CloudRaid

Ein sicherer Raid-Manager für freie Cloud Storages

Bachelor-Thesis von Klaus Wilhelmi

Sommersemester 2012

TECHNISCHE UNIVERSITÄT DARMSTADT

CloudRaid
Ein sicherer Raid-Manager für freie Cloud Storages

Vorgelegte Bachelor-Thesis von Klaus Wilhelmi

Prüfer: Prof. Dr. Michael Waidner
Betreuer: Lukas Kalabis

Tag der Einreichung:

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 2. August 2012

(K. Wilhelmi)

Zusammenfassung

Kostenlose Cloud-Storage-Dienste erfreuen sich aktuell wachsender Beliebtheit. Sie bieten aber teilweise nur eingeschränkten Speicherplatz, fehlende Funktionalität und unzureichende Sicherheit, was einen großen Raum für Verbesserungen offen lässt. Diese Arbeit beschäftigt sich damit, kostenlose Dienste um eine RAID ähnliche Funktionalität zu erweitern und mögliche Verbesserungen seitens Funktionalität und Sicherheit zu untersuchen. Dazu wurden drei Dienste exemplarisch ausgewählt.

Zusätzlich zur Auswahl und Evaluation der drei Dienste wurde eine Anwendung implementiert, die mit beliebig vielen Accounts dieser Anbieter arbeiten kann. Der Speicherplatz der Accounts wird dazu verwendet, um zwei Speicherstrategien zu realisieren, was Vorteile bezüglich Sicherheit und Verfügbarkeit mit sich bringt. Durch die ständige Entwicklung im privaten Cloud-Storage-Bereich wurde bei der Implementierung besonderer Wert auf Erweiterbarkeit gelegt. Dazu zählen unter anderem die Unterstützung von neuen Diensten, die Integration in andere Programme und die Erweiterung um neue Funktionalitäten, was die Anwendung für eine zukünftige Weiterentwicklung über diese Arbeit hinaus interessant macht.

Inhaltsverzeichnis

1 Einführung

1.1 Problemstellung

In der letzten Zeit erfreuen sich Cloud-Storage-Dienste wie Dropbox[1] oder Ubuntu One[2] wachsender Beliebtheit. Diese kostenlosen Dienste bieten dem Anwender eine begrenzte Menge an Speicherplatz, der genutzt werden kann, um Dateien auf dem lokalen Rechner in der Cloud zu speichern. Üblicherweise stehen dabei Datenmengen zwischen 2 und 5 GB zur Verfügung. Möchte ein Benutzer über zusätzlichen Speicherplatz verfügen, wird meist eine monatliche Gebühr fällig. Sollen Daten wie Bilder oder Videos dort hinterlegt werden, sind die Grenzen schnell erreicht.

Anwender nutzen diese Dienste gerne um persönliche Daten zu sichern. Dies bietet den Vorteil, dass Daten einfach auf mehreren Computern, Smartphones oder Tablet-PCs abgerufen werden können. Ebenso bieten alle Dienste Weboberflächen, die den Zugriff über einen Browser ermöglichen. Diese Dienste sind aber aus zweierlei Hinsicht problematisch. Zum einen ist meist nicht ersichtlich, wo die Daten physikalisch gespeichert werden und welche Personen darauf zugreifen können. Zum anderen können Probleme auftreten, wenn Dienste ihr Angebot einstellen oder aus anderen Gründen (z. B. Hacker-Angriffe oder rechtliche Gründe) schließen müssen. Ein solcher Fall ist 2011 aufgetreten im Zusammenhang mit der Schließung der Sharehosting Plattform *Megaupload*.[3] Viele Nutzer die den Dienst gewerblich verwendeten, konnten nicht mehr auf ihre Daten zugreifen.[4] Um solchen Problemen vorzubeugen, hat ein Benutzer zum Beispiel die Möglichkeit, Daten manuell bei mehreren Diensten zu speichern. Dies ist nur mit einem hohen administrativen Aufwand möglich.

Zusammengefasst behandelt diese Arbeit das Problem des begrenzten Speicherplatzes solcher Dienste und der fehlenden Funktionalität, Daten auf unkomplizierte Art und Weise bei mehreren Anbietern zu hinterlegen, um weiterhin auf diese zugreifen zu können, falls einer der Provider ausfällt.

1.2 Motivation

Die Idee ist, mehrere Accounts bei Cloud-Storage-Providern zu kombinieren und diese gemeinsam einzusetzen. Der Benutzer hat dazu die Möglichkeit, zwischen zwei verschiedenen RAID ähnlichen Speicherstrategien zu wählen.

Die erste Strategie realisiert eine Spiegelung der Daten (ähnlich RAID-1 [23]). Die Daten eines Nutzers werden automatisch bei Provider A und Provider B hinterlegt. Falls Provider A ausfällt, kann über Provider B weiterhin auf die Daten zugegriffen werden. Dieses Beispiel ist nicht nur auf zwei Provider beschränkt, sondern kann beliebig erweitert werden, indem neue Provider unterstützt werden.

Die zweite Strategie zielt darauf ab, den begrenzen Speicherplatz mehrerer Accounts zu bündeln und so einen großen Speicherplatz zur Verfügung zu stellen (ähnlich RAID-5 [23]). Das bedeutet, dass der Benutzer nicht die Wahl hat, welcher Account zur Speicherung der Daten benutzt wird, sondern die Daten automatisch auf dem verfügbaren Speicherplatz verteilt werden.

[1] http://www.dropbox.com
[2] http://one.ubuntu.com
[3] http://www.megaupload.com
[4] http://www.golem.de/news/megaupload-us-regierung-will-cloud-nutzer-seine-daten-nicht-zurueckgeben-1206-92512.html

1.3 Ziel

Das Ziel der Arbeit ist die Implementierung einer Client-Anwendung, die in der Lage ist, mehrere Accounts von Cloud-Storage-Diensten zu verwalten. Unter Betrachtung der zukünftigen Entwicklung soll es möglich sein, die Anwendung leicht um weitere Dienste zu erweitern oder im Fall einer API Änderung die bestehende Implementierung anzupassen. Viele Dienste bieten eine *REST API*, die mit nahezu jeder Programmiersprache mit Hilfe von HTTP-Anfragen [6] angesprochen werden kann.

Die Software benutzt den Speicherplatz, der durch die Accounts der Cloud-Storage-Dienste bereitgestellt wird, um die vorher beschriebenen Speicherstrategien umzusetzen. Die Implementierung erfolgt als Webanwendung, die auf dem lokalen Rechner des Benutzers ausgeführt wird. Um die Benutzbarkeit auf dem lokalen Rechner zu erhöhen, werden ebenfalls einige Anpassungen (z.B. ein Symbol in der Taskleiste) vorgenommen, wie man sie von den Client Programmen der Cloud-Storage-Anbieter kennt.

Da die Software mit sensiblen Daten, wie unter anderem Benutzernamen und Passwörtern des Benutzers umgeht, wird ein besonderer Fokus auf die Sicherheit der Implementierung gelegt. Als Erstes sollte die Verbindung zu den Cloud-Storage-Anbietern deshalb gegen das Abhören der Kommunikation gesichert sein. Es sollte also mindestens HTTPS [28] verwendet werden, was bei der Kommunikation bei den meisten Providern aber sowieso als Standard vorausgesetzt wird. Des Weiteren sollte die Anwendung gegen unbefugte Zugriffe durch die Eingabe von Benutzername und Passwort geschützt sein.

Neben dem Aspekt der Sicherheit soll auch die Benutzbarkeit der Implementierung sichergestellt werden. Die Anwendung soll von einem Benutzer mit durchschnittlichen Vorkenntnissen einfach zu bedienen sein. Daher sollte die Benutzeroberfläche einfach und intuitiv gestaltet sein.

2 Technische Hintergründe

2.1 World Wide Web

Das World Wide Web (WWW) [31] ist ein System aus vielen miteinander verknüpften Hypertext Dokumenten. Es existieren Webserver, die Dokumente bereitstellen und Clients, zum Beispiel in Form eines Browsers, die diese Dokumente abrufen. Das meist verwendete Hypertext-Format ist HTML[1], das von einem Browser direkt angezeigt werden kann. Außer HTML existieren im World Wide Web noch viele weitere Dateitypen, die nicht mit dem Browser angezeigt werden können. Hierfür wird bei jeder Datei, die von einem Server an den Benutzer geschickt wird, ein MIME-Typ [8] angegeben, der Informationen über das Format der Datei liefert. Als Protokolle für den Austausch zwischen Client und Server wird HTTP [6] beziehungsweise HTTPS [28] verwendet, auf das im weiteren Verlauf noch genauer eingegangen wird.

Die Adressierung im World Wide Web erfolgt über eine sogenannte URL[2]. Gibt ein Benutzer beispielsweise die URL `http://www.tu-darmstadt.de/studieren/index.de.jsp` an, so kann der Browser ermitteln,

- welche Seite der Benutzer sehen möchte (`/studieren/index.de.jsp`),
- auf welchem Computer sich die Seite befindet (`www.tu-darmstadt.de`) und
- mit welchem Protokoll er die Seite anfordern will (`http`).

2.1.1 HTTP (Hypertext Transfer Protocol)

Die von den Cloud-Storage-Anbietern bereitgestellten APIs basieren in der Regel auf REST und dieses arbeitet wiederum auf Basis von HTTP. Aus diesem Grund ist es sinnvoll HTTP im folgende Abschnitt etwas detaillierter zu betrachten. Wie zuvor schon erwähnt, handelt es sich bei HTTP um das Standardprotokoll zur Kommunikation im WWW. Es existieren Clients, die Anfragen senden und Server, die diese beantworten. Eine typische Anfrage (Request) die generiert wird, wenn ein Benutzer über seinen Browser eine Seite aufruft, besteht aus folgenden Teilen [12]:

- Die Anforderungszeile (Request-Line) enthält an der ersten Stelle die auszuführende Request Methode, die betreffende Ressource und die Protokollversion (Abbildung 2.1, Zeile 1). Eine Übersicht über die Request Methoden ist auf Seite 9 dargestellt. Die Ressource beschreibt die Datei, die der Benutzer abrufen möchte. Im Beispiel handelt es sich hier um das Basisverzeichnis („/"). Eine Ressource deutet nicht immer auf eine Datei sondern kann auch auf ein Programm deuten, welches dynamisch Inhalte erzeugt. Handelt es sich um ein Verzeichnis und nicht um eine konkrete Datei, so wird eine beim Webserver festgelegte Indexdatei abgerufen (meist index.htm, index.html, etc.).
- Der Anforderungskopf (Request-Header) wird in der Regel dazu benutzt, zusätzliche Informationen zu übermitteln. Er ist in Form einer Liste aufgebaut. Jedes Element der Liste enthält einen Namen und einen dazugehörigen Wert. Ein Beispiel dafür ist in Abbildung 2.1 von Zeile 2 bis 8 zu sehen.

[1] Hypertext Markup Language
[2] Uniform Resource Locator

- Im Anforderungsbody (Request-Body) können zusätzliche Inhalte untergebracht sein. Dies hängt aber von der jeweiligen Request Methode ab. Wie man im Beispiel sieht, ist der Body bei einer GET Anfrage leer. Beispielsweise kann er in einer POST Anfrage genutzt werden, um Daten an den Webserver zu übertragen.

```
1 GET / HTTP/1.1
2 Host: www.tu-darmstadt.de
3 Connection: keep-alive
4 User-Agent: Mozilla/5.0 (X11; Linux i686) AppleWebKit/535.11 (KHTML, like Gecko) Ubuntu
      /10.04 Chromium/17.0.963.79 Chrome/17.0.963.79 Safari/535.11
5 Accept: text/html,application/xhtml+xml,application/xml;
6 Accept-Encoding: gzip,deflate,sdch
7 Accept-Language: de-DE,de;q=0.8,en-US;q=0.6,en;q=0.4
8 Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
```

Abbildung 2.1: Beispiel einer HTTP-Anfrage

Nachdem der Server die Anfrage erhalten hat, verarbeitet er diese und verschickt eine Antwort (Response), welche folgendermaßen aufgebaut ist:

- Die Statuszeile (Status-Line) enthält Informationen über Verarbeitung der Anfrage beim Server. Sie enthält als Erstes die Protokollversion gefolgt von einem dreistelligen Statuscode und einer kurzen Beschreibung des Statuscodes in Textform (Abbildung 2.2, Zeile 1).
- Der Antwortkopf (Response-Header) erlaubt dem Server, zusätzliche Informationen zu geben, die nicht in der Statuszeile enthalten sind. Hierzu zählen auch Informationen über den MIME-Typ oder die Länge von übertragenen Inhalten (Abbildung 2.2, Zeile 2 bis 8).
- Im Antwortbody (Response-Body) befinden sich die eigentlichen Nutzdaten. Diese sollten vom im Antwortkopf definierten Typ sein. Im Beispiel einer GET-Anfrage, wird der Antwortbody die angeforderte HTML Datei beinhalten (Abbildung 2.2, ab Zeile 10).

```
1 HTTP/1.1 200 OK
2 Date: Wed, 23 May 2012 09:05:15 GMT
3 Server: Apache/2
4 Set-Cookie: JSESSIONID=A3BC832717EDB16403E47542E304C9AD; Path=/
5 Vary: Accept-Encoding
6 Transfer-Encoding: chunked
7 Content-Type: text/html;charset=UTF-8
8 Content-Language: de
9
10 <!DOCTYPE html>
11 <html lang="de">
12 <head>
13 <title>Willkommen an der TU Darmstadt</title>
14 <meta charset="utf-8"/>
15 ....
```

Abbildung 2.2: Beispiel einer HTTP-Antwort

Request Methoden

Wie im vorherigen Absatz vorgestellt, enthält jede Anfrage eine bestimmte Methode, welche direkt an ihrem Anfang steht. In Tabelle 2.1 sind alle in HTTP 1.1 definierten Methoden enthalten [31].

Methode	Beschreibung
GET	Anforderung zum Lesen einer Webseite
HEAD	Anforderung zum Lesen des Headers einer Webseite
PUT	Anforderung zum Speichern einer Webseite
POST	Anhängen an eine benannte Ressource (z.B. eine Webseite)
DELETE	Entfernen einer Webseite
TRACE	Echo für die eingehende Anforderung
CONNECT	Für zukünftige Verwendung reserviert
OPTIONS	Abfrage bestimmter Optionen

Tabelle 2.1: HTTP Methoden

Die **GET-Methode** sendet eine Anforderung zum Lesen einer Webseite an den Server. Der Aufbau besteht aus dem Schlüsselwort GET, der angeforderten Ressource und der Protokollversion. Eine GET-Methode erfordert keinen weiteren Inhalt außer der vorher beschriebenen Anforderungszeile und optionalen Headern (Abbildung 2.1).

Die **PUT-Methode** fordert die Speicherung einer Ressource an. Die Anfrage enthält das Schlüsselwort PUT, gefolgt von der Ressource die gespeichert werden soll, sowie der Protokollversion. Im Header sind zusätzliche Felder wie *Content-Type* oder *Content-Lenght* enthalten, die Informationen über den übertragenen Inhalt angeben. Die Prüfung ob eine PUT Anfrage durchgeführt werden darf, erfolgt auf der Serverseite, indem das Feld *Authorization* ausgewertet wird.

Die **POST-Methode** hängt bei einer Anfrage Daten an eine Ressource an, ohne diese zu überschreiben. Als Anwendungsfall ist zum Beispiel eine Kommentarfunktion auf einer Webseite denkbar, bei der ein neuer Kommentar die Alten natürlich nicht überschreiben soll.

Mit Hilfe der **DELETE-Methode** wird versucht, eine betreffende Ressource zu löschen. Hier wird wieder über das *Authorization* Feld im Header geregelt, ob die Anfrage berechtigt ist, die Ressource zu löschen.

