Single Sign-On Lösung für Django

Semesterarbeit in Informatik

Studierender - Marc Egli

Auftraggeber - Silvan Spross

Projektbetreuer - Lukas Eppler

HSZ-T - Technische Hochschule Zürich

Februar 2012 bis Juli 2012

Zusammenfassung

Diese Semesterarbeit widmet sich einem den meisten Webagenturen, welche für ihre Kunden viele Webseiten betreuen, bekannten Problem: Zugangsdaten werden oftmals zentral und allen Mitarbeitern, welche Zugang zu den Webseiten benötigen, zugänglich abgelegt um den schnellen Zugriff auf Administrationsbereiche zu gewährleisten. Dies geschieht zum Wohle des Kunden und im Vertrauen darauf, dass alle Mitarbeiter mit guter Absicht handeln und auch nach dem Verlassen der Agentur nicht die ihnen anvertrauten Zugangsdaten missbrauchen.

Als Ergebnis dieser Arbeit entstand django-admin-sso. Es handelt sich dabei um eine Django Applikation welche Mitarbeiter über ein zentrales System autorisiert und dadurch eine Lösung für das zuvor erwähnte Problem bietet.

Inhaltsverzeichnis

1. Personalienblatt

Name, Vorname:	**Egli, Marc**
Adresse:	**Altstetterstrasse 257**
PLZ, Wohnort:	**8048 Zürich**
Geburtsdatum:	**13.11.1983**
Heimatort:	**Winterthur**

2. Bestätigung

Hiermit bestätige ich, Marc Egli, dass ich die vorliegende Semesterarbeit “Single Sign-On Lösung für Django” im Rahmen der geltenden Reglemente selbstständig erarbeitet habe.

Zürich, den 17. Juli 2012

Marc Egli

3. Einleitung

3.1. Ausgangslage

Die Agentur allink Betreibt für ihre Kunden diverse Django basierte Webapplikationen. Diese Applikationen besitzen jeweils einen Administrations-Bereich, in welchem man sich mit Benutzername und Passwort einloggen kann. Da allink anfangs 2012 fast 100 solche Webapplikationen in Betrieb hatte, wurden vielfach die selben Login-Daten wiederverwendet.

3.2. Problemstellung

Nicht alle verwendeten Login-Daten sind sauber dokumentiert. Darum muss man teilweise den Entwickler einer Applikation nach den Login-Daten fragen oder die gängigsten Passwörter durchprobieren um Zugriff auf den Administrations-Bereich zu erhalten. Zudem gibt es keine Möglichkeit einem Mitarbeiter welcher die Agentur verlässt den Zugang zu sperren, ohne dass für sämtliche Applikationen die Zugangsdaten geändert werden müssen.

3.3. Zielsetzung

Das Hauptziel dieser Arbeit besteht darin den Zugang zu den Administrations-Bereichen über ein zentrales System zu Regeln. Dadurch kann sich jeder Mitarbeiter mit seinen eigenen Zugangsdaten an allen Administrations-Bereichen anmelden.

4. Planung

Zu Beginn der Arbeit standen zwei Termine fest. Vom 16.4.2012 bis am 4.5.2012 musste ich die Arbeit unterbrechen, da ich in dieser Zeit meinen letzten "Fortbildungsdienst der Truppe"[1] bei der Schweizer Armee leistete. Die Arbeit sollte spätestens in der Woche vom 16.7.2012 abgeben werden, da ich danach drei Wochen abwesend sein werde.

4.1. Projektplan

Das Projekt wurde nur auf Wochen genau geplant, da zu Beginn der Arbeit nicht klar war, wann ich wie viel Zeit in die Arbeit investieren kann. Diese Planung ist in Abbildung 4.1 zu sehen. Die gesetzten Meilensteine wurden nicht alle termingerecht erreicht. Jedoch war die Implementierung mit einer Woche Verspätung fertig und auch bereits in der allink im Einsatz.

4.2. Zeitaufwände

Die Tabelle 4.1 zeigt die für die Semesterarbeit aufgewendete Zeit. Nicht in dieser Tabelle enthalten sind diverse kleinere Aufwände, Sitzungen und Schulungen in der allink und der Lightningtalk an der Djangoconeu 2012.

[1] `http://www.vtg.admin.ch/internet/vtg/de/home/militaerdienst/dienstleistende/dienstleistungspflicht/sdt.html`

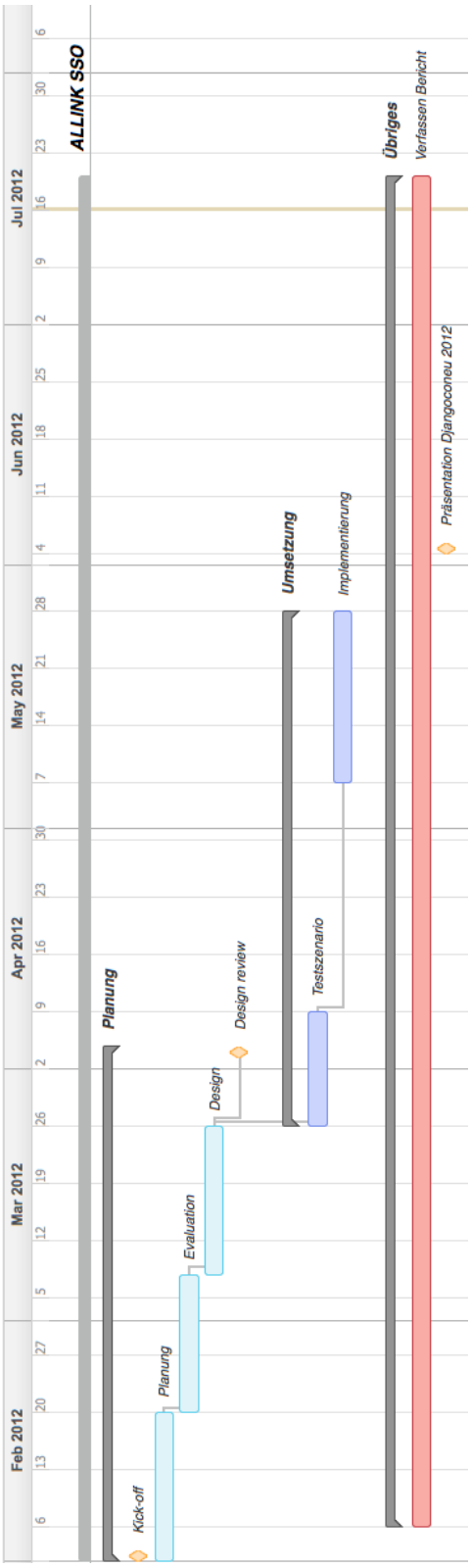

Abbildung 4.1.: Zeitplan

Planung	14h
Evaluation	10h
Design	12h
Testing	4h
Implementation	31h
Dokumentation	50h
Total	121h

Tabelle 4.1.: Zeitabrechnung

5. Analyse

In diesem Kapitel wird die Situation der allink vor Aufnahme dieser Semesterarbeit beschrieben. Das Diagramm in Abbildung 5.1 zeigt alle bei allink benutzten Informationssysteme.

