Konzeption und Entwicklung einer Cloud-basierten, redundanten Speicherlösung

Melchior Jurczyk

BACHELORARBEIT

MatrikelNr. 627924

eingereicht am

Bachelorstudiengang

INFORMATIK

an der Hochschule Mannheim

im April 2011

Betreuer: M.Sc. Christian Baun

Zweitkorrektor: Prof. Dr. Georg Winterstein

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Bachelorarbeit

Konzeption und Entwicklung einer

Cloud-basierten, redundanten Speicherlösung

selbständig und ohne fremde Hilfe angefertigt habe. Dabei wurden nur die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet.

Hochschule Mannheim, den 04.04.2011

Melchior Jurczyk

Danksagung

Ich möchte mich an dieser Stelle bei M.Sc. Christian Baun für seine Unterstützung und Betreuung im Rahmen dieser Bachelorthesis bedanken.

Vielen Dank auch an Marc Fröhlich für das Korrekturlesen dieser Arbeit.

Ein ganz besonderer Dank geht an meine Mutter Edwarda Jurczyk, die mich während meines Studiums unermüdlich und mit vollem Herzen unterstützt hat.

Kurzbeschreibung

Ziel der Arbeit ist die Realisierung einer Cloud-basierten, anbieterunabhängigen Speicherlösung. Über diese können die Kunden ihre Daten transparent auf den Speicherressourcen mehrerer unterschiedlicher Dienstanbieter ablegen und verwalten. Die Speicherlösung verwendet die objektbasierten Speicherdienste Amazon Simple Storage Service (S3) und Google Storage. Eine Erweiterung für private Cloud-Installationen mit zu S3 kompatibler Schnittstelle, auf Basis von Eucalyptus und Cumulus ist möglich. Die Speicherlösung läuft als Webanwendung (Dienst) im öffentlich verfügbaren Cloud-Plattformdienst Google App Engine und wird in Python realisiert.

Die Nutzdaten der Kunden werden in den sogenannten Blobstore der Google App Engine übertragen und dort zwischengespeichert. Von dort aus verteilt die Speicherlösung alle Nutzdaten zu denjenigen Speicherdiensten, für die die Kunden Zugangsdaten importiert haben. Daraus ergibt sich eine redundante Speicherlösung.

Die Architektur ist darauf ausgelegt, dass es für die Kunden möglich ist, weitere Speicherdienste verschiedener Anbieter hinzuzufügen bzw. bestehende Dienste zu entfernen.

Hinweis

Der Begriff Cloud Computing sowie viele damit verbundene Technologien und Begriffe sind in den letzten Jahren neu aufgekommen. Wie in jeder Entstehungsphase neuer Technologien gibt es in vielen Bereichen weder präzise Definitionen noch genau definierte Standards.

Daher werden in dieser Arbeit die derzeit üblichen, in Forschung und Industrie sowie in deren sprachlichem Umfeld umgangssprachlich genutzten Begriffe verwendet. Es wird stets auf den Zusammenhang geachtet, sodass es für den Leser – falls dieser einen Begriff bzw. eine Begebenheit bisher aus einem anderen Kontext heraus kennt – verständlich und nachvollziehbar ist. Die Begriffsnutzung entspricht jedoch größtenteils dem derzeit vorherrschenden Verständnis.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

1 Einleitung

1.1 Motivation

In der Industrie und im privaten Umfeld ist es Stand der Technik, dass Daten in digitaler Form vorliegen. Die Wichtigkeit dieser Daten ist vom konkreten Einsatzzweck und Inhalt abhängig. Datensicherung ist mit zusätzlichem Aufwand verbunden und wird daher häufig vermieden. Gründe für nachlässige Datensicherung sind u.a. Unwissenheit und Bequemlichkeit. Unregelmäßige oder gar fehlende Datensicherung führt zur Gefahr von Datenverlust, was insbesondere im wirtschaftlichen Umfeld existenzgefährdende Auswirkungen haben kann.

Daher ist es wichtig, sich mit Datensicherungsmethoden auseinanderzusetzen. Generell existieren zwei Möglichkeiten zur Datensicherung:

1. Die Datenhaltung kann entweder vor Ort geschehen.
2. Die Datenhaltung kann an einem anderen Ort stattfinden.

Beide Vorgehensweisen haben verschiedene Vor- und Nachteile. Wird die Datensicherung vor Ort vorgehalten, ist der Zugriff jederzeit möglich. Allerdings setzt sich der Eigentümer der Daten der Gefahr aus, dass beispielsweise bei Elementarschäden, Diebstahl oder Sabotage wichtige Daten verloren gehen.

Wird die Datensicherung an einem geschützten, externen Ort aufbewahrt, riskiert der Kunde, nicht zu jeder Zeit direkten Zugriff auf seine Daten zu haben.

Eine Lösungsmöglichkeit, Datensicherung extern und hochverfügbar zu gewährleisten, ist der Einsatz von Online-Backup-Diensten. Die hohen Sicherheitsstandards, die die Anbieter solcher Speicherdienste in ihren Rechenzentren garantieren, ermöglichen es den Kunden, die Vorteile beider zuvor genannten Lösungen zur Datensicherung zu vereinen. Die Rechenzentren der Anbieter bieten den Nutzdaten der Kunden einen höheren Schutz vor Elementarschäden und anderen Gefahren, als dieses die Kunden selbst bei vertretbarem Aufwand lokal garantieren können. Voraussetzung für die Nutzung solcher externer Speicherdienste ist eine zuverlässige Netzwerkverbindung auf Seiten der Kunden.

1.2 Aufgabenstellung

Ziel der vorliegenden Arbeit ist die Entwicklung einer Cloud-basierten anbieterunabhängigen Speicherlösung. Ziel der Speicherlösung ist die Realisierung eines hohen Grades an Datensicherheit und Verfügbarkeit. Beide Eigenschaften werden in erster Linie dadurch realisiert, dass die Daten redundant auf den Ressourcen unterschiedlicher Speicherdienstanbieter abgelegt werden. Schwerpunktmäßig beschäftig sich diese Arbeit mit den Speicherdiensten Amazon S3 und Google Storage.

Die Architektur der Anwendung erlaubt eine nachträgliche Erweiterung bzw. Löschung von einzelnen Speicherdiensten. Insbesondere soll eine Erweiterung um zukünftig verfügbare Speicherdienste mit zu S3 kompatibler Schnittstelle möglich sein. Auch die Nutzung von Speicherdiensten in privaten Cloud-Infrastrukturen auf Basis von Eucalyptus oder Nimbus ist für zukünftige Erweiterungen möglich.

Die im Rahmen dieser Bachelorarbeit entwickelte Speicherlösung wird als Webanwendung konzipiert. Diese soll im öffentlich verfügbaren Plattformdienst Google App Engine laufen und wird in der Programmiersprache Python implementiert. Als temporärer Zwischenspeicher dient der Blobstore, ein in der App Engine nutzbarer Speicherdient für Binärdaten. Dadurch ist die Verteilung der Daten von den Servern der App Engine möglich und der Bedarf an Netzwerkbandbreite auf Seiten der Endanwender wird reduziert.

Die in dieser Arbeit entwickelte Speicherlösung wird untersucht und mit anderen am Markt etablierten Cloud-basierten Lösungen verglichen.

Kapitel 2

2 Cloud Computing

Eine feste Definition des Begriffs Cloud Computing existiert nicht. Stellvertretend sei hier die Definition aus [1] genannt:

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

Allgemein lässt sich feststellen, dass Cloud Computing die Nutzung von über das Internet zur Verfügung gestellten Ressourcen bezeichnet. Der Ursprung des Begriffs Cloud Computing ergibt sich daraus, dass das Internet in Netzwerkdiagrammen als Wolke (Cloud) dargestellt wird und dass die Wege zwischen den einzelnen Teilnehmern und Ressourcen häufig unbestimmt sind. Ziel des Cloud Computing ist, dass die Kunden aus der Wolke exakt die Ressourcen beziehen können, die sie zur Laufzeit benötigen. Die bestehenden Cloud-Dienste können anhand der zu Grunde liegenden Prinzipien, Kategorien und Architekturen unterschieden werden.

2.1 Cloud Prinzipien

Es existieren verschiedene Merkmale, u.a. Abstraktion, Flexibilität, bedarfsgerechte Abrechnung, Skalierbarkeit, Automatisierung und Dienstleistung, die allen Cloud-Diensten zu Grunde liegen.

2.1.1 Abstraktion

Die existierenden Cloud-Dienste können in einem mehrstufigen Modell (siehe Cloud Stack, Kapitel 2.2) in Stufen unterschiedlicher Granularität dargestellt werden. In jeder Stufe liegt ein höherer Unabhängigkeitsgrad von der darunterliegenden Hardware-Infrastruktur vor.

2.1.2 Flexibilität

Die unterschiedlichen Abrechnungsmodelle und die Möglichkeit, in kurzer Zeit verschiedene Ressourcen zu allozieren, bieten den Kunden ein hohes Maß an Flexibilität. Nachfolgend sind Beispiele für die Umsetzung der Flexibilität angeführt:

- Die Abrechnung erfolgt nach Verbrauch („pay-as-you-go“ Prinzip).
- Feste Vertragslaufzeiten sind unüblich, aber bei bestimmten Angeboten, wie z.B. den Amazon EC2 Reserved Instances möglich.
- Die Kunden nutzen die Dienste im gewünschten Umfang und der benötigten Zeitspanne, wie es ihrem Bedarf entspricht. Die Bereitstellung erfolgt somit on-demand.
- Während der Resssourcennutzung ist es für die Kunden jederzeit möglich die Menge der verwendeten Ressourcen den eigenen, sich eventuell verändernden Bedürfnissen anzupassen.
- Die Infrastrukturen der Dienstanbieter müssen skalierbar sein. Das bedeutet, dass sie auch plötzlich auftretende Lastspitzen (englisch: Peaks) bedienen können. Siehe hierzu auch Abschnitt 2.1.4.

2.1.3 Bedarfsgerechte Abrechnung

Aus der vorhandenen Flexibilität resultieren Vorteile, wie beispielsweise die zuvor erwähnte bedarfsgerechte Abrechnung (pay-as-you-go) genannt. Ein Entgelt für den Zugang bzw. die reine Bereitstellung ist unüblich und wird nicht erhoben.

Diese Form der Abrechnung ermöglicht den Zugang zu komplexer Software, die im traditionellem Verkaufs- bzw. Abrechnungssystem sehr kostenintensiv für die Kunden wäre. Der Kauf von Softwarelizenzen ist nicht immer empfehlenswert, da die betreffenden Anwendungen nur für einen kurzen Zeitraum benötigt werden. In solchen Anwendungsfällen ist ein flexibles Mietsystem, wie es in Cloud-Infrastrukturen möglich ist, für die Kunden sinnvoller.

2.1.4 Skalierbarkeit

Skalierbarkeit bezeichnet die Anpassungsfähigkeit eines Systems an veränderte Anforderungen. Beim traditionellen IT-Modell, das auf dem Beschaffen und Installieren von Hardware basiert, ist Skalierbarkeit nur dadurch zu erreichen, dass viel mehr Ressourcen betrieben werden, als es für den durchschnittlichen Bedarf nötig wäre. Der Betrieb von eigentlich überflüssiger Hardware verursacht laufende Kosten. Ein hohes Maß an Skalierbarkeit bieten u.a. öffentlich verfügbare Cloud-Infrastrukturdienste wie EC2.

2.1.5 Automatisierung

Software zum Betrieb von Cloud-Infrastrukturen ist darauf ausgelegt in hohem Maße zu automatisieren. Das betrifft u.a. die Ressourcenzuteilung, die Verwaltung der Kundenkonten, sowie die Messung des Ressourcenverbrauchs und spätere Abrechnung der verbrauchten Ressourcen.

2.1.6 Dienstleistung

Ressourcen aller Art werden in Form von Dienstleistungen den Kunden bereitgestellt. Diese Dienstleistung beinhaltet die der Bereitstellung eines Produkts zu Grunde liegenden Anschaffungen und Aufgaben genauso wie die kontinuierliche Pflege des Produkts. Ziel der Dienste ist eine höchstmögliche Benutzerfreundlichkeit, um möglichst viele Kundenschichten erreichen zu können.

Abbildung 2.1: Dienstleistungs-Prinzip als Werbebotschaft [8]

2.2 Cloud Stack

Die verschiedenen Cloud-Dienste lassen sich in unterschiedliche Kategorien unterteilen und in einen Cloud Stack einordnen. In diesem Modell unterscheiden sich die einzelnen Schichten bezüglich der Güte ihrer Dienste. Die Dienstgüte sagt aus, wie sich die Granularität der Abstraktion der Anwenderschicht vom zugrundeliegenden Dienst darstellt.

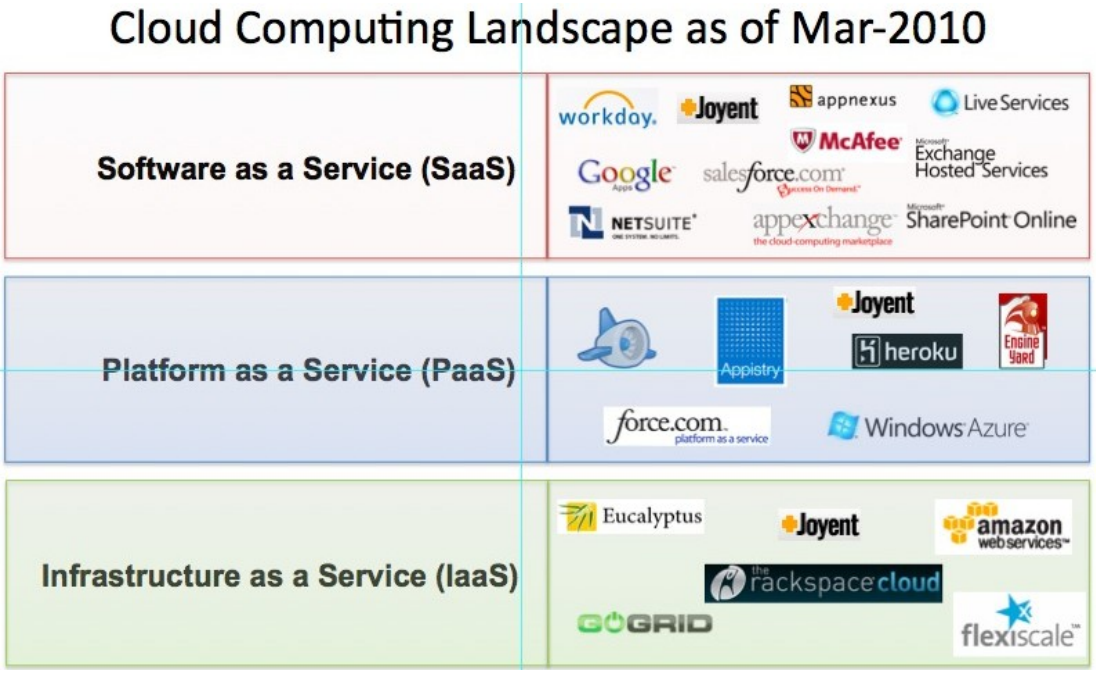

Abbildung 2.2: Populärste Cloud-Dienst Kategorien und Anbieterauswahl [9]

Die einzelnen Ebenen im Schichtenmodell werden nach der Funktionalität der Dienste unterschieden. Im Folgenden wird auf die einzelnen Schichten bzw. Kategorien eingegangen.

2.2.1 Infrastructure-as-a-Service (IaaS)

Die Infrastrukturdienste ermöglichen den Kunden den Betrieb virtueller Serverinstanzen mit beinahe beliebigen Betriebssystemen.

Server können über verschiedene Werkzeuge in verschiedenen Leistungskategorien gestartet und nach wenigen Minuten sofort verwendet werden. Auf diese Art lassen sich Cluster und sogar komplette Rechenzentren virtuell in einer Cloud-Infrastruktur realisieren. Die Kunden haben in ihren Instanzen volle Administrationsrechte, aber keinerlei Zugriff auf die physische Hardware. Die Bereitstellung, Verwaltung und

Wartung der Hardware ist die alleinige Aufgabe der Betreiber, die nicht zwingend notwendig auch die Anbieter der Dienste sein müssen.

Die Kunden haben die Möglichkeit, vorkonfigurierte Images, sogenannte Appliances zu nutzen. Eine Appliance enthält für einen bestimmten Verwendungszweck vorinstallierte und konfigurierte Softwarepakete, wie z.B. einen Webserver oder Datenbanksysteme.

Für den Kunden von Infrastrukturdiensten ergeben sich u.a. folgende Vorteile:

- Die Kunden müssen keine eigene Hardware betreiben und administrieren.
- Lastspitzen der eigenen Infrastruktur können ausgelagert werden.
- Plötzlich auftretendes Wachstum ist problemlos möglich (Skalierbarkeit).
- Softwaretests können ad-hoc auf verschiedenen Systemkonfigurationen gleichzeitig gefahren werden.
- Nicht mehr benötigte Ressourcen können die Kunden wieder freigeben.
- Die Einstiegshürde für kleinere Unternehmen und Startups ist dank flexibler Abrechnung äußerst gering.

Populäre Anbieter von Infrastrukturdiensten sind unter anderem Amazon EC2, Rackspace und GoGrid.

2.2.2 Platform-as-a-Service (PaaS)

Plattformdienste bieten Entwicklungs- und Laufzeitumgebungen als Dienste an und richten sich primär an Softwareentwickler.

Welche Programmiersprachen und Bibliotheken ein Plattformdienst unterstützt, hängt vom Anbieter ab.

Für die Kunden ergeben sich folgende Vorteile:

- Kein Administrationsaufwand der physischen Hardware und Systemsoftware.
- Skalierung auf den Anbieter-Servern erfolgt transparent.
- Bereitstellung von Bibliotheken für die Webentwicklung.

Es ist von Nachteil, dass jeder Plattformdienst spezifische Eigenschaften und Einschränkungen hat, an die sich die Entwickler anpassen müssen.

Zudem sind die Schnittstellen und Werkzeuge der unterschiedlichen Plattformdienste in der Regel nicht zueinander kompatibel.

Für den populären Plattformdienst Google App Engine existieren bereits freie Reimplementierungen. Deren Existenz verhindern, dass es zum Lock-In kommen kann, da die Kunden ihre Webanwendungen auch in einem privaten Kontext betreiben können.

Bekannte Plattformdienste sind u.a. die Google App Engine, Microsoft Azure, Heroku und Engine Yard

2.2.3 Software-as-a-Service (SaaS)

Bei Softwarediensten werden Anwendungen auf den Servern eines Dienstleisters betrieben und die Kunden nutzen diese Anwendungen mit einem Webbrowser über das Internet. Software muss von den Kunden nicht lokal installiert werden. Häufig ist eine lokale Installation auch nicht möglich.

Der Dienst des Anbieters liegt neben der Bereitstellung, Wartung und Instandhaltung vor allen Dingen in der Weiterentwickelung und Aktualisierung der angebotenen Webanwendungen. Der Kunde muss keine Installations- oder Updatezyklen durchlaufen oder beachten.

Grundvoraussetzung ist, dass der Kunde dem Anbieter in Bezug auf Datensicherheit und Erreichbarkeit des Dienstes vertrauen kann. Für unternehmenskritische Anwendungsfälle gibt es auch SaaS Anwendungen, die sich innerhalb der eigenen Infrastruktur betreiben lassen.

Mit SaaS wird eine Plattformunabhängigkeit, sowie die mobile Nutzung über internetfähige Geräte erreicht.

