelberg, Berlin, Deutschland: Springer-Verlag.

Eilebrecht, K., & Starke, G. (2013). Strategy. In *Patterns kompakt. Entwurfsmuster für effektive Software-Entwicklung* (S. 66-68). Heidelberg, Berlin, Deutschland: Springer-Verlag.

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures.* IRVINE: UNIVERSITY OF CALIFORNIA.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, L., & Berners-Lee, T. (Juni 1999). *RFC 2616 Hypertext Transfer Protocol -- HTTP/1.1.* Abgerufen am 8. Oktober 2013 von http://tools.ietf.org/html/rfc2616

Garfinkel, S. (3. Oktober 2011). *MIT Technology Review*. Abgerufen am 22. April 2013 von http://www.technologyreview.com/news/425623/the-cloud-imperative/

Google Cloud Platform. (2013). (Google) Abgerufen am 04. Mai 2013 von https://cloud.google.com/

Google Cloud Storage Interoperability. (27. April 2013). (Google) Abgerufen am 04. Mai 2013 von https://developers.google.com/storage/docs/migrate-to-gcs

Google Inc. (kein Datum). *Using OAuth 2.0 to Access Google APIs.* Abgerufen am 18. September 2013 von https://developers.google.com/accounts/docs/OAuth2

Hansen, M. (01. Juni 2012). Vertraulichkeit und Integrität von Daten und IT-Systemen im Cloud-Zeitalter. *Datenschutz und Datensicherheit - DuD*, S. 407-412.

Hardt, D. (October 2012). *The OAuth 2.0 Authorization Framework.* Abgerufen am 23. August 2013 von http://tools.ietf.org/pdf/rfc6749.pdf

Hogan, M., Liu, F., Sokol, A., & Jin, T. (10. August 2011). NIST Cloud Computing Standards Roadmap Version 1.0. In *Special Publication 500-291* (S. 27-28). National Institute of Standards and Technology.

Höllwarth, T. (2012). *Cloud Migration.* Heidelberg: mitp Verlag.

How to use Blob Storage from Java. (kein Datum). (Microsoft) Abgerufen am 04. Mai 2013 von http://www.windowsazure.com/en-us/develop/java/how-to-guides/blob-storage/?fb=de-de

Introducing Windows Azure. (kein Datum). (Microsoft) Abgerufen am 04. Mai 2013 von http://www.windowsazure.com/en-us/develop/net/fundamentals/intro-to-windows-azure/?fb=de-de

jClouds BLOBSTORE GUIDE. (kein Datum). (THE APACHE SOFTWARE FOUNDATION) Abgerufen am 11. 06 2013 von http://jclouds.incubator.apache.org/documentation/userguide/blobstore-guide/

jclouds Developer Setup instructions for Eclipse. (kein Datum). (The Apache Software Foundation) Abgerufen am 05. August 2013 von http://jclouds.incubator.apache.org/documentation/devguides/using-eclipse/

Lenk, A., Klems, M., Nimis, J., Tai, S., & Sandholm, T. (2009). What's inside the Cloud? An architectural map of the Cloud landscape. Washington, DC, USA: IEEE Computer Society.

Microsoft. (2012). *Windows Azure Poster.* Abgerufen am 17. Mai 2013 von http://www.microsoft.com/en-us/download/details.aspx?id=35473

National Institute of Standards and Technology. (2011). *The NIST Definition of Cloud Computing.* Von http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf abgerufen

National Institute of Standards and Technology. (September 2011). *The NIST Definition of Cloud Computing.* Abgerufen am 5. April 2013 von http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

Petcu, D. (2013). Multi-Cloud. Expectations and Current Approaches. In *Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds* (S. 1-6). ACM New York. doi:10.1145/2462326.2462328

Rodriguez, A. (06. November 2008). *RESTful Web services. The basics.* Abgerufen am 03. Mai 2013 von https://www.ibm.com/developerworks/webservices/library/ws-restful/ws-restful-pdf.pdf

Schatten, A., Demolsky, M., Winkler, D., Biffl, S., Gostischa-Franta, E., & Östreicher, T. (2010). Dependency-Injection. In *Best Practice Software-Engineering* (S. 315-321). Heidelberg: Spektrum Akademischer Verlag.

Simple Cloud API. (kein Datum). Abgerufen am 05. Mai 2013 von http://framework.zend.com/manual/1.12/de/zend.cloud.html

Spillner, A., & Linz, T. (2012). Allgemeine Prinzipien des Softwaretestens. In *Basiswissen Softwaretest* (S. 37). Heidelberg: dpunkt.verlag GmbH.

Spillner, A., & Linz, T. (2012). Statische Analyse. In *Basiswissen Softwaretest* (S. 99-108). Heidelberg: dpunkt.verlag GmbH.

Umsatz mit Cloud Computing steigt über 5 Milliarden Euro. (4. März 2012). (BITKOM) Abgerufen am 12. April 2013 von https://www.bitkom.org/de/presse/74532_71376.aspx

Varia, J. (Janaur 2010). *Cloud-Architektur – Best Practices.* Abgerufen am 15. Mai 2013 von http://d36cz9buwru1tt.cloudfront.net/de/AWS_Cloud_Best_Practices_05252010.pdf

Varia, J., & Mathew, S. (März 2013). *Overview of Amazon Web Services.* Abgerufen am 03. Mai 2013 von http://d36cz9buwru1tt.cloudfront.net/AWS_Overview.pdf

Was ist Dropbox? (kein Datum). (Dropbox Inc.) Abgerufen am 04. Mai 2013 von https://www.dropbox.com/news/company-info

Weinhardt, C., Anandasivam, A., Blau, B., Borissov, N., Mein, T., Michalk, W., & Stößer, J. (11. Mai 2009). Cloud-Computing - Eine Abgrenzung, Geschäftsmodelle und Forschungsgebiete. *WIRTSCHAFTSINFORMATIK*, S. 453-462.

What is JClouds? (2011). Abgerufen am 05. Mai 2013 von http://www.jclouds.org/documentation/gettingstarted/what-is-jclouds/

What is Maven? (kein Datum). (The Apache Software Foundation) Abgerufen am 05. August 2013 von http://maven.apache.org/what-is-maven.html

Where does Dropbox store everyone's data? (kein Datum). Abgerufen am 04. Mai 2013 von https://www.dropbox.com/help/7/en

Versicherung über Selbstständigkeit

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

Hamburg, den ___14.11.2013___ _____

Aleksandr Nosov