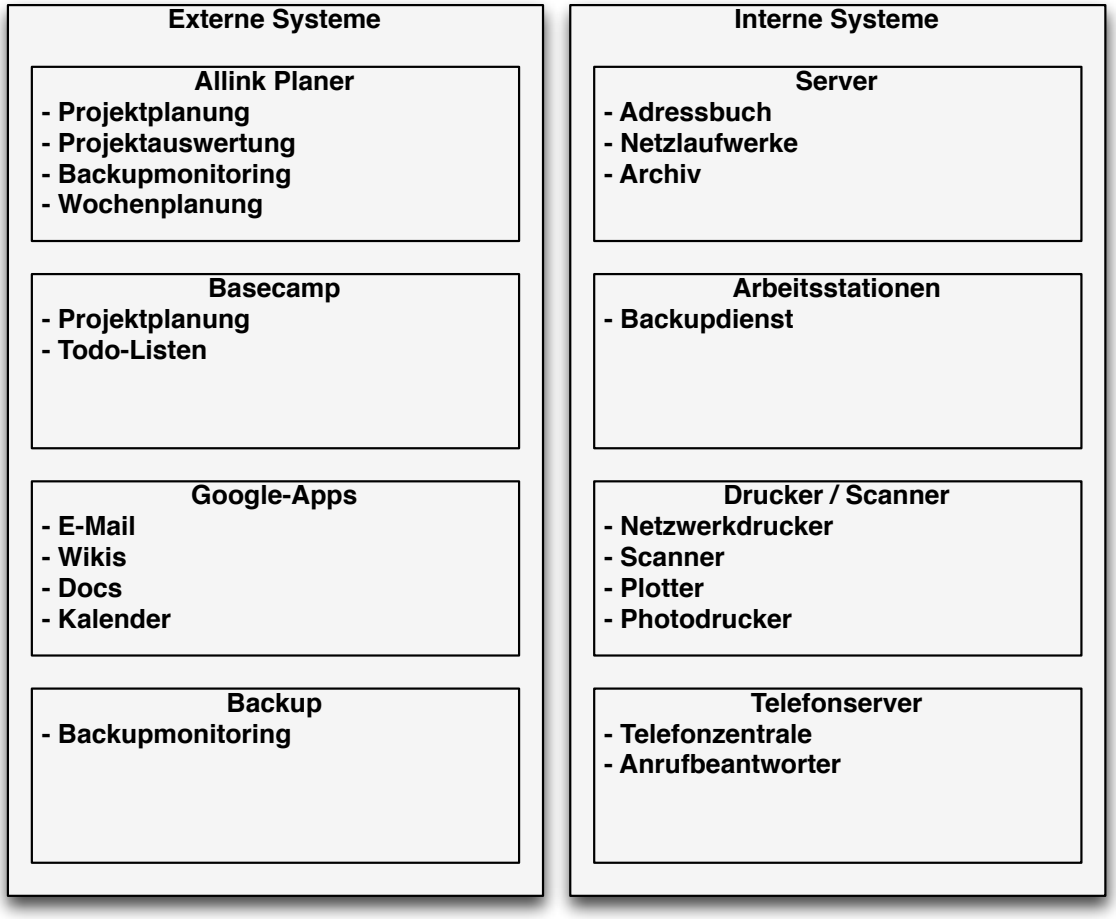

Abbildung 5.1.: IT-Systeme in der allink

5.1. Systeme mit Benutzermanagement

Viele Systeme welche in der allink verwendet werden, sind nicht für die Verwendung als Single-Sign-On System geeignet. So hat zum Beispiel der Telefon Server sehr wohl Benutzerkonten, jedoch sind diese in den Telefon Handgeräten durch die Informatik konfiguriert und die meisten Mitarbeiter wissen somit nicht, dass sie ein Benutzername und ein Passwort für den Telefondienst besitzen. Nachfolgend wird auf drei Systeme eingegangen welche von den Mitarbeitern bewusst verwendet werden und über eine Benutzerverwaltung verfügen.

5.1.1. Google Apps

allink verwendet aus Google Apps folgende Tools.

- Google Mail
- Google Calendar
- Google Sites
- Google Docs
- Google Analytics

Google Apps sind bei allink schon seit mehreren Jahren im Einsatz. Google Docs hat Microsoft Office für den internen Gebrauch vollständig abgelöst.

5.1.2. Basecamp

Mit Basecamp werden Milestones und Todo-Listen für sämtliche Projekte verwaltet. Zudem werden sämtliche Arbeitsaufwände darin erfasst um am Ende eines Projekts einen Überblick über alle Arbeitsleistungen zu haben. Basecamp ist seit mehr als zwei Jahren im Einsatz, wird jedoch nicht von allen Mitarbeitern konsequent genutzt.

5.1.3. Mac OS X Server

Allink verwendet ausschliesslich Desktop und Notebook Computer der Marke Apple. Für die zentrale Dateiablage wird ein Mac Mini verwendet, auf welchem neben der Dateiablage auch noch ein Adress-Server und weitere Dienste laufen. Die meisten Dienste welche ursprünglich auch auf diesem Server liefen, wurden durch Dienste von Google Apps ersetzt.

System	Anzahl
Google Apps	10
Basecamp	5
Mac OS X Server	4

Tabelle 5.1.: Anzahl Mitarbeiter welche ihre Logindaten kennen

5.2. Mitarbeiter

Da die Mitarbeiter von allink die zukünftigen Benutzer des zu entwickelnden Systems sein werden, wurden 11 Mitarbeiter befragt, welche Passwörter und Logindaten ihnen bekannt sind. Aus den Resultaten der Befragung, welche in Tabelle 5.1 zu sehen sind, geht hervor, dass die meisten Benutzer ihre Logindaten für Google Apps auswendig kennen und nur wenige Mitarbeiter sich bewusst sind, dass sie auf dem zentralen Fileserver ein Konto besitzen.