Statuscodes

Die erste Zeile eines Antwort-Headers enthält Informationen über den Status der Anfrage, die der Antwort voraus gegangen ist. Dieser Status wird in Form eines Statuscodes dargestellt, durch den der Client leicht nachvollziehen kann, ob die abgesetzte Anfrage erfolgreich war [7] . Es existieren fünf verschiedene Gruppierungen von Statuscodes:

Code	Beschreibung
1xx - Information	Anfrage erhalten und bereit die Anforderung weiter zu verarbeiten
2xx - Success	Die Anfrage war erfolgreich
3xx - Redirection	Es ist eine weitere Aktion notwendig um die Anfrage abzuschließen
4xx - Client Error	Die Anfrage enthält Fehler oder kann nicht erfolgreich durchgeführt werden
5xx - Server Error	Der Server war nicht in der Lage, die Anfrage zu bearbeiten

Tabelle 2.2: Gruppierungen der Statuscodes

Die Bedeutung der Statuscodes innerhalb einer Gruppe kann unter Umständen abweichen. So ist es möglich, dass Serverhersteller eigene Statuscodes einführen, die nicht als Standard definiert sind. Zu den wichtigsten Codes zählen:

- 200 - die Anfrage war erfolgreich und die angeforderte Ressource ist im Nachrichtenbody enthalten
- 400 - die Anfrage war fehlerhaft
- 401 - es wird eine Authentifizierung benötigt
- 404 - die angeforderte Ressource wurde nicht gefunden
- 500 - es ist ein Fehler beim Bearbeiten der Anfrage auf dem Server aufgetreten

2.1.2 HTTPS (Hypertext Transfer Protocol Secure)

Das HTTPS-Protokoll [28] erweitert das zuvor vorgestellte HTTP-Protokoll um Verschlüsselung der Kommunikation zwischen Webbrowser und Webserver. Hierzu werden jegliche Anfragen und Antworten vor dem Verschicken über das Internet verschlüsselt. Um dies zu ermöglichen, wurde eine zusätzliche Schicht zwischen der Anwendungs- und Transportschicht eingeführt. Anstatt unverschlüsselte Anfragen direkt über TCP zu verschicken, werden sie vorher an die neue Schicht weitergegeben, welche die Anfragen verschlüsselt. Dies führt dazu, dass die Kommunikation in der Regel nicht abgehört oder verändert werden kann [4]. HTTPS wird nahezu von jedem Cloud-Storage-Anbieter verwendet, um die Kommunikation mit den Benutzern gegen Angreifer abzusichern. Es gibt aber auch Ausnahmen, wie den Dienst CloudMe[3], welcher den Datenverkehr zwischen Benutzer und Dienst nicht verschlüsselt [2].

2.1.3 REST (Representational State Transfer)

Bei dem Begriff REST handelt es sich um keinen definierten Standard sondern eher um ein Designrezept für Webanwendungen [29]. Bei diesen Anwendungen muss es sich auch nicht typischerweise um browserbasierte Anwendungen handeln. Zum Beispiel nutzen viele Anbieter REST um Programmierschnittstellen, sogenannte APIs, ihrer Dienste zur Verfügung zu stellen. Im Abschnitt über HTTP wurde bereits über Ressourcen gesprochen, welche nahezu alles darstellen können:

- Eine Liste von Büchern eines bestimmten Autors
- Eine Datei
- Der letzte Eintrag eines Blogs im März 2009

Beim Konzept von REST existiert für jede Ressource eine eindeutige URL um diese ohne Umwege ansprechen zu können. Für die oben genannten Beispiele könnten die URLs folgendermaßen aufgebaut sein:

- http://example.com/books/by-author/name
- http://example.com/files/42
- http://example.com/blog/2009/03/last

[3] http://www.cloudme.com

Um mit einer Ressource zu arbeiten, wird auf die HTTP Methoden zurückgegriffen. Hierzu stehen in erster Linie die Methoden GET (hole eine Ressource), PUT (lege eine Ressource an oder verändere sie), POST (lege eine Ressource an) und DELETE (lösche eine Ressource) zur Verfügung.

Zustandslosigkeit

In Hinsicht auf REST bedeutet Zustandslosigkeit, dass jede HTTP-Anfrage isoliert voneinander zu betrachten ist [30]. Das heißt, dass zum Durchführen einer Anfrage alle benötigten Informationen mitgeliefert werden und keine zusätzlichen Informationen, zum Beispiel aus einer vorherigen Anfrage, benötigt werden. Eine URL darf nicht abhängig von einem Zustand sein, sondern muss immer zugreifbar sein. Um dies genauer zu verdeutlichen, kann man die Anfrage an eine Suchmaschine betrachten. Die Ressource „/search/book“ liefert ein Suchergebnis mit 100 Einträgen zurück, welche auf 10 Seiten verteilt sind. Ein Aufruf der Ressource „/search/book/5“ liefert jetzt direkt die 5. Seite der Suchergebnisse, unabhängig der vorher durchgeführten Anfragen.

Ein weiterer Aspekt der Zustandslosigkeit ist die Mehrfachausführung von Aktionen. In diesem Zusammenhang ist davon die Rede, dass eine GET oder HEAD Anfrage „sicher“ sein muss. Das bedeutet, dass ein Benutzer der eine beliebige, unbekannte URL eines Webservices aufruft sicher sein kann, dass keine Seiteneffekte auftreten. Das heißt, dass diese Methoden nur Daten liefern und in keinem Fall Daten verändern. Es darf also am Zustand nichts verändern, wenn eine GET Anfrage mehrfach hintereinander, einmal oder nie aufgerufen wird [29].

Betrachtet man das Verhalten von Methoden, wie PUT und DELETE, die Ressourcen manipulieren, so darf eine mehrfache Ausführung hintereinander ebenfalls immer nur das gleiche Ergebnis liefern. Löscht man mit DELETE eine Ressource fünf Mal hintereinander, so ist die Ressource nach dem fünften Aufruf immer noch gelöscht. Erstellt man eine Ressource mit PUT gilt auch, dass die Ressource nach dem fünften Aufruf immer noch den gleichen Inhalt besitzt wie nach dem ersten Aufruf.

Darstellungsweisen

Eine REST basierte Webanwendung arbeitet in der Regel mit unterschiedlichen Darstellungsformen. Ruft man die Anwendung beispielsweise über einen Browser auf, so werden in der Regel HTML Daten zurück gegeben. Existiert eine Ressource in mehreren Sprachen, so gelten diese auch als eigene Darstellungen. Arbeitet man mit einer REST API, werden in der Regel Formate wie XML oder JSON verwendet, um Daten auszutauschen. Über diese verschiedenen Darstellungsformen kann über die URL oder auch über den Header der Anfrage entschieden werden. So könnte zum Beispiel über die URL „/books/show/1.de“ die deutsche Version eines Buches aufgerufen werden und über „/books/show/1.en“ die englische Version. Eine Entscheidung über den Anfrage-Header wird zum Beispiel bei der Benutzung eines Browsers durchgeführt, ohne dass der Benutzer davon etwas merkt. Jeder Aufruf mit einem Browser enthält das Feld „Accept-Language“, welches die vom Benutzer akzeptierten Sprachen enthält. Ob eine Auswertung erfolgt, hängt dann vom jeweiligen Webservice ab.

2.1.4 XML (Extensible Markup Language)

Zu Beginn des WWW wurden alle Inhalte in statischen HTML Dateien abgelegt. Es gab also keine Trennung zwischen Informationen und Formatierung. Die Formatierung war dafür gedacht, die Inhalte für den Betrachter möglichst gut darzustellen. Für Computer ist diese Form von Informationen aber schwer zu verarbeiten. Daher wurde vom World Wide Web Consortium[4] das Format XML entwickelt [3]. XML dient zur reinen Präsentation der Daten und enthält keine Informationen über deren Darstellung.

```
<?xml version="1.0" encoding="UTF-8"?>
<apartment>
  <room>
    <name>Kueche</name>
    <size>10 qm</size>
  </room>
  <room>
    <name>Wohnzimmer</name>
    <size>15 qm</size>
  </room>
</apartment>
```

Abbildung 2.3: Beispieldatei in XML

2.1.5 JSON (JavaScript Object Notation)

Das Format JSON wurde als die „fettfreie“ Alternative zu XML vorgestellt.[5] Es bietet ebenfalls die Möglichkeit Daten zu repräsentieren aber es ist bedingt durch seinen Aufbau für den Menschen leichter lesbar als XML. Es existieren vier primitive Datentypen: Number, String, Boolean und Null. Zum strukturellen Design existieren die Datentypen Object und Array [5].

```
{
    "apartment": [
      {
        "name": "Kueche",
        "size": 10,
        "clean": true
      },
      {
        "name": "Wohnzimmer",
        "size": 15,
        "clean": false
      }
    ]
}
```

Abbildung 2.4: Beispieldatei in JSON

[4] http://www.w3.org
[5] http://www.json.org/xml.html

2.2 Authentifizierung

Das HTTP-Protokoll liefert nur zwei eingebaute Möglichkeiten eine Authentifizierung zu realisieren. Die *Basic Authentication* und die *Digest Authentication* basieren beide auf dem klassischen Authentifizierungsmodell mit Client und Server [9]. Das bedeutet, dass dem Benutzer Zugangsdaten in Form eines Benutzernamens und eines Passworts vorliegen und er sich mit diesen beim Server authentifiziert. Dies ist in manchen Anwendungsfällen zu unflexibel, weshalb sich alternative Protokolle wie *OAuth* etabliert haben.

2.2.1 Basic Access Authentication

Die *Basic Access Authentication* ist die einfachste Methode eine Ressource mit einem Benutzernamen und einem Passwort zu schützen [12]. Ein Authentifizierungsvorgang läuft folgendermaßen ab:

- Ein Client ruft eine URL auf, die durch *Basic Access Authentication* geschützt ist.
- Der Server antwortet mit dem Statuscode 401 und der Nachricht „Authorization Required“. Die Antwort enthält die Art der Authentifizierung sowie ein „Authentication Realm“, welcher die Menge der Ressourcen definiert, die durch die Zugangsdaten geschützt sind.
- Der Client, typischerweise ein Browser, fordert den Benutzer auf, einen Benutzernamen und ein Passwort einzugeben. Der Benutzer kann an dieser Stelle den Vorgang abbrechen. Erfolgt eine Eingabe, so wird der String „username:password“ mit Base64 [21] codiert, als Authentication Header in die ursprüngliche Anfrage eingefügt und zurück an den Server gesendet.
- Der Server prüft nun den mit Base64 kodierten übertragenen String, indem er Benutzername und Passwort mit seinen lokalen Daten vergleicht. Handelt es sich um die korrekten Zugangsdaten, sendet der Server die angeforderte Ressource an den Benutzer zurück.

Wird dieser Vorgang über eine unverschlüsselte HTTP Verbindung durchgeführt, besteht die Gefahr, dass die Zugangsdaten von einem Angreifer ausgelesen werden können. In der Praxis ist *Basic Authentication* sehr einfach und flexibel zu realisieren aber nur durch zusätzliche Maßnahmen wie den Einsatz von HTTPS wirklich sicher.

2.2.2 Digest Access Authentication

Die *Digest Access Authentication* bietet deutlich mehr Schutz als die *Basic Access Authentication*, ist aber auch deutlich komplexer. Im Vergleich zum vorher vorgestellten Verfahren werden keine Passwörter im Klartext über das Netzwerk übertragen. Es existieren außerdem Schutzmechanismen gegen diverse Angriffsmethoden auf die aber hier nicht weiter eingegangen wird. Der vereinfachte Ablauf eines Authentifizierungsvorgangs sieht folgendermaßen aus:

- Der Client ruft eine URL auf, die durch *Digest Access Authentication* geschützt ist.
- Der Server fordert den Client auf, den Benutzernamen und einen berechneten Digest des Passworts zu senden. Der Digest ist ein Hashwert aus verschiedenen verbindungsbezogenen Daten kombiniert mit dem Passwort. Daher kann ein Angreifer beim Abfangen eines Digests keine Rückschlüsse auf das Passwort ziehen.
- Der Client sendet die angeforderten Daten an den Server, der diese dann verifiziert. Der Server ist in Besitz der Passwörter aller Benutzer und kann so ebenfalls den Digest berechnen und vergleicht diesen dann mit dem empfangenen Digest. Stimmen Beide überein, so sendet der Server die angeforderte Ressource an den Client zurück.

2.2.3 OAuth 1.0

OAuth [16] bietet eine Alternative zum klassischen Authentifizierungsmodell mit Client und Server. Wie in Abbildung 2.5 zu sehen ist, existieren drei Rollen: *Client*, *Server* und *Resource Owner*. Der Vorteil bei OAuth liegt darin, dass ein Benutzer (*Resource Owner*) einer dritten Partei (*Client*) erlauben kann, auf eigene Daten bei einem Dienst (*Server*) zuzugreifen, ohne dass die dritte Partei die eigentlichen Zugangsdaten des Benutzers kennt [17]. Es werden hierzu lediglich sogenannte *Token Credentials* erstellt, mit denen dann im weiteren Verlauf gearbeitet wird. Bei Client und Server handelt es sich nach Definition um einen HTTP-Client und einen HTTP-Server, was bedeutet, dass die Kommunikation zwischen den einzelnen Rollen rein über HTTP(S) durchgeführt wird. Es wird zwar in den Spezifikationen von OAuth 1.0 nicht festgelegt aber es ist empfehlenswert, die Kommunikation immer über HTTPS durchzuführen, um das Mitlesen eines Angreifers zu verhindern.

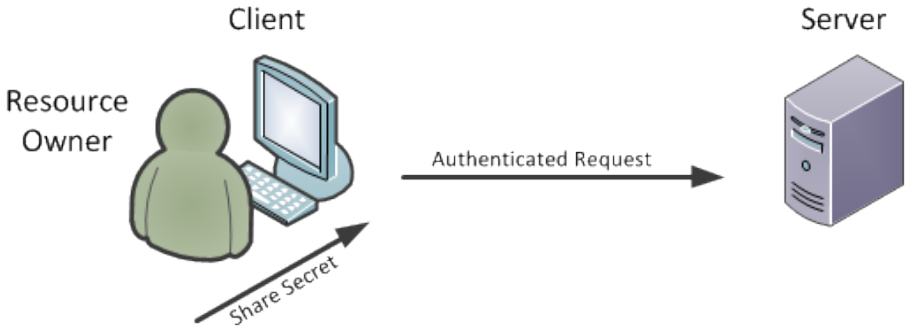

Abbildung 2.5: Rollenmodell von OAuth 1.0 [14]

OAuth wird von großen Anbietern wie Google oder Facebook verwendet, die auch dessen Entwicklung voran treiben [13]. Ebenso war bei der Entwicklung der Anwendung das Verständnis von OAuth von großer Bedeutung, da es auch im Cloud-Storage-Bereich zur Authentifizierung eingesetzt wird. Um die Funktionsweise genauer zu verdeutlichen, betrachten wir folgendes Beispiel: Der Benutzer Bob (*Resource Owner*) möchte auf dem Internetportal mybooks.com (*Client*) mehrere eBooks kaufen und diese auf seinem Cloud-Storage-Dienst mycloud.com (*Server*) abspeichern. Um dies zu ermöglichen hat das Buchportal beim Cloud-Dienst eigene *Client Credentials* angefordert und nutzt die bereitgestellten Protokollendpunkte des Cloud-Dienstes:

```
https://mycloud.com/request_token
https://mycloud.com/authorize
https://mycloud.com/access_token
```

Nun fordert das Buchportal im ersten Schritt ein Tupel mit temporären Zugangsdaten an, welche benötigt werden, um die Zugriffsanfrage zu identifizieren:

```
1 POST  /request_token HTTP/1.1
2 Host: mycloud.com
3 Authorization: OAuth realm="MyCloud",
4     oauth_consumer_key="912ec803b2ce49e",
5     oauth_signature_method="PLAINTEXT",
6     oauth_timestamp="633701682600000000",
7     oauth_nonce="akEkaAeQ",
8     oauth_callback="http://mybooks.com/ok",
9     oauth_signature="74KNZJeDHnMBp0EMJ9ZHt%2FXKycU%3D"
```

Der Cloud-Dienst verifiziert die Anfrage und sendet das angeforderte Tupel zurück:

```
HTTP/1.1 200 OK
Content-Type: application/x-www-form-urlencoded

oauth_token=8ba46f039d270&oauth_token_secret=eb891f645f059&oauth_callbackconfirmed=true
```

Im Anschluss wird Bob auf die Seiten des Cloud-Dienstes weitergeleitet. Dort wird er aufgefordert, sich mit Benutzernamen und Passwort anzumelden und die Zugriffsanfrage des Buchportals zu bestätigen. War dies erfolgreich, wird er auf die in der Anfrage angegebene Callback URL zurück auf die Seiten des Buchportals geleitet. Wurde zuvor keine Callback URL angegeben, wird er nicht weitergeleitet. Jetzt kann das Buchportal die endgültigen Zugangsdaten beim Cloud-Dienst anfordern:

```
Post /access_token HTTP/1.1
Host: mycloud.com
Authorization: OAuth realm="MyCloud",
    oauth_consumer_key="912ec803b2ce49e",
    oauth_token="8ba46f039d275920";
    oauth_signature_method="PLAINTEXT",
    oauth_timestamp="633701682600978474",
    oauth_nonce="dkAkadET",
    oauth_signature="gKgrFCywp7rO0OXSjdot%2FIHF7IU%3D"
```

Der Cloud-Dienst verifiziert wie zuvor die Anfrage und sendet ein Tupel mit den Token Credentials zurück:

```
HTTP/1.1 200 OK
Content-Type: application/x-www-form-urlencoded

oauth_token=e206a54e97690cce&oauth_token_secret=50cc872dd70ee896
```

Das Buchportal ist nun in der Lage mit Hilfe dieser Token signierte Anfragen an den Cloud-Dienst zu senden ohne Benutzername oder Passwort von Bob zu kennen. Die Token verlieren ihre Gültigkeit, sobald Bob den Zugriff beim Cloud-Dienst widerruft.