Typische SaaS Anwendungen sind u.a.: Salesforce.com, Google Docs oder Zoho.com

2.2.4 Everything-as-a-Service

Die Bezeichnung Everything-as-a-Service beschreibt kein konkretes Konzept, sondern den Ansatz eines dienstleistungsorientierten Vertriebsmodells.

Ein Produkt wird in Form einer Dienstleistung bereit gestellt. Um das Produkt entsteht daher kein oder nur minimaler Verwaltungs-Overhead. Alles was die Verwendung bzw.

den Betrieb des Produkts erst möglich macht, ist im Dienstangebot inbegriffen. Kunden können sich ausschließlich auf den Verwendungszweck des Produktes konzentrieren. Siehe hierzu auch Dienstleistungsprinzip Abschnitt 2.1.6.

Typische Merkmale solcher Dienste sind die Flexibilität in der Bereitstellung (on-demand) und eine verbrauchsabhängige Abrechnung (pay-as-you-go).

Neben den zuvor genannten populären Kategorien gibt es auch noch weitere:

- Humans-as-a-Service

  Menschen werden als Ressourcen eingesetzt [1]. Dies geschieht für Tätigkeiten, die keine hohe Qualifikation erfordern. Tätigkeiten die aber ein Computersystem nur schlecht oder überhaupt nicht erledigen kann. Typische Anwendungsfälle liegen in der Bilderkennung oder in der Textübersetzung. Ein bekanntes Beispiel ist Amazon Mechanical Turk.

- Storage-as-a-Service

  Kunden wird in Form einer Webanwendung Speicherplatz über das Internet zur Verfügung gestellt.

  Im Prinzip wird ein bestehender Dienst (Cloud Storage) veredelt und mit einem Mehrwert in Form eines höherwertigen Dienstes (Storage-as-a-Service) an die Kunden weitergereicht. Der Nutzen für die Storage-as-a-Service Anbieter ist, dass sie selbst kein Rechenzentrum bzw. Hardware vorhalten müssen, sondern nur den benötigten Speicher bedarfsgerecht anmieten. Der Kunde erhält eine Webanwendung zur Nutzung seines Online-Speichers.

  Ein Beispiel hierfür ist der Speicherdienst Dropbox, welcher den Cloud Storage von Amazon nutzt [10].

  Häufig werden die Begriffe Storage-as-a-Service und Cloud Storage umgangssprachlich synonym verwendet und nicht voneinander abgegrenzt.

2.2.5 Cloud Storage

Beim Cloud Storage wird mittels Programmierschnittstelle (API) Speicherplatz über das Internet zur Verfügung gestellt [11]. Auf die Hardware hat der Kunde keinen Zugriff. In Folge dessen ist keine Administration notwendig.

Cloud Storage wird flexibel verfügbar gemacht (on-demand) und nach Verbrauch abgerechnet (pay-as-you-go). Entstanden ist die Idee, aus der Notwendigkeit,

ungenutzte Kapazitäten an den Markt zu bringen. Mittlerweile ist der Cloud Storage Markt zu einem wichtigen Standbein für Cloud Computing Anbieter geworden.

Beispiele hierfür sind z.B. S3 von Amazon, Google Storage oder Rackspace Cloud Files

2.3 Cloud Architekturen

Neben der Einordnung nach technischen Gesichtspunkten, wie sie beim Cloud Stack vorgenommen wurde, ist auch eine Unterscheidung nach organisatorischen Merkmalen sinnvoll. Es können drei Typen unterschieden werden:

- Öffentliche Infrastrukturen (Public Cloud)
- Private Infrastrukturen (Private Cloud)
- Hybride Infrastrukturen (Hybrid Cloud)

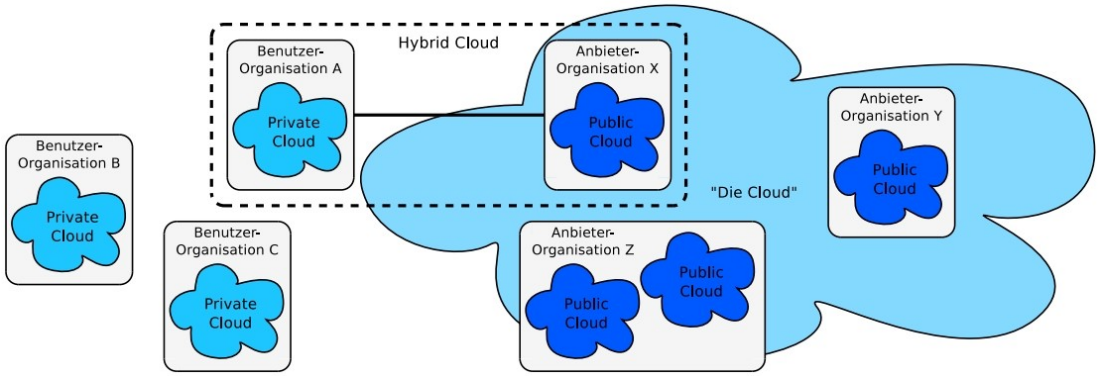

Abbildung 2.3: Cloud Organisationsstruktur [5]

2.3.1 Öffentliche Infrastrukturen (Public Cloud)

Bei der Public Cloud Architektur liegt die gesamte Cloud-Infrastruktur bei einem externen Anbieter. Diese Infrastruktur wird den Kunden bereitgestellt und nach dem „pay-as-you-go“ Prinzip abgerechnet. Der Begriff public bezieht sich dabei auf die Verfügbarkeit des Angebots über das öffentlich zugängliche Internet. In einer öffentlichen Infrastruktur werden die Ressourcen von vielen Kunden gemeinsam benutzt. Das Angebot hat ein hohes Maß an Transparenz, sodass den Kunden viele technische Details verborgen bleiben. Die Verwaltung und die Zuweisung der verfügbaren Ressourcen erfolgt vollautomatisch.

Der Kunde kann auf das Management und die Unterhaltung von eigener Hard- und Software verzichten. Neben der bedarfsgerechten Abrechnung wird auch von den anderen in Kapitel 2.1 genannten Vorteilen profitiert.

Als kritisch wird diese Architektur unter datenschutztechnischen Gründen angesehen. In Firmen mit restriktiver Datenschutzpolitik kommt sie daher nicht zur Anwendung. Gleichwohl besteht das Risiko eines Anbieter Lock-Ins. Diese muss jedoch nach Anbieter differenziert werden.

2.3.2 Private Infrastrukturen (Private Cloud)

Die Private Cloud ist eine mit Cloud-Technologien unternehmensintern aufgebaute Infrastruktur. Anbieter und Kunde gehören der gleichen Organisationseinheit an. Eine Private Cloud kommt beispielsweise dann zum Einsatz, wenn eine bereits vorhandene Infrastruktur konsolidiert wird um z.B. Cloud Prinzipien wie Skalierung und Automatisierung umzusetzen. Oder wenn datenschutzrechtliche Gründe eine Rolle spielen und die Nutzung einer öffentlichen Infrastruktur nicht in Frage kommt.

Daten verlassen nicht die unternehmensinterne Infrastruktur da sie hinter der eigenen Firewall bleiben. Es wird darauf Wert gelegt, die gleichen technischen Schnittstellen wie in einer öffentlichen Infrastruktur zu realisieren. Dadurch wird die Option offen gehalten Anwendungen in eine öffentliche Infrastruktur zu skalieren.

Der Nachteil bei einer privaten Infrastruktur liegt darin, dass die Kosten für die Anschaffung, das Management und die Wartung von Hard- und Software ähnlich hoch sind, wie die konventioneller Systemarchitekturen.

2.3.3 Hybrid Cloud

Die Hybrid Cloud ist eine Mischform der beiden zuvor erläuterten Architekturen. Hier versucht man die Vorteile beider Architekturen zu kombinieren und dadurch die Nachteile zu relativieren.

Für diese Architektur wird eine unternehmensinterne Infrastruktur vorgehalten, in der man unternehmenskritische Daten verarbeitet. Über eine kompatible Schnittstelle wird eine Anbindung an öffentliche Infrastrukturen realisiert. Anwendungen und Prozesse, die keine sensiblen Daten verarbeiten, können in eine öffentliche Cloud ausgelagert werden. Häufig wird diese Architektur genutzt, um Lastspitzen oder periodisch auftretende Lasten, wie beispielsweise zu Zeiten des Weihnachtsgeschäfts zu

überbrücken. Ebenso kann sie Verwendung finden um z.B. automatisiert Backups auszulagern.

Die Vorteile entsprechen den Vorteilen der zuvor vorgestellten Architekturen. Die Kosten für die eigene Infrastruktur können auf das notwendige Minimum reduziert werden. Die verfügbare Gesamtinfrastruktur ist trotzdem flexibel, hochautomatisiert und belastbar. Unternehmenskritische Daten verlassen die eigene Infrastruktur nicht und unterliegen der eigenen Kontrolle.

Eine Hybrid Cloud stellt für den Betreiber eine Kompromisslösung zwischen der Minimierung des Risikos und der Maximierung des Nutzens dar.

Kapitel 3

3 Beschreibung der Speicherlösung

Im Rahmen der vorliegenden Abschlussarbeit wird eine Speicherlösung entwickelt, die ihren Nutzern eine möglichst hohe Datensicherheit und Flexibilität bietet. Dies soll durch verteilte Redundanz der Datenhaltung und durch die Umsetzung der Software als Cloud-basierte Webanwendung gewährleistet werden.

Die Anwendung ermöglicht das Herauf- und Herunterladen von Dateien beliebigen Formats. Die Datenablage wird bei mehreren unterschiedlichen Speicherdienstanbietern realisiert. Zur Verwendung der Lösung werden lediglich gültige Benutzerkonten bei allen verwendeten Speicherdienstanbietern benötigt. Kosten fallen ausschließlich für die Nutzung der Speicherdienstanbieter an.

Die Verteilung der Daten geschieht automatisiert und transparent für den Benutzer. Vor der Verteilung werden die Daten von der Anwendung zwischengespeichert. Sie dient als Gateway, welches eine leistungsstarke Datenübertragung an mehrere Speicherorte bewerkstelligt.

Die Architektur ermöglicht es, Speicherdienstanbieter mit geringem Aufwand auszutauschen, hinzuzufügen oder zu entfernen. Weiterhin erlaubt sie den einfachen Austausch der Bedienoberfläche.

Durch die geographisch verteilte Datenablage und die daraus folgende Redundanz werden sensible und wichtige Daten besonders sicher verwahrt. Zwar ist der Einsatz der Anwendung zur zentralen Datenablage denkbar. Jedoch liegt ihr Haupteinsatzgebiet im Backup-Bereich.

Aufgrund der Konzeption als Webanwendung ist die Lösung öffentlich zugänglich und kann von Firmen- und Privatkunden gleichermaßen verwendet werden.

Die Anwendung kann in einer Private- oder Hybrid-Cloud eingesetzt werden. Dies ermöglicht die Geheimhaltung der Zugangsdaten zu externen Speicherdienstanbietern trotz externer Datenhaltung.

In den nachfolgenden Abschnitten wird detailliert auf die zu entwickelnde Anwendung und deren besondere Merkmale eingegangen.

3.1 Anforderungsanalyse

Die im Rahmen dieser Arbeit entstehende Anwendung soll den Kunden ermöglichen, Daten sicher und auf möglichst einfache Art und Weise zu speichern und zu verwalten.

Die folgenden Anforderungen sind maßgebend:

1. Sicherheit
2. Verfügbarkeit
3. Anbieterunabhängigkeit
4. Transparenz
5. Benutzerfreundlichkeit

In Kapitel 4 wird die Umsetzung der Anforderungen anhand eines Prototyps erläutert.

3.1.1 Sicherheit

Sicherheitsaspekte sind für jede Anwendung bedeutsam, die im Unternehmensrahmen eingesetzt werden kann. Der Begriff Sicherheit fasst viele Kriterien zusammen. In diesem Unterkapitel wird hauptsächlich auf die Daten- bzw. Datenausfallsicherheit eingegangen.

Daten müssen vor Diebstahl, Brand, höherer Gewalt, mutwilliger Beschädigung und sonstigen unvorhersehbaren Ereignissen geschützt werden.

Im privaten Umfeld ist dies kaum realisierbar. Datensicherungen auf mobilen Datenträgern sind zwar vor einem Ausfall einer PC-internen Festplatte geschützt, nicht jedoch vor Diebstahl oder Beschädigungen. Zum Schutz wäre die Lagerung des Datenträgers an einem entfernten Ort denkbar. Dies erhöht zwar die Sicherheit, jedoch könnten zuvor genannte Gefahren auch an jedem anderen Ort auftreten.

Zuverlässige Sicherheitsmaßnahmen werden von Speicherdienstanbieter geboten. Diese leben davon, ihren Kunden einen höchstmöglichen Grad an Datensicherheit zu bieten. Sie nehmen für den Bau und Betrieb ihrer Rechenzentren hohe Kosten auf sich, die für Kleinunternehmen und Privatanwender undenkbar wären.

Anlagen dieser Art können nur nach dem Passieren strenger Sicherheitsvorkehrungen betreten werden, was den Schutz vor Manipulation, Diebstahl oder Zerstörung ermöglicht. Weiterhin besitzen sie einzelne, in sich abgeschlossene Schutzbereiche, die

u.a. mit autonomen Lösch-, Belüftungs- und Stromkreisläufen zum Schutz vor höheren Gewalten abgesichert sind. Die Duplizierung wichtiger Daten an andere Standorte bietet zudem Schutz vor geographisch ausgedehnten Elementarschäden, die Totalausfälle bewirken könnten.

Deshalb verwendet die zu entwickelnde Software für die Datenhaltung ausschließlich auf Datenhaltung spezialisierte Anbieter.

3.1.2 Verfügbarkeit

Bei einer Datenablage, welche auf Speicherdienstanbietern basiert, muss geprüft werden, ob der Datenzugriff jederzeit gewährleistet ist.

Deshalb ist zum einen die Verfügbarkeit des Dienstes zu betrachten und zum anderen die Verfügbarkeit der gespeicherten Daten.

- Die Verfügbarkeit des Dienstes

  Für den Kunden stellt die Erreichbarkeit des Speicherdienstanbieters ein grundlegendes Kriterium bei der Wahl eines Anbieters dar.

  Die Lagerung und der Zugriff über das Internet vereinen die Vorteile der externen Datenhaltung mit der Nutzerfreundlichkeit eines lokalen Zugriffs - eine breitbandige Netzanbindung vorausgesetzt.

  Intelligente Routingverfahren erlauben einen Zugriff auf Daten in Rechenzentren selbst dann, wenn Teile des Internets ausfallen würden.

  Ein Richtwert für Erreichbarkeit eines Dienstes ist die sogenannte Uptime. Dies ist die zugesicherte Erreichbarkeit der Speicherinfrastruktur eines Speicherdienstanbieters. In den SLAs der Anbieter werden garantierte Erreichbarkeiten in Prozenten angegeben. Werte von 99.99% sind üblich und garantieren eine max. Nichterreichbarkeit von 52:36 Minuten pro Jahr.

  Die Anwendung nutzt ausschließlich Anbieter, die eine Erreichbarkeit von 99.99% garantieren.

- Die Verfügbarkeit der Daten

  Neben der Diensterreichbarkeit spielt die Zuverlässigkeit der Datenhaltung eine wesentliche Rolle. Bei der Wahl des Speicherdienstanbieters sollten die Antworten auf folgende Fragen entscheidend sein:

    - Besteht das Risiko, dass Daten verloren gehen?

- Wenn ja, wie hoch ist es?
- Kann man immer lesend und schreibend zugreifen?
- Gibt es Wartungsfenster, in denen der Zugriff nicht möglich ist?

Auf Grund dieser Kriterien verwenden Speicherdienstanbieter Technologien die ihren Kunden eine Objektzuverlässigkeit von 99,99% Uptime und mehr garantieren.

Die Anwendung nutzt ausschließlich Speicherdienstanbieter, deren SLAs eine Zuverlässigkeit von mindestens 99.99% garantieren.

3.1.3 Anbieterunabhängigkeit

Die hohen Maßstäbe an Sicherheit und Verfügbarkeit, von auf Datenhaltung spezialisierten Unternehmen sind vielversprechend. Jedoch muss beachten werden, dass trotz des deutlichen Zugewinns an Sicherheit und Verfügbarkeit im Gegensatz zur eigenen Speicher- bzw. Backup-Lösung, das Risiko eines Anbieter Lock-Ins steigt. Hierbei können zwei Arten von Lock-Ins unterschieden werden:

- API Lock-In

  Jeder Anbieter stellt eine andere Schnittstelle (API) zur Nutzung seines Speicherdienstes zur Verfügung. Die Entscheidung für einen Anbieter führt zwangsläufig zu einer engen Integration mit den eigenen Systemen und einem damit verbundenem Verlust an Flexibilität.

  Es wird gegen die Anbieter-API programmiert, was bedeutet, dass Änderungen an Dieser zeitnahe Änderungen an der eigenen Software notwendig machen. Bei umfangreichen Projekten verursacht eine Portierung zu anderen Anbietern mit einer anderen API einen enormen Aufwand. Deshalb wird häufig auf einen Wechsel verzichtet, oder dieser nur bei absoluter Notwendigkeit unternommen. Es entsteht also eine Abhängigkeit von der API – ein sogenannter API Lock-In.

- Daten Lock-In

  Ein weiterer Aspekt ist die anfallende Datenmenge. Je länger eine Speicherlösung in einem Unternehmen Bestand hat, desto größer wird das benötigte Datenvolumen. Bei Datenmengen im Tera- und Petabyte Bereich reicht zur Übertragung eine breitbandige Internetanbindung nicht mehr aus. Der Bandbreiten- und Zugriffsverlust während großer Dateitransfers stellt für viele Kunden ein großes Problem dar. Zwar bieten manche Speicherdienstanbieter

einen Festplatten-basierten Import/Export an. Jedoch kostet auch Kopie, Versand und Wiederaufspielen der Daten Zeit. Handelt es sich dabei um Backups, ist dies unkritisch. Handelt es sich jedoch um aktiv genutzte Daten, so werden diese Vorgänge für viele Firmen zu einem komplizierten Unterfangen.

Ein steigendes Datenvolumen steigert den Aufwand eines Wechsels des Speicherdienstanbieters. Es entsteht eine Anbieterabhängigkeit auf Grund der vorhandenen Datenmenge – ein sogenannter Daten Lock-In.

Trotz des Lock-In-Risikos muss nicht auf Flexibilität und Leistungsumfang verzichtet werden. Durch die Einführung einer anbieterunabhängigen Abstraktionsschicht kann eine Loslösung von der Anbieter-API erreicht werden.

Daten Lock-Ins können kaum vermieden werden. Jedoch sollte zur Minderung der Auswirkungen ein Anbieter, der Festplatten basierten Import bzw. Export von Daten anbietet gewählt werden.

Die zu entwickelnde Anwendung nutzt die Boto API als anbieterunabhängige Abstraktionsschicht. Mehr zur Verwendung von Boto im Abschnitt 4.2.4.

3.1.4 Transparenz

Im Gegensatz zum umgangssprachlichen Gebrauch, beschreibt der Begriff Transparenz in der IT den vom Anwender nicht wahrnehmbaren Ablauf bestimmter Programmteile.

Ein hoher Grad an Automatisierung soll dies gewährleisten. Der Kunde soll sich auf die Kernfunktionalität einer Anwendung konzentrieren können und alle zusätzlichen Aufgaben weitestgehend der Software überlassen.