5.3. Login Prozess

Die meisten Mitarbeiter kennen nicht die Passwörter sämtlicher Webapplikationen. Darum ist ein zeitraubendes Nachfragen der Daten notwendig falls man nicht mehr eingeloggt ist. Eine Darstellung dieses Vorgehens ist in Abbildung 5.2 zu sehen.

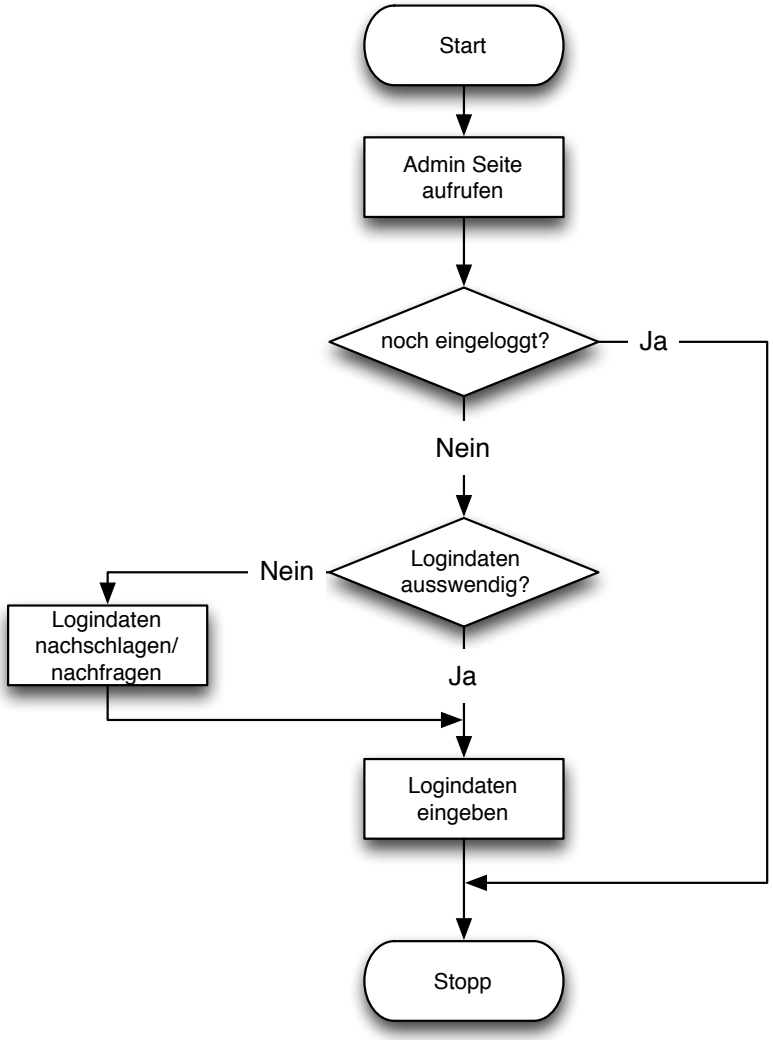

Abbildung 5.2.: Login Prozess vor dieser Arbeit

6. Anforderungen

Die Anforderungen an eine Single-sign-On Lösung wurden bereits lange vor dieser Arbeit durch die technische Leitung der allink festgelegt. In diesen Anforderungen wird unter Benutzer ein Mitarbeiter der allink verstanden und unter Administrator ein Mitglied der technischen Leitung.

6.1. Funktionale Anforderungen

Die Single-sign-On Lösung sollte mindestens folgende Funktionalitäten erfüllen:

1. Ein Benutzer kann sich mit seinen globalen Login-Daten an sämtlichen Webapplikationen der allink anmelden für die er berechtigt ist.
2. Grundsätzlich sind alle Benutzer berechtigt sich an allen Webapplikationen anzumelden. Dies kann für einzelne Applikationen weiter eingeschränkt werden.
3. Lokale Benutzerkonten, welche mit einem eigenen Passwort funktionieren, werden nur noch für die Kunden von allink verwendet.
4. Ein neuer Benutzer kann an einem zentralen System erfasst werden. Er hat sofort die Rechte für alle Webapplikationen welche keine besonderen Beschränkungen haben.
5. Ein Benutzer kann zentral gesperrt werden. Danach ist es ihm nicht mehr möglich sich anzumelden. Bestehende Sitzungen werden nicht abgebrochen.

6.2. Technische Anforderungen

Da diese SSO-Lösung für sehr viele Projekte eingesetzt werden soll, ist es wichtig, dass der nötige Aufwand um sie in einzelne Projekte zu integrieren möglichst klein gehalten wird.

7. Evaluation

Anwendungen welche mit externen Systemen kommunizieren, müssen auf einem Standard aufbauen welcher auch zukünftig noch unterstützt wird. Darum wurden zuerst die verschiedenen Standards betrachtet und danach mittels einer Nutzwert-Analyse ein System ausgewählt. Ein wichtiger Punkt für die Wahl eines Standards ist, dass die Integration in ein bestehendes Projekt möglichst einfach mit wenig Konfigurationsaufwand erreicht werden kann.

7.1. Wahl eines Systems

Es gibt diverse Standards mit welchen man einen Benutzer identifizieren kann. Jedoch sind nicht alle davon für Webapplikationen geeignet. Um die Übersicht zu wahren, wird hier nur auf zwei Typische Web Standards eingegangen. Damit auch eine Authentifizierung über Mac OS X Server erreicht werden könnte, wird zusätzlich noch kurz auf LDAP eingegangen.

OpenID

Der offene Standard Openid wird von der OpenID Foundation[1] entwickelt. Diese Stiftung vermarktet, schützt und pflegt die OpenID Gemeinschaft und deren Technologien. Der Standard wurde dafür entwickelt, es einem Benutzer welcher über eine OpenID verfügt zu ermöglichen, sich an Webseiten welche OpenID unterstützen anzumelden, ohne dass er sich einen neuen Benutzername und ein neues Passwort ausdenken muss. Das System welches die OpenID zur Verfügung stellt, wird dabei OpenID-Provider und die Webseiten an der sich der Benutzer anmeldet OpenID-Relying-Parties genannt.