2.2.4 OAuth 2.0

Bei OAuth 2.0 [13] handelt es sich um ein komplett neues Protokoll, welches nicht abwärtskompatibel sein wird. Zwar befindet es sich noch in der Entwicklung aber wird schon von einigen Anbietern wie zum Beispiel den APIs von Google und Facebook verwendet. Die Vorgängerversion erwies sich in bestimmten Szenarien als unflexibel, weshalb in der Version 2.0 von OAuth sogenannte *Flows* eingeführt werden sollen, die bestimmte Anwendungsfälle für Entwickler erleichtern sollen [15]. In Version 1.0 war es nicht möglich, einfache Skripte zum Beispiel mit cURL[6] zu schreiben, da jede Anfrage individuell signiert werden musste. Dies erforderte meist die Verwendung einer zusätzlichen Bibliothek. In der Zukunft soll es nun möglich sein, sich in einem solchen Fall auch mit Benutzernamen und Passwort zu authentifizieren.

[6] Client for URLs - ein Programm für die Kommandozeile zum Übertragen von Dateien

2.3 Verteiltes Speichern

Dieser Absatz beschäftigt sich mit RAID und den für die Arbeit relevanten Aspekten. *RAID (Redundant Array of Independent Disks)* ist ein System zur Verbesserung der Ausfallsicherheit und der Leistungsfähigkeit in einem Festplattenverbund [23]. Meist kommt es im Serverbereich zum Einsatz, wenn es um wichtige Daten oder Anwendungen geht.

Ein Grund für den Einsatz von RAID kann eine Steigerung der Performance in Form von schnelleren Zugriffszeiten sein. Außerdem ist es möglich, die Ausfallsicherheit eines Systems zu verbessern, indem Daten mehrfach gespeichert werden, um so dem Verlust von Daten vorzubeugen. Ein RAID-Verbund erscheint aus Anwendersicht als eine Festplatte und die verschiedenen Volumes die zum Einsatz kommen, bleiben verborgen. Die Verwendung von RAID ist durch zusätzliche Hardware und den erhöhten Verwaltungsaufwand natürlich auch mit höheren Kosten verbunden.

Da die Anwendung, die in dieser Arbeit beschrieben wird, Funktionalitäten von RAID nachahmen soll, ist es wichtig, die Arbeitsweise von RAID zu betrachten, allerdings nicht die technischen Hintergründe. Die Funktion von Festplatten nehmen einzelne Accounts bei Cloud-Storage-Anbietern ein, die einen begrenzten Speicherplatz zur Verfügung stellen. Dieser Speicherplatz wird dazu verwendet, Speicherstrategien zu implementieren, die RAID-1 und RAID-5 ähneln.

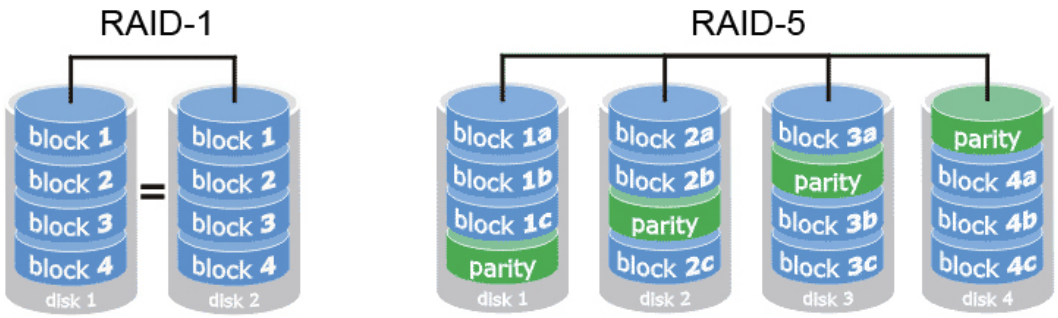

Abbildung 2.6: RAID-1 und RAID-5[7]

Es existieren die RAID-Level 0 bis 6, die zusätzlich noch unterschiedlich miteinander kombiniert werden können. Je nach Anwendungsfall entscheidet sich, welche Art am besten geeignet ist.

Ein RAID vom Typ 1 wird auch als *Mirroring* oder *Spiegelung* bezeichnet. Wie der Name schon vermuten lässt, wird eine Spiegelung der Daten vorgenommen. Das heißt, dass Dateien auf mindestens zwei Laufwerken parallel hinterlegt werden. Im Fall von zwei Laufwerken halbiert sich also der tatsächlich nutzbare Speicherplatz aber es wird gewährleistet, dass auch bei Ausfall einer Festplatte die Daten weiterhin verfügbar sind. Ein weiterer Vorteil ist die Verbesserung der Lese- und Schreibgeschwindigkeit, da Zugriffe auf mehrere Festplatten verteilt werden.

Bei RAID-5 werden Datenblöcke auf verschiedenen Festplatten verteilt [24]. Es entsteht also ein großer Speicherplatz, der aus mehreren Volumes besteht. Nun könnte man denken, dass dadurch beim Versagen einer Festplatte die Daten darauf verloren gehen. RAID-5 ist aber so konzipiert, dass eine der verwendeten Festplatten ausfallen kann. Der Inhalt des ausgefallenen Volumes kann dann durch die anderen Festplatten wieder hergestellt werden. Außerdem bietet RAID-5 ebenfalls wie RAID-1 durch die redundante Speicherung der Daten einen gesteigerten Datendurchsatz bei Lese- und Schreiboperationen.

[7] http://msi-forum.de

3 Evaluation der Cloud-Storage Dienste

3.1 Cloud-Computing

Das Thema Cloud-Computing erfreut sich in den letzten Jahren einer sehr großen Beliebtheit und ist nach wie vor ein aktuelles Thema in der IT-Branche.[1] Die gängige Definition von Cloud-Computing [10], beschreibt die Bereitstellung von Rechenressourcen über ein Netz. Ein Anbieter stellt zum Beispiel Rechenleistung, Speicherplatz, Anwendungen oder Betriebssysteme zur Verfügung, die von einem Nutzer in Anspruch genommen werden können [1]. Die Ressourcen werden dabei allein vom Provider in Form eines Pools verwaltet auf den mehrere Nutzer zugreifen. Kapazitäten können ohne großen Aufwand und durch möglichst geringe Interaktion mit dem Anbieter dynamisch angepasst werden. Dies bringt sowohl Vorteile als auch Nachteile mit sich. Nutzt ein Anwender eine Anwendung in der Cloud, liegt der komplette Verwaltungsaufwand für Hardware und Betriebssystem beim Anbieter der Anwendung. Der Benutzer muss sich nicht um Softwareupdates kümmern, sondern kann sich ausschließlich auf die Benutzung der Anwendung konzentrieren [18]. Der größte Kritikpunkt an diesem Modell ist die Transparenz für den Benutzer. Er kann nicht immer nachvollziehen, was mit seinen Daten passiert, an welchem Ort diese physikalisch gespeichert sind und wer darauf zugreifen kann.

Nach der Definition des National Institute of Standards and Technology existieren folgende Charakteristiken im Cloud-Computing [25]:

- Ein Benutzer ist in der Lage, selbstständig Ressourcen zu erlangen, ohne persönlich mit dem Serviceprovider in Kontakt treten zu müssen. Dazu könnte eine Erweiterung des Speicherplatzes eines Benutzers oder ähnliches zählen.
- Die Angebote der Provider sind mit gängigen Plattformen und Mitteln über das Internet zu erreichen.
- Es existiert ein Pool an Ressourcen (Speicherplatz, Rechenleistung, etc.) beim Serviceprovider, der von den Benutzern in Anspruch genommen werden kann. Der Benutzer hat dabei keinen Einfluss, auf welchem konkreten Server er welche Daten speichert. Lediglich der Ort der Speicherung der Daten kann in Form von einem Land, einer Stadt oder einem Rechenzentrum vertraglich festgelegt werden.
- Die Ressourcen können für den Anwender dynamisch zur Verfügung gestellt werden, so dass sie für den Benutzer unbegrenzt erscheinen. Diese Ressourcen können Hardware in Form von Servern oder Rechenzentren sein, die der Anwender nutzt.
- Die Nutzung der Ressourcen kann durch den Provider gemessen und überwacht werden und ebenfalls dem Benutzer zur Verfügung gestellt werden. Im Beispiel eines Cloud-Storage-Anbieters kann ein Benutzer einsehen, wie viel Speicherplatz sein Account einnimmt aber nicht den gesamt zur Verfügung stehenden Speicherplatz.

[1] http://www.motiviti.com/blog/gartner-hype-cycle-for-2012-and-beyond/

Cloud-Computing-Dienste können in drei Servicemodelle eingeteilt werden [25]. Diese sind auch als eine Art Abstraktionsebene zu sehen, welche gerne als Schichten in einer Pyramide dargestellt werden.[2] Um dies zu verdeutlichen, betrachten wir nun die drei Servicemodelle. Als oberste Ebene fungiert das *Software-as-a-Service (SaaS)* Modell. Hierzu zählen Dienste, die dem Benutzer Zugriff auf eine Anwendung bieten. Ein Beispiel ist Google Docs[3], welches ein Office Paket in Form eines SaaS-Dienstes bereit stellt. Der Benutzer installiert dafür keine Anwendung auf seinem Computer, sondern greift mit Hilfe des Browsers auf die Anwendung in der Cloud zu. Bei diesem Modell hat der Benutzer keinen Einfluss auf die unterliegenden Schichten wie Betriebssystem oder Hardware, anders als bei der Installation auf dem lokalen Rechner.

Der Begriff *Platform-as-a-Service (PaaS)* beschreibt die Ebene unterhalb der Anwendungen. Hier wird dem Benutzer die Infrastruktur wie Hardware und Betriebssystem zur Verfügung gestellt und er hat nur Zugriff auf die Anwendungsebene. Er kann Dinge wie verwendete Software, Programmiersprachen oder Bibliotheken verwalten, hat aber keinen Zugriff auf die Wahl des Betriebssystems oder die darunter liegenden Schichten.

Die unterste Schicht, welche dem Benutzer ermöglicht, Anwendung und Plattform zu kontrollieren wird *Infrastructure-as-a-Service (IaaS)* genannt. Hier wird von einem Serviceprovider Rechenleistung, Speicher und Netzwerkanbindung zum Beispiel in Form einer bestimmten Bandbreite zur Verfügung gestellt. Der Benutzer hat die Möglichkeit Betriebssystem und Anwendungen zu bestimmen.

3.2 Cloud-Storage

Cloud-Storage, auch oft als *Storage-as-a-Service* bezeichnet, beschreibt mehrere miteinander verbundenen Rechenzentren, die den Benutzern ein Interface in Form einer Client-Anwendung oder einer Weboberfläche zur Verfügung stellen, um Dateien bei diesen zu speichern. Dabei kommen die im Abschnitt über Cloud-Computing beschriebenen Charakteristiken zum Einsatz, was bedeutet, dass die hinter dem Interface liegende Infrastruktur dem Benutzer verborgen bleibt [2].

Auf dem Markt existieren zwei generelle Arten von Cloud-Storage-Anbietern. Es gibt kommerzielle Dienste wie Amazon S3[4], die ihre Ressourcen als *Business-to-Business* Lösung für andere Unternehmen zur Verfügung stellen. Dienste wie Dropbox oder Ubuntu One, deren Kundenstamm vorzugsweise aus Privatanwendern besteht, bewegen sich im *Business-to-Consumer* Bereich. Letztere bieten meist kostenlose Zugänge mit einem begrenzten Speicherplatz an. Jeder Dienst bietet auch verschiedene Bezahlmodelle an, auf die hier aber nicht weiter eingegangen wird.

Da Cloud-Storage-Dienste von Benutzern verwendet werden, um teils persönliche Daten zu speichern, spielen die Sicherheitsanforderungen hier eine große Rolle. In der Studie des Fraunhofer SIT [2] wurden dafür folgende Anforderungen aufgeführt, die an einen Cloud-Storage-Dienst gestellt werden sollten:

- Während des Registrierungsvorganges sollte die Identität des Benutzers überprüft werden. Dies kann in Form einer Bestätigungsmail erfolgen, um zu vermeiden, dass ein Angreifer eine Mailadresse eines anderen Benutzers verwendet um sich so als dieser auszugeben.
- Alle Interaktionen zwischen Benutzer und Cloud-Storage-Dienst sollten verschlüsselt stattfinden, um ein eventuelles mitlesen eines Angreifers unmöglich zu machen. Dazu zählen der Registrierungs- und der Login-Vorgang sowie der Austausch von Dateien während der Nutzung des Dienstes.
- Die Daten sollten auf den Servern des Dienstes in verschlüsselter Form abgespeichert werden. Auch hier gibt es Unterschiede. Manche Dienste verschlüsseln die Daten des Benutzers erst nach dem Empfang auf dem Server. Meist mit einem Schlüssel der dem Dienst bekannt ist, was dazu

[2] http://pyramid.gogrid.com
[3] http://docs.google.com
[4] http://aws.amazon.com/de/s3/

führt, dass die Daten gegen externe Angreifer geschützt aber trotzdem für den Anbieter lesbar sind. Werden die Schlüssel entwendet oder möchte ein Mitarbeiter, der in Besitz dieser Schlüssel ist, auf die Daten zugreifen, so ist dies ohne Weiteres möglich. Eine Verschlüsselung auf der Seite des Clients würde dies verhindern aber keiner der bekannteren Anbieter, bietet dies aktuell ohne den Einsatz von Zusatzsoftware an.[5]

- Ein beliebtes Feature ist das Teilen von Dateien mit anderen Benutzern. Hier ist darauf zu achten, dass Dateien wirklich nur für die Personen zugänglich sind, die der Benutzer wünscht. Außerdem sollten Dateien die durch eine öffentliche URL geteilt werden, nicht über eine Suchmaschine oder durch raten einer URL zu finden sein.

- Ein großer Vorteil ist die Benutzung eines Cloud-Storage-Dienstes auf mehreren Geräten wie verschiedenen Computern oder Smartsphones zur gleichen Zeit. Der Benutzer kann so immer auf den gleichen Datenbestand zugreifen. Im Fall eines Diebstahls muss der Benutzer in der Lage sein, zum Beispiel über die Accountverwaltung dem gestohlenen Gerät die Zugriffsrechte zu entziehen.

- Der Anbieter sollte klar zu erkennen geben, in welchem Land oder in welcher Stadt die Daten der Benutzer gespeichert werden. Im Idealfall könnte er den Benutzer informieren, welche rechtlichen Bestimmungen im entsprechenden Land gelten.

Dies ist ein grober Überblick über die behandelten Sicherheitsanforderungen an Cloud-Storage-Dienste. Er zeigt, dass der Aspekt der Sicherheit in diesem Anwendungsfeld entscheidend ist.

3.3 Dienste

Für Privatanwender und auch kommerzielle Anwender existiert eine Vielzahl von Cloud-Storage-Diensten. Diese Arbeit setzt sich ausschließlich mit kostenlosen Diensten auseinander. Hierzu wurden drei Anbieter ausgewählt.

Um die Anwendung für möglichst viele Benutzer zugänglich zu machen, wurde bei der Auswahl der Dienste darauf Wert gelegt, möglichst bekannte Dienste auszuwählen. Trotzdem sollte die Auswahl, für die auch die Studie des Fraunhofer SIT [2] hinzugezogen wurde, möglichst unterschiedlich sein und die ausgewählten Anbieter sollten eine API anbieten. Die Wahl fiel auf folgende Dienste:

- Dropbox ist mit über 25 Millionen Nutzern (Stand 2011) einer der Anbieter mit den größten Nutzerzahlen im Cloud-Storage Bereich für Privatanwender.[6]

- SugarSync wurde auf Grund seiner sehr gut dokumentierten API ausgewählt. Die Dokumentation war nicht nur hilfreich um die API zu verstehen, sondern auch um sich mit den technischen Hintergründen vertraut zu machen.

- Ubuntu One ist ein Dienst der ursprünglich für das gleichnamige Betriebssystem gedacht war und daher im Umfeld er Anwender von Ubuntu beliebt ist. Außerdem bietet der Dienst eine API und eine akzeptable Menge an kostenlosem Speicherplatz.

Alle ausgewählten Dienste verfügen über eine REST API. Das bedeutet, dass der Austausch von Daten über einzelne HTTP-Anfragen initiiert wird. Um sicher zu stellen, dass Daten, die zwischen dem Benutzer und dem Cloud-Storage-Dienst übertragen werden nicht abgehört oder verändert werden können, erfolgt die Kommunikation ausschließlich über HTTPS. Dies setzten alle drei Dienste standardmäßig voraus. Detaillierte Informationen über die APIs der drei Dienste finden sich im späteren Verlauf ab Seite 21.

[5] https://www.boxcryptor.com

[6] http://www.heise.de/newsticker/meldung/Dropbox-vermeldet-25-Millionen-Nutzer-1229650.html

	Dropbox	SugarSync	Ubuntu One
Kostenloser Speicherplatz	2 GB	5 GB	5 GB
API	REST	REST	REST
Format zum Datenaustausch	JSON	XML	JSON
Authentifizierungsmethode	OAuth 1.0	Eigene	OAuth 1.0
Synchronisierung beliebiger Verzeichnisse	✗	✓	✗
Versionierung von Dateien	✓	✓	✗
Unterstützung für alle gängigen Betriebssysteme	✓	✓	✗
Unterstützung für alle gängigen Smartphones	✓	✓	✓

Tabelle 3.1: Vergleich der ausgewählten Cloud-Storage-Dienste

3.3.1 Dropbox

Bei Dropbox[7] handelt es sich wohl um den bekanntesten Anbieter im Bereich der kostenlosen Cloud-Storage-Dienste. Der Benutzer erhält pro registriertem Account 2 GB Speicherplatz. Diese können durch eine Art Belohnungssystem kostenlos erweitert werden. Zusätzlicher Speicherplatz wird zum Beispiel für das Einladen von Freunden, das Freigeben von Ordnern oder auch die Installation von Betaversionen der Client-Software zur Verfügung gestellt.