3.1.5 Benutzerfreundlichkeit

Ein Kunde sollte sich schnell in einem System zurechtfinden und die gegebene Funktionalität unkompliziert und möglichst ohne umfangreiche Anleitung nutzen können.

Daher sollte die Benutzeroberfläche selbsterklärend und einfach in der Handhabung konzipiert werden. Unnützen Details sowie umständliche Menüführungen sollte vermieden und eine schlichte und zweckmäßige GUI entwickelt werden.

Die Farben sollten hell und positiv, die gesamte Anwendung einladend und freundlich wirken.

3.2 Beschreibung der Zielplattformen

Die zu entwickelnde Software läuft als Webanwendung auf einer Public Cloud Plattform und nutzt Cloud Storage Dienste zur Datenhaltung. Im Folgenden werden die verwendeten Anbieter und deren Dienste vorgestellt.

3.2.1 Amazon S3

Amazon S3 ist ein Speicherdienst aus der Familie der Amazon Web Services (AWS), die das Public Cloud Angebot von Amazon darstellen [12]. Er wurde 2006 als „Speicher für das Internet“ gestartet, mit dem Gedanken skalierbare Webentwicklung zu erleichtern [13].

Daher ist er auf Skalierbarkeit, Hochverfügbarkeit und schnelle Zugriffszeiten ausgelegt. Abgerechnet wird nach dem „pay-as-you-go“ Prinzip. Bei der Anmeldung zu S3 kann zwischen einem der vier Rechenzentren, USA Ost- bzw. Westküste, Singapur und Irland gewählt werden.

Dateien werden in einem Key/Value basierten System gespeichert und in sogenannten Buckets organisiert. Die Buckets müssen global über Amazon S3 eindeutig sein und können nicht geschachtelt werden. Pro Benutzer dürfen maximal 100 Buckets angelegt werden. Durch geschickte Namensgebung kann eine konventionelle Ordnerstruktur konstruiert werden. Buckets sowie darin enthaltene Dateien können als URLs adressiert und abgerufen werden. Der Zugriff kann über SOAP, REST oder auch BitTorrent erfolgen. Für viele populäre Programmiersprachen stehen hierfür Bibliotheken zu Verfügung.

Einzelne Dateien können bis zu 5 Terabytes groß sein. Größere Datenmengen können alternativ zur Internetübertragung per Festplatte über den Postweg eingesendet bzw. angefordert werden.

Seit Februar 2011 besteht die Möglichkeit, in einem Bucket statische HTML Seiten bereit zu stellen und S3 somit als Hoster zu verwenden.

Vielversprechend ist die von Amazon garantierte Zuverlässigkeit bzw. Haltbarkeit der gespeicherten Daten. Amazon garantiert eine Objektzuverlässigkeit von

99.999999999% pro Jahr. Werden beispielsweise 10.000 Objekte in S3 ablegt, geht im Schnitt alle 10.000.000 Jahre ein Objekt verloren. Dieser hohe Grad an Zuverlässigkeit wird durch verteilte, redundante Datenablage erreicht.

Für Daten kleinerer Priorität besteht die Möglichkeit, Daten auf einen S3 Reduced Redundancy Storage abzulegen. Dieser Dienst bietet nur 99.99% Zuverlässigkeit und wird kostengünstiger zur Verfügung gestellt. Laut Amazon werden die durch diesen Dienst verwalteten Daten nicht häufig repliziert, sind jedoch etwa 400 Mal zuverlässiger gespeichert, als auf einer herkömmlichen Festplatte [14].

Amazon sichert die Erreichbarkeit seiner Dienste mit 99.99% Uptime pro Jahr zu. Somit beläuft sich die maximal erlaubte Ausfallzeit auf 52:56 Minuten.

Amazon war der erste Cloud Storage Anbieter. Daraus lässt sich ein hoher Reifegrad des Dienstes und die quasi Marktführung ableiten. Laut Angaben von Amazon wurden im Jahr 2010 rund 100 Milliarden Objekte gespeichert, auf denen rund 120.000 Operationen pro Sekunde ausgeführt wurden [13].

3.2.2 Google Storage

Google bietet seit Sommer 2010 ebenfalls einen Cloud Storage an [15].

Der Dienst ist auf einen hohen Datendurchsatz und maximale Skalierbarkeit ausgelegt. Aufgrund von Replizierungsmechanismen wird eine hohe Datenbeständigkeit bzw. Integrität zugesichert, jedoch nicht genauer beziffert.

In vielen Bereichen ähnelt der Google Dienst dem Konkurrenzprodukt S3 von Amazon. Auch hier werden Daten per REST übermittelt, in einem Key/Value basiertem System gespeichert und in Buckets organisiert. Wie bei Amazons S3 müssen Buckets systemweit eindeutig sein und sind nicht schachtelbar, jedoch sind sie im Gegensatz zu S3 nicht in der Anzahl beschränkt.

Bei der Betrachtung von Google Storage sind besonders zwei Tatsachen hervorstechend:

1. Kompatibilität zu Amazon S3

   Google Storage kann aufgrund des Key/Value Mechanismus aus der gleichen API heraus genutzt werden, wie Amazon S3. Diese Tatsache macht es zu seinem größten Konkurrenzprodukt kompatibel. Diese Kompatibilität ist beabsichtigt und bleibt daher mit an Sicherheit grenzender Wahrscheinlichkeit zukünftig bestehen. Ein API Lock-In wird dadurch vermindert, was die Nutzung beider Dienste interessant macht.

2. Integration in das Google Ökosystem

   Bereits jetzt kann man Google Accounts dazu benutzen, Nutzer- oder Gruppenauthentifizierungen vorzunehmen. Weitere Möglichkeiten sind denkbar und werden im Laufe der Zeit sicherlich folgen. Absehbar ist vor allen Dingen die Verflechtung von Google Storage mit anderen Google Diensten. Dies ermöglicht neue Nutzungsszenarien und erhöht den Nutzwert des gesamten Google Ökosystems für dessen Kunden.

Da Google Storage in direkter Konkurrenz zu Amazon S3 steht, sind ähnliche Funktionen zu erwarten, wie sie S3 zur Verfügung stellt. Dies bietet den Kunden einen Wiedererkennungswert und fördert die Akzeptanz des Dienstes am Markt. Ein Nutzer muss sich ein Nutzer bei einem Wechsel zwischen den Anbietern nicht an eine neue Programmiertechnik oder Abrechnungspolitik gewöhnen.

3.2.3 Google App Engine

Die Google App Engine ist ein von Google angebotener Plattformdienst [16]. Im Cloud Stack kann er der Kategorie Platform-as-a-Service zugeordnet werden. Die App Engine bietet eine komplette Laufzeitumgebung (sog. Sandbox) zum Hosting von Webanwendungen auf der Google Infrastruktur an.

Alle betriebsbedingt notwendigen Ressourcen wie Hardware, Softwareinstallation und Konfiguration der Webserver werden von Google verwaltet. Die Lastenverteilung und das komplette Management der Laufzeitumgebung wird automatisiert und transparent umgesetzt. Darüber hinaus bietet Google ein eigenes Web-Framework an, welches die Entwicklung von Anwendungen für die App Engine ermöglicht. Dieses ist sowohl für Python, als auch für Java verfügbar. Dank der Java-Unterstützung lassen sich für die Entwicklung auch alle anderen Programmiersprachen einsetzen die Bytecode generieren.

Der verfügbare Funktions- bzw. Bibliothekenumfang ist programmiersprachen-abhängig. Google erweitert für beide Sprachen kontinuierlich das Angebot an verwendbaren Funktionen. Viele davon werden intern bereits längere Zeit verwendet, bevor sie im Plattformdienst integriert und angeboten werden.

Es werden folgende Funktionen in Form von APIs bereitgestellt:

- Datastore

  Darunter versteht man eine verteilte Key/Value Datenbank. Diese kann entweder objektorientiert oder per GQL (SQL ähnliche Auszeichnungssprache) abgerufen

werden. Transaktionen sind ebenso verfügbar, wie alle im Datenbankumfeld gängigen Datentypen.

- Memcache

  Ist ein hochperformanter Kurzspeicher, der als temporärer Hauptspeicher dient. Mithilfe dieser API kann man Daten zwischen multiplen Instanzen einer Anwendung mit äußerst niedrigen Zugriffszeiten austauschen (cachen).

- URL Fetch

  Ermöglicht die Einbindung von Internet-Ressourcen und damit das Senden und Empfangen von Daten über die hochperformante Google Netzwerkinfrastruktur.

- Mail

  Ermöglicht Anwendungen, die Mail-Funktionen von Google zu nutzen. Durch die Verwendung dieser Schnittstelle ist es möglich, über unterschiedliche Mail Adressen zu empfangen und zu senden.

- XMPP

  Mittels der XMPP API kann mit allen XMPP kompatiblen Instant Messaging Anwendungen über das XMPP Protokoll kommuniziert werden.

- Images

  Mit Hilfe der Image API wird eine rudimentäre Bearbeitung von JPG und PNG Dateiformaten ermöglicht. Es werden die Funktionalitäten drehen, spiegeln, skalieren und zuschneiden von Bildern bereitgestellt.

- Google Accounts

  Bietet Kunden einer App Engine Anwendung die Möglichkeit, sich mit einem Googlekonto oder einem OpenID-Account zu authentifizieren. Die komplette Accountabwicklung wird von Google übernommen.

- Task

  Aufgaben die unabhängig von einem Web-Request ausgeführt werden sollen, können als Tasks definiert und einer Queue übergeben werden. Die App Engine erkennt neue Tasks und führt diese automatisch aus. Aufgaben die nicht den vollen Umfang der Task API benötigen, können mit der deferred API, welche auf der Task API aufsetzt, realisiert werden.

- Blobstore

  Im Blobstore können große Binärdaten, sogenannte Blobs, abgelegt werden. Jedes Blob wird automatisch über einen BlobInfo-Eintrag im Datastore referenziert.

Google stellt ein sehr mächtiges API- und Laufzeit-Ökosystem zur Verfügung. Es muss jedoch beachtet werden, dass es Sandbox-Einschränkungen gibt. Diese begründen sich zum Teil durch das Wesen stark verteilter, skalierbarer Webanwendungen und in strategischen bzw. sicherheitstechnischen Aspekten. Jede der zuvor genannten APIs hat spezifische Einschränkungen – sogenannte Quotas [17]. Diese werden von Google in unregelmäßigen Abständen angepasst und sollten vor jedem Projekt mit den eigenen Anforderungen abgeglichen werden.

Es gilt folgende Sandbox-Einschränkungen zu beachten:

- Es besteht nur lesender Zugriff auf das Dateisystem. Dateien können mit dem Projekt hochgeladen und gelesen, jedoch nicht verändert oder gänzlich neu erstellt werden.
- Direkter Zugriff auf andere Netzwerkressourcen ist genauso wie das Öffnen von Sockets nicht möglich. Kommunikation mit anderen Ressourcen wird nur anhand der URL Fetch API und den HTTP/HTTPS Ports 80/443 ermöglicht.
- Das Starten von Sub-Prozessen und Multi-Threading ist nicht möglich.
- Domänen ohne www, werden genauso wenig unterstützt wie SSL/HTTPS über eigene Domänen.
- Es wird nur ein Teil der Java SE Klassen unterstützt, die Python Unterstützung ist umfangreicher.

Die Dienstnutzung ist bis zu den in den SLAs festgelegten Quoten kostenfrei. Im Falle einer Überschreitung wird die Nutzung nach verbrauchten Einheiten in Rechnung gestellt. Trotz Freikontingent erfordern einige Dienste vor ihrer Inanspruchnahme die Freischaltung der Abrechungsfunktion (z.B. Blobstore).

Die im Rahmen dieser Arbeit entwickelte Anwendung soll in einem hohen Maße verfügbar und belastbar umgesetzt werden. Aufgrund der zuvor beschriebenen Untersuchungen eignet sich die Google App Engine als Plattform. Technische Details, die bei der Anwendungsentwicklung auf der App Engine eine Rolle spielen sind in Kapitel 4.3.1 vorzufinden.

3.3 Anwendungsfälle

Betrachtet man die Anwendung hinsichtlich der geplanten Funktionen, lassen folgende, in Abbildung 4 dargestellte Anwendungsfälle benennen:

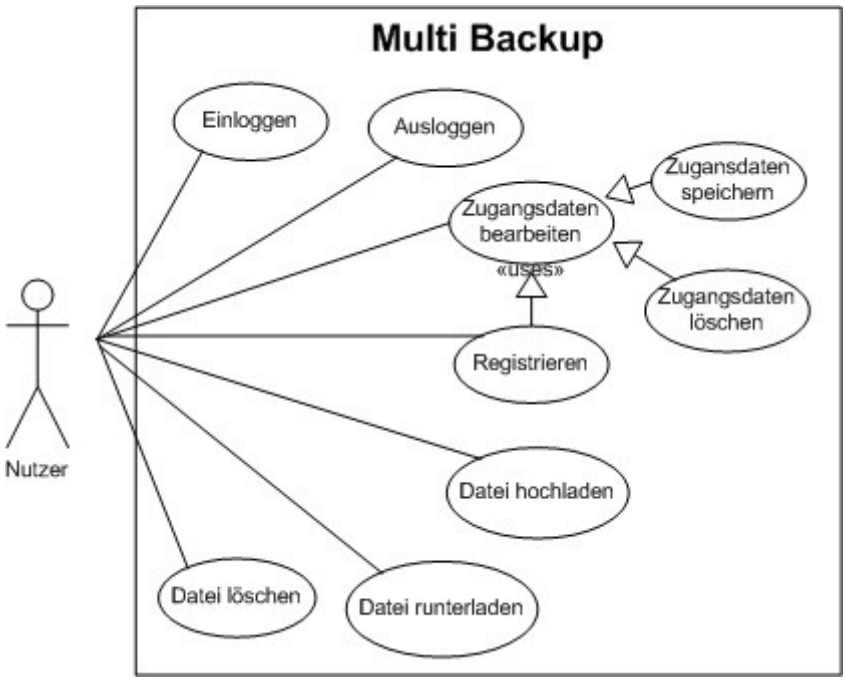

Abbildung 3.3: Anwendungsfälle von MultiBackup

Beim Aufruf der Webanwendung besteht die Möglichkeit sich entweder einzuloggen oder zu registrieren.

Möchte sich der Kunde registrieren, wird er auf eine Seite weitergeleitet, die ihm die Möglichkeit gibt, die Zugangsdaten seiner Speicherdienstanbieter zu speichern oder zu löschen. Daraufhin kann er sich auf die Upload Seite weiterleiten lassen.

Die Kernfunktionalität der Anwendung stellt der Up- und Download von Dateien dar. Die bereits hochgeladenen Dateien werden über die gesamte Dauer der Anwendungsnutzung als Liste unter dem Upload-Bereich aufgeführt. Von dort aus kann jede Datei einzeln gelöscht oder heruntergeladen werden.

Ein Button zum Ausloggen wird sofern der Kunde sich einloggt eingeblendet.

Kapitel 4

4 Beschreibung der Implementierung

In Kapitel 4 wird im Detail auf die Implementierung der während dieser Arbeit entstandenen Anwendung eingegangen. Basierend auf den Grundlagen, die in Kapitel 3 besprochenen wurden, soll nachfolgend die technische Umsetzung dargelegt werden.

4.1 Architektur

Die Anwendung wurde nach dem Model-View-Controller (MVC) Pattern entworfen und lässt sich auf eine klassische 4-Tier Web-Architektur abbilden. Somit ist die Präsentationsschicht von der Anwendungslogik getrennt, was die Erweiterbarkeit der Programm-Logik und die Portierung der GUI vereinfacht.

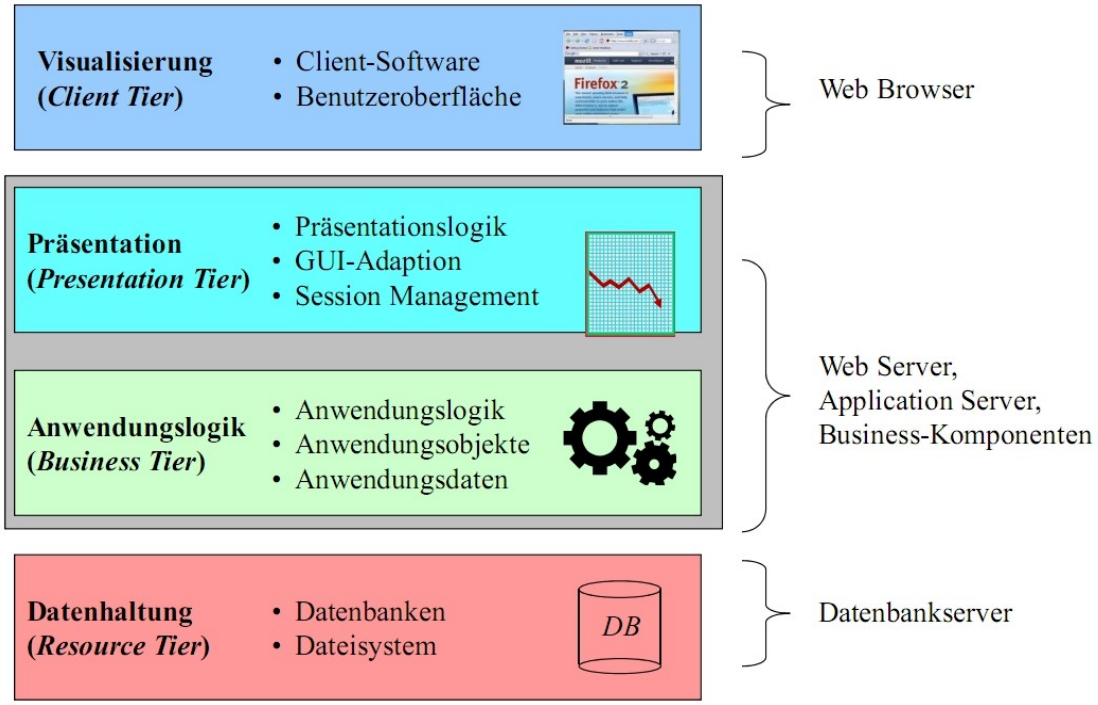

Abbildung 4.1: Darstellung einer 4-Tier Architektur [6]

Nachfolgend werden die einzelnen Schichten im Bezug auf die entstandene Anwendung erläutert:

- Client Tier

  Die Visualisierung wird durch den Browser des Kunden vorgenommen und erfolgt daher Client-seitig. Die Benutzeroberfläche beinhaltet alle notwendigen Elemente zum Ein- bzw. Ausloggen, zum Registrieren von Speicherdienstanbietern und zur Bedienung des Dateimanagements. Der clientseitige Code besteht aus einzelnen HTML Dateien und einer global verwendeten CSS Datei. JavaScript wird nicht verwendet (mehr dazu in Kapitel 4.2.3).

- Presentation Tier

  Die Präsentation ist serverseitig implementiert und stellt die Rendering- und Zugriffsschnittstelle der Visualisierungsschicht dar. Zum Einsatz kommt das von der App Engine nativ unterstützte Django Web-Framework, dessen Renderer dynamisch HTML Code generiert und der Visualisierungsschicht übergibt. Die Umleitung der Aufrufe aus der Visualisierungsschicht zur Anwendungslogik wird vom Google eigenen Webapp Framework übernommen. Auf Django und Webapp wird in Kapitel 4.2.4 näher eingegangen.