OpenID ist weit verbreitet und wird von vielen grösseren Webportalen unterstützt. Normalerweise erhält die OpenID-Relying-Party nachdem sich ein Benutzer über OpenID eingeloggt hat, einen Identifier in form einer URL. Mit der attribute exchange Erweiterung lassen sich weitere Informationen über den Benutzer vom OpenID-Provider beziehen.

[1] `http://openid.net/foundation/`

OAuth

Auch OAuth[2] ist ein offener Standard und ist zur Zeit in der Version 1.0 im RFC 5849(HL10) aktuell. Jedoch verwenden die meisten Dienste heute schon OAuth 2.0 welches als Working Draft[3] verfügbar ist. OAuth ist nicht für Authentifizierung gebaut sondern für Autorisierung. Es ermöglicht einem Benutzers einer Webapplikation die Vergabe von Rechten an einen weiteren Dienst, damit dieser bei der ersten Webapplikation API Funktionen aufrufen kann, auf welche er ohne das Einverständnis des Benutzers keinen Zugriff hätte. Es lässt sich mit OAuth eine Pseudo-Authentifizierung durchführen, welche darauf beruht, dass nur der Benutzer diese Rechte erteilen kann. Dieses System wird von Facebook für die "Login with Facebook" verwendet.

LDAP

Obwohl es unüblich ist für Webapplikationen LDAP(Ser06) zu verwenden, wurde diese Variante ebenfalls geprüft, da dies der einzige praktikable Weg wäre, um eine Single-Sign-On Lösung mit einem Mac OS X Server als Kontenverwaltungsdienst zu erstellen.

7.2. Nutzwert-Analyse

Da mehrere Systeme in der allink vorhanden sind, welche sich für ein zentrales Login eignen, wurde eine Nutzwert-Analyse durchgeführt um die einzelnen Systeme miteinander zu vergleichen. Dabei wurde vor allem Wert darauf gelegt, dass sich die Mitarbeiter nicht ein weiteres Login merken müssen.

7.2.1. Gewichtung

Die Gewichtung der einzelnen Punkte wurde in Zusammenarbeit mit der allink Geschäftsleitung erarbeitet. Jedoch war zu diesem Zeitpunkt noch nicht bekannt, welches System bei welchem Punkt seine Stärken ausspielen kann. Die so festgelegte Gewichtung ist in Abbildung 7.1 zu sehen.

[2] `http://oauth.net`
[3] `http://tools.ietf.org/html/draft-ietf-oauth-v2-26`

1	Adaption innerhalb des Betriebs	60%	1.1	Mittarbeiter kennen ihre Credentials	50%
			1.2	Tool wird von den Mitarbeitern verwendet	30%
			1.3	Adminbereich ist bekannt	20%
2	Technische Möglichkeiten	40%	2.1	OpenID	45%
			2.2	OAuth	35%
			2.3	anderes System	20%

Abbildung 7.1.: Gewichtung der einzelnen Faktoren

7.2.2. Bewertung

Die zuvor beschriebenen Systeme wurden gemäss der in der Gewichtung genannten Punkte bewertet. Jeder Punkt der Nutzwert-Analyse wurde nach der Skala in Tabelle 7.1 klassiert. Nachfolgend wird auf die einzelnen Punkte eingegangen.

0	Punkt nicht erfüllt
1	Punkt ungenügend erfüllt
2	Punkt genügend erfüllt
3	Punkt erfüllt
4	Punkt mehr als erfüllt

Tabelle 7.1.: Skala für die Punktevergabe

Mitarbeiter kennen ihre Credentials

Das System soll den Mitarbeitern die Arbeit erleichtern. Die Bewertung für diesen Punkt wurde anhand der Umfrage aus Kapitel 5.2 erstellt.

Google Apps	3
Basecamp	2
Mac OS X Server	1

Tool wird von den Mitarbeitern verwendet

Unter diesem Punkt wurde beurteilt von wie vielen Mitarbeitern das geprüfte System bereits im Arbeitsalltag eingesetzt wird.

Google Apps wird zur Zeit von allen Mitarbeitern verwendet, da darüber die E-Mail Dienste laufen. Zudem verwendet die IT und die Geschäftsleitung Google Docs und Google Sites.

Basecamp wird vor allem von der Projektleitung verwendet. Der Rest der Mitarbeiter verwendet Basecamp vor allem für die Zeiterfassung, welche aber über ein Widget geschieht und so nicht viel mit Basecamp zu tun hat.

Mac OS X Server wird von allen Mitarbeitern für den Austausch von Dateien und für das Backup verwendet. Jedoch sind sich die meisten Mitarbeiter nicht bewusst wo sie dieses System verwenden da sie sich dafür nie anmelden müssen.

Google Apps	4
Basecamp	2
Mac OS X Server	2

Administrations-Bereich ist bekannt

Bei jedem Zu- und Abgang eines Mitarbeiters bei allink muss ein Benutzerkonto errichtet bzw. ein Benutzerkonto wieder gesperrt werden. Darum ist es notwendig, dass zumindest die Geschäftsleitung mit den entsprechenden Tools vertraut ist.

Google Apps wird von der Informatik Leitung regelmässig verwendet. Daneben wird gelegentlich Google Apps für Kunden eingerichtet und jeder Informatiker kennt sich im Administrationsbereich aus. Zudem sollte auch die gesamte Geschäftsleitung über Grundkenntnisse verfügen.

Basecamp wird vor allem von der Projektleitung verwaltet. Das Erstellen eines neuen Mitarbeiters ist jeweils die Aufgabe der Informatik Leitung.

Mac OS X Server wird nur von der Informatik Leitung konfiguriert und wird sonst von niemandem verwendet.

Google Apps	4
Basecamp	2
Mac OS X Server	1

OpenID

OpenID ist die erste Wahl, da es in Python schon unterstützt wird und es dafür konzipiert ist Benutzer über ein zentrales System anzumelden.

Google Apps unterstützt OpenID und wird auch auf der Webseite der OpenID Foundation gelistet.

Basecamp unterstützte früher OpenID hat jedoch die Unterstützung am 1. Mai 2011 [4] eingestellt.

Mac OS X Server unterstützt ohne zusätzliche Software kein OpenID.