Die Nutzung von Dropbox erfolgt über den Browser oder eine entsprechende Client-Software, die aktuell für alle gängigen Betriebssysteme erhältlich ist. Ebenso existieren Versionen für Smartphones und Tablets. Entscheidet sich der Benutzer für die Client-Software, so wird ein lokales Verzeichnis angelegt, dessen Inhalt automatisch mit dem Online-Speicherplatz synchronisiert wird.

Aktuell existiert eine technische Schwachstelle, die mit der Registrierung zusammen hängt [2]. Erstellt ein Benutzer bei Dropbox ein neues Konto, so wird bei der Registrierung über die Webseite keine Prüfung der Identität vorgenommen. Ein Benutzer kann sich mit einer falschen E-Mail-Adresse bei Dropbox registrieren und der Zugang ist sofort aktiv. Dies wurde in der Studie des Fraunhofer SIT bemängelt und war ebenfalls zum Zeitpunkt der Erstellung dieser Arbeit noch der Fall [2].

Ebenfalls kritisch zu sehen ist, dass die Verschlüsselung der Daten erst auf den Servern von Dropbox erfolgt. Dadurch kann der Benutzer nicht sicher sein, was mit seinen Daten beim Anbieter genau passiert. Eine Lösung hierfür wäre eine clientseitige Verschlüsselung der Daten, bevor sie an Dropbox übertragen werden [2].

3.3.2 SugarSync

SugarSync[8] bietet in der kostenlosen Variante 5 GB Speicherplatz. Es existieren Client-Anwendungen für alle gängigen Betriebssysteme und Smartphone Plattformen. SugarSync bietet ein wesentliches Feature, welches die anderen Anbieter nicht unterstützen. Der Benutzer hat die Möglichkeit einzelne Ordner auf seiner Festplatte auszuwählen, die dann automatisch mit der Cloud synchronisiert werden. Für den Fall, dass ein Benutzer eine Datei überschreibt oder aus Versehen löscht, hat er die Möglichkeit die 5 letzten Versionen der Datei wieder herzustellen.

Im Gegensatz zu Dropbox validiert SugarSync die E-Mail-Adresse eines Benutzers bei der Registrierung, durch den Versand einer Bestätigungsmail. Dies scheint im Moment ein generelles Problem bei mehreren Anbietern darzustellen.[9] Nach Angaben von SugarSync hat das Unternehmen Nutzerzahlen

[7] http://www.dropbox.com
[8] http://www.sugarsync.com
[9] http://www.sit.fraunhofer.de/de/medien-publikationen/pressemitteilungen/2012/Cloud-Schwachstelle.html

im Millionenbereich.[10] Trotz dieser Größe existieren bisher keine Untersuchungen bezüglich der Sicherheit des Dienstes. Auf der Webseite des Anbieters findet sich lediglich ein Hinweis darüber, dass die Daten zwischen den Benutzern und SugarSync verschlüsselt übertragen und verschlüsselt abgespeichert werden.[11]

3.3.3 Ubuntu One

Der dritte ausgewählte Dienst ist Ubuntu One[12]. Wie der Name schon vermuten lässt, wird der Dienst von der gleichen Firma betrieben, die auch an der Entwicklung von Ubuntu Linux beteiligt ist. Nach einer kostenlosen Registrierung stehen auch hier 5 GB an Speicherplatz zur Verfügung. Es werden die gleichen Basisfunktionalitäten wie bei den anderen Anbietern bereitgestellt, wobei Features wie eine Versionierung oder die Synchronisation beliebiger Verzeichnisse fehlen. Eine Übersicht dazu findet sich in Tabelle 3.1.

Anfänglich war Ubuntu One auch in erster Linie für Benutzer des eigenen Betriebssystems gedacht. Daher existiert auch eine Client-Anwendung für Ubuntu mit einer guten Integration in den Desktop. Im Gegenzug existiert kein Client für Mac OS X. Die Client-Anwendung für Windows bietet nicht den gleichen Funktionsumfang wie die Variante für Linux [2]. Im Smartphonebereich gibt es Anwendungen für iOS und Android.

Der Registrierungsvorgang ist durch den Einsatz von *Captchas* und der Bestätigung der Registrierung mittels E-Mail geschützt [2]. Außerdem erfolgt die Übertragung der Daten während des Logins sowie der Registrierung verschlüsselt über HTTPS. Die Daten der Benutzer in der Cloud werden allerdings komplett unverschlüsselt hinterlegt.[13]

3.3.4 Vergleich der APIs

Die von **Dropbox** bereitgestellte API [19] ist sehr gut strukturiert und bietet viele wichtige Funktionen. Die Dokumentation bietet leider nur spärliche Informationen und wenige Beispiele, wie die API zu nutzen ist. Dies liegt wahrscheinlich auch daran, dass Dropbox eigene Entwicklerpakete für *Java*, *Python*, *Ruby* und *iOS* anbietet und empfiehlt, vorzugsweise mit diesen zu arbeiten. Als Protokoll zur Authentifizierung wird OAuth 1.0 verwendet, wie in RFC 5849 im Abschnitt „Redirection-Based Authorization“ beschrieben [16]. Als Format zum Austausch von Daten wird JSON verwendet.

SugarSync bietet eine sehr gut dokumentierte API [20] und wartet mit vielen in Java geschriebenen Beispielen und Diagrammen auf. Zu jeder Funktion existiert eine Beschreibung, die in Request und Response aufgeteilt ist. Die Dokumentation beschreibt detailliert, wie zum Beispiel HTTP-Anfragen aufgebaut sein müssen, wie die entsprechenden Antworten der API aussehen und welche Response Codes für die einzelnen API-Calls existieren. Als Austauschformat kommt XML zum Einsatz.

Die API von **Ubuntu One** [22] ist auf Grund ihrer sehr kurz gehalten Dokumentation schwer ohne Vorkenntnisse zu benutzen. Sie bietet außerdem deutlich weniger Funktionen als die APIs der anderen Anbieter. Wie bei der Programmierschnittstelle von Dropbox wird als Authentifizierungsmodell OAuth 1.0 und als Containerformat JSON verwendet.

[10] http://techcrunch.com/2012/02/29/cloud-based-file-syncing-platform-sugarsync-raises-15m-to-add-another-data-center-to-the-mix/
[11] http://www.sugarsync.com/products/security.html
[12] http://one.ubuntu.com
[13] https://one.ubuntu.com/help/faq/are-my-files-stored-on-the-server-encrypted/

Wenn es um die Sicherheit der verwendeten APIs geht, sind in erster Linie zwei Dinge zu beachten. Welche Rechte erhält die Anwendung, wenn der Benutzer Zugriff gewährt und wie sicher ist die Übertragung von Daten zwischen der Anwendung und dem Cloud-Storage-Anbieter. Ersteres ist innerhalb der drei ausgewählten Anbieter recht unterschiedlich realisiert. Für die Durchführung von Anfragen ist allerdings bei allen Diensten die Verwendung von HTTPS vorausgesetzt, was bedeutet, dass unverschlüsselte Anfragen nicht durchgeführt werden können.

Abbildung 3.1: Dropbox Authentifizierungsvorgang

Nachdem man sich bei Dropbox als Entwickler registriert hat, besteht die Möglichkeit eine Anwendung anzulegen. Hier kann der Entwickler nun wählen, ob die Anwendung Zugriff auf die gesamte Dropbox des Benutzers oder nur auf einen für die Anwendung bestimmten Ordner in der Dropbox des Benutzers erhält. Dies bekommt der Benutzer der Anwendung dann angezeigt, sobald er seinen Account mit der Anwendung verknüpfen möchte (Abbildung 3.1). Zum Abschluss enthält der Entwickler einen „App Key“ und ein „App Secret“ welche den „Client Credentials“ im Kapitel über OAuth 1.0 entsprechen. Diese dienen auch dazu, dass Dropbox identifizieren kann, welche API-Calls von welcher Anwendung gesendet werden. Wird eine Anwendung neu angelegt, befindet sie sich im Entwicklungsstatus und kann nur von dem Entwickler selbst und bis zu 5 zusätzlichen Benutzern verwendet werden. Die Anzahl der verfügbaren API-Calls ist in diesem Stadium auf eine Höchstanzahl begrenzt, die erhöht wird, sobald die Anwendung sich im Produktivstatus befindet.

Abbildung 3.2: Ubuntu One Authentifizierungsvorgang

Obwohl die API von Ubuntu One ebenfalls OAuth 1.0 als Authentifizierungsmodell verwendet, gibt es einen gravierenden Unterschied zur Vorgehensweise von Dropbox. Ein Entwickler muss weder sich noch seine Anwendung explizit bei Ubuntu registrieren, sondern alle Entwickler verwenden die gleichen Client Credentials (der Consumer Key ist „ubuntuone“ und das Consumer Secret ist „hammertime“). Dies hat zur Folge, dass die API von Ubuntu One Anfragen einem Benutzer zuordnen kann aber nicht mit welcher Anwendung er diese durchgeführt hat. Dies sieht man gut in Abbildung 3.2, welche die Seite nach der Anmeldung eines Benutzers mit seinen Ubuntu One Zugangsdaten zeigt. Die Anwendung wird dort als „Computer“ hinterlegt. Wird dieser später vom Benutzer wieder entfernt, so verliert die Anwendung ihre Rechte auf den Account des Benutzers zuzugreifen. Ebenfalls gibt es nicht wie bei Dropbox eine Auswahlmöglichkeit ob die Anwendung Zugriff auf die gesamten Daten eines Benutzers erhält oder nur Zugriff auf einen für die Anwendung bestimmten Ordner erhält. Erteilt also ein Benutzer einer Anwendung Zugriff, so kann die Anwendung alle Daten die der Benutzer online über Ubuntu One gespeichert hat auslesen und manipulieren.

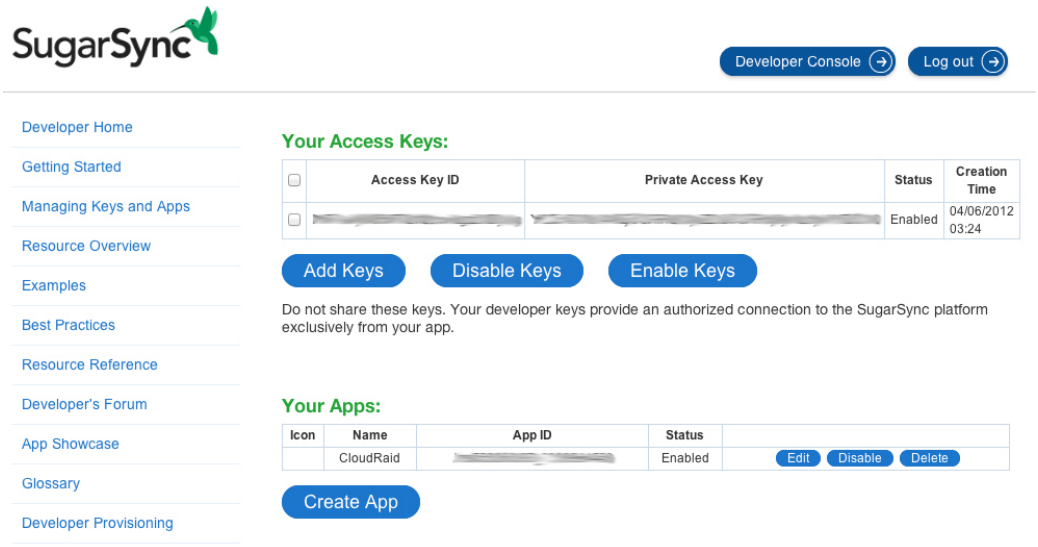

Abbildung 3.3: SugarSync Developer Console

Um Zugang zur API von SugarSync zu erhalten, muss man sich als Entwickler mit seinem Account auf der Entwicklerseite registrieren. Hier kann nun ein *Access Key Paar* und eine Anwendung erstellt werden (Abbildung 3.3). Bei der Authentifizierung kommt nicht OAuth zum Einsatz sondern ein eigenes Verfahren. Verwendet ein Benutzer zum ersten Mal eine Anwendung, so wird ein *RefreshToken* generiert. Hierzu werden der Benutzername, das Passwort, die *App ID* und das *Access Key Paar* an die SugarSync API übergeben, welche dann einen gültigen *RefreshToken* zurück liefert. Dieser wird dann zur weiteren Verwendung von der Anwendung abgespeichert. Dies hat den Vorteil, dass die Zugangsdaten des Benutzers nur einmalig benötigt werden und im weiteren Verlauf nicht gespeichert werden. Um endgültig eine Anfrage an die API senden zu können, muss im letzten Schritt ein *AccessToken* angefordert werden. Dazu wird eine HTTP-Anfrage mit dem *RefreshToken* und dem *Access Key Paar* verschickt. Sind diese gültig, wird ein *AccessToken* zurück gegeben, der in der Regel eine Gültigkeitsdauer von 1 Stunde besitzt. Dieser wird bei einer Anfrage in den Header im Feld „Authorization“ eingefügt. Die API kann dann überprüfen, ob es sich um eine gültige Anfrage handelt.

3.4 Zusammenfassung

Die Betrachtung der ausgewählten Provider ergab, dass alle Dienste in der kostenlosen Variante ähnliche Funktionalitäten bieten. Insbesondere stellt keiner der betrachteten Anbieter eine lokale Verschlüsselung der Daten sowie einen Schutz gegen Ausfallsicherheit bereit. Außerdem ergaben die Betrachtungen der Programmierschnittstellen und deren verwendete Authentifizierungsverfahren Schwachstellen bezüglich der Sicherheit.

Verschlüsselung

Ein lobenswertes Sicherheitsfeature wäre eine Verschlüsselung der Daten auf der Anwenderseite, also bevor sie zum Provider gesendet werden. Dropbox verschlüsselt die Daten erst nach Empfang mit einem ihnen bekannten Schlüssel, so dass der Benutzer nicht sicher sein kann, was der Dienst mit den Daten anstellt. Im Fall von Ubuntu One werden die Daten bei der Speicherung überhaupt nicht verschlüsselt. Bei SugarSync ist es unklar, wie die Verschlüsselung der Daten von statten geht. Da es zum aktuellen Zeitpunkt keine wissenschaftlichen Arbeiten oder Untersuchungen bezüglich der Sicherheit von SugarSync gibt, geben nur die Angaben des Providers selbst Auskunft. Dieser gibt an, dass die Daten direkt nach Erhalt in zwei verschiedenen Rechenzentren verschlüsselt abgespeichert werden. Ein Zugriff auf die Daten ist nur durch wenige Mitarbeiter möglich und erfolgt nur, wenn es „absolut notwendig“ erscheint.[14] Das deutet darauf hin, dass ein ähnliches Vorgehen wie bei Dropbox verfolgt wird. Positiv zu erwähnen ist, dass SugarSync eine Anleitung anbietet, wie der Benutzer eine clientseitige Verschlüsselung seiner Daten mit TrueCrypt[15] einrichten kann.[16]

Ausfallsicherheit

Die zweite Funktionalität, die allgemein von Cloud-Storage-Anbietern nicht abgedeckt wird, ist eine Sicherheit gegen den Ausfall eines gesamten Dienstes. Die Dienste selbst speichern die Daten der Benutzer zwar mehrfach in verschiedenen Rechenzentren ab, aber im Falle, dass ein Dienst zum Beispiel wegen Gesetzesverstößen komplett eingestellt wird, hat der Benutzer keine Möglichkeit mehr auf seine Daten zuzugreifen. Um einem solchen Szenario vorzubeugen, ist der Anwender gefragt, mehrere Dienste parallel zu nutzen.

Programmierschnittstellen

Individuelle Dinge bezüglich der Sicherheit der Dienste sind bei der genaueren Betrachtung der bereitgestellten Programmierschnittstellen aufgefallen. Eine Anwendung die diese nutzt, muss vom Benutzer autorisiert werden, um auf dessen Daten zugreifen zu können. Betrachtet man als Anwendungsfall eine Bookmarkverwaltung, die die Lesezeichen des Benutzers in dessen Cloud-Storage-Account hinterlegt, so wäre es wünschenswert, wenn die Anwendung nur Zugriff auf den Ordner erhält in dem die Lesezeichen gespeichert werden und nicht auf den gesamten Datenbestand des Benutzers. Leider ist dies nur bei Dropbox möglich. Der Entwickler hat dort die Möglichkeit anzugeben, ob seine Anwendung nur auf einen Ordner oder auf die gesamte Dropbox Zugriff benötigt. Eine Anwendung die mit den APIs von SugarSync oder Ubuntu One arbeitet, hat immer Zugriff auf den kompletten Datenbestand eines Benutzers, was unter Umständen von einem Angreifer ausgenutzt werden kann. Außerdem muss ein Benutzer darauf vertrauen, dass der Entwickler keine Fehler gemacht hat, die die eigenen Daten manipulieren oder löschen könnten.