- Business Tier

  Die Anwendungslogik ist ebenfalls serverseitig implementiert und stellt den Kern der Anwendung dar. In einzelnen Klassen gekapselt, werden Methoden zur Verarbeitung der Nutzerinteraktionen ausgeführt. Deren Resultate werden an die Präsentationsschicht bzw. den Renderer weitergegeben. Weiterhin werden aus dieser Schicht heraus die Operationen zu den Speicherdienstanbietern realisiert und sämtliche internen Datenbankzugriffe vorgenommen. Als Programmiersprache wurde ausschließlich Python verwendet.

- Resource Tier

  Die Datenhaltung wird an Hand des in der App Engine integrierten Datenbanksystems umgesetzt. Der sogenannte Datastore kann entweder über ein objektorientiertes Interface oder über eine SQL ähnliche Syntax (GQL) abgefragt werden. Die Eigenschaften des Datastores wurden in Kapitel 3.2.3 besprochen, auf das verwendete Datenmodell wird in Kapitel 4.3.2 eingegangen.

4.1.1 Lösung als Dienst

Die Umsetzung erfolgt als Webanwendung und wird im Cloud Stack der Software-as-a-Service Schicht zugeordnet (siehe Kapitel 2.2).

Hauptkriterium für diese Entscheidung ist die fast allgegenwärtige Verfügbarkeit von internetfähigen Geräten. Der Kunde kann somit unabhängig von seinem Aufenthaltsort über sämtliche seiner Daten verfügen.

Die Vorteile der Lösung als Dienst wurden bereits in Kapitel 2.2.3, Software-as-a-Service erläutert.

Die Umsetzung als Desktop-Anwendung könnte in Betracht gezogen werden. Die Software würde als Installationsmedium, samt Handbüchern an den Kunden ausgeliefert werden. Im Gegensatz zu einer SaaS-Anwendung Anwendung würden dem Kunden dadurch aber eine Reihe von Nachteilen entstehen:

1. Der Kunde muss sich um die Installation der Software bzw. um die Neuinstallation bei Systemwechsel oder Crash kümmern.
2. Die Durchführung von Updates muss regelmäßig vorgenommen werden.
3. Der Kunde ist an das Gerät gebunden, auf dem die Software installiert ist. Daraus folgt, dass ihm eine Lokalität aufgezwungen wird, die Software dort zu nutzen wo sie installiert ist und nicht etwa dort wo er sie evtl. bräuchte.
4. Der Kunde zahlt für die Software als Komplettpaket, unabhängig von der tatsächlichen Häufigkeit und Anzahl der Nutzung aller bereitgestellten Funktionen.

Die zuvor genannten Nachteile treffen auf eine SaaS-Lösung nicht zu. Daher erschließt sich, dass für die Realisierung der in Kapitel 1.2 definierten Speicherlösung eine Desktop-Anwendung keine Alternative darstellt. Als Ergänzung aber denkbar wäre (siehe Kapitel 6).

4.1.2 Integration von Cloud Storage

Für die persistente Datenhaltung werden Speicherdienste von professionellen Drittanbietern genutzt (siehe Kapitel 3.2.1 und 3.2.2). Es bestehen zwei Möglichkeiten der Überführung der Kunden-Uploads zu den verwendeten Speicherdiensten.

1. Vom Browser direkt zum Speicherdienstanbieter

   Der Kunde lädt seine Daten direkt zu einem oder zu mehreren Speicherdienstanbietern. Die Anwendung stellt ein Upload-Formular bereit, welches den Daten-Upload direkt an den entsprechenden Bucket des Speicherdienstanbieters zulässt. Die Anwendung dient als Vermittlungsstelle.

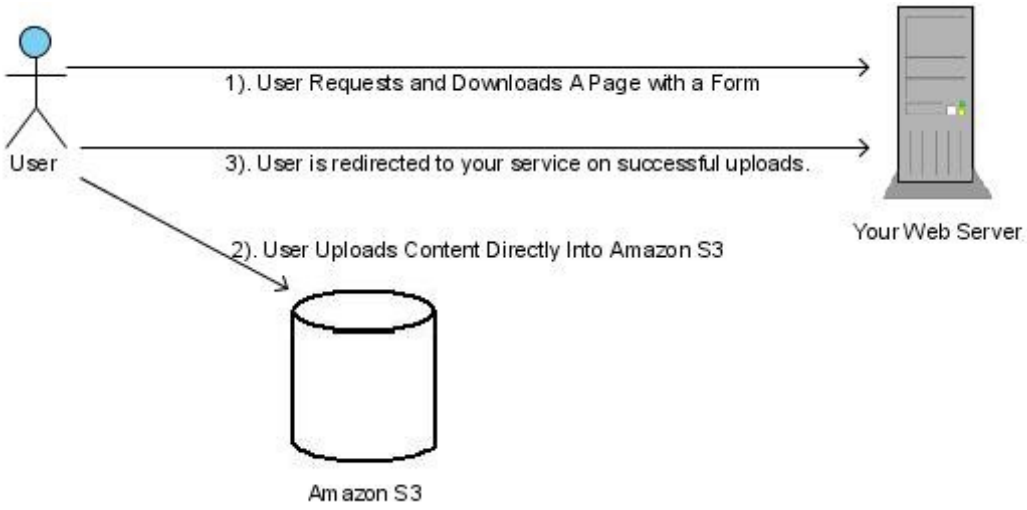

Abbildung 4.2: Ablauf eines direkten Uploads [18]

   Der Vorteil dieser Variante liegt darin, dass dem Betreiber der Anwendung kein Upload Traffic entsteht.

   Den Nachteil hat der Kunde, für den Fall, dass mehrere Speicherdienstanbieter adressiert werden. Die benötigte Bandbreite vervielfacht sich um die Anzahl der Speicherdienstanbieter.

2. Vom Browser über einen Server zum Speicherdienstanbieter

   Die Daten werden vom Kunden an die Webanwendung übertragen. Daraufhin wird die weitere Übertragung an den Speicherdienstanbieter vorgenommen. Die Anwendung dient dabei als Proxy und temporärer Zwischenspeicher.

   Diese Vorgehensweise wird unter zwei Bedingungen eingesetzt:

   1. Wenn die Daten vor der dauerhaften Speicherung bearbeitet werden sollen.

   2. Wenn mehrere Speicherdienstanbieter genutzt werden.

   Der Vorteil besteht darin, dass die Upload-Dauer und der Traffic unabhängig von der Anzahl der Speicherdienstanbieter sind.

Der Nachteil liegt bei dem Betreiber der Anwendung, da dieser zusätzliche Bandbreite bereitstellen muss.

Da bei der vorliegenden Arbeit mehrere Speicherdienstanbieter zum Einsatz kommen, wird die zweite Variante verwendet.

4.1.3 Cloud-basierte Umsetzung

Eine Webanwendung benötigt eine Laufzeitumgebung, in der sie ausgeführt wird. Hardware, ein Betriebssystem, ein Webserver und programmiersprachenabhängige Erweiterungen bilden die Betriebsgrundlage.

Diese Komponenten müssen gekauft oder gemietet sowie konfiguriert werden. Dies bindet Arbeitskraft und verfügbare Zeit. Der daraus entstehende Nachteil ist allerdings nicht so gravierend, wie der Folgende:

Infrastrukturen sind aufgrund eines begrenzten Budgets bei ihrer Erstanschaffung oft minimalistisch eingerichtet. Ein möglicher Slashdot-Effek ist die Folge, da Skalierbarkeit nur in einem begrenzten Maße möglich ist.

Da es sich um eine datenintensive Anwendung handelt, wird die Netzwerkinfrastruktur stark beansprucht. Die benötigte Bandbreite ist von der Anzahl gleichzeitig zu bedienender Kunden abhängig und kann daher sehr schwanken.

Aufgrund dessen und der in Kapitel 2.2.2 vorgestellten Vorteile, fiel die Entscheidung auf einen Plattformdienst.

4.2 Entwurf

In diesem Kapitel wird auf technische Randbedingungen eingegangen, die für den Lösungsentwurf von Einfluss waren.

1. Umsetzung als Webanwendung

   Der Entwurf einer Webanwendung unterscheidet sich in wesentlich von dem einer Desktopanwendung. Der Hauptunterschied ist, dass Abläufe vom Nutzer über einen Browser als Anfragen (request) gestartet werden und vom Server in Form einer Antwort (response) darauf reagiert wird.

2. Nutzung der Google App Engine

   Die Google App Engine besitzt produktspezifische Eigenschaften, die es zu beachten gibt (siehe Kapitel 3.2.3). Besonders die datenverarbeitenden APIs und deren Einschränkungen spielen eine Rolle.

3. Anwendung von Python

   Aufgrund der Anwendung einer objektorientierten Sprache wurden klassische OO-Paradigmen angewendet.

Darüber hinaus übte der Entwurf mit Modell-View-Controller (MVC)-Pattern Einfluss aus. Eine gesonderte Betrachtung des MVC erfolgt in Abschnitt 4.2.2.

4.2.1 Aufbau

Es wird der Ablauf eines beliebigen Anwendungsfalles dargestellt, um die Charakteristik eines Aufrufs von der App Engine zu verdeutlichen.

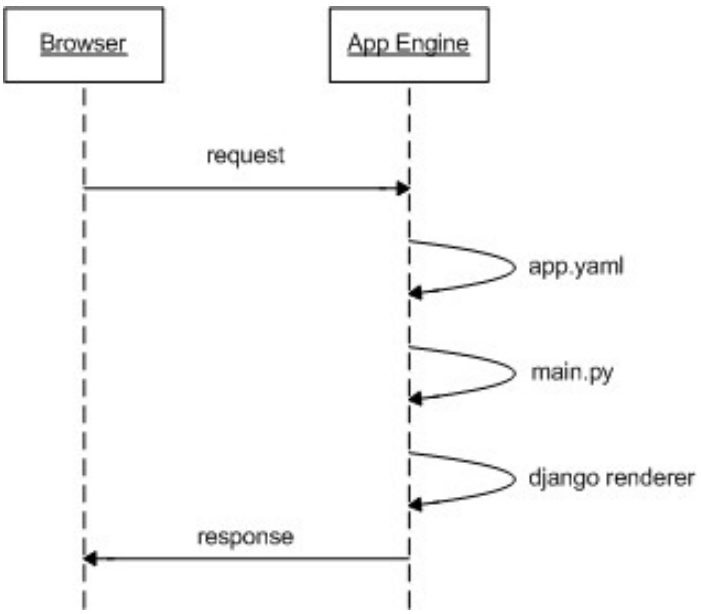

Abbildung 4.3: Charakteristik eines Requests

Erfolgt eine Aktion des Kunden, wird eine Anfrage (request) an die Webanwendung gestellt. Die App Engine überprüft daraufhin anhand der Konfigurationsdatei app.yaml, ob der Ressourcenzugriff berechtigt ist. Liegt eine Berechtigung vor, wird in der Datei main.py überprüft welche Klasse zur Verarbeitung des Aufrufs benötigt wird. Daraufhin wird die entsprechende Klasse instanziiert und der Methodenaufruf durchgeführt. Am Ende des Methodenblocks wird der Renderer aufgerufen. Dieser

generiert eine HTML Seite, welche als Antwort (response), an den Nutzer zurückgesendet wird.

Sollte in der app.yaml keine Berechtigung vorliegen, wird eine Fehlermeldung ausgegeben.

Die Anwendung ist so aufgebaut, dass das Hochladen der Nutzdaten von der Verteilung auf die verwendeten Speicherdienstleister getrennt ist. Durch diese Trennung ergeben sich drei Vorteile:

1. Abarbeitung im Hintergrund

   Alle Methoden müssen durch eine Aktion vom Kunden (request) gestartet werden. Es besteht jedoch die Möglichkeit, Methodenaufrufe jeweils einzeln in eine interne Queue einzureihen und erst dann abarbeiten zu lassen. Dies geschieht im Hintergrund, sodass der Programmlauf zu keinem Zeitpunkt von dem Verteilungsmechanismus aufgehalten wird. Hierfür wird die Task API eingesetzt, die die Kapselung der Programmlogik in einer eigenen Klasse in einer separaten Datei erforderlich macht.

2. Erweiterbarkeit

   Die Logik zur Datenverteilung ist nach Speicherdienstanbietern getrennt in einzelnen Methoden implementiert. Sowohl das Hinzufügen als auch das Entfernen eines Speicherdienstanbieters erfordert nicht mehr, als der Klasse die jeweilige Methode hinzuzufügen bzw. zu entfernen. Das Hauptprogramm bleibt, bis auf den spezifischen Methodenaufruf, unangetastet.

3. Loskopplung von der GUI-Interaktion

   Durch die Abarbeitung der Queue im Hintergrund ergibt sich eine vollständige Entkopplung der Aufgabenabarbeitung von der GUI-Interaktion. Der Kunde wartet somit nicht auf die Abarbeitung der Verteilung.

4.2.2 Model View Controller (MVC)

Bei Anwendung des MVC-Patterns wird der ausführende Code von dem darstellenden Code getrennt.

Daraus folgt eine lose Kopplung, die eine unabhängige Entwicklung der einzelnen Softwarekomponenten ermöglicht. Die Übersichtlichkeit des Quellcodes und die Wiederverwendbarkeit steigen, während das Risiko von unerwünschten Seiteneffekten sinkt.

Die View wurde in HTML Templates verfasst, was zur Folge hat, dass die Programmlogik frei von GUI-spezifischem Programmcode ist.

Der Controller steht bereits in der App Engine zur Verfügung. Er bildet die Zugriffs- und Vermittlungsstelle in Form des Django Renderers (Logik zu GUI) und des Webapp Frameworks (GUI zu Logik).

Das Model enthält die Verarbeitung, Datenhaltung und die Programmlogik. Da die Portabilität auf nicht Cloud-basierte Systemarchitekturen nicht Teil der Anforderungen ist, wurde die Datenhaltung und die Programmlogik nicht anhand eines zusätzlichen Interfaces abstrahiert.

4.2.3 JavaScript

JavaScript ist eine clientseitige Skriptsprache zur Manipulation des DOM Baumes und ermöglicht das Generieren und Nachladen von Inhalten aus dem Browser heraus. Folglich wird ein Teil der Programmlogik zum Client ausgelagert. Dies steigert die Komplexität des Quellcodes und senkt die Überschaubarkeit des Projektes. Diese Effekte sprechen gegen den Einsatz von JavaScript - dennoch wird es häufig eingesetzt, um ein Desktop-ähnliches Verhalten nachzuempfinden.

Durch ausgewählte Programmiermethoden kann unter Umständen ein JavaScript - typisches Verhalten serverseitig implementiert werden. Dementsprechend konnte bei der Entwicklung der entstandenen Anwendung auf JavaScript verzichtet.

Der Blobstore der App Engine erfordert, dass dessen Zugriffsmethoden von vorgegebenen Up- und Download Handlern erben. Zusätzlich sieht der Blobstore unterschiedliche Klassen für das Herauf- bzw. Herunterladen vor. Deshalb muss diese Funktionalität gesondert von der normalen Requestverarbeitung implementiert werden.

JavaScript könnte in diesem Fall in Erwägung gezogen werden. Während der Dateiauswahl eines Kunden könnte mit JavaScript eine serverseitige Upload-Methode aufgerufen werden welche dann einen fallspezifischen Upload Link bereitstellen könnte. Dieser würde während der Dateiauswahl im Hintergrund eingebettet werden.

Diese Funktionalität lässt sich jedoch auch mit einem eingebetteten Frame realisieren. Beim Laden der Upload-Seite wird für den Upload statt eines HTML Form-Tags ein Iframe-Tag geladen. Das Iframe-Tag kann beliebigen Inhalt (beispielsweise Formulare, Bilder, etc.) in den Body einer HTML-Datei einfügen. Als Ziel-URL wird ein Link zu einer Klasse der eigenen Anwendung mitgegeben. Diese Klasse erbt vom blobstore. UploadHandler und gibt einen Upload-spezifischen Link zurück. Dieser wird daraufhin mit einem gesonderten Form-Template an den Renderer übergeben. Der Renderer bettet

den Link als Ziel-URL eines Action-Attributes innerhalb eines Form-Templates ein und sendet ihn an das Iframe zurück. Dem Kunden wird die Upload Seite mit einem für den Upload generierten Upload-Link präsentiert. Zudem hat das Iframe die Eigenschaft, dass die HTML-Seite schon angezeigt werden kann, während der Inhalt des Iframe noch geladen wird. Infolge dessen kann das Upload Feld im Falle ungeplanter Latenzen später angezeigt werden, während die Seite schon dargestellt ist. Auf diese Art und Weise entstehen für den Kunden keine unangenehmen Verzögerungen beim Laden der Webseite.

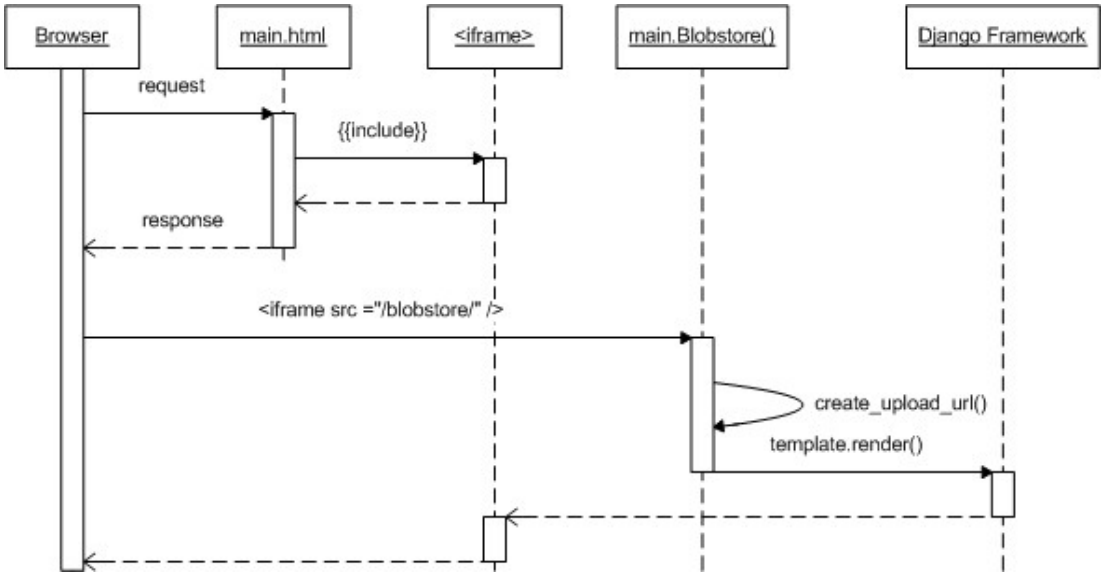

Abbildung 4.4: Ablaufschema Upload-Link Bereitstellung

Mehr zu den Randbedingungen und der Nutzung des Blobstores in Kapitel 4.3.2 und 4.4.3.

4.2.4 Verwendete Bibliotheken

Die Umsetzung der aus der Anforderungsanalyse extrahierten Aufgaben erfordert den Einsatz mehrerer Bibliotheken. Diese ermöglichen die Lauffähigkeit der Lösung auf den Zielplattformen wie sie in Kapitel 3.2 vorgestellt wurden.

In den folgenden Unterkapiteln werden drei Bibliotheken und deren Verwendung innerhalb des vorliegenden Projektes vorgestellt.

- Google WebApp Framework

  Ein für die App Engine entwickeltes Web-Framework von Google, das den Versand und Empfang von HTTP/HTTPS Anfragen ermöglicht [19]. Es stellt die Basis für die Entwicklung auf der App Engine dar. Es lässt sich durch jedes

CGI/WSGI-konforme Framework ersetzen und orientiert sich an dem Django Framework, dessen Rendering Engine übernommen wurde.