Google Apps	3
Basecamp	0
Mac OS X Server	1

OAuth

Obwohl OAuth eigentlich nicht als Authentifizierungs-Protokoll gedacht ist. Kann es dafür verwendet werden indem man vom Provider die entsprechenden Daten bezieht. Dies ist jedoch zweite Wahl falls sich keine Möglichkeit bietet OpenID zu verwenden.

Google Apps lässt über OAuth Zugriff auf Benutzerdaten und Schnittstellen zu.

Basecamp lässt über OAuth Zugriff auf die gesamte API zu.

Mac OS X Server unterstützt kein OAuth.

Google Apps	3
Basecamp	3
Mac OS X Server	0

[4] `http://productblogarchive.37signals.com/products/2011/01/well-be-retiring-our-support-of-openid-on-may-1.html`

Anderes System

Für den Fall, dass keines unserer Systeme geeignet ist um einen Benutzer mit OpenID oder OAuth zu authentifizieren, wurden weitere Möglichkeiten unserer Systeme überprüft.

Google Apps bietet neben den schon erwähnten Möglichkeiten noch Unterstützung für SAML[5] jedoch zur Zeit nur um Benutzer von Google Diensten über ein Drittsystem zu authentifizieren.

Basecamp bietet keine weitere Möglichkeiten an.

Mac OS X Server hat standardmässig einen LDAP Dienst integriert. Darüber könnte man die nötige Authentifizierung und Autorisierung tätigen.

Google Apps	1
Basecamp	0
Mac OS X Server	3

7.3. Auswertung

Wenn man die obigen Bewertungen gewichtet und summiert zeit sich Google Apps als klarer Sieger. Dies lässt sich dadurch erklären, dass Google die eigene Platform bewusst mit den nötigen Schnittstellen ausrüstet, weil es für Google Apps mittlerweile sogar einen Marktplatz für Zusatztools gibt.

System	1.1	0.3	1.2	0.18	1.3	0.12	2.1	0.18	2.2	0.14	2.3	0.08	Total
Google Apps	3	0.9	4	0.72	4	0.48	3	0.54	3	0.42	1	0.08	3.14
Basecamp	2	0.6	2	0.36	2	0.24	0	0	3	0.42	0	0	1.62
Mac OS X Server	1	0.3	2	0.36	1	0.12	1	0.18	0	0	3	0.24	1.2

Tabelle 7.2.: Auswertung der Nutzwertanalyse

[5] `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security`

8. OpenID

Da die Wahl des zu verwendenden Protokolls auf OpenID fällt, wird an dieser Stelle die Funktionsweise von OpenID erklärt. Es ist nicht Bestandteil dieser Arbeit das OpenID Protokoll in Python zu implementieren, da dazu bereits eine Library existiert welche auch gewartet wird. Ryan Boyd von Google empfiehlt in einem Talk ausdrücklich davon abzusehen OpenID selber zu implementieren (BP10, 0:16:08).

8.1. Übersicht

OpenID wurde entwickelt damit sich Benutzer an fremden Webseiten anmelden können, ohne dass sie ein neues Passwort kreieren müssen. OpenID ermöglicht es Benutzern eines OpenID-Providers sich an Webseiten welche als OpenID-Relying-Party fungieren anzumelden.

8.2. Ablauf

Abbildung 8.1 zeigt den Ablauf der Authentisierung über OpenID. Es handelt sich hierbei um den einfachsten Ablauf. Es gibt noch weitere Abläufe, in denen zum Beispiel der Provider und die Relying-Party ein Shared Secret mittels Diffie Hellman(Res99) vereinbaren, um damit die OpenID-Response mit einem HMAC(KBC97) zu versehen. Damit sind die Schritte 9 und 10 nicht mehr nötig, da nach dem Schritt 8 die Relying-Party bereits sicherstellen kann, dass die Response nicht verändert wurde und vom Provider stammt.

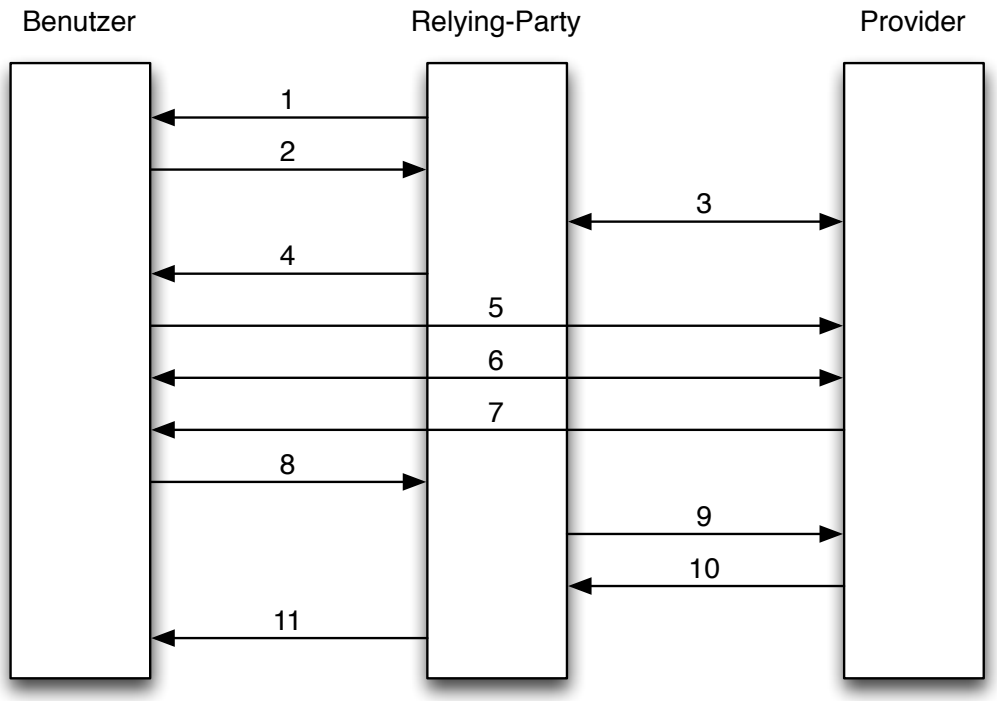

Abbildung 8.1.: OpenID Ablauf

1. Relying-Party fordert den Benutzer auf sich anzumelden
2. der Benutzer teilt der Relying-Party mit welcher Provider er verwenden möchte
3. die Relying-Party führt eine Discovery durch um den Provider zu finden
4. die Relying-Party übergibt dem Benutzer einen OpenID-Request
5. der OpenID-Request wird vom Benutzer welcher weitergeleitet wird an den Provider geliefert
6. falls nötig fordert der Provider den Benutzer auf sich anzumelden
7. der Provider leitet den Benutzer mit der OpenID-Response an die Relying-Party weiter
8. der Benutzer übergibt der Relying-Party die OpenID-Response
9. die Relying-Party sendet dem Provider die OpenID-Response zur Bestätigung
10. der Provider bestätigt der Relying-Party, dass die OpenID-Response von ihr verfasst wurde
11. die Relying-Party leitet den Benutzer weiter auf eine Welcome-Seite