[14] https://sugarsync.custhelp.com/app/answers/detail/a_id/201/
[15] http://www.truecrypt.org
[16] https://sugarsync.custhelp.com/app/answers/detail/a_id/367/

Eine weitere Problematik besteht im Zusammenhang mit der Autorisierung der API von SugarSync. Autorisierungsverfahren wie OAuth bieten den Vorteil, dass ein Benutzer einer Anwendung die Rechte erteilen kann auf seine Daten zuzugreifen, ohne dass die Anwendung in Besitz der Zugangsdaten des Benutzers kommt. Das von SugarSync verwendete Verfahren erfordert, dass der Benutzer die Zugangsdaten in der Anwendung selbst eingibt. Die Anwendung fordert dann einen *RefreshToken* an, mit dem sie zukünftig arbeitet. Es ist also nicht nötig, die Zugangsdaten des Benutzers dauerhaft zu speichern. Allerdings kann der Benutzer nicht nachvollziehen, was die Anwendung vielleicht im Hintergrund mit den Zugangsdaten macht.

Zusammengefasst sei gesagt, dass ein Entwickler die Möglichkeit hätte, auf Daten des Benutzers zuzugreifen, sowie im Falle von SugarSync, die Zugangsdaten des Benutzers gegen dessen Willen auszulesen. Einzig Dropbox bietet eine gute Lösung an. Anwender sollten daher mit Vorsicht entscheiden, welchen Anwendungen sie Zugriff gewähren.

4 CloudRaid - Beschreibung der Anwendung

Wie zu Beginn der Arbeit beschrieben, handelt es sich bei der Implementierung um eine Anwendung, die mehrere Accounts bei Cloud-Storage-Diensten verwendet, um RAID ähnliche Speicherstrategien umzusetzen. Statt der manuellen Speicherung von Dateien bei verschiedenen Diensten, fügt der Benutzer seine Accounts in der Anwendung hinzu und wählt eine Speicherstrategie. Entsprechend dieser nimmt die Software die Verteilung der Dateien automatisch vor.

Die Anwendung ist als Webanwendung realisiert, die auf dem lokalen Rechner ausgeführt wird. Die Implementierung erfolgte durch Einsatz des Play Frameworks.[1] Dieses basiert auf Java und bietet so den Vorteil, dass Entwickler auf eine Vielzahl von externen Bibliotheken zurückgreifen können [27]. Des Weiteren wurden zusätzliche Bibliotheken wie die Apache Commons Library[2] für die Überwachung des lokalen Dateisystems, der Apache HttpClient[3] für die Kommunikation mit den Cloud-Storage-Diensten und Twitter Bootstrap[4] für die Gestaltung der Benutzeroberfläche eingesetzt.

Wie zu Beginn der Arbeit beschrieben, wurde ein besonderer Fokus auf die Sicherheit und die Benutzbarkeit der Software gelegt. Deshalb wird die Anwendung durch die Eingabe eines vierstelligen Sicherheitscodes ge- und entsperrt. Dies bietet zwar nur bedingt Sicherheit, ist aber ausreichend, da die Anwendung zur jetzigen Zeit nur lokal ausgeführt wird.

4.1 Begrifflichkeiten

Bevor auf die einzelnen Aspekte der Anwendung eingegangen wird, werden in diesem Abschnitt Begriffe eingeführt, die im weiteren Verlauf verwendet werden.

Die Anwendung enthält ein *Basisverzeichnis* mit dem gearbeitet wird. Alle Änderungen die innerhalb dieses Verzeichnisses vorgenommen werden, werden von der Anwendung erkannt und entsprechend verarbeitet. Der Inhalt des Basisverzeichnisses wird im Weiteren als *lokale Daten* bezeichnet.

Die Bezeichnung *Account*, die in der Anwendung verwendet wird, beschreibt einen real existierenden Account bei einem Cloud-Storage-Dienst. Ein *Account* kann entweder unbenutzt oder einem *Array* zugewiesen sein. Es ist ebenfalls möglich, mehrere Accounts bei einem Anbieter mit der Anwendung zu benutzen.

Im Weiteren wird der Begriff *Array* verwendet, welcher eine Art Profil darstellt. Ein *Array* besteht aus einem Bezeichner, aus mehreren *Accounts* und aus einer Speicherstrategie. Im *Basisverzeichnis* der Anwendung wird ein Verzeichnis mit dem Namen des *Arrays* angelegt. Alle Änderungen die während der Laufzeit der Anwendung an diesem Verzeichnis erfolgen, werden entsprechend der Speicherstrategie an die verknüpften *Accounts* weitergegeben. Existiert zum Beispiel ein *Array* mit dem Namen „Uni“, den verknüpften Accounts „Account 1“ und „Account 2“ und dem Mirroring-Modus als Speicherstrategie, so wird eine Datei, die in das Verzeichnis des *Arrays* kopiert wird, automatisch an die entsprechenden Accounts weitergegeben.

1 http://www.playframework.org
2 http://commons.apache.org
3 http://hc.apache.org/httpcomponents-client-ga/
4 http://twitter.github.com/bootstrap/

4.2 Speicherstrategien

Der Benutzer hat die Möglichkeit, zwischen zwei Speicherstrategien in der Anwendung zu wählen, die im Folgenden beschrieben werden. Des Weiteren ist die Anwendung so aufgebaut, dass sie leicht um zusätzliche Speicherstrategien erweitert werden kann. Wie dies möglich ist, wird am Ende des Kapitels beschrieben.

4.2.1 Mirroring-Modus

Der sogenannte „Mirroring-Mode“ ermöglicht es, Daten automatisch bei mehreren Anbietern zu speichern. Der Benutzer wählt dazu mehrere Accounts aus und die Daten werden dann parallel auf diese übertragen. Es wird so also eine Ausfallsicherheit hergestellt, die es im Falle einer Störung eines Providers weiterhin ermöglicht, auf die Daten zuzugreifen. Beim Einsatz von RAID-1 wird durch die Verwendung von mehreren Festplatten ein Geschwindigkeitsgewinn erzielt, da einzelne Zugriffe auf die unterschiedlichen Platten verteilt werden können. Dies wurde bei der Realisierung des „Mirroring-Mode“ allerdings nicht umgesetzt.

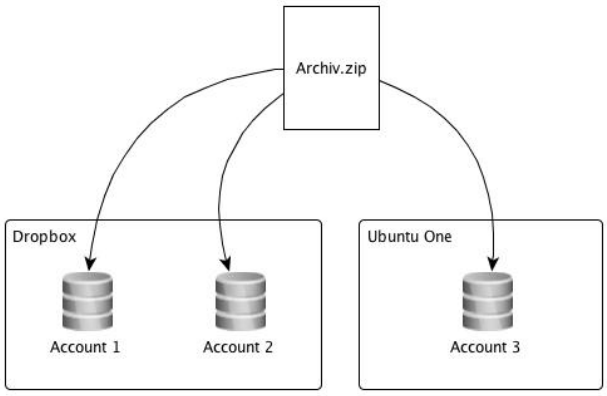

Abbildung 4.1: Mirroring-Modus

Der Abgleich der Daten erfolgt einseitig. Wenn eine lokale Datei angelegt wird, wird diese automatisch auf die Accounts bei den Cloud-Storage-Anbietern verteilt. Für den Fall, dass Daten bei einem der Cloud-Storage-Dienste gelöscht werden, bleiben sie auf der lokalen Festplatte, sowie auf den anderen Accounts erhalten. Änderungen an den lokalen Daten, die außerhalb der Laufzeit der Anwendung vorgenommen werden, werden beim Start der Anwendung erkannt und es erfolgt ein automatischer Datenabgleich mit den Cloud-Storage-Accounts. Werden in einem Array, dass den Mirroring-Modus nutzt, Accounts mit unterschiedlichen Größen verwendet, erfolgt eine Beschränkung des Speicherplatzes. Es wird als Limit die Größe des Accounts mit dem geringsten Speicherplatz gewählt.

4.2.2 Striping-Modus

Beim „Striping-Mode“ werden mehrere Accounts gebündelt um sie als einen großen Speicherplatz zu nutzen. Da alle untersuchten Anbieter eine Limitierung auf 2 oder 5 GB vorweisen, ermöglicht dieser Modus größere Datenmengen in der Cloud zu speichern ohne diese manuell auf mehreren Accounts zu verteilen. Die Funktionsweise ähnelt RAID-5, mit dem Unterschied, dass keine Ausfallsicherheit und keine Verbesserung der Performance erreicht wird.

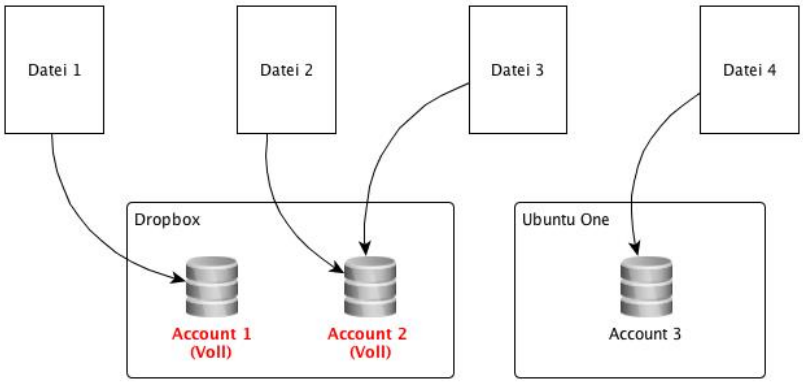

Abbildung 4.2: Striping-Modus

Der Anwendung steht eine Liste mit mehreren Cloud-Storage-Accounts zur Verfügung. Der erste freie Account in dieser Liste wird dazu verwendet, um Daten zu speichern. Vor jedem Kopiervorgang wird der Account auf seinen verfügbaren Speicherplatz geprüft. Unterschreitet der freie Speicherplatz einen gewissen Wert, wird automatisch der nächste Account in der Liste zum Speichern verwendet. Verfügt keiner der Accounts mehr über ausreichend freien Speicherplatz, erhält der Anwender eine Fehlermeldung. Genau wie beim „Mirroring-Mode“ wird beim Start der Anwendung eine Datenabgleich der lokalen Daten mit allen angeschlossenen Accounts vorgenommen. Die Dateien in der Cloud werden so angepasst, dass sie dem Stand der lokalen Daten entsprechen.

4.3 Verwalten von Accounts

Ausgangspunkt für die Verwaltung ist die Account Übersicht, welche in Abbildung 4.3 zu sehen ist. Dies ist die zentrale Ansicht der Anwendung, auf die man direkt nach dem Start gelangt. Von hier aus können Accounts hinzugefügt, editiert oder gelöscht werden. Man erhält ebenfalls einen Überblick über die Auslastung der einzelnen Accounts und über den Speicherplatz den alle Accounts zusammengerechnet bereitstellen.

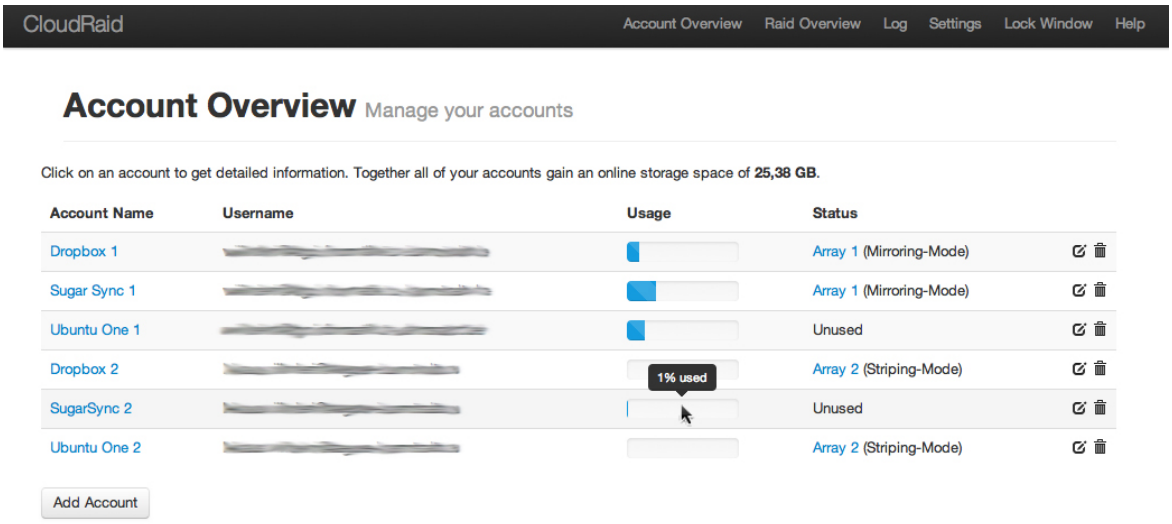

Abbildung 4.3: Account Übersicht

In der Spalte „Usage“ wird die allgemeine Auslastung des Accounts angezeigt. Sie bezieht sich nicht nur auf den Speicherplatz, der durch die Anwendung belegt ist, sondern auf den gesamten Platz. Fährt der Benutzer mit dem Mauszeiger über einen Eintrag in der Spalte, erscheint ein Tooltip, der die aktuelle Belegung anzeigt. Die Spalte „Status“ zeigt an, ob ein Account von einem Array benutzt wird. Ist das nicht der Fall, wird er als „Unused“ gekennzeichnet. Über die Navigationsleiste am Anfang der Seite hat der Benutzer die Möglichkeit, durch die Anwendung zu navigieren und auf die anderen Funktionalitäten zuzugreifen.

4.3.1 Account hinzufügen

Das Hinzufügen eines neuen Accounts erfolgt über den Button „Add Account“ in der Account Übersicht und beinhaltet drei Schritte. Zu Beginn wird der Benutzer aufgefordert, einen Provider auszuwählen, einen Benutzernamen einzugeben und einen entsprechenden Bezeichner für den Account einzugeben, was in Abbildung 4.4 zu sehen ist.

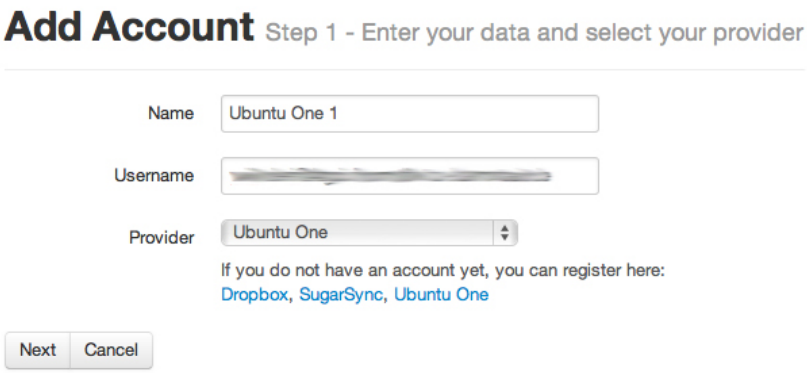

Abbildung 4.4: Eingabe des Benutzernamens und Auswahl des Providers

Entsprechend des ausgewählten Providers erfolgt als nächster Schritt die Weiterleitung auf die Anmeldeseite des Providers. Auf dieser muss sich der Benutzer anmelden, um der Anwendung Zugriff zu gewähren (Abbildung 4.5). Wählt der Benutzer SugarSync als Provider aus, erfolgt keine Weiterleitung und der Benutzer wird aufgefordert, sein Passwort direkt einzugeben. Dies hat den Hintergrund, dass SugarSync ein eigenes Authentifizierungsmodell verwendet.

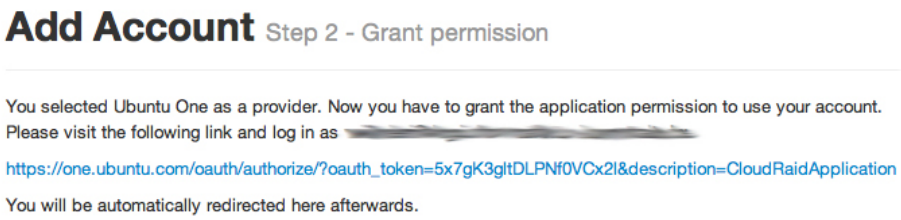

Abbildung 4.5: Weiterleitung zur Anmeldeseite des Providers

War die Anmeldung auf der Seite des Providers erfolgreich, muss der Benutzer bestätigen, dass er der Anwendung erlaubt, auf seine Ressourcen beim Provider zuzugreifen. Das entspricht dem Ablauf des OAuth Protokolls, was zuvor vorgestellt wurde. Wurde dieser Schritt ebenfalls durchgeführt, wird der Benutzer wieder auf die Seite der Anwendung zurück geleitet (Abbildung 4.6). Erst nach diesem letzten Aufruf ist der Vorgang abgeschlossen und der Account mit der Anwendung verbunden.