In der Anwendungslogik übernimmt das WebApp Framework mehrere Aufgaben. Neben der Vermittlung von Anfragen ist es auch für den Teil des Controllers zuständig, der für die Vermittlung von der GUI zur Logik verantwortlich ist. Der Import der WepApp-Bibliothek wird zur Nutzung vorausgesetzt.

- Django Web Framework Version 1.2

  Das Django Framework ist darauf ausgelegt, sehr schnell Ergebnisse und lauffähige Webanwendungen zu erzielen [20]. Im vorliegenden Projekt findet es in Form der von WebApp genutzten Rendering-Engine nur implizit Verwendung. Der Einsatz des Django Frameworks muss trotzdem erläutert werden, um zu verdeutlichen, welche Technologien für die erfolgreiche Umsetzung des Projektes nötig waren.

  Von der Anwendungslogik können Werte unterschiedlichen Datentyps übergeben und dann in der darzustellenden Seite eingebettet werden. Die genutzte Django Version muss vor der Nutzung deklariert werden. Ein Import ist nicht notwendig.

- Boto Version 2.0b3

  Boto ist eine leichtgewichtige Python Bibliothek zur Nutzung der Amazon Web Services (AWS) [21]. Sie wurde in Python 2.6.5 implementiert und zielt darauf, die Nutzung der AWS so effektiv und unkompliziert wie möglich zu gestalten. Boto unterstützt auch zu den AWS kompatible Dienste wie Google Storage, archive.org und Eucalyptus [22].

  Boto bildet in der Anwendungslogik eine Abstraktionsschicht, welche es ermöglicht, aus einer API heraus gegen mehrere Dienste zu programmieren. Die verwendeten Dienste können im Nachhinein ausgetauscht, entfernt oder neu hinzugefügt werden, ohne dass großer Anpassungsaufwand entsteht. Ähnlich wie bei einem OR-Mapper gewinnt man durch diese interne Kapselung Unabhängigkeit von der darunter liegenden Schicht – in diesem Falle von den Speicherdienstanbietern.

4.3 Implementierungsdetails

Die Kernfunktionalität der Anwendung besteht aus der Verarbeitung und Verteilung von Nutzdaten auf unterschiedliche Speicherdienstanbieter. Weiterhin wird das Anzeigen, Herunterladen und Löschen von bereits heraufgeladenen Daten ermöglicht. Die Anwendung gliedert sich in unterschiedliche Klassen, welche die jeweilige Funktionalität in sich gekapselt bereitstellen.

In Python dürfen laut Konvention mehrere Klassen in einem sogenanntem Modul (Datei) platziert werden. Diese Möglichkeit wurde genutzt und die Anwendung in nur drei Module gegliedert:

- main.py

Der Einstiegspunkt in die Anwendung. Das Modul enthält die Main-Funktion, welche vom WebApp Framework im beim Zugriff auf eine URL gestartet wird und an die entsprechenden Funktionen weiterleitet.

- lib.py

In diesem Modul ist die Anwendungslogik umgesetzt. Es ist in Klassen und deren POST / GET Methoden strukturiert.

- transfer.py

Dieses Modul enthält den Mechanismus für die Datenverteilung an die externen Speicherdienstanbieter.

Zur Ausführung der Anwendung sind zusätzlich folgende Komponenten erforderlich:

- HTML Templates

Zur Förderung der Übersichtlichkeit und Wiederverwendbarkeit wurden zur Gestaltung keine reinen HTML Dateien verwendet. Stattdessen wurden HTML Templates verfasst, die gesammelt in einem Unterverzeichnis vorliegen.

- CSS Layout-Dokument

In einer style.css Datei wird die Darstellung der HTML-Tags zentral festgelegt. Dies erleichtert die Anpassung des Designs und hält die in HTML festgelegte Seitenstruktur frei von Darstellungselementen.

- app.yaml

Dies ist die Konfigurationsdatei der App Engine, in der Eigenschaften der Anwendung und Berechtigungen für Unterverzeichnisse vermerkt werden müssen. Ohne diese Datei kann die Anwendung nicht ausgeführt werden.

- boto.cfg

Die boto.cfg stellt eine optionale Konfigurationsdatei für Boto dar. Wird jedoch von Google Storage für die Zugangsdaten vorausgesetzt.

- Bilder

Verwendete Grafiken werden in einem Unterverzeichnis gesondert aufbewahrt.

Bei der Entwicklung mit dem WebApp Framework muss jede Klasse, die HTTP-Anfragen verarbeitet, von der Klasse webapp.RequestHandler erben. Weiterhin muss sie entweder eine get oder eine post Methode implementieren, abhängig von der HTTP-Anfrage. Somit enthält jede Klasse maximal zwei Methoden, welche von außen aufgerufen werden können und optional weitere, nur aus der eigenen Klasse aufrufbare Methoden. In den meisten Fällen folgt auf das Ende einer Methode ein Redirect, ein Aufruf des Renderers oder eine direkte Ausgabe von HTML Code.

In Abbildung 9 sind die Zusammenhänge zwischen den einzelnen Klassen an Hand eines Klassendiagrammes dargestellt.

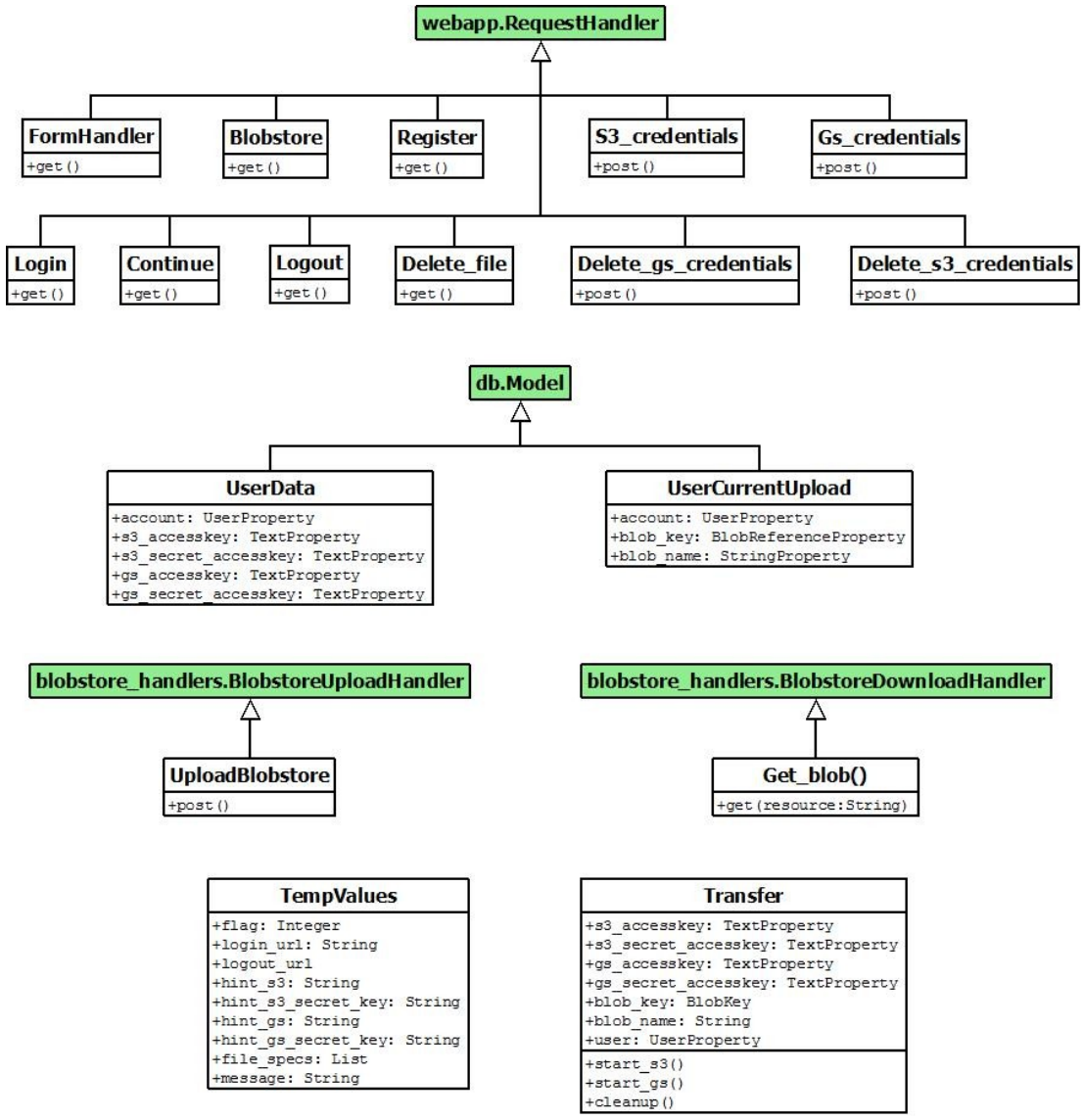

Abbildung 4.5: Klassendiagramm von MultiBackup

4.3.1 Tag-System

Für das individuelle Zusammenfügen der Templates wurde ein Tag-System entwickelt. Dieser Abschnitt erläutert die Grundlagen dessen.

Die Anwendung besteht aus Anwendersicht aus drei Webseiten:

- der Startseite
- der Konfigurationsseite
- der Hauptseite

Intern wird jedoch nur eine HTML Seite genutzt, die der Renderer bei jedem Aufruf unterschiedlich zusammensetzt. Diese trägt den Namen „main.html“ und besteht aus drei Bereichen, die als Tabellenzeilen umgesetzt sind. In jedem Bereich ist mit der Django-eigenen Templatesprache eine Abfragelogik implementiert. Diese prüft die an den Renderer übergebenen Werte. Einer der Werte gibt Aufschluss darüber, welche der drei Seiten zusammengestellt werden soll. Während dieses Rendering-Prozesses werden auch die restlichen Übergabewerte ausgewertet und in der HTML Seite platziert.

Die Anwendung des Tag-Systems bietet folgende Vorteile:

- Flexibilität bei der Zusammensetzung der Seiteninhalte. Die Bereiche der Seite können bei Bedarf ausgetauscht werden. Ein Nachladen oder verzögertes Laden, wie es bei dem Iframe mit dem Upload-Formular geschieht, wird damit ermöglicht. Nimmt der Nutzer auf einer der drei Seiten Änderungen vor, so muss er ausschließlich auf das Nachladen des geänderten Bereiches warten, nicht auf das Nachladen der ganzen Seite.
- Es wird zentral festgelegt, was unter welcher Bedingung angezeigt werden soll. Änderungen werden zentral vorgenommen, ein Suchen in unterschiedlichen HTML Dateien erübrigt sich, der Quellcode der View ist gut wartbar. So werden unangenehme Seiteneffekte beim Vornehmen von Änderungen verhindert.
- Der Entwickler der Anwendungslogik bleibt von Darstellungsdetails weitestgehend unangetastet. Er übergibt lediglich den Tag und die Werte, die er auf der entsprechenden Seite dargestellt haben möchte.

4.3.2 Programmstruktur

In diesem Unterkapitel wird detailliert auf die einzelnen Klassen und deren Funktionalität eingegangen.

Die Anwendung kann in folgenden Bestandteilen beschrieben werden:

- main()

  Diese statische Methode ist der Einstiegspunkt der Webanwendung. Die Python Laufzeitumgebung der App Engine ruft diese zum Starten der Anwendung auf. In ihr findet die Umsetzung von URL-Endungen in entsprechende Methodenaufrufe statt.

- Formhandler(webapp.RequestHandler)

  Die Formhandler Klasse implementiert die Hauptfunktionen der Anwendung. Sie erbt vom RequestHandler und enthält eine get-Methode. Wird die Anwendungs-URL ohne eine gültige Endung aufgerufen, welche auf eine Methode deutet, entscheidet der FormHandler, was geschehen soll. Zunächst wird geprüft, ob der Kunde bereits eingeloggt ist. Ist dies nicht der Fall, wird ihm die Startseite der Anwendung angezeigt. Handelt es sich dabei um die Hauptseite, werden die bereits hochgeladenen Dateien abgerufen – unter der Voraussetzung, dass bereits Zugangsdaten und Dateien vorliegen.

  Sollte die Klasse intern aufgerufen werden, wird auf eventuell vorhandene Tag-Werte geprüft und diese verarbeitet.

- Blobstore(webapp.RequestHandler)

  Auf diese Klasse wurde bereits in Kapitel 4.2.3 eingegangen. Diese wird Iframe als Ziel-URL referenziert. Sie generiert eine individuelle Upload-URL für den Blobstore und übergibt diese zusammen mit einem Form-Template an den Renderer. Das Ergebnis wird an den Browser gesendet und dort in die Seite eingebettet. Die Klasse erbt vom RequestHandler und enthält eine get-Methode.

- UploadBlobstore(blobstore_handlers.BlobstoreUploadHandler)

  In dieser Klasse wird über die zuvor generierte URL das in den Blobstore hochgeladene Blob verarbeitet. Um auf den Blobstore Zugriff zu haben, muss die Klasse vom BlobstoreUploadHandler erben. In einer post-Methode wird zunächst das BlobInfo-Objekt der hochgeladenen Datei abgefragt, um dieses daraufhin als Parameter an die deferred API zu übergeben.

Dies geschieht folgendermaßen: Ein Transferobjekt wird erzeugt, dessen Konstruktor zuvor abgerufene Speicherdienst-Zugangsdaten und eine Referenz auf das BlobInfo Objekt erhält. Auf diesem Objekt werden dann die einzelnen, zuvor implementierten Upload-Methoden der Speicherdienste aufgerufen. Letztlich wird eine Cleanup-Methode aufgerufen, welche das nicht mehr benötigte Blob aus dem Blobstore entfernt. Diese erhält als Übergabeparameter eine Zeitspanne, nach deren Ablauf die Ausführung der Methode beginnt.

Durch den Einsatz der deferred API werden die auf dem Transferobjekt aufgerufenen Methoden in eine Task Queue eingereiht. Dadurch wird der Ablauf der Methode nicht durch den Datentransfer zu den Speicherdiensten aufgehalten. Die deferred API ist Teil der umfangreichen Task API, die für Hintergrundarbeiten konzipiert wurde.

In einem begrenzten Maß wird auf diese Weise auf der App Engine ein Multi-Threading-Verhalten nachempfunden, da der Programmlauf nicht angehalten wird, um auf Rückgabewerte bzw. ein Abschließen der aufgerufenen Methode zu warten.

- Get_blob(blobstore_handlers.BlobstoreDownloadHandler)

  Diese Klasse erhält über die URL einen BlobKey-String und sendet dessen Blob als Antwort zurück. Um diese Funktionalität umzusetzen, muss sie vom BlobstoreDownloadHandler erben. Sie enthält eine get-Methode, die einen Übergabeparameter erwartet.

- UserData(db.Model)

  Dies ist eine Datenbank-Modell-Klasse, hierfür muss sie von db.Model erben. In ihr werden die Zugangsdaten der Kunden zu den einzelnen Speicherdienstanbietern gespeichert.

- UserCurrentUpload(db.Model)

  Dies ist eine Datenbank-Modell Klasse, welche ebenfalls von db.Model erben muss. Sie enthält Informationen zum aktuellem Upload.

- TempValues()

  TempValues dient als Hilfklasse zur Speicherung temporärer Werte. Sie wird bei jedem Renderer-Aufruf mit übergeben, sodass dieser die für die Darstellung benötigten Werte auslesen und in die HTML Seite einbetten kann.

- Transfer()

  Diese Klasse enthält drei Methoden:

  - start_s3()
  - start_gs()
  - cleanup()

  Damit das Transferobjekt mit der deferred API verwendet werden kann, muss es in einer einzelnen Datei verfasst und serialisierbar sein. Die Größe darf nach der Serialisierung 1MB nicht überschreiten. Je Anbieter ist eine Methode zur Abwicklung des Uploads vorgesehen. Die Vorgehensweise ist bei beiden verwendeten Anbietern gleich.

  Der Ablauf gestaltet sich wie folgt: Zuerst wird ein BlobReader Objekt erzeugt, welches einen BlobKey als Übergabeparameter erhält. Auf diesem Objekt wird ein Reader geöffnet, der den Blob byteweise ausliest. Dieser Reader wird einem StringIO Objekt übergeben, welches ein Schreiben in eine Datei simuliert. Das StringIO Objekt ist in Python ein dateiähnliches Objekt, welches wie eine normale Datei geöffnet, gelesen und geschrieben werden kann. Es wird eingesetzt, da Schreib-Operationen auf der App Engine nicht erlaubt sind (siehe Kapitel 3.2.3). Anschließend wird eine Verbindung zum Speicherdienstanbieter hergestellt und auf dieser Verbindung ein Bucket erzeugt. Nachfolgend wird ein Key mit dem gerade erstelltem Bucket als Übergabeparamter instanziiert. Auf dem Key Objekt wird der Methode set_contents_from_file() das StringIO Objekt übergeben und somit an den Anbieter übermittelt.

  Zusätzlich wurde eine alternative Vorgehensweise entwickelt, um die temporär gespeicherte Datei aus dem Blobstore abzurufen. Dabei wird ein URL Fetch auf die eigene Anwendung durchgeführt. Mit der zuvor beschriebenen Hilfsklasse Get_blob() wird das Blob als response zurückgesendet. Dieses lässt sich mittels URL Fetch als Rückgabewert abrufen und an die Speicherdienstanbieter weiterreichen. Da das Blob als String-Objekt vorliegt, muss die set_contents_from_string() Methode verwendet werden. Diese Methode wird von Google Storage nicht unterstützt. Daher wurde der Blob-Abruf über den BlobReader realisiert, sodass die Vorgehensweise für beide Speicherdienstanbieter gleich gestaltet werden kann. Bei der URL Fetch Vorgehensweise besteht das Risiko eines Fetch-Timeouts, weshalb die Blobreader Variante vorzuziehen ist.

Die zuvor erfolgte Aufstellung stellt ausschließlich die für den Programmablauf wesentichen Klassen dar. Es wurden weitere Hilfsklassen implementiert, die jedoch entweder für das Verständnis der Anwendung unbedeutend oder durch den Namen

selbsterklärend für die enthaltene Logik sind. Eine vollständige Aufstellung aller Klassen ist Abbildung 9 zu entnehmen.

4.3.3 Verwendete App Engine APIs

Die Google App Engine wurde als Plattformdienst bereits in Kapitel 3.2.3 vorgestellt. In diesem Unterkapitel wird auf die verwendeten APIs eingegangen, die für die Anwendungsentwicklung von Relevanz waren.

Neben der Kenntnis der APIs haben auch deren Limitierungen einen hohen Stellenwert. Dahingehend werden vor allem die Restriktionen und die daraus resultierende Art der Verwendung vorgestellt.

- Deferred API

  Die deferred API ist essenzieller Bestandteil der Anwendung, mit deren Hilfe die Verteilung transparent im Hintergrund bewerkstelligt wird. Sie ist Teil der Task API und für schnelle und unkomplizierte Hintergrundarbeiten konzipiert, bei denen nicht der komplette Umfang der Task API notwendig ist [23].

  Es gibt zwei Motive für den Einsatz der deferred API. Einerseits um den Anwendungsfluss nicht durch das Verteilen zu blockieren. Andererseits, weil die Antwortdauer auf requests durch die App Engine auf 30 Sekunden beschränkt ist. Eine Methode die auf eine Browseranfrage antwortet, kann maximal 30 Sekunden ausgeführt werden. Ist die Abarbeitung der Programmlogik dann nicht abgeschlossen, wird ihr Prozess terminiert. Sowohl die deferred- als auch die Task API geben dagegen kein Laufzeit-Limit vor. Aus diesem Grund eignen sie sich hervorragend zum Verteilen von Daten – ein Vorgang, der bei größeren Dateien viel Zeit in Anspruch nehmen kann.