8.3. Sicherheit

Die Sicherheit von OpenID beruht vor allem auf dem Vertrauen in den OpenID-Provider. Da sich diese Arbeit darauf verlässt, dass die E-Mail-Adresse, welche vom OpenID-Provider geliefert wird, stimmt wurde auf die Schritte 1-3 aus Abbildung 8.1 verzichtet und stattdessen ein OpenID-Provider in der Konfiguration hinterlegt.

8.4. Erweiterungen

Um über OpenID an die E-Mail-Adresse eines Benutzers zu gelangen, können zwei verschiedene Erweiterungen verwendet werden. Dies sind Simple Registration(HDR06) und Attribute Exchange(HBH07). Falls vorhanden wird Attribute Exchange vorgezogen, da Simple Registration einst als Minimalansatz entwickelt wurde und Attribute Exchange um einiges umfangreicher ist.

9. Design

9.1. Entscheidungen

9.1.1. Zu verwendendes System

Aus der Evaluation geht hervor, dass eine Anbindung an die OpenID Dienste von Google wohl am meisten Sinn macht. Google Apps ist ein fester Bestandteil der Arbeitsmittel jedes Mitarbeiters der allink.

9.1.2. Anmelde-Mechanismus

In den meisten Fällen ist keine granulare Unterteilung der Benutzerrechte jedes einzelnen Mitarbeiters nötig. Den es geht viel mehr darum, jedem Mitarbeiter von allink Zugriff zu den Administrations-Bereichen sämtlicher Webapplikationen zu gewähren und beim Enden des Arbeitsverhältnisses diese Rechte wieder zu entziehen. Aufgrund dessen wird wie bisher für jede Webapplikation ein Benutzer mit vollen Administrator-Rechten erstellt. Bei erfolgreichem Anmelden über Google Apps ist der Mitarbeiter danach im Administrations-Bereich mit diesem Benutzer angemeldet und bleibt für die Dauer seiner Session zugelassen. Es ist möglich Regeln für die automatische Zuweisung von lokalen Benutzerkonten zu OpenID Benutzern zu erstellen. Diese Regeln können unterschiedlich gewichtet werden um Fälle in denen zum Beispiel wenige OpenID Benutzer Administrator-Rechte haben und alle übrigen Mitarbeiter Autoren-Rechte erhalten sollen.

9.1.3. Wahl eines OpenID-Providers

Die Applikation benötigt um lauffähig zu sein einen OpenID-Provider. Theoretisch lässt sie sich mit jedem beliebigen OpenID-Provider verwenden. Da wir die Mitarbeiter mit Hilfe der vom OpenID-Provider verlangten E-Mail-Adresse identifizieren und ein OpenID-Provider diese E-Mail-Adresse fälschen könnte, kann der OpenID-Provider nicht während des Login-Prozesses

gewählt werden. Der Provider muss in den Einstellungen der Webapplikation hinterlegt werden. Falls jedoch kein Provider hinterlegt wurde, wird das Portal von Google als Standard verwendet.

9.1.4. Abhängigkeiten

Da die zu erstellende Applikation in bestehende Projekte integriert werden soll, ist es wichtig die externen Abhängigkeiten so gering wie möglich zu gestalten. So wird nur für die Implementation von OpenID auf eine Bibliothek zurückgegriffen.

Python OpenID

Da die korrekte Implementation von OpenID wichtig ist um die Sicherheit zu gewährleisten, ist es nicht ratsam OpenID ohne eine Library zu verwenden. In Python existiert zur Zeit nur eine Implementation von OpenID welche auch weiter entwickelt wird. Diese Library nennt sich python-openid und ist über den Python-Packet-Index[1] installierbar. Weiterentwickelt wird pyton-openid über das zugehörige Github Repository[2].

Django

Da ein Paket für das Django Webframework erstellt werden soll gehört Django auch zu den Abhängigkeiten. Um 'Class Based Views' verwenden zu können, benötigt man mindestens Version 1.3, welche zum Zeitpunkt dieser Arbeit bereits nicht mehr die aktuellste Version ist.

[1] `http://pypi.python.org/pypi/python-openid/`
[2] `https://github.com/openid/python-openid`

10. Tests

10.1. Unittests

Die Logik welche die OpenID Konten mit den lokalen Benutzerkonten verbindet wurde erst programmiert, nachdem Unittests dafür programmiert wurden. Diese Unittests laufen auch nachdem das Packet in ein Projekt eingebunden wurde noch und lassen somit ein Test unter Produktivbedingungen zu. Das Protokoll eines Testlaufs ist in Listing 10.1 zu sehen.

```
$ ./manage.py test -v2 admin_sso
Creating test database for alias 'default' (':memory:')...
Creating tables ...
Creating table auth_permission
Creating table auth_group_permissions
Creating table auth_group
Creating table auth_user_user_permissions
Creating table auth_user_groups
Creating table auth_user
Creating table django_content_type
Creating table django_session
Creating table django_site
Creating table django_admin_log
Creating table admin_sso_assignment
Creating table admin_sso_openiduser
Creating table admin_sso_nonce
Creating table admin_sso_association
Installing custom SQL ...
Installing indexes ...
Installed 0 object(s) from 0 fixture(s)
test_change_weight (admin_sso.tests.AuthModuleTests) ... ok
test_domain_matches (admin_sso.tests.AuthModuleTests) ... ok
test_domain_matches_and_username_doesnt_begin_with_foo (admin_sso.tests.AuthModuleTests
    ) ... ok
test_domain_matches_and_username_ends_with_bar (admin_sso.tests.AuthModuleTests) ... ok
test_invalid_domain (admin_sso.tests.AuthModuleTests) ... ok
test_login_twice_and_reuse_stored_openid (admin_sso.tests.AuthModuleTests) ... ok

----------------------------------------------------------------------
Ran 6 tests in 0.113s

OK
```

Listing 10.1: Resultat der Unittests

10.2. Integrationstests

Da ein Single-Sign-On System aus mindestens zwei Systemen besteht und in diesem Fall das eine bestehend ist, wird darauf verzichtet automatisierte Integrationstests zu erstellen. Stattdessen werden die wichtigsten Vorgänge mit manuellen Testverfahren getestet.