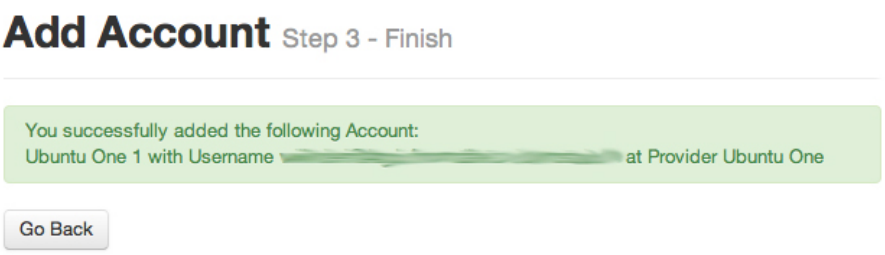

Abbildung 4.6: Bestätigung, dass der Account hinzugefügt wurde

4.3.2 Account editieren

Um einen bestehenden Account zu editieren, stehen zwei Wege zur Verfügung. Entweder durch Klicken auf den entsprechenden Button in der Account Übersicht (Abbildung 4.3) oder durch den Button „Edit Account“ in der Detailansicht eines Accounts (Abbildung 4.10).

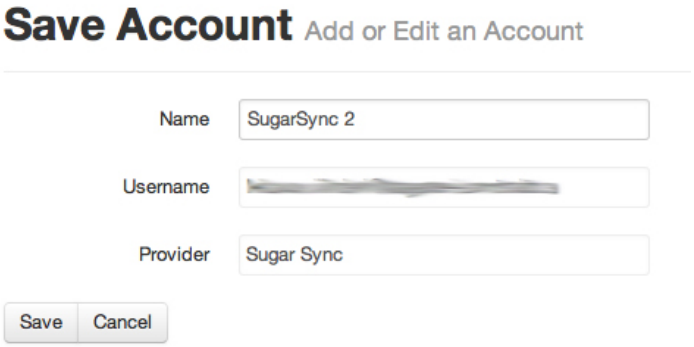

Abbildung 4.7: Account editieren

Es ist nur möglich den Bezeichner eines Accounts zu editieren (Abbildung 4.7). Möchte man die Zugangsdaten eines Accounts verändern, hat man nur die Möglichkeit, den Account zu löschen und erneut hinzuzufügen. Dies hat den Hintergrund, dass die Anwendung keine Zugangsdaten speichert, sondern sogenannte *Access Token*, die im Falle einer Änderung neu generiert und gespeichert werden müssen.

4.3.3 Account löschen

Das Löschen eines Accounts kann analog zum Editieren entweder über einen Button in der Account Übersicht oder in der Detailansicht des entsprechenden Accounts durchgeführt werden. Accounts die einem Array zugewiesen sind müssen erst aus diesem entfernt werden, bevor das Löschen durchgeführt werden kann. Der Benutzer wird darauf durch eine entsprechende Meldung aufmerksam gemacht, wenn er versucht einen Account zu löschen (Abbildung 4.8). Damit soll vermieden werden, dass ein Anwender versehentlich einen Account löscht, der sich noch in Benutzung befindet und er so nicht mehr auf Daten zugreifen kann.

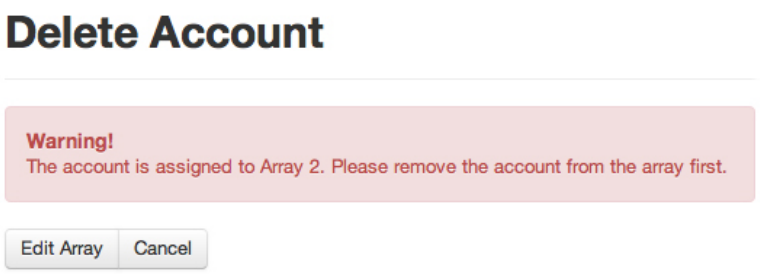

Abbildung 4.8: Fehler beim Löschen eines Accounts

Wird ein Account in der Anwendung entfernt, steht er nicht mehr zur Verfügung und die entsprechenden Zugangsdaten werden gelöscht. Der Benutzer hat dann die Möglichkeit, den Account wieder neu hinzuzufügen. Wurden durch die Anwendung Daten auf dem Account gespeichert, bleiben diese nach wie vor erhalten. Sollte ein Benutzer einen Account versehentlich löschen, gehen keine Daten verloren. Die Entfernung eines Accounts aus der Anwendung hat keinen Einfluss auf den Status beim Provider. Der Account ist weiterhin außerhalb der Anwendung benutzbar und es werden lediglich die Rechte gelöscht, sodass die Anwendung nicht mehr auf den Account zugreifen kann.

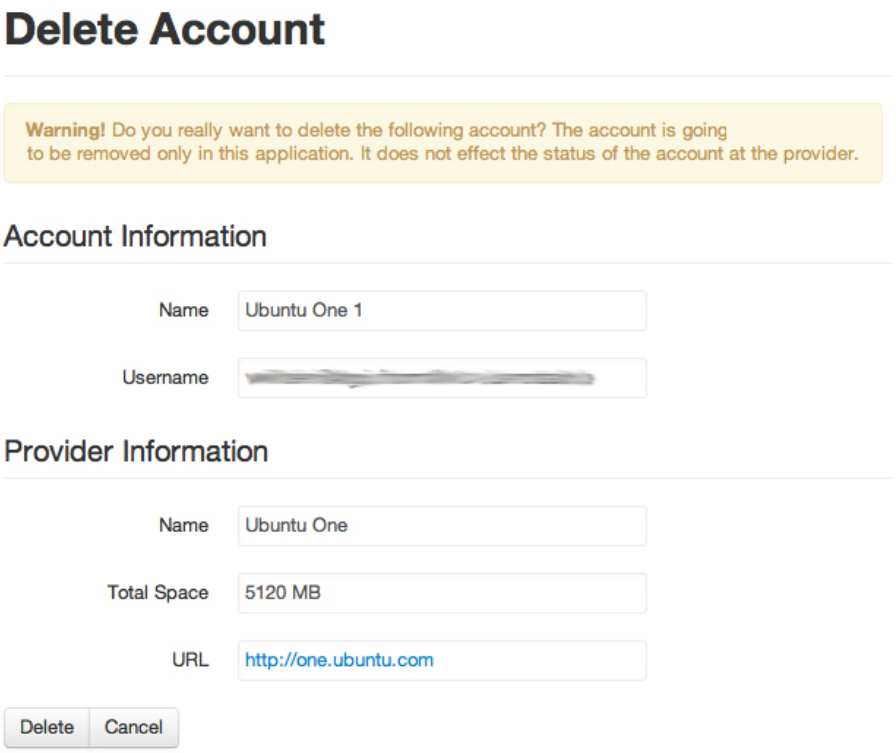

Abbildung 4.9: Bestätigung beim Löschen eines Accounts

4.3.4 Detailansicht eines Accounts

Die Detailansicht eines Accounts dient dazu, zusätzliche Informationen anzuzeigen, die in der Account Übersicht nicht enthalten sind. Dazu zählen Informationen über den Provider und eine detailliertere Anzeige über die Auslastung des Accounts (Abbildung 4.10).

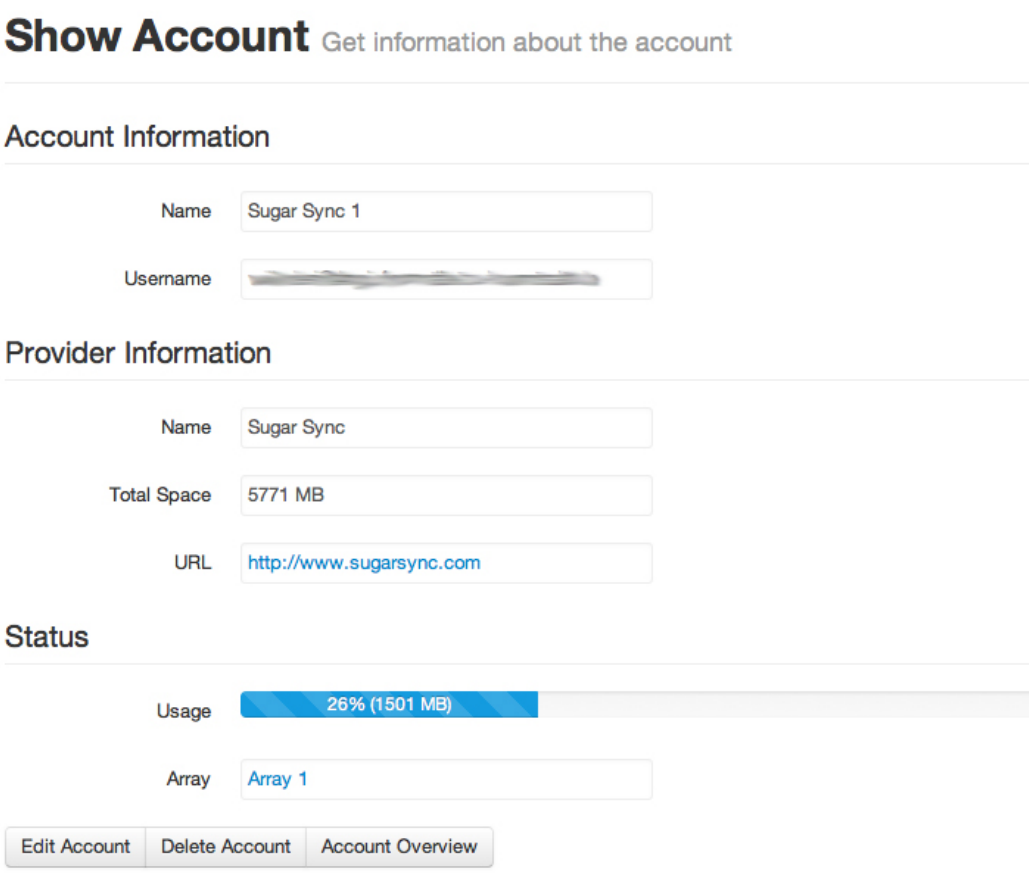

Abbildung 4.10: Detailansicht eines Accounts

4.4 Verwalten von Arrays

Das Verwalten von Arrays erfolgt über den Menüpunkt „Raid Overview“. Diese Übersicht bietet ähnlich wie die Account Übersicht eine Liste der aktuell definierten Arrays. Der Benutzer kann neue Arrays anlegen und bestehende editieren oder löschen. Zu jedem hier angezeigten Array befindet sich im Basisverzeichnis ein Ordner mit dem Namen des Arrays.

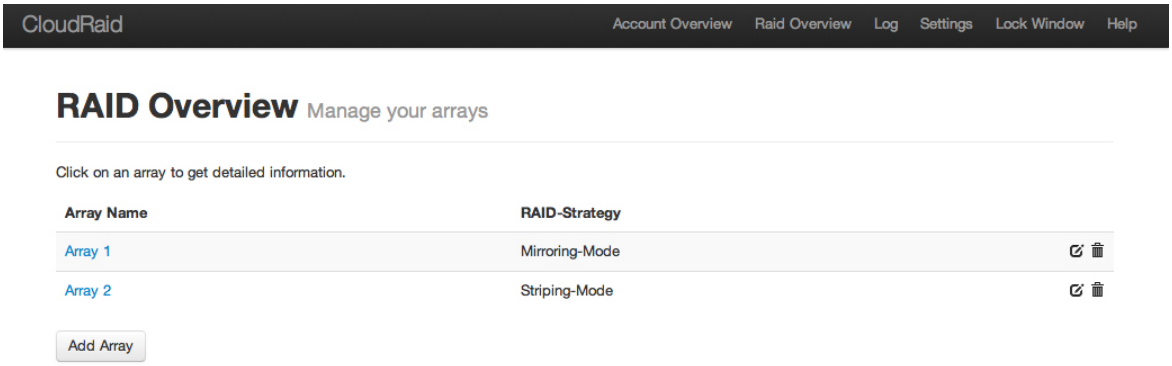

Abbildung 4.11: Array Übersicht

4.4.1 Array hinzufügen

Ein neues Array kann über den Button „Add Array“ innerhalb der Array Übersicht angelegt werden. Der Name dient zur Identifikation. Als „RAID Strategy“ wählt man eine der vorher vorgestellten Speicherstrategien. In der Accountliste darunter befinden sich alle Accounts, die sich aktuell nicht in Benutzung befinden. Es muss mindestens ein Account ausgewählt werden.

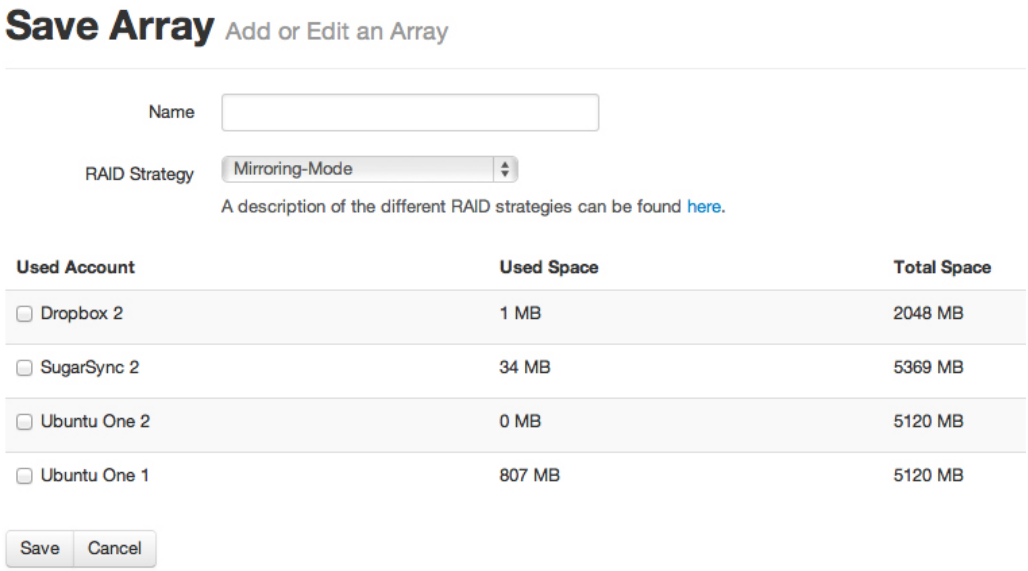

Abbildung 4.12: Array hinzufügen

Legt der Benutzer ein neues Array an, werden die ausgewählten Accounts diesem zugewiesen und erscheinen nicht mehr in der Liste der unbenutzten Accounts. Außerdem wird im Basisverzeichnis der Anwendung ein Ordner mit dem Namen des Arrays angelegt, welches während der Laufzeit der Anwendung auf Änderungen überwacht wird. Werden durch den Benutzer Dateien in das Verzeichnis kopiert, werden diese anhand der Speicherstrategie automatisch verarbeitet.

4.4.2 Array editieren

Die Editierung eines Array kann analog zur Editierung eines Accounts auf zwei Arten durchgeführt werden. Entweder durch den entsprechenden Button in der Array Übersicht oder über die Detailansicht des Arrays über den Button „Edit Array“. Der Benutzer erhält die gleiche Eingabemaske wie beim Hinzufügen eines Arrays (Abbildung 4.12). Er hat dort die Möglichkeit, den Namen eines Arrays oder die Liste der verwendeten Accounts zu editieren. Wird der Name des Arrays verändert, so wird der Name des Verzeichnisses im Basisverzeichnis angepasst. Passt der Benutzer die Liste der verwendeten Accounts an, so können Accounts dem Array hinzugefügt oder entfernt werden. Dies ist eine sehr praktische Funktionalität, wenn man den Striping-Mode verwendet. Sind zum Beispiel dem Array zwei Accounts zugewiesen, deren Speicherplatz voll belegt ist, können einfach weitere Accounts hinzugefügt werden und der Speicherplatz so auf einfachste Weise erweitert werden. Wird ein Account von einem Array entfernt, bleiben die in der Cloud hinterlegten Daten so lange erhalten, bis der Account einem neuen Array zugewiesen wird. Gleiches gilt bei der Löschung eines Arrays, was im nächsten Absatz beschrieben wird.

4.4.3 Array löschen

Möchte der Benutzer ein definiertes Array löschen, geschieht dies über die Array Übersicht oder über den Button „Delete Array“ in der Detailansicht. Bevor das Array gelöscht wird, erhält der Benutzer eine Übersicht mit Informationen über den Account und muss das Löschen dann explizit bestätigen, wie man in Abbildung 4.13 sehen kann.