  Die Verwendung der deferred API gestaltet sich unkompliziert. Es wird eine Instanz der gewünschten Klasse erzeugt und deren Methoden einem statischen deferred-Methodenaufruf als Parameter übergeben. Diese werden dann in einer Anwendungseigenen Queue eingereiht und abgearbeitet, sofern es die Ressourcenverfügbarkeit zulässt.

  Auch die Nutzung der deferred-API gibt besondere Rahmenbedingungen vor. Klasseninstanzen, welche über diese in eine Abarbeitungs-Queue übergeben werden, müssen gesondert und in einer eigenen Datei implementiert werden. Außerdem müssen sie serialisierbar und in serialisiertem Zustand nicht größer als 1MB sein.

- BlobStore API

  Wie bereits in Kapitel 4.1.3 dargelegt wurde, werden die Daten von der Anwendung vor der Verteilung an die Speicherdienstanbieter zwischengespeichert. Dies geschieht über den Blobstore Dienst der App Engine [24]. Der Blobstore wurde zum Speichern von Binärdaten entworfen und ist somit als Ergänzung des Datastores zu sehen. Während BlobProperty-Objekte im Datastore nur bis zu 1MB groß sein dürfen, kann man im Blobstore Dateien bis zu einer Größe von 2GB speichern. Voraussetzung für die Nutzung des Blobstores ist die Freischaltung der Abrechungsfunktion.

  Der Upload zum Blobstore erfolgt über spezielle URLs die individuell für jeden Upload erstellt werden müssen. Der statischen Blobstore-Methode create_upload_url() wird zunächst eine URL-Endung der eigenen Anwendung übergeben, die diese Klasse referenziert die vom BlobstoreUploadHandler erbt. Diese spezielle Klasse wird danach den Upload entgegennehmen und entsprechend der darin implementierten Logik bearbeiten. Als Rückgabeparameter erhält man eine Upload-URL zur einmaligen Verwendung. Diese kann in das Action-Attribut einer HML Form eingebettet werden.

  Der Download vom Blobstore erfolgt über eine Klasse, die vom BlobstoreDownloadHandler erben muss. Dazu muss deren post- oder get-Methode einen zusätzlichen Parameter ermöglichen. Dieser ist der Key eines BlobInfo-Objektes in String-Darstellung. Die Übergabe erfolgt, indem der String an das Ende der URL angehangen wird. Diese URL kann dann z.B. als Download-Link im Browser angezeigt werden. Bei deren Aufruf wird ein GET-Befehl gesendet, welcher von der App Engine an die entsprechende Methode weitergeleitet wird. Diese liest den Key aus der URL aus und durchsucht den Blobstore nach diesem. Wird das Blob gefunden, kann es mittels send_blob() Methode als Antwort an den Browser gesendet werden.

  Soll aus der eigenen Anwendung heraus auf ein Blob im Blobstore zugegriffen werden, so besteht keine Möglichkeit, dieses als Ganzes abzurufen. Die App Engine stellt für solche Fälle ein BlobReader-Interface bereit. Über dieses kann ein Stream auf das Blob geöffnet werden und dieser dann in jeweils 1MB großen Teilen ausgelesen werden. Die Sicherheitsvorschrift, dass auf der App Engine keine Datei zum Schreiben geöffnet werden darf (siehe Kapitel 3.2.3), kann zu Komplikationen führen. Diesen Umstand kann man mit einem StringIO Objekt umgehen. Die konkrete Verwendung des Blobreaders wurde bereits in Kapitel 4.3.2 besprochen.

  Zusammenfassend gestaltet sich der Gebrauch des Blobstores aufgrund einiger Restriktionen und Sicherheitsmechanismen verhältnismäßig umständlich. Zur Überprüfung der Gebrauchstauglichkeit beider Vorgehensweisen wurden beide

Zugriffsmethoden implementiert. Der Zugriff über den BlobReader stellte sich als vorteilhafter heraus. Da ein Zugriff über eine URL über einen URL-Fetch-Aufruf erfolgen muss und dies einen Timeout herbeiführen kann.

- URL Fetch API

  Über die URL Fetch API können Anwendungen der App Engine mit anderen Servern kommunizieren [25]. Es können per HTTP/HTTPS Anfragen gesendet und empfangen werden. Sockets andere Zugriffsmethoden auf andere Server sind aufgrund der Sandbox-Restriktionen nicht möglich.

  Da die Blobs aus dem Blobstore über eine URL bereitgestellt werden, wurde u.a. der Ansatz des Abrufs über einen URL Fetch verfolgt. Das während der Entwicklungszeit bestehende Limit von 32MB für URL-Abrufe erschien ausreichend. Während der Entwicklung der Anwendung stellte sich jedoch heraus, dass trotz der Tatsache, dass Quelle und Ziel jeweils innerhalb der Google Infrastruktur lagen, Timeouts auftraten. Zwar kann man die Wartezeit vor einem Timeout selbst bestimmten. Jedoch zeigte sich, dass der Zugriff mit starken Verzögerungen (bis zu 30 Sekunden) verbunden war. Aus diesem Grund wurde zum Auslesen der Blobs die alternative Vorgehensweise mittels Blobreader bevorzugt.

  Anfragen über URL Fetch sind auf 1 MB Größe beschränkt. Alle sonstigen Bibliotheken zur Serverkommunikation, beispielsweise urllib, urllib2 oder httplib werden in App Engine mittels URL Fetch realisiert und erzeugen so Anfragen, die unter das 1 MB Limit fallen. Boto verwendet httplib und kann deshalb nur 1MB große Dateien an Speicherdienstanbieter übermitteln. Dies stellte sich erst während der Implementierung heraus. Das Verteilungskonzept, wie es in Kapitel 4.1.2 beschrieben wurde, wurde trotzdem aus folgenden Gründen nicht geändert:

  - Die AppEngine-Limitierungen werden in unregelmäßigen Abständen gelockert.
  - Die Anwendung kann auch innerhalb einer Private Cloud verwendet werden. Dort spielen Einschränkungen bezüglich der App Engine keine Rolle.
  - Für die gewünschte Art von Anwendung steht das indirekte Verteilen ohne Alternative da.

- DataStore API

  Der Datastore dient zur langfristigen Speicherung von Daten [26]. Alle bereitgestellten Datentypen unterliegen einem 1MB Größenlimit. Die API wurde

benutzt, um zwei Datenbankmodelle zu persistieren (siehe Kapitel 4.3.2). Eines ist dauerhaft auf die Speicherung von Nutzerdaten ausgelegt (UserData), das Andere temporär um Upload Details während der Verteilung zwischen zu speichern. (UserCurrentUpload).

- Logging Modul

  Die App Engine nutzt für Fehlermeldungen eine modifizierte Version des Python Logging-Moduls [27]. An Hand diesem können fünf unterschiedlich kritische Arten von Log-Events ausgegeben werden.

  Namentlich sind dies:

  - Debug
  - Info
  - Warning
  - Error
  - Critical

  Über die statische logging Klasse kann man die gewünschte Log-Art in Form einer gleichnamigen Methode aufrufen. Diesem Statement wird zusätzlich eine selbst definierte Nachricht übergeben. So lässt sich der Anwendungsablauf dokumentieren und sehr gut nachvollziehen. Die generierten Meldungen sind entweder in der Logdatei des Entwicklungsservers oder – für den Fall, dass die Anwendung schon online ist, im Dashboard unter dem Reiter mit der Bezeichnung Logs einsehbar. Sollte man eine IDE wie z.B. Aptana [28] benutzen, so werden sie in der Konsole mit ausgegeben

4.3.4 Werkzeuge

Während der Anwendungsentwicklung wurden Entwicklungswerkzeuge eingesetzt, die zusammen mit der angewendeten Programmiersprache in diesem Abschnitt vorgestellt werden.

- Python 2.5.4

  Da die Python-Laufzeitumgebung der App Engine genutzt wurde, musste Python in der Version 2.5.2 verwendet werden [29]. Alle aktuelleren Python-Versionen werden von der App Engine nicht unterstützt [30]. Aufgrund des sehr hohen Reifegrads von Python 2.5.2 ist dies allerdings für die Entwicklung mit keinerlei Problemen oder Einschränkungen verbunden.

- App Engine Development Server

  Google stellt über sein App Engine SDK einen lokalen Entwicklungsserver bereit [31]. Dieser simuliert Googles App Engine Infrastruktur, sodass Anwendungen auf dem Entwicklungs-PC ausgeführt und getestet werden können. Änderungen im Quellcode werden fortwährend erkannt und während des Betriebs eingebunden. Ein Neustarten des Entwicklungsservers ist somit nicht notwendig.

  Django ist in die Laufzeitumgebung der Python App Engine integriert. Daher muss es nicht gesondert installiert werden. Da die Versionen 0.96 und 1.2 unterstützt werden, muss darauf geachtet werden die genutzte Version in der Anwendung gesondert auszuzeichnen.

- Aptana 3

  Aptana ist eine auf Eclipse basierende IDE [28, 32]. Sie ist auf die Entwicklung von Webanwendungen ausgerichtet. Sie enthält das PyDev Plugin und ist damit hervorragend für ein Python Webprojekt geeignet. Python- und HTML- Code Completion, sowie ein interaktiver Debugger erleichtern die Entwicklung. Besonders gelungen ist die Nutzung des App Engine Entwicklungsservers, der innerhalb der IDE gestartet werden kann. Während der Laufzeit sind dadurch alle Serverstatus- und Lognachrichten über die Konsole übersichtlich nachzuvollziehen.

4.4 Vorstellung der Anwendung

Die aus der vorliegenden Arbeit entstandene Anwendung wird im folgenden Kapitel vorgestellt.

Die Anwendung trägt die Bezeichnung MultiBackup und ist unter der URL: http://multi-backup.appspot.com zu erreichen.

Die Anwendung ist in drei Bereiche unterteilt, die deutlich voneinander abgegrenzt sind. Die Motivation der Aufteilung in diese drei Teilbereiche wurde in Kapitel 4.3.1 erläutert. Beim graphischen Entwurf wurde Wert darauf gelegt, dass die Bereiche sich jeweils nach einem bestimmten semantischen Bereich voneinander abheben.

- Der obere Bereich stellt den Infobereich dar. In diesem werden dem Nutzer seitenspezifische Informationen angezeigt. Dies sind meist Hinweise zur Verwendung der aktuellen Seite, Fehlermeldungen oder Buttons zur Orientierung innerhalb der Anwendung.

- Der mittlere Bereich wurde als Aktionsbereich definiert. Hier findet der Nutzer Schaltflächen, die zur eigentlichen Verwendung der der Anwendung dienen.

- Der untere Bereich der Seite ist der Darstellungsbereich. Hier werden seitenspezifische Daten angezeigt, welche die Reaktion auf Nutzeraktionen anzeigen.

Beim Erscheinungsbild wurde darauf geachtet, dass die Bereiche der Benutzerführung dienlich sind. Durch die konsequente Einhaltung der Bereiche soll eine unterbewusste Navigationshilfe für den Nutzer entstehen. Der Anwender muss nicht auf jeder Seite erneut umdenken, da die Elemente wie zu erwarten platziert sind.

Gleichermaßen wurde bei der Gestaltung der Bewegungsradius der Maus beachtet. Die komplette Anwendung besitzt ein fixes Layout, welches auf dem Bildschirm zentriert ist. Desweiteren wurden innerhalb der Anwendung alle Elemente zentral platziert. Diese Maßnahmen minimieren die Abstände zwischen den Elementen, die vom Nutzer bedient werden, sodass dieser während eines Arbeitsschrittes keine langen Mausbewegungen ausführen muss. Durch die Reduktion der notwendigen Mausbewegungen sollen die Arbeitsabläufe flüssiger und angenehmer gestaltet werden. Hinsichtlich der Durchdringung der Internetnutzung mittels Netbooks und anderer mobiler Geräte müssen Anwendungen für alle Nutzer gut bedienbar entworfen werden. Bei diesen steht meist entweder ein kleines Display zur Verfügung oder ein Mousepad, dass die Navigation über den gesamten Bildschirmbereich schwierig macht.

Nach dem Aufruf der Anwendung erscheint die Hauptseite. Dort hat der Nutzer die Möglichkeit, sich entweder direkt einzuloggen oder – falls er die Anwendung erstmalig

aufruft, seine Zugangsdaten zu registrieren. Sollte er sich nicht gleich anmelden wollen, so wird ihm die Möglichkeit angeboten sich über die verwendeten Dienste zu informieren. Im unteren Teil sind Buttons in Form von Grafiken hinterlegt, welche zu den Webseiten der Speicherdienstanbieter führen.

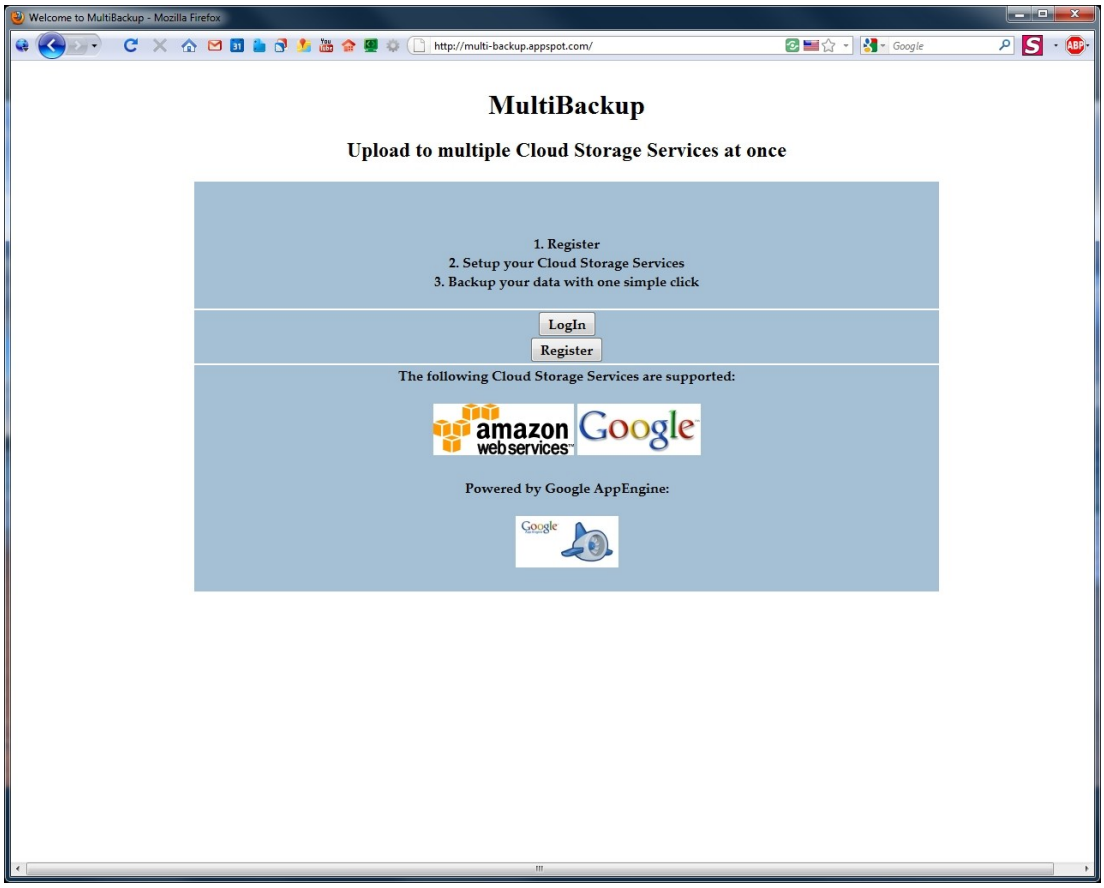

Abbildung 4.6: Einstiegs-Seite von MultiBackup

Gleich, welche Aktion der Nutzer auslöst, wird er zuerst auf die Google Accounts Login-Seite umgeleitet. Nach der Anmeldung wird er auf die Hauptseite umgeleitet oder, falls er den Register – Button bedient hat, zur Konfigurationsseite. War der Kunde gleichzeitig bereits bei einem anderen Google-Dienst angemeldet, wird die Anmeldung übersprungen und er gelangt automatisch auf die gewünschte Seite.

Im Falle einer Erstanwendung der Software durch einen Kunden, welcher sich registrieren möchte, wird die Konfigurationsseite der Anwendung angezeigt. Auf dieser wird oben ein kurzer Infotext eingeblendet und darüber hinaus zwei Weiterleitungs-Buttons. An dieser Stelle besteht für den Kunden die Möglichkeit, sich nach Eingabe seiner Daten auf die Hauptseite weiterleiten zu lassen oder sich aus dem System auszuloggen.

Die Eingabe der Zugangsdaten zu den Speicherdienstanbietern erfolgt gesondert und muss mit einem Klick auf die Schaltfläche „Save Credentials“ bestätigt werden.

Daraufhin werden die Daten übermittelt und der betreffende Bereich neu geladen. Dies geschieht mit dem zuvor eingegebenen Schlüssel und einer symbolischen Darstellung des geheimen Schlüssels. Da nur dieser Bereich neu geladen wird, bleibt die gesamte Seite weiterhin navigierbar. So kann der Kunde direkt mit der Eingabe der restlichen Zugangsdaten fortfahren. Dieses Verhalten ist analog zum Anwendungsverhalten beim Löschen von Zugangsdaten.

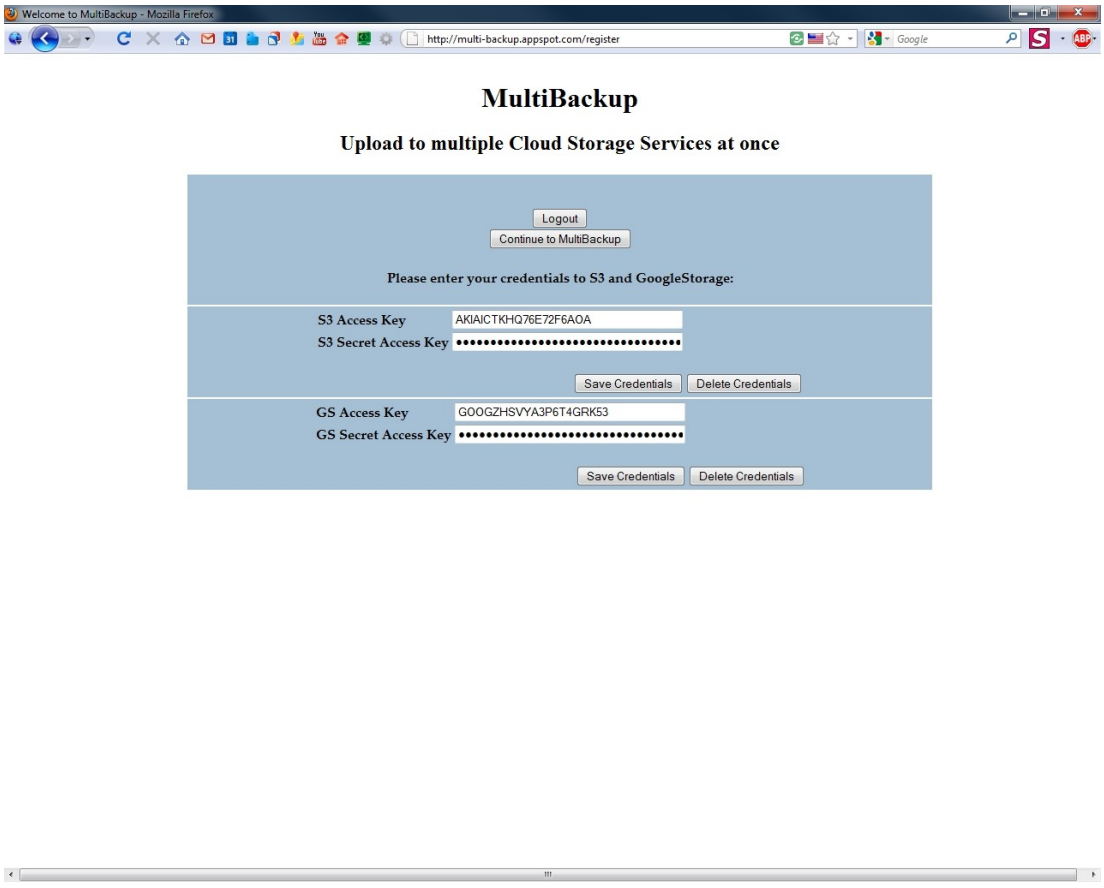

Abbildung 4.7: Konfigurations-Seite von MultiBackup

Nach der Eingabe der Zugangsdaten bzw. nach dem Login eines bereits registrierten Kunden gelangt man auf die Hauptseite der Anwendung.