10.2.1. Ausgangslage

Um die Reproduzierbarkeit dieser Tests zu gewährleisten, müssen vor Beginn folgende Schritte getätigt werden.

- Das Cache des Browsers muss geleert werden.
- Es darf keine offene Google Apps Sitzung bestehen. Falls ein Benutzer angemeldet ist, muss dieser abgemeldet werden.
- Die Datenbank des Testsystems muss neu erstellt werden.
- Sämtliche Cookies welche zum Testsystem gehören müssen gelöscht werden.
- Ein Testbenutzer ist in Google Apps erstellt und aktiv.

10.2.2. Testablauf

Aktion	Erwartetes Resultat
Drücken des “Login mit OpenID” Buttons im Admin	Der Browser wechselt zum Google Login
Eingeben der Benutzerdaten	Django Admin Startseite wird angezeigt und der zugewiesene Benutzer ist angemeldet
Löschen der Session des Testsystems und neu laden der Seite	Die Login-Seite erscheint
Sperren des Benutzers in Google Apps mit einem anderen Browser	Kein Resultat erwartet
Drücken des “Login mit OpenID” Buttons im Admin	Der Browser wechselt zum Google Login
Eingeben der Benutzerdaten	Benutzer darf sich nicht anmelden können

11. Umsetzung

Dieser Teil befasst sich mit der eigentlichen Programmierarbeit und allem was zusätzlich nötig war um diese Applikation zu erstellen.

11.1. Namensgebung

Der Name eines Softwareprojekts kann grossen Einfluss darauf haben, wie viel es von anderen Entwicklern verwendet wird. Darum wurde für dieses Projekt ein Name gesucht, der die Funktion beschreibt und den man sich einfach merken kann. Aus diesen Überlegungen entstand der Name django-admin-sso.

11.2. Opensource

Es gehört zur Philosophie von allink, entwickelte Tools welche auch für andere Agenturen und Webentwickler von Nutzen sein könnten, zu veröffentlichen. Dies aus zwei verschiedenen Gründen auf welche hier kurz eingegangen wird.

Steigerung der Reputation Um allink für potentielle Mitarbeiter im Webbereich attraktiver zu machen und zu zeigen, dass allink im Stande ist solide Applikationen zu entwickeln.

Chance auf Mitarbeit Dritter Wenn ein von allink veröffentlichtes Tool von Dritten genutzt wird, besteht die Chance, dass diese das Tool weiterentwickeln, Fehler melden, oder diese beheben.

11.2.1. Lizenz

Um ein Tool für Dritte nutzbar zu machen und um zu gewährleisten, dass diese auch den Quellcode des Tools ändern dürfen, muss das ganze Tool unter einer Softwarelizenz, welche dies gestattet, veröffentlicht werden. Die drei Klausel BSD Lizenz[1] lässt dem Lizenznehmer

[1] `http://www.opensource.org/licenses/BSD-3-Clause`

genügend Freiheiten um den Quellcode zu verwenden wo er will. Das Framework Django ist ebenfalls unter der drei Klausel BSD Lizenz veröffentlich worden.

11.2.2. Bestandteile aus anderen Projekten

In dieser Arbeit wurde zusätzlich zu Python OpenID die Implementierung eines OpenID Store-Backends aus django-openid-auth zurückgegriffen. Diese Datei wurde ebenfalls unter der drei Klausel BSD Lizenz veröffentlicht und ist somit lizenzrechtlich kompatibel[2] zu django-admin-sso. Um den Lizenzbedingungen Rechnung zu tragen wurde die Datei mit komplettem Lizenz-Text und Herkunftsangaben übernommen.

11.2.3. Versionsverwaltung

Es wurde Git für die Versionierung verwendet, da dies dem Standard der allink gehört. Es wurde ein zentrales Repository[3] auf Github eingerichtet um den Quellcode zu veröffentlichen.

11.3. Python Paket

Es gibt zwei grundsätzlich verschiedene Distributionsarten für Python Pakete. “Source” Distributionen sind Archive mit den benötigten Python Dateien zusammen mit einem Setup-Script. “Builded” Distributionen hingegen sind für ein bestimmtes System gebaut und lassen sich typischerweise über den systemeigenen Packet Manager oder im Fall von Windows, über ein Installationsprogramm installieren. Python Pakete welche für den Gebrauch in Webapplikationen konzipiert sind, sind normalerweise als “Source”-Distribution erhältlich. Diese “Source”-Distributionen werden dann mit easy_install[4] oder pip[5] installiert. Alternativ können “Source”-Distributionen auch durch simples Entpacken und Ausführen des Installations-Skripts installiert werden. Da django-admin-sso nur für die Verwendung in Webapplikationen vorgesehen ist, wird nur eine “Source”-Distribution erstellt.

11.3.1. Paketerstellung

Ein Python-Paket wird durch verschiedene Dateien charakterisiert. Auf die für die Distribution verwendete Dateien wird in Tabelle 11.1 eingegangen. Im Manifest wurden zusätzlich noch nicht

[2] `http://www.dwheeler.com/essays/floss-license-slide.html`
[3] `https://github.com/frog32/django-admin-sso`
[4] `http://packages.python.org/distribute/easy_install.html`
[5] `http://www.pip-installer.org/`

existierende Dateien erwähnt, welche für eine Übersetzung des Paketes in mehrere Sprachen verwendet werden.

Dateiname	Zwingend	Inhalt
AUTHORS	Nein	eine Liste aller Authoren
LICENSE	Nein	die verwendete Softwarelizenz
MANIFEST.in	Nein	eine Liste von nicht Python Dateien welche berücksichtigt werden muss
setup.cfg	Nein	Einstellungen für Dokumentationstools
setup.py	Ja	Konfiguration des Installationsprogramms

Tabelle 11.1.: Dateien und ihre Funktion für eine Distribution

11.3.2. Registration

Der "Python Packet Index"[6] bildet die zentrale Registrierungsstelle für alle Python Pakete. Es ist ratsam jedes Paket, welches für den freien Gebrauch bestimmt ist, in diesem Index zu registrieren.[7]

Zusätzlich zum "Python Packet Index" wurde das Paket auch in den Index von "Django Packages"[8] eingetragen. Diese Platform wird von vielen Django-Entwicklern verwendet um neue Pakete zu evaluieren.