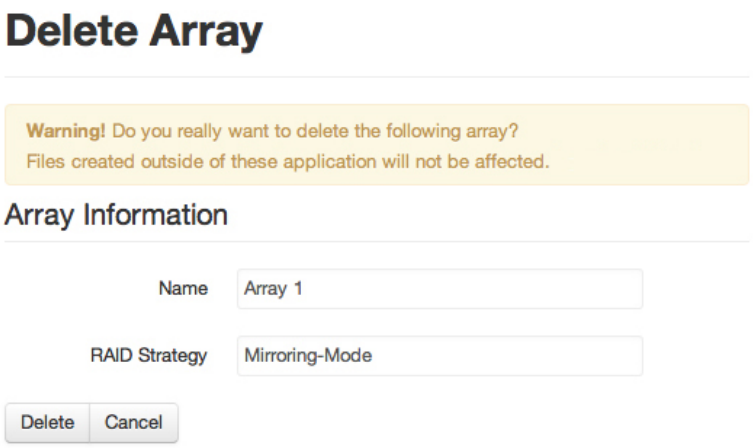

Abbildung 4.13: Array löschen

Die Accounts die dem Array zugewiesen sind, werden im Anschluss freigegeben und sind wieder für andere Arrays verfügbar. Dabei ist zu beachten, dass Daten, die durch die Anwendung auf dem Account gespeichert wurden, weiterhin dort verbleiben. Die lokalen Daten des Arrays im Basisverzeichnis bleiben ebenfalls erhalten. Das Verzeichnis wird lediglich nicht mehr von der Anwendung überwacht. Wird der Account zu einem späteren Zeitpunkt wieder einem Array zugewiesen, wird dieser mit dem neuen Array synchronisiert und die alten Daten werden dabei gelöscht.

Dieses Vorgehen hat den Vorteil, dass beim versehentlichen Löschen eines Arrays oder eines Accounts dieser nur wieder neu angelegt werden muss, ohne die lokalen Daten neu übertragen zu müssen.

4.4.4 Detailansicht eines Arrays

Abbildung 4.14 zeigt die Detailansicht eines Arrays. Darin erhält der Benutzer Informationen über den Namen und die Speicherstrategie des Arrays. Im Gegensatz zur Array Übersicht werden zusätzlich die verwendeten Accounts und deren Belegung angezeigt.

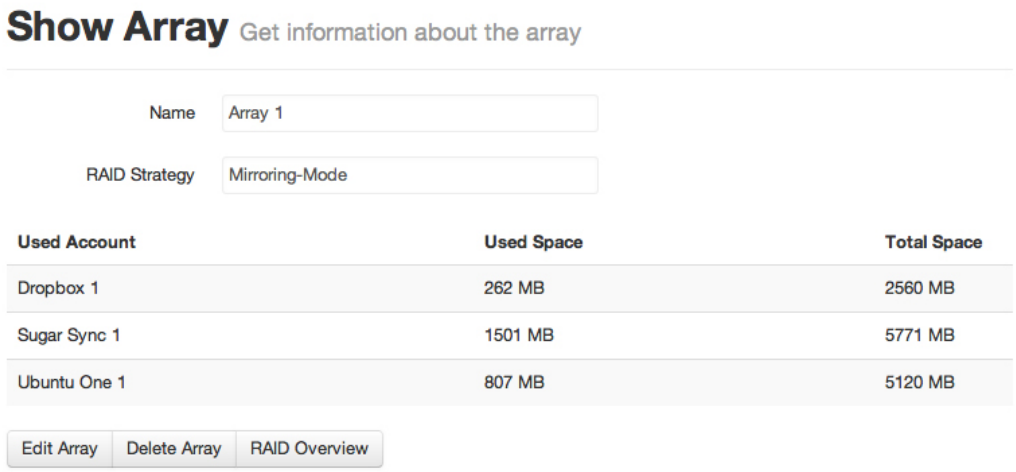

Abbildung 4.14: Detailansicht eines Arrays

4.5 Logging

Die Protokollansicht der Anwendung dient dazu, Informationen über die Vorgänge die sich im Hintergrund abspielen zu erhalten. Es werden Fehler, Warnungen und Interaktionen der Anwendung angezeigt. Dazu zählen das Erstellen und Löschen von Dateien und Ordnern sowie die Manipulation von Accounts und Arrays. Wird in einen der überwachten Ordner eine Datei kopiert, die größer als 2 MB ist, erscheint in der Protokollansicht ein Fortschrittsbalken, der den Fortschritt des Uploads der Datei anzeigt. Ist eine Datei kleiner als 2 MB entfällt dieser.

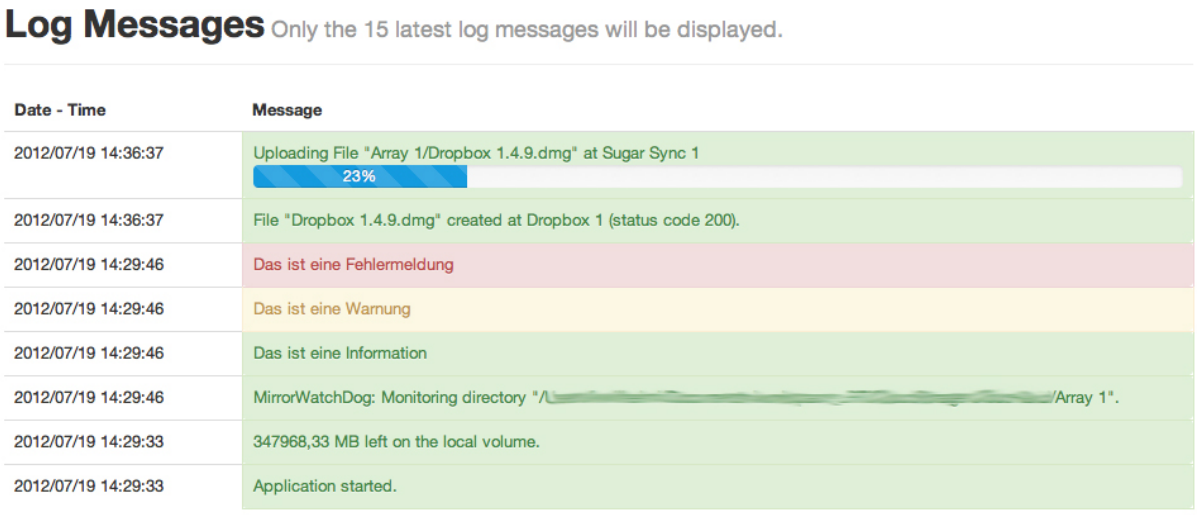

Abbildung 4.15: Protokollansicht

Mit den in Abbildung 4.16 aufgeführten Aufrufen lassen sich Protokolleinträge generieren. Die Klasse *LogEngine* setzt auf der Logging-Klasse des Play Frameworks auf, was bedeutet, dass Meldungen sowohl auf der Konsole des Frameworks als auch in der Protokollansicht der Anwendung angezeigt werden.

```
1 LogEngine.get().info("Das ist eine Information");
2 LogEngine.get().warn("Das ist eine Warnung");
3 LogEngine.get().error("Das ist eine Fehlermeldung");
```

Abbildung 4.16: Beispielcode zum Generieren von Protokolleinträgen

4.6 Einstellungen

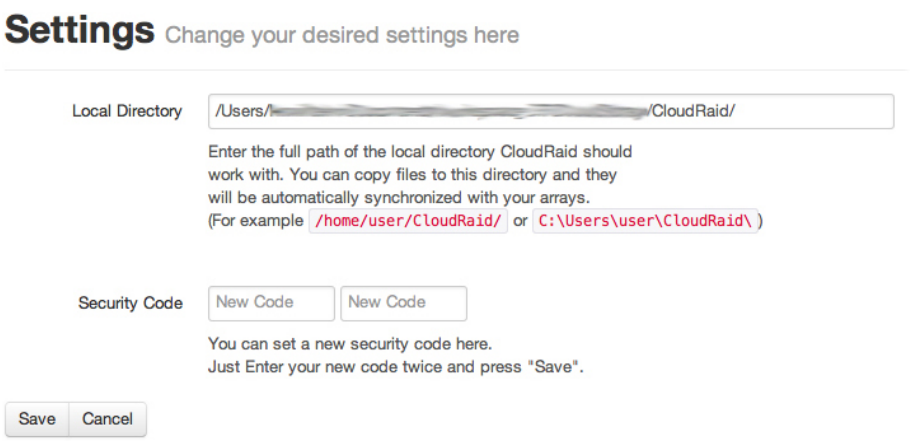

Abbildung 4.17: Einstellungsansicht

Basisverzeichnis ändern

Im Einstellungsfenster (Abbildung 4.17) ist unter dem Punkt „Local Directory“ das aktuelle Basisverzeichnis der Anwendung zu finden. Gibt der Benutzer hier ein neues Verzeichnis an, prüft die Anwendung ob das angegebene Verzeichnis existiert und legt es falls notwendig an. Beim ersten Start der Anwendung wird automatisch ein Ordner „CloudRaid“ im Verzeichnis in dem die Anwendung ausgeführt wird erstellt, welcher als Basisverzeichnis verwendet wird.

Im Fall einer Änderung des Basisverzeichnisses wird der Benutzer auf folgende Schritte aufmerksam gemacht, die durchgeführt werden müssen um mit dem neuen Verzeichnis zu arbeiten:

1. Die Anwendung anhalten
2. Falls notwendig Dateien manuell aus dem alten Basisverzeichnis in das Neue kopieren
3. Die Anwendung erneut starten

Sicherheitscode ändern

Die Anwendung ist durch einen vierstelligen Sicherheitscode gesichert, der beim Start der Anwendung abgefragt wird. Der Benutzer kann den Bildschirm jeder Zeit über die obere Navigationsleiste wieder sperren. Um den Code zu ändern, existieren im Abschnitt „Security Code“ zwei Eingabefelder, mit dessen Hilfe ein neuer Sicherheitscode festgelegt werden kann.

4.7 Integration neuer Dienste

Das zur Realisierung der Anwendung verwendete Framework arbeitet nach dem MVC Prinzip [11]. Daher sind Änderungen leicht zu realisieren und der Aufbau ist für Entwickler, die das erste Mal mit der Anwendung arbeiten, leicht zu verstehen.[5] Bei der Implementierung wurde versucht, den Code, der für die Interaktion mit den Cloud-Storage-Diensten zuständig ist, möglichst unabhängig vom Framework zu erstellen. Dies soll ermöglichen, den Code auch ohne das Framework zu benutzen. Um die Anwendung um einen neuen Dienst zu erweitern, muss das Interface *StorageProvider* implementiert werden. Die zu implementierenden Methoden, welche in Abbildung 4.18 zu sehen sind, sind im Interface selbst dokumentiert. Außerdem bieten die Klassen der bereits implementierten Provider einen Einblick in die Funktionsweise.

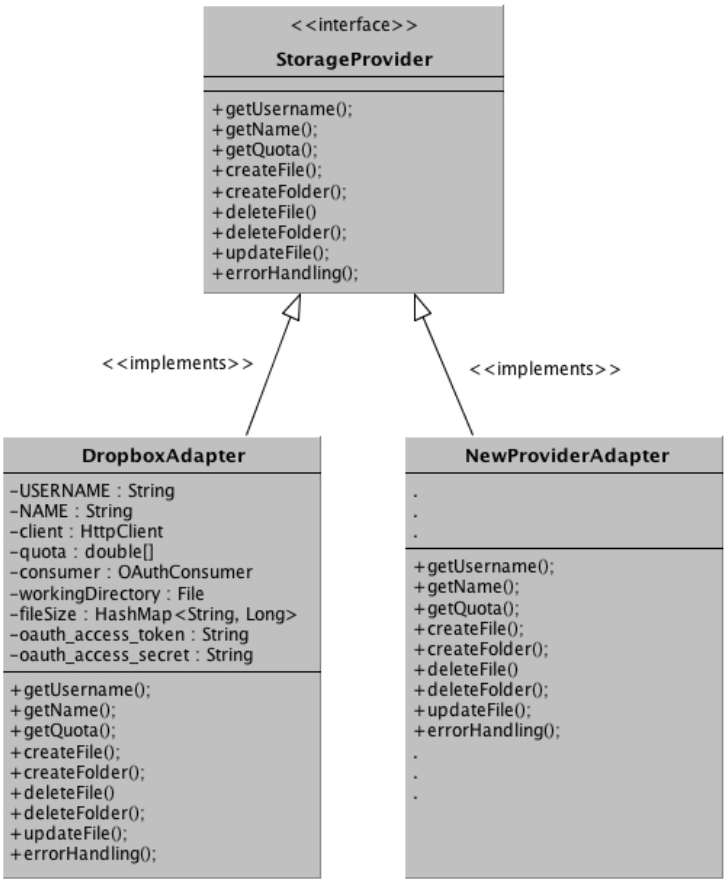

Abbildung 4.18: Interface zur Integration eines neuen Dienstes

Wie man anhand der Methoden in Abbildung 4.18 sieht, deckt das Interface *StorageProvider* nicht die Authentifizierung ab sondern nur die Dateioperationen sowie das Bestimmen der aktuellen Quota eines Accounts. Da der Authentifizierungsvorgang durch die verschiedenen eingesetzten Verfahren stark variieren kann, wurde für jeden Cloud Dienst eine eigene Klasse angelegt. Dies erwies sich am besten, da die Abläufe bei den einzelnen Providern zu unterschiedlich waren, um daraus einen einheitlichen Prozess in Form eines Interfaces oder einer abstrakten Klasse zu erstellen [32]. Beim Hinzufügen eines neuen Dienstes ist also individuell zu beachten, welches Verfahren zur Authentifizierung verwendet wird. Es muss eine entsprechende Klasse implementiert werden und ebenfalls der Assistent zum Hinzufügen von Accounts angepasst werden.

[5] http://www.playframework.org/documentation/1.2.4/main

4.8 Implementierung neuer Speicherstrategien

Abbildung 4.19 zeigt die Implementierung der Speicherstrategien, wobei es sich bei der Klasse *MirrorWatchDog* um den Mirroring-Mode und bei der Klasse *StripeWatchDog* um den Striping-Mode handelt. Das Integrieren einer neuen Speicherstrategie erfolgt durch Erstellen einer neuen Klasse, die von der abstrakten Klasse *WatchDog* abgeleitet ist und zusätzlich das Interface *FileAlterationListener* implementiert. Das Interface ist Teil des Apache Commons Pakets[6] und ist für die Überwachung der lokalen Verzeichnisse verantwortlich.

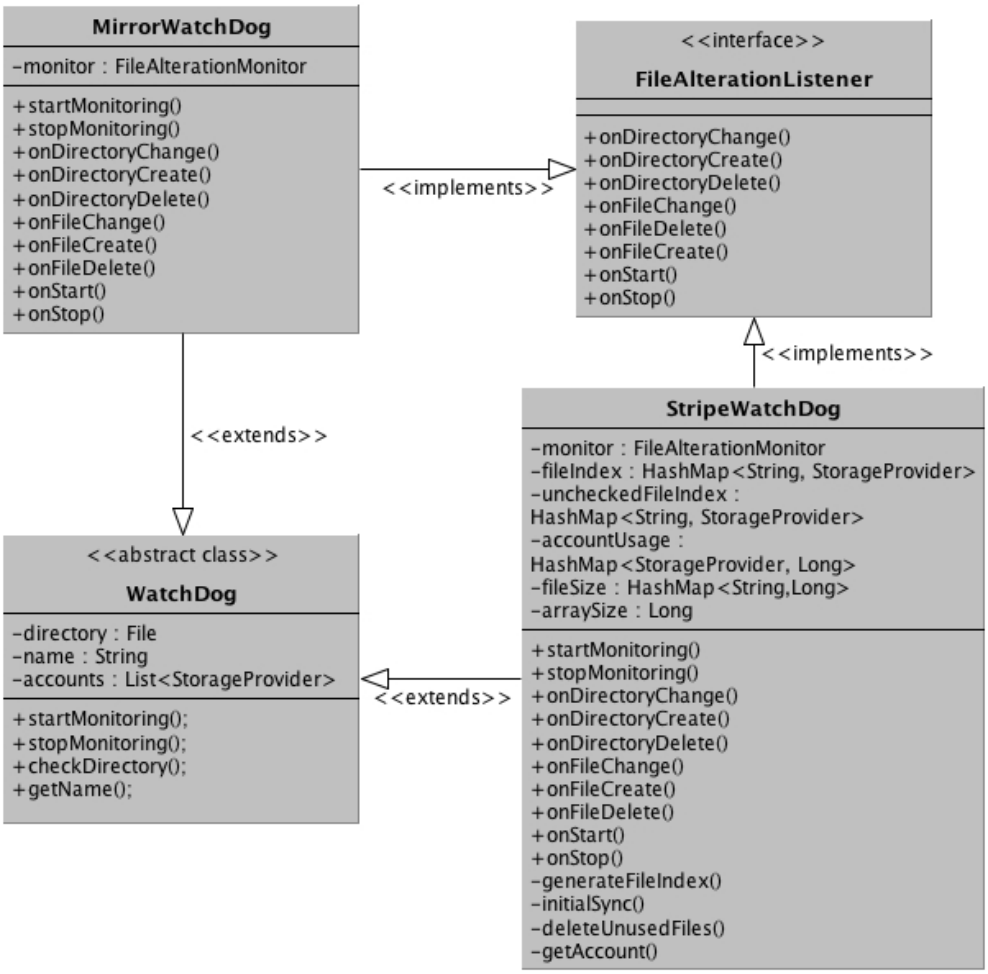

Abbildung 4.19: Realisierung der Speicherstrategien

In der neu erstellten Klasse müssen folgende Methoden, die von der abstrakten Klasse WatchDog geerbt wurden, implementiert werden:

- In der Methode *startMonitoring()* werden alle nötigen Schritte vorgenommen um die Überwachung des lokalen Verzeichnisses zu starten. Hier kann auch festgelegt werden, in welchem Zyklus auf Änderungen am Dateisystem geprüft wird.
- Die Methode *stopMonitoring()* sorgt dafür, dass beim Löschen eines Arrays oder beim Stoppen der Anwendung die Überwachung des Dateisystems korrekt beendet wird.