Nach dem bereits eingeführten Schema wird im oberen Bereich ein kurzer Infotext eingeblendet, sowie zwei Weiterleitungsbuttons. Der Kunde hat nun die Möglichkeit, sich auf die Konfigurationsseite weiterleiten zu lassen oder sich auszuloggen. Hat der Kunde ungültige Zugangsdaten eingegeben, so erscheint an dieser Stelle eine rot eingefärbte Warnmeldung. Diese gibt dem Kunden Aufschluss darüber, dass keine Datenoperationen möglich sind und er die Zugangsdaten überprüfen sollte.

Im mittleren Teil der Hauptseite befindet sich ein Dateiauswahlfeld und ein Button (Submit) zum Starten der Dateiübermittlung. Nachdem eine Datei ausgewählt und der Submit Button gedrückt wurde, wird die Datei übertragen und der mittlere Bereich der

Anwendung neu geladen. Der Rest der Anwendung bleibt navigierbar. Wurde der mittlere Bereich neu geladen, lässt sich die nächste Datei auswählen und hochladen.

Im unteren Bereich der Seite wird eine Liste bereits hochgeladener Dateien eingeblendet. Jede verfügt über einen gesonderten Lösch- und Download-Link. Diese sind hinter einer entsprechenden Grafik eingebettet, sodass der Kunde aufgrund des Wiedererkennungswerts intuitiv handeln kann. Entsprechend der Nutzeraktion wird nach einem Klick die passende Datei heruntergeladen oder gelöscht und die Zeile aus der Auflistung entfernt.

Die Anwendung wurde auf Kompatibilität mit den Browsern Mozilla Firefox 3.6, Google Chrome 10 und Apple Safari 5 überprüft. Sie stellt einen technisch voll funktionsfähigen Prototypen dar. Dieser soll als Machbarkeitsbeweis der Realisierung eines redundanten Speicherdienstes auf der Google App Engine dienen. Erweiterungen, Änderungen und weitere Funktionalitäten sind denkbar und werden in Kapitel 6.2 besprochen.

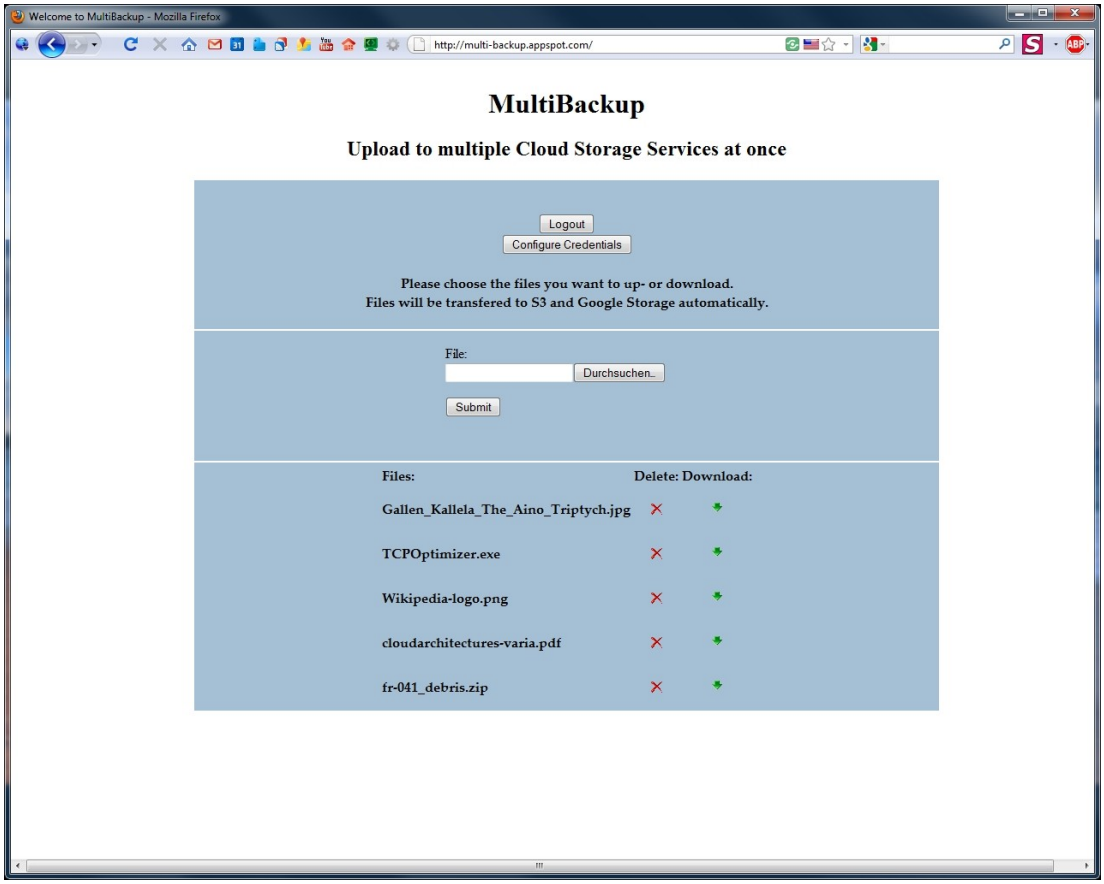

Abbildung 4.8: Haupt-Seite von MultiBackup

Kapitel 5

5. Vergleich mit existierenden Speicherlösungen

Im folgenden Kapitel wird auf Speicherdienste und deren Eigenschaften eingegangen, die sich bereits auf dem Markt befinden. Betrachtungen zur Motivation und zur Wirtschaftlichkeit werden an dieser Stelle nicht vorgenommen. Vorrangig werden technische Aspekte sowie Alleinstellungsmerkmale erläutert. Anschließend werden Vor- und Nachteile im Gesamtkontext herausgearbeitet.

5.1 Marktanalyse

- Dropbox

  Dropbox ist ein Backup- und Synchronisierungsdienst [33]. Dessen Funktionalität wird über eine Webanwendung sowie über Clients für unterschiedliche Betriebssysteme bereitgestellt. Der Dienst ist auf die Nutzung durch Privatkunden ausgelegt.

  Es werden 2GB Speicherplatz kostenlos bereit gestellt. Durch das Werben von Kunden ist es möglich, bis zu 8GB kostenlosen Speicher zur Verfügung gestellt zu bekommen. Wünscht ein Kunde mehr Kapazität, so ist es möglich, bis zu 100GB Speicherplatz dazuzukaufen. Dropbox nutzt den Dienst Amazon S3 zur Speicherung und AES256 zur Verschlüsselung der Daten [10].

  Die Stärke von Dropbox liegt in der Synchronisation der Daten. Bei der Installation des Clients wird ein Dropbox Ordner eingerichtet. Alle Dateien, die darin gespeichert werden, werden automatisch erkannt und transparent für den Nutzer zu Amazon S3 übertragen. Dies macht die Software für jeden - auch für technisch nicht versierte Nutzer - sehr leicht nutzbar.

  Darüber hinaus ist es möglich, Ordner mit anderen Dropbox Kunden zu teilen. Diese erhalten in diesem Fall eine Einladungsmail und können sich falls noch kein Dropbox Benutzerkonto vorhanden ist registrieren. Der Versender und der Empfänger der Einladung erhalten dafür jeweils 250MB mehr Kontingent. Dieser Mechanismus führte nach der Markteinführung von Dropbox zu einer schnellen Verbreitung.

Fazit: Dropbox ist ein für Privatkunden sehr gut geeigneter Synchronisations- und Backupdienst. Die Tarife unterscheiden sich nur in der bereitgestellten Speichergröße. „Easy-to-use“ und „Collaboration“ steht stark im Vordergrund. Primär als Synchronisationsdienst für mehrere Geräte gedacht, sind sämtliche Daten auch ohne Netzanbindung verfügbar.

- Strato HiDrive

  HiDrive ist ein Speicherdienst der Firma Strato AG [34]. Er zielt auf Privat- sowie auf Geschäftskunden ab. Kunden können daher zwischen den Tarifen HiDrive Media und HiDrive Pro wählen, die in verschiedenen Speicherplatzvolumina verfügbar sind.

  Es wird ein Online-Dateimanager bereitgestellt. Einen Desktop-Client gibt es nicht, jedoch kann man Programme von Drittanbietern nutzen, da unterschiedliche Protokolle wie z.B. FTP/SFTP, WebDav, Rsync unterstützt werden. Hervorzuheben ist die Unterstützung von SMB/CIFS, die es ermöglicht HiDrive unter Windows und unter Linux als Netzlaufwerk einzubinden.

  Neben dem verfügbaren Speicherplatz unterscheiden sich die Tarife u.a. auch in der maximalen Anzahl der verfügbaren Nutzerkonten und der maximalen Anzahl der Downloads per Freigabelink. Einerseits kann man eigene Daten per Downloadlink bereit stellen. Andererseits kann man Dritten eigene eingeschränkte Konten einrichten. Daten werden demzufolge über gegenseitiges Herauf- und Herunterladen zur Verfügung gestellt. Eine Synchronisationsfunktion von gemeinsam bearbeiteten oder zur Verfügung stehenden Dateien gibt es nicht.

  Abhängig vom gewählten Tarif und dessen Speichervolumen wird entweder ein kostenloses Streaming-Gerät oder ein NAS zu einem vergünstigten Preis angeboten. Das Streaming-Gerät dient zum Abspielen von online gespeicherten Dateien auf Fernsehgeräten. Der Einsatz eines NAS ermöglicht einen von der Internetverbindung unabhängigen Zugriff, da vor Ort gespeichert wird. Durch die im NAS integrierte Synchronisationsfunktion wird der Inhalt intervallgesteuert mit HiDrive abgeglichen.

  Fazit: HiDrive bietet unterschiedliche Tarife, die sich sowohl im Preis als auch in der Leistung unterscheiden. So werden möglichste viele Anwendertypen abgedeckt. Die Attraktivität wird durch Kombination mit abgestimmter Hardware gesteigert. Der Dienst ist primär als Online-Festplatte gedacht. Der Zugriff auf die Daten funktioniert daher nur bei aktiver Internetverbindung (Ausnahme: NAS Kombination).

- Mozy

  Mozy ist eine Tochterfirma der EMC Corporation [35]. EMC ist ein Anbieter von Hardware-Speicherlösungen und Lösungen rund um die Datenhaltungs- und Informationsinfrastruktur [36]. Mozy adressiert mit den Tarifen MozyHome und MozyPro, sowohl Privat- als auch Business-Kunden. Eine 2GB Version wird für Privatanwender kostenlos zur Verfügung gestellt.

  Mozy bietet einen lokalen Client an, der Daten aus vorher definierten Verzeichnissen verschlüsselt und in beliebig einstellbaren Intervallen ins Mozy-Rechenzentrum überträgt. Eine Webanwendung dient als zentrales Verwaltungszentrum, ermöglicht jedoch nicht den Zugriff auf die Daten. Mozy verwendet eine 448-Bit Blowfish-Verschlüsselung und ermöglicht es wahlweise einen eigenen privaten Schlüssel anzugeben. Dies ist dahingehend vorteilhaft, da die Daten mit der Software eines Drittanbieters ausgelesen werden können. Bei der Übertragung der Daten ans Mozy Rechenzentrum wird mit 256-Bit SSL verschlüsselt. Es besteht die Möglichkeit Daten postalisch auf einem Datenträger anzufordern.

  Mozy ist nicht als Datentausch- oder Archivierungsdienst konzipiert. Werden Daten aus den definierten Backup-Ordnern gelöscht, werden diese auch in den Rechenzentren entfernt. Da Mozy aber über eine Versionierungs-Unterstützung verfügt, können gelöschte Dateien innerhalb von 30 Tagen wieder hergestellt werden. Freigabefunktionen jeglicher Art werden nicht unterstützt.

  Fazit: Primär ist Mozy zum sicheren und voll automatisierten Erstellen von Backups konzipiert. Die Daten sind auch ohne Netzanbindung verfügbar, die Sicherungen natürlich nicht. Mozy sticht mit der seltenen Eigenschaft hervor, dass die Daten auf dem Client-PC vor der Übertragung verschlüsselt werden.

- Box.net

  Box.net ist ein Speicherdienst mit Fokus auf das Teilen und gemeinsame Nutzen von Daten [37]. Neben einem Speicher- und Synchronisationsdienst werden Dokumentenversionierung, Nutzerprofile, Kommentar- und Suchfunktion sowie eine umfangreiche Administrationskonsole zur Verfügung gestellt.

  Es wird eine kostenfreie Version mit 5GB verfügbarem Speicherplatz für Privatkunden und zwei unterschiedliche Varianten für Geschäftskunden angeboten. Box.net sieht sich selbst als Online-Speicher und Cloud-Content-Management-System [38]. Desweiteren positioniert sich Box.net als Cloud-basierte Microsoft Sharepoint Alternative [8]. Argumente hierfür seien hauptsächlich die Kostenersparnis und die Ausfallsicherheit gegenüber einer „In-house“ Lösung.

Das Angebot ist auf Businesskunden ausgerichtet und bietet diesen ein umfangreiches Dokumentenmanagement und Redaktionssystem an. Ein Desktop Client zur Synchronisation und eine Integration der Google Apps sowie der Salesforce.com Plattform ermöglichen ein einheitliches Teamwork über mehrere Abteilungen einer Firma. Dokumentenserver oder z.B. die Aufbewahrung von Dokumenten innerhalb eines Mailkontos sollen durch die zentrale Ablage auf box.net ersetzt werden. Der Kunde erhält eine eigene Subdomain und weiterhin die Möglichkeit, das Layout seinem Corporate Design anzupassen.

Mit ECM Cloud Connect besteht die Möglichkeit, box.net in eine Sharepoint [39] oder Documentum [40] Infrastruktur einzubinden. Datenaustausch und Synchronisation ist in beide Richtungen realisierbar. Dies ermöglicht Firmen eine hybride Infrastruktur (siehe Kapitel 2.3.3) für ihre Dokumentenverwaltung aufzubauen.

Die freie Variante für Privatkunden beinhaltet neben dem kostenlosen 5GB Angebot die Möglichkeit, den verfügbaren Speicherplatz kostenpflichtig auf bis zu 50GB zu erweitern. Zusätzlich werden Freigabefunktionen und eine App für den Zugriff portabler Endgeräte wie Handys oder iPads bereit gestellt.

Fazit: Das Angebot ist auf Firmen zugeschnitten, die nach einer einfachen aber mächtigen Lösung für eine zentrale Datenverwaltung und allen damit verbundenen Aufgaben suchen.

- Microsoft SkyDrive/ LiveMesh

  Microsoft stellt mit SkyDrive eine kostenlose Online-Festplatte zur Verfügung [41]. Über eine Webanwendung können Dateien herauf- und herunterladen sowie Freigaben bearbeitet werden.

  Zu Speicherung der Daten werden 25GB zur Verfügung gestellt. Davon können bis zu 5GB von dem Synchronisationsdienst Microsoft LiveMesh genutzt werden [42]. Hierfür muss der LiveMesh Client installiert werden. Daraufhin können beliebige Ordner zur Synchronisation hinzugefügt werden. Neue, gelöschte oder geänderte Dateien werden automatisch erkannt und synchronisiert. Skydrive beschränkt die Größe einzelner Dateien auf 50MB, während LiveMesh mit beliebig großen Dateien umgehen kann.

  Für SkyDrive wird kein Desktop-Client bereitgestellt, aber es besteht die Möglichkeit aus der Office 2010 Produktfamilie heraus auf SkyDrive zu speichern. Die Dateien sind online verfügbar und können über den in Windows Live enthaltenen SaaS-Dienst Office Live im Browser bearbeitet werden [43, 44]. Das Hochladen von Dateien über die Webseite gestaltet sich sehr umständlich, aber die Integration in die Microsoft Desktop-Produktfamilie ist

gut umgesetzt. Aus diesem Grund kann angenommen werden, dass Microsoft diesen Dienst als Erweiterung zur Desktop Produktstrategie und nicht als reinen Cloud-Dienst sieht.

Es lassen sich Download Links für Dateien und Berechtigungen für Ordner erstellen. Ein gemeinsames Arbeiten an dem gleichen Office Dokument ist möglich, sofern man Office 2010 einsetzt.

Fazit: Skydrive bietet sich als Online-Festplatte für Office Dokumente an. Es besteht die Option, auch andere Datentypen über eine Weboberfläche zu speichern. Ein Teil des bereitgestellten Speichers steht zu Synchronisations-zwecken zwischen Windows Rechnern zur Verfügung. Funktionen zum gemeinsamen Arbeiten sind gut integriert. Der Dienst ist kostenlos und hauptsächlich an Privatnutzer gerichtet.

5.2 Vor- und Nachteile

Die unterschiedlichen Ansätze der vorgestellten Speicherlösungen machen deutlich wie divergent der Markt der Speicherdienste ist. Bei der Auswahl der Firmen wurde darauf geachtet, exemplarische Typen von Speicherdiensten gegenüberzustellen.

Es lassen sich folgende Kategorien von Speicherdiensten ausmachen:

- Synchronisation
- Online-Festplatte
- Backup Lösung
- Dokumenten- und Datenmanagement
- Speicher- und Anwendungs-Ökosystem

Die Anbieter haben für ihre Lösungen unterschiedliche Zielgruppen und Einsatzgebiete. Unabhängig davon lassen sich anbieterübergreifend Mängel ausmachen.

Im Allgemeinen kann festgestellt werden, dass das Vertrauen der Kunden im Bezug auf Datenspeicherung ein sehr ernst zu nehmendes Thema ist. In dieser Hinsicht hebt sich besonders Mozy hervor. Daten werden bereits auf dem PC verschlüsselt und dann über eine verschlüsselte Verbindung übertragen. Der verwendete Verschlüsselungs-algorithmus ist offen gelegt und die Angabe eines eigenen privaten Schlüssels ist möglich. Dieses Vorgehen ist einzigartig unter den angeführten Speicherdienstanbietern. Deutlich im Nachteil liegt hier z.B. HiDrive. Dort wird das

Passwort verschlüsselt übertragen, die Daten selbst aber nicht [45]. Andere Anbieter halten sich mit Informationen bezüglich der Datensicherheit entweder bedeckt oder versichern nur dass die Daten sicher sind. „Security through obscurity“ [46] scheint bei Speicherdienstanbietern weit verbreitet. Dies ist ein Mangel, mit dem sich die Speicherdienstanbieter in Zukunft beschäftigen sollten.