[6] `http://pypi.python.org/`
[7] `http://pypi.python.org/pypi/django-admin-sso/`
[8] `http://www.djangopackages.com/packages/p/django-admin-sso/`

12. Resultate

Das sicher wichtigste Resultat dieser Arbeit bildet die funktionsfähige Applikation, welche es ermöglicht sich mit einer OpenID an einem Administrationsbereich anzumelden. Diese Applikation wird bei jeder neuen Webseite eingesetzt und wurde zudem in den wichtigsten bestehenden Webseiten bereits eingebaut.

12.1. Erfüllung der Anforderungen

Django-admin-sso erfüllt alle in Kapitel 6.1 definierten Anforderungen und lässt sich einfach in bestehende und neue Projekte integrieren. Die Installation des Paketes ist über den Python Packet Index und einem Python Paket-Management sehr einfach. Die technischen Anforderungen aus Kapitel 6.2 werden meines Erachtens gut erfüllt.

12.2. Resultat des Integrationstests

Die manuellen Tests welche in Kapitel 10.2 mehrfach durchgeführt und die Ergebnisse waren jeweils wie erwartet. Zusätzlich haben einige an dem Projekt interessierte Entwickler das Projekt selbst ausprobiert und dabei zumindest einen Teil der Funktionalität überprüft.

12.3. Kurze Dokumentation der Installation der Applikation

Die Dokumentation der Installation befindet sich in der README-Datei des Projektes. Dieses ist im Python Packet Index[1] oder auf der Projektseite[2] auf Github zu sehen. Die Anweisungen wie die Zuweisungen zu erstellen sind, wurden freundlicherweise von George Hickman ergänzt.

[1] `http://pypi.python.org/pypi/django-admin-sso/0.1.1`
[2] `https://github.com/frog32/django-admin-sso`

12.4. Reaktionen

Am 4.Juni 2012 habe ich spontan das Projekt an der Djangoconeu 2012[3] während eines Lightning Talks vorgestellt. Der Talk bestand im wesentlichen in einer während der Konferenz vorbereiteten live Demonstration. Nach dem Talk haben sich mehrere Personen bei mir gemeldet und mir erzählt, dass auch sie schon lange auf der Suche nach einer solchen Lösung waren.

Das Projekt hat zur Zeit[4] 15 Watchers auf Github und und bisher 4 Pull Requests mit kleineren Anpassungen. Das Paket hat bereits 396 Downloads über den Python Packet Index. Auf meine Anfrage ob sie django-admin-sso produktiv einsetzen, bekam ich von Matthias Kestenholz und Marc Tamlyn folgende Antworten:

Matthias Kestenholz Programmierung & Qualitätssicherung bei FEINHEIT GmbH

> Hi Marc
>
> Unfortunately we aren't using sso yet, but I'd very much like to.
>
> It certainly gave us a few good ideas how to solve the problem where former employees still have access to many websites just because they know the passwords we use. I think we should start using it on all sites except for a few special ones (not that I'd think of any projects in particular, but there are pages which have a different audience, especially webapps or plattforms).
>
> Thanks
> Matthias

Marc Tamlyn Developer bei INCUNA LTD.

> Hi Marc,
>
> We have implemented admin-sso into our base project, and it is being used on all new applications. There may be some apps where we can't use it due to clients having odd requirements on security, but it is not enabled by default.
>
> Thanks for such a useful project!
>
> Marc

[3] http://2012.djangocon.eu/
[4] Stand vom 16.Juli 2012

13. Fazit

Das Szenario, dass sich ehemalige Mitarbeiter an Passwörter erinnern und mit diesem Wissen versuchen der Agentur oder einem Kunden Schaden zuzufügen, hat sich bisher glücklicherweise nicht ergeben. Jedoch können wir nun mit einfachen Mitteln ehemaligen Mitarbeitern den Zugang zu von uns betreuten Webseiten sperren. Auch wenn in der Praxis wahrscheinlich nie Probleme entstehen würden, lässt uns der Einsatz von django-admin-sso besser schlafen.

Diese Arbeit hat ihren Ursprung im Herbst 2011, als wir uns in der allink ein Zeitbudget von zwei Tagen reserviert hatten um alle Mitarbeiter über ein zentrales System zu autorisieren. Schnell hatte sich damals gezeigt, dass in den zwei Tagen keine praktikable Lösung erzielt werden konnte. Daraufhin entstand die Idee meine Semesterarbeit dem Thema zu widmen.

Ich konnte durch diese Arbeit an der Djangoconeu 2012 viele Kontakte zu Webentwicklern knüpfen, welche das gleiche Problem hatten wie wir. Von diesen setzen heute zumindest einige meine Applikation ein. Ich konnte zudem im Zuge der Arbeit meine Kenntnisse in diversen weniger oft verwendeten Gebieten auffrischen oder vertiefen. Zu diesen Gebieten gehören unter anderem das erstellen eines Python Paketes, erstellen von Dokumentationen mit LaTeX und das Anwenden von Projektplanungstools.

A. Abbildungsverzeichnis

B. Tabellenverzeichnis

C. Listings

D. Glossar

Authentifikation Identitätsprüfung eines Benutzers als Zugangs- und Rechtekontrolle für ein System (z.B. durch Passwort) [1].

Autorisierung Synonym für Berechtigung oder Erlaubnis [2].

Git Ein Werkzeug mit welchem man Quellcode versionieren kann. `http://git-scm.com/`.

OpenID-Provider Ein Dienst welcher seinen Benutzern je eine OpenID zur Verfügung stellt. Bekannte OpenID-Provider sind Google, Yahoo und flickr.

OpenID-Relying-Party Eine Applikation welches seinen Benutzern die Anmeldung mit einer OpenID ermöglicht.

[1] `http://www.duden.de/rechtschreibung/Authentifikation`
[2] `http://www.duden.de/rechtschreibung/Autorisierung`

E. Akronyme

API Application Programming Interface.

URL Uniform resource locator.

F. Literaturverzeichnis

[BLFF96] Berners-Lee, T