[6] http://commons.apache.org/io/

Das Interface *FileAlterationListener* erfordert die Implementierung folgender Methoden, die jeweils bei den verschiedenen Dateioperationen automatisch aufgerufen werden:

- Die Methoden *onFileChange()*, *onFileCreate()* und *onFileDelete()* werden aufgerufen, wenn eine Datei auf der lokalen Festplatte angelegt, verändert oder gelöscht wird.
- Die Methoden *onDirectoryChange()*, *onDirectoryCreate()* und *onDirectoryDelete()* werden aufgerufen, wenn ein Verzeichnis auf der lokalen Festplatte angelegt, verändert oder gelöscht wird.
- Die Methoden *onStart()* und *onStop()* werden jeweils vor und nach einer der zuvor beschriebenen Methoden aufgerufen. Diese Methoden können unter anderem dazu verwendet werden um redundanten Code zu vermeiden.

4.9 Integration in den Desktop

Zur Integration der Webanwendung in den Desktop wurde ein Prototyp entworfen, welcher in Abbildung 4.20 zu sehen ist. Dieser besteht aus einem Fenster, das die Anwendung enthält und einem Icon im System Tray des Betriebssystems. Dies bietet den Vorteil, dass der Anwender nicht in Kontakt mit der Konsole kommt und ein Programm hat, dass über eine Verknüpfung gestartet werden kann. Ohne die Desktoperweiterung müsste der Benutzer die Anwendung über eine Verknüpfung oder einen Aufruf in der Konsole starten und sie dann in einem Browser öffnen.

Zur Erstellung einer plattformunabhängigen Anwendung wurde das Standard Widget Toolkit[7] verwendet. Dadurch können verschiedene Pakete für alle gängigen Betriebssysteme erstellt werden, ohne für jedes eine eigene Implementierung erstellen zu müssen.

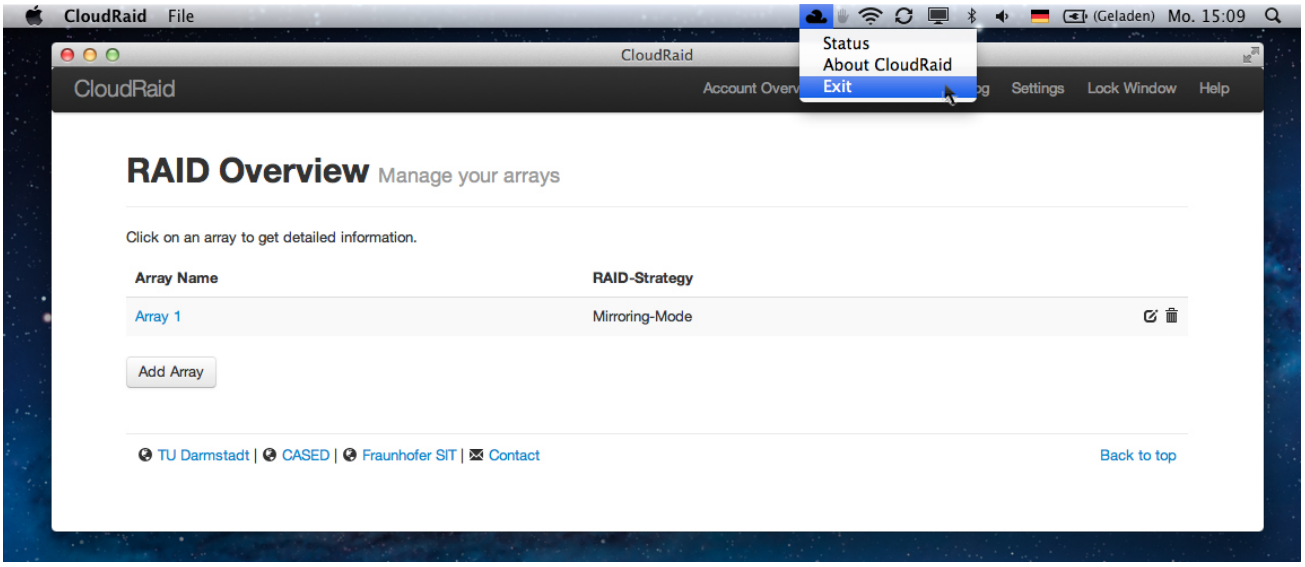

Abbildung 4.20: Prototyp der Desktopanwendung

[7] http://www.eclipse.org/swt/

5 Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde die in Kapitel 4 beschriebene Anwendung entwickelt, was eine Einarbeitung in die technischen Hintergründe erforderte. Darüber hinaus erfolgte vor Beginn der Implementierung eine Auswahl der verwendeten Hilfsmittel wie Bibliotheken und des Frameworks. Während der Entwicklungsphase erfolgte eine Auswahl von Cloud-Storage-Diensten, die von der Anwendung unterstützt werden sollten. Die hierfür durchgeführte Evaluation ergab, dass diese hinsichtlich der Features und des angebotenen Speicherplatzes sehr ähnlich sind. Bei der genauen Betrachtung stellten sich aber deutliche Unterschiede heraus. Hervorzuheben wären hier die fehlende Kontrolle der E-Mail-Adresse bei Dropbox und die fehlende Verschlüsselung der Daten auf den Servern von Ubuntu One.

Deutliche Mängel zeigten sich bei der Betrachtung der Programmierschnittstellen. Einzig bei der API von Dropbox fielen keine Probleme auf. Durch den Authentifizierungsvorgang der API von SugarSync begünstigt ließe sich eine Phishing-Anwendung erstellen, die die Zugangsdaten des Benutzers auslesen könnte [26]. Bei Ubuntu One könnte ein Angreifer zwar nicht die Zugangsdaten des Benutzers auslesen aber er könnte ebenfalls eine Anwendung erstellen, welche die online gespeicherten Daten des Benutzers auslesen könnte. Ein Benutzer muss zwar auf den Seiten von Ubuntu One einer Anwendung Zugriff gewähren aber er sieht dort nicht, um welche Anwendung es sich genau handelt und auf welche seiner Daten diese zugreift.

An die Implementierung der Anwendung wurde die Anforderung gestellt, leicht um neue Cloud-Storage-Dienste erweiterbar zu sein. Das sollte durch eine einheitliche Strukturierung erreicht werden. Nach genauerer Betrachtung der Dienste stellte sich aber heraus, dass diese recht unterschiedlich sind und eine einheitliche Implementierung nur bedingt möglich ist. In erster Linie lag das an den unterschiedlichen Authentifizierungsverfahren, welche beim Verknüpfen der Accounts mit der Anwendung verschiedene Interaktionen mit dem Benutzer erforderten. Das Hinzufügen eines neuen Providers ist nicht trivial aber gut umsetzbar.

Die Anwendung unterstützt aktuell die Anbieter Dropbox, SugarSync und Ubuntu One und die folgenden zwei Speicherstrategien. Der „Mirroring-Mode“ kopiert Dateien in einem lokalen Verzeichnis auf der Festplatte automatisch auf beliebig viele Cloud-Storage-Accounts. So wird eine Ausfallsicherheit einzelner Accounts hergestellt. Der „Striping-Mode“ kombiniert mehrere Accounts zu einem großen Speicherplatz. Die Auslastung der Accounts wird überwacht und Dateien werden automatisch auf die Accounts verteilt. Zusätzlich zu der bestehenden Anwendung gibt es noch weitere praktische Features, die Aufgrund der begrenzten Bearbeitungszeit nicht realisiert werden konnten. Einige davon werden jetzt im Folgenden erläutert.

5.1 Bandbreitenmessung

Webservices bieten unterschiedliche Bandbreiten und Reaktionszeiten, was bei Cloud-Storage-Diensten zu längeren Zeiten beim Up- oder Download von Dateien führt. Bei Benutzung der Anwendung war deutlich sichtbar, dass der Upload einer identischen Datei bei einem Dienst schneller ging als bei einem Anderen. Durch die Implementierung einer Bandbreitenmessung könnte so die Performance der Anwendung deutlich gesteigert werden. Die Anwendung könnte dann zum Beispiel nach dem Start eine Messung durchführen und zum Up- oder Download einer Datei den Provider mit dem besten Durchsatz wählen.

5.2 Verbesserte Synchronisierung

Die Benutzung der meisten Cloud-Storage-Anbieter erlaubt es, den Dienst auf mehreren Geräten gleichzeitig zu benutzen. Wird dabei beispielsweise über das Smartphone eine Datei gelöscht, wird der Computer des Anwenders über die Änderung informiert und der Datenbestand auf den neusten Stand gebracht. Dies funktioniert mit der in dieser Arbeit entwickelten Anwendung nicht. Sie kann nur von einem Computer genutzt werden und spiegelt die lokalen Daten entsprechend der Speicherstrategie auf die Accounts bei der Cloud-Diensten. Im Falle einer Weiterentwicklung der Anwendung wäre eine beidseitige Synchronisierung wünschenswert.

5.3 Datensicherheit

Eine interessante Erweiterung der Anwendung wäre eine Verschlüsselung der Daten, bevor diese zu den Diensten übertragen werden. Eine lokale Verschlüsselung der Daten wird von keinem der Provider angeboten, was sicher auch daran liegt, dass so der Einsatz von *Data Deduplication [33]* für den Provider unmöglich wäre. Wäre eine effiziente Verschlüsselung auch größerer Dateien in Java umsetzbar, könnte eine lokale Verschlüsselung durch eine neue Speicherstrategie leicht in die Anwendung integriert werden. Denkbar wäre eine Realisierung mit TruPax[1]. Es ermöglicht TrueCrypt kompatible Container-Dateien zu erstellen, ohne dass TrueCrypt auf dem Computer selbst installiert sein muss.

In der aktuellen Version der Anwendung verbleiben Dateien, die auf den Cloud-Storage-Accounts gespeichert wurden dort, wenn ein Account wieder aus der Anwendung entfernt wird. Hier könnte man dem Benutzer mehr Kontrolle darüber geben, indem man eine Auswahlmöglichkeit anbietet, die Daten beim Entfernen des Accounts von diesem zu löschen.

5.4 Erweiterung um neue Speicherstrategien

Die Implementierung neuer Speicherstrategien bietet eine Vielfalt an Erweiterungsmöglichkeiten. Die Anwendung lässt sich wie in Abschnitt 4.8 auf Seite 38 beschrieben, leicht um neue Strategien erweitern. Denkbar wären neben anderen RAID-Leveln zum Beispiel ein „Split-Mode“, welcher einzelne Dateien in mehrere Teile aufteilt und diese dann bei unterschiedlichen Cloud-Storage-Anbietern speichert. Somit wären alle Accounts erforderlich, um die Datei wieder herzustellen. Dies wäre ein naiver Ansatz der darauf abzielt, Daten vor dem Kompromittieren von einzelnen Accounts durch Angreifer zu schützen.

[1] http://www.coderslagoon.com

Literaturverzeichnis

[1] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A View of Cloud Computing. *Commun. ACM*, 53(4):50–58, April 2010.

[2] Moritz Borgmann, Tobias Hahn, Michael Herfert, Thomas Kunz, Marcel Richter, Ursula Viebeg, and Sven Vowé. On the Security of Cloud Storage Services. Technical report, Fraunhofer SIT, 2012.

[3] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible Markup Language (XML) 1.0 (Fifth Edition). `http://www.w3.org/TR/2008/REC-xml-20081126/`, 2008. [Online; Zugegriffen am 20.6.2012].

[4] F. Callegati, W. Cerroni, and M. Ramilli. Man-in-the-Middle Attack to the HTTPS Protocol. *Security Privacy, IEEE*, 7(1):78 –81, jan.-feb. 2009.

[5] D. Crockford. The application/json Media Type for JavaScript Object Notation (JSON). RFC 4627 (Informational), July 2006.

[6] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. Updated by RFCs 2817, 5785, 6266, 6585.

[7] The Internet Corporation for Assigned Names and Numbers. Hypertext Transfer Protocol (HTTP) Status Code Registry. `http://www.iana.org/assignments/http-status-codes/http-status-codes.xml`, 2012. [Online; Zugegriffen am 4.7.2012].

[8] The Internet Corporation for Assigned Names and Numbers. MIME Media Types. `http://www.iana.org/assignments/media-types/index.html`, 2012. [Online; Zugegriffen am 4.7.2012].

[9] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart. HTTP Authentication: Basic and Digest Access Authentication. RFC 2617 (Draft Standard), June 1999.

[10] Bundesamt für Sicherheit in der Informationstechnik. Sicherheitsempfehlungen für Cloud-Computing Anbieter. `https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Mindestanforderungen/Eckpunktepapier-Sicherheitsempfehlungen-CloudComputing-Anbieter.pdf`, 2012.

[11] Erich Gamma, Richard Helm, and Ralph E. Johnson. *Design Patterns - Elements of Reusable Object-Oriented Software*. Pearson Education, Amsterdam, 1995.

[12] David Gourley and Brian Totty. *Http - The Definitive Guide*. O'Reilly Media, Inc., Sebastopol, CA, 2002.

[13] E. Hammer, D. Recordon, and D. Hardt. The OAuth 2.0 Authorization Framework - draft-ietf-oauth-v2-26. `http://tools.ietf.org/html/draft-ietf-oauth-v2-26`, 2012. [Online; Zugegriffen am 26.6.2012].

[14] Eran Hammer. The OAuth 1.0 Guide. `http://hueniverse.com/oauth/guide/terminology/`, 2009. [Online; Zugegriffen am 25.6.2012].

[15] Eran Hammer. Introducing OAuth 2.0. http://hueniverse.com/2010/05/introducing-oauth-2-0/, 2010. [Online; Zugegriffen am 25.6.2012].

[16] E. Hammer-Lahav. The OAuth 1.0 Protocol. RFC 5849 (Informational), April 2010.

[17] Adrian Hannah. A Primer to the OAuth Protocol. *Linux J.*, 2011(206), June 2011.

[18] Brian Hayes. Cloud Computing. *Communications of the ACM*, 51(7):9–11, July 2008.

[19] Dropbox Inc. Dropbox - REST API Reference. http://www.dropbox.com/developers/reference/api, 2012. [Online; Zugegriffen am 30.6.2012].

[20] SugarSync Inc. SugarSync for Developers. http://www.sugarsync.com/dev/home.html, 2012. [Online; Zugegriffen am 22.6.2012].

[21] S. Josefsson. The Base16, Base32, and Base64 Data Encodings. RFC 4648 (Proposed Standard), October 2006.

[22] Canonical Ltd. Ubuntu One Developer Documentation Contents. http://one.ubuntu.com/developer/contents, 2012. [Online; Zugegriffen am 1.7.2012].

[23] Jörg Luther, Christian Vilsbeck, and Bernhard Haluschak. Raid im Überblick. http://www.tecchannel.de/storage/extra/401665/raid_sicherheit_level_server_storage_performance_festplatten_controller/, 2011. [Online; Zugegriffen am 11.7.2012].

[24] Michael Lyle. Redundancy in Data Storage. http://www.definethecloud.net/redundancy-1-raid-levels, 2010. [Online; Zugegriffen am 10.7.2012].

[25] Peter Mell and Tim Grance. The NIST Definition of Cloud Computing. Technical report, July 2009.

[26] Tyler Moore and Richard Clayton. Examining the impact of website take-down on phishing. In *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, eCrime '07, pages 1–13, New York, NY, USA, 2007. ACM.

[27] Alexander Reelsen. *Play Framework Cookbook*. Packt Publishing Ltd, Birmingham, UK, 2011.

[28] E. Rescorla. HTTP Over TLS. RFC 2818 (Informational), May 2000. Updated by RFC 5785.

[29] Leonard Richardson and Sam Ruby. *RESTful Web Services*. O'Reilly Media, Inc., Sebastopol, CA, 2007.

[30] Alex Rodriguez. RESTful Web Services: The Basics. https://www.ibm.com/developerworks/webservices/library/ws-restful/, 2012. [Online; Zugegriffen am 15.6.2012].

[31] Andrew S. Tanenbaum. *Computernetzwerke*. Pearson Studium, München, 4. überarb. A. edition, 2003.

[32] Christian Ullenboom. *Java ist auch eine Insel - Programmieren mit der Java Platform, Standard-Edition 6*. Galileo Press, Bonn, 7. aufl. edition, 2008.

[33] Wenying Zeng, Yuelong Zhao, Kairi Ou, and Wei Song. Research on Cloud Storage Architecture and Key Technologies. In *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, ICIS '09, pages 1044–1048, New York, NY, USA, 2009. ACM.

Abbildungsverzeichnis

Tabellenverzeichnis