Es zeichnet sich ab, dass Anbieter zwei unterschiedliche Ansätze der Datenhaltung angehen. Einerseits wird die Datenhaltung lokal und bei dem Anbieter vorgenommen. Andererseits ausschließlich bei dem Anbieter. Wird eine Online-Festplatte als Netzlaufwerk eingebunden, wie es beispielsweise HiDrive ermöglicht, so ist der Zugriff auf die Daten nur bei aktiver Internetverbindung gegeben. Sobald ein Nutzer nicht mehr online ist, ist das Laufwerk nicht eingebunden und der Datenzugriff nicht mehr möglich. Dennoch hat diese Art und Weise einer Dienstumsetzung den Vorteil, dass sich im Falle eines Diebstahls eines mobilen Gerätes keinerlei persönliche Daten auf dem Gerät befinden. Andere Lösungen, wie z.B. Dropbox halten auf allen Endgeräten eine Kopie des Datensatzes. Besteht keine Internetverbindung, kann zwar nicht synchronisiert, jedoch immer noch auf die Daten zugegriffen werden. Sollte das Gerät gestohlen werden, wären persönliche Daten für Dritte einsehbar. Dieses Manko ließe sich durch einen passwortgeschützten Zugriff auf den Synchronisationsordner beheben. Jedoch wurde dies bisher von keinem Anbieter realisiert.

Ein Problem, dass im Prinzip der Synchronisation liegt ist, dass auf jedem Gerät ausreichend Speicherplatz verfügbar sein muss. Dies kann bei Netbooks und anderen mobilen Geräten ein Problem darstellen. Zwar lassen sich bei manchen Diensten (z.B. Dropbox) die zu synchronisierenden Ordner selektieren. Jedoch kann der Nutzer dabei den Überblick verlieren, welche Daten auf welchen Geräten gespeichert sind. Daten sind dadurch häufig nicht auf dem benötigten Gerät. Im Bezug auf mobile Geräte mit kleinen Speichern ist eine Online-Festplatte wie etwa HiDrive im Vorteil, da diese Geräte für eine dauerhafte Internetverbindung ausgelegt sind. Die Anbieterwahl ist also stark vom Verwendungszweck abhängig.

Manche Dienste zwingen dem Kunden eine Ordnerstruktur auf. Dropbox synchronisiert beispielsweise nur Dateien, welche im Dropbox-Ordner abgelegt sind. Anwender sind gezwungen, diesen Ordner für die zu synchronisierenden Daten zu nutzen. Dies führt zu einer Zersplitterung der eigenen Daten- bzw. Ordnerstruktur. Mozy hingegen lässt die benutzerdefinierte Definition der Ordner zum Backup zu, erkennt und synchronisiert Daten jedoch nicht automatisch. Sicherungen führt es intervallgesteuert aus. LiveMesh geht den Mittelweg und verfolgt beide Ansätze. Auch diese unterschiedlichen Herangehensweisen müssen differenziert betrachtet werden.

Ein gesonderter Ordner macht dem Anwender klar, welche Dateien synchronisiert werden und welche nicht – dies schafft Klarheit. Vor allem da viele Nutzer oft unübersichtlich viele Ordner pflegen. Eine Backup-Software wie Mozy muss zur

Synchronisation nicht ständig Bandbreite verbrauchen, sondern kann Daten in nachvollziehbaren Versionen absichern. Dem Kunden soll damit die Sorge um die Sicherheit seiner Daten genommen werden und ihm dabei einen möglichst einfachen Umgang mit der Backup Thematik bieten. Eine präzise Auswahl der Daten für die Sicherung ist dahingehend selbstverständlich. Vor allem im Hinblick darauf, dass es sich häufig um größere Datenmengen handelt, will man den Nutzer langwieriges Kopieren von Daten in einen gesonderten Ordner ersparen. Anwendungsfreundlichkeit ist wichtig um den Nutzer nicht zu frustrieren, sodass er aus Frust keine Backups mehr macht.

Ein Problem, welches bei Speicherdienstanbietern auftaucht ist, dass versucht wird, so viele Möglichkeiten, Anwendungsfälle und Kundentypen wie möglich abzudecken. Die Dienste verlieren an Übersichtlichkeit und sind nur umständlich zu bedienen. Ein Präzedenzfall hierfür ist Skydrive. Die SkyDrive Webanwendung ist verwoben mit Office Live und weiteren Live Diensten aus dem Hause Microsoft. Es wird so viel Funktionalität von anderen Diensten integriert, dass ein Nutzer die von ihm gewünschten Funktionalitäten nur schwierig ausmachen kann. Die Usability leidet und die Dauer der Eingewöhnungsphase steigt. Box.net hingegen ist das Gegenbeispiel. Es wird eine enorme Vielfalt an Funktionalität geboten und darüber hinaus werden Dienste anderer Anbietern integriert. Trotzdem bleibt der Dienst einheitlich und unkompliziert zu bedienen.

Schlussendlich lässt sich feststellen, dass eine genaue Betrachtung eines Anbieters unter Bezug auf den eigenen Verwendungszweck unerlässlich ist. Man sollte insbesondere Themen wie Sicherheit, die zu verwendeten Geräte und den Anwendungsfall genau definieren. Erst dann lassen sich von Vor- und Nachteile einzelner Anbieter diskutieren.

5.3 Vergleich mit MultiBackup

Alle besprochenen Speicherdienstanbieter stellen Speicher-, Backup- oder Synchronisationsdienste zur Verfügung. Infrastruktur und Software werden als Dienst bereitgestellt. MultiBackup stellt selbst keinen Speicherplatz zur Verfügung, sondern bietet Zugriff auf Cloud Storage Anbieter. Somit tritt MultiBackup als Vermittler auf und bietet Speicherplatz nur indirekt an.

Im Vergleich mit den vorgestellten Lösungen hat das eine Reihe von Vorteilen:

- Unabhängigkeit

  Durch die Nutzung von MultiBackup ist ein Kunde nicht auf einen Speicherdienstanbieter oder auf MultiBackup angewiesen. Dies ist bei den vorhandenen Lösungen nicht der Fall.

- Redundanz

  Es wird Datenredundanz durch Verteilung der Daten auf mehrere Cloud Storage Anbieter geboten. Dies wird bisher von keinem Speicherdienstanbieter umgesetzt.

- Erweiterbarkeit

  MultiBackup ist so ausgelegt, dass es möglich ist beliebige Anbieter hinzuzufügen bzw. zu entfernen. Eine Selbstbestimmung der Cloud Storage Anbieter ist bei keiner der vorhandenen Lösungen möglich.

- Zugang

  Der Kunde hat für den Fall, dass MultiBackup seinen Dienst einstellt oder nicht verfügbar sein sollte weiterhin Zugang zu seinen Daten. Ebenfalls besteht die Möglichkeit, die Speicherdienstanbieter direkt in Anspruch zu nehmen. Bei allen vorgestellten Anbietern besteht ausschließlich die Möglichkeit auf die vom Anbieter bestimmte Art auf seine Daten zuzugreifen.

- Kosten

  Der Kunde zahlt nur für den verwendeten Speicherplatz bei den Speicherdiensanbietern. Neben der flexiblen Abrechnung kommt er dabei vor allem in den Vorteil der preiswerten Tarife. Dies ermöglicht kein Anbieter.

Daher kann MultiBackup mehr als „Webanwendung zur Nutzung von Cloud Storage“ angesehen werden als ein reiner Speicherdienst.

Bedingt vergleichen kann man Desktop Clients von Drittanbietern. Diese lokal zu installierenden Programme erlauben den Zugriff auf Cloud Storage Anbieter aus einer Explorer-ähnlichen Anwendung heraus. Im Gegensatz zu MultiBackup oder anderen zuvor vorgestellten Anbietern entspricht ein Desktop Client nicht den Cloud Prinzipien (siehe Kapitel 2.1). Daraus ergibt sich eine Reihe von Nachteilen gegenüber einer Cloud-basierten Lösung (siehe Kapitel 4.1.1). Würde man die Cloud Prinzipien in der Betrachtung außen vor lassen, würde die Redundanz der Verteilung eine rein Desktop-basierte Lösung unbrauchbar machen. Diese Aussage kann aufgrund der durch die Redundanz bedingten Vervielfachung der benötigten Bandbreite getroffen werden. Es wäre denkbar einen Desktop Client als Erweiterung zu einer Webanwendung bereitzustellen. Wie am Beispiel Dropbox zu beobachten, wo sich Desktop Client und Webanwendung gegenseitig ergänzen.

Kapitel 6

6. Zusammenfassung und Ausblick

Wie im vorherigen Kapitel hervorgehoben wurde, existieren vielfältige Lösungen, die je nach Einsatzgebiet mehr oder weniger zur Nutzung in Betracht gezogen werden können. Da Unternehmen einen immer größeren Datenzuwachs verzeichnen, steigt auch die Notwendigkeit, sich mit nachhaltigen Storage-Strategien zu befassen. Dies umfasst die Bereitstellung und Verwaltung, genauso wie die Sicherung der Daten.

Von dieser Notwendigkeit profitieren Hardwarehersteller und Dienstleister gleichermaßen. Durch die starke Verbreitung von digitalen Medien steigt auch in Privathaushalten die Notwendigkeit, Daten zu sichern.

MultiBackup verfolgt einen anderen Ansatz, als die meisten Anbieter. Es soll Zugang und Nutzung mehrerer Cloud Storage Anbieter ermöglichen und somit eine höchstmögliche Sicherheit durch redundant abgelegte Daten gewährleisten.

6.1 Zusammenfassung

Da MultiBackup als Webdienst konzipiert wurde, ermöglicht es einen standortunabhängigen Zugang. Der Kunde kann unabhängig von MultiBackup auf seine Daten zugreifen, es besteht kein Lock-In gegenüber MultiBackup. Durch die redundante Ablage der Daten wird sichergestellt, dass die Daten sich, selbst bei einem Komplettausfall eines Cloud Storage Anbieters, noch an einem anderen Ort befinden.

Die Entwicklung unter Verwendung der Google App Engine erwies sich als vorteilhaft. Diese bietet mit vielseitig verwendbaren APIs ein professionelles Ökosystem, welches die unkomplizierte Entwicklung anspruchsvoller Webanwendungen ermöglicht. Aufgrund des geringen Bandbreitenlimits des ein- und ausgehenden Datenverkehrs über URL-Fetch erscheint sie jedoch nicht als optimale Plattform für datenintensive Anwendungen. Die Entwicklung der Anwendung auf einem eigenen Server hätte den Gesamtaufwand und die Entwicklungszeit um ein hohes Maß gesteigert. Ungeachtet dessen zeigt die Entwicklung von MultiBackup wie derartige Anwendungen funktionieren können. Das Resultat stellt einen geeigneten Machbarkeitsbeweis dar. Insbesondere unter dem Gesichtspunkt, dass Google die bestehenden Limitierungen häufig auflockert und MultiBackup auch innerhalb einer Private-Cloud anwendbar ist, ist die Machbarkeitsanalyse einer solchen Software als positiv einzustufen.

6.2 Ausblick

MultiBackup stellt einen technisch voll funktionsfähigen Prototypen dar. Die Basisfunktionalitäten Speichern, Runterladen und Löschen wurden für S3 und Google Storage umgesetzt. Die dafür notwendige Zugangsdatenverwaltung wurde gleichermaßen umgesetzt. Durch die Verwendung von Google Accounts ist Multibackup mehrbenutzerfähig. Der nächste Schritt wäre die Anbindung von Eucalyptus und Cumulus von Nimbus. Dies gäbe Kunden die Möglichkeit, Daten sowohl intern als auch extern über die gleiche Anwendung zu verteilen.

Das Konzept ist tragfähig und Erweiterungen sind denkbar. Nachfolgend wird eine denkbare Release-Roadmap vorgestellt, bei der die Reihenfolge keine Prioritäten repräsentieren soll:

- Upload mehrerer Dateien gleichzeitig
- Erweiterte Konfigurationsmöglichkeiten der Cloud Storage Anbieter
- Integration zusätzlicher Cloud Storage Anbieter
- Lokaler Client für:
    - Automatisierte Backups von Desktop Systemen
    - Automatisierte Backups von Server Systemen
    - Synchronisationsdienst
- Erhöhung der Sicherheit durch:
    - SSL-Übertragung
    - Client-seitige Verschlüsselung der Daten

Das Ziel wäre es, eine Cloud-basierte Anwendung wie MultiBackup, kombiniert mit einem Client wie Mozy, einem Ordner-basierten Auswahlverfahren wie SkyDrive und einer Synchronisationfunktion wie Dropbox bereit zu stellen.

Kapitel 7

7. Begriffsverzeichnis

Appliance	Anwendung mit einem Minimum an benötigtem Betriebssystem, meist in Form einer Virtuellen Maschine bereitgestellt.
AWS	Amazon Web Services, Name der von Amazon angebotenen Cloud-Dienste [12]
BLOB	Binary Large Object, Binärdatentyp in Datenbanksystemen.
Collaboration	Zusammenarbeit mehrerer Personen oder Personengruppen. In der Informatik häufig als beschreibendes Merkmal von Software aufgeführt. Beschreibt die Unterstützung und Vereinfachung von Teamarbeit und aller damit zusammenhängender Abläufe und Prozesse.
DOM Baum	Document Object Model, Baum Struktur eines HTML-Dokuments
Easy-to-use	Meint eine einfache und selbsterklärende Bedienung.
GQL	Proprietäre von Google für Datastore Datenbankabfragen entwickelte SQL-ähnliche Auszeichnungssprache [47].
In-house	Etwas wird innerhalb eines Unternehmens umgesetzt.
Lock-In	Abhängigkeit von einer Begebenheit, einem Umstand oder einem Anbieter.
OO-Paradigmen	Grundlegende Konzepte objektorientierter Programmiersprachen. U.a. Vererbung, Polymorphie, Kapselung.
Key/Value	Datenstruktur, dabei wird ein Wert (Value) einem Schlüssel (Key) zugeordnet und ist über diesen auffindbar.
KMU	Kleine und mittlere Unternehmen.
Reader	Oft auch Input- oder Filereader, liest Daten von einer Ressource.
Request/Response	Senden von Anfragen und Antworten auf Anfragen im Web-Umfeld.

REST	Representational State Transfer, Softwarearchitekturstil.
Security through obscurity	Bezeichnet den Umstand, dass Sicherheit durch Verschleierung bzw. Geheimhaltung zu erreichen versucht wird [46].
Slashdot Effekt	Servicelevel Agreement, wenn eine populäre Berichterstattung auf eine kleine Webseite hinweist und diese daraufhin durch einen Benutzeransturm nicht erreichbar wird.
SLA	Bezeichnet einen Vertrag zwischen Auftraggeber und Dienstleister für wiederkehrende Dienstleistungen.
SOAP	Simple Object Access Protocol, Netzwerkprotokoll für entfernte Methodenaufrufe.
Stream	Kontinuierliche Übertragung von Daten.
Transparent	Beschreibt in der Informatik den Umstand, dass etwas nicht wahrnehmbar bzw. unsichtbar ist.
Usability	Nutzungsqualität bei der Interaktion mit einem System.

Kapitel 8

8. Literaturverzeichnis

8.1 Literatur

[1] BAUN, C.: Cloud Computing: Web-basierte dynamische IT-Services, Springer 2010

[2] RHOTON, J.: Cloud Computing Explained: Implementation Handbook for Enterprises, Recursive Press 2010

[3] SANDERSON, D.: Programming Google App Engine, O'Reilly 2009

[4] CIURANA, E.: Developing with App Engine, Apress 2008

[5] BAUN, C.: Cluster-, Grid- und Cloud Computing, Vorlesung Hochschule Mannheim, 15. Oktober 2010

[6] SPECHT, T.: Webtechnologien, Vorlesung Hochschule Mannheim, 16. März 2010

8.2 Web-Quellen

[7] http://www.cio.de/was_ist_cloud_computing/2231267, Version: April 2010

[8] http://techcrunch.com/2010/04/07/box-net-raises-15-million-to-take-on-microsoft-sharepoint-in-the-cloud, Version: April 2010

[9] http://cloudcomputing.sys-con.com/node/1338492, Version: April 2010

[10] https://www.dropbox.com/help/27, Version: April 2011

[11] http://cloudstoragestrategy.com/2009/03/defining-cloud-storage.html, Version: März 2009

[12] http://aws.amazon.com/de, Version: April 2011

[13] http://www.allthingsdistributed.com/2010/05/amazon_s3_reduced_redundancy_storage.html, Version: Mai 2010

[14] http://aws.amazon.com/de/s3/faqs/#What_is_the_durability_of_Amazon_S3_w
en_using_RRS, Version: April 2011

[15] http://code.google.com/intl/de-DE/apis/storage, Version: April 2011

[16] http://code.google.com/intl/de-DE/appengine, Version: April 2011

[17] http://code.google.com/intl/de-DE/appengine/docs/quotas.html,
Version: April 2011

[18] http://doc.s3.amazonaws.com/proposals/post.html, Version: April 2011

[19] http://code.google.com/intl/de-DE/appengine/docs/python/tools/webapp,
Version: April 2011

[20] http://www.djangoproject.com, Version: April 2011

[21] http://code.google.com/p/boto, Version: April 2011

[22] http://www.eucalyptus.com, Version: April 2011

[23] http://code.google.com/intl/de-DE/appengine/articles/deferred.html,
Version: Oktober 2009

[24] http://code.google.com/intl/de-DE/appengine/docs/python/blobstore,
Version: April 2011

[25] http://code.google.com/intl/de-DE/appengine/docs/python/urlfetch,
Version April 2011

[26] http://code.google.com/intl/de-DE/appengine/docs/python/datastore,
Version: April 2011

[27] http://code.google.com/intl/de-DE/appengine/articles/logging.html,
Version: April 2008

[28] http://www.aptana.com/products/studio2, Version: April 2011

[29] http://www.python.org/download/releases/2.5, Version: April 2011

[30] http://code.google.com/intl/de-DE/appengine/docs/whatisgoogleappengine.html,
Version: April 2011

[31] http://code.google.com/intl/de-DE/appengine/downloads.html,
Version: April 2011

[32] http://www.eclipse.org, Version: April 2011

[33] https://www.dropbox.com, Version: April 2011

[34] http://www.strato.de/online-speicher/index.html, Version: April 2011

[35] http://mozy.de, Version: April 2011

[36] http://www.emc.com, Version: April 2011

[37] http://www.box.net, Version: April 2011

[38] http://www.box.net/solutions, Version: April 2011

[39] http://sharepoint.microsoft.com, Version: April 2011

[40] http://www.emc.com/domains/documentum/index.htm, Version: April 2011

[41] http://skydrive.live.com, Version: April 2011

[42] http://www.mesh.com, Version: April 2011

[43] http://www.windowslive.de, Version: April 2011

[44] http://www.officelive.com, Version: April 2011

[45] http://www.zdnet.de/bildergalerien_so_laesst_sich_stratos_online_festplatte_hidrive_einsetzen_story-39002391-41529678-1.htm, Version: März 2010

[46] http://www.uni-protokolle.de/Lexikon/Security_through_Obscurity.html, Version: April 2011

[47] http://code.google.com/intl/de-DE/appengine/docs/python/datastore/gqlreference.html, Version: April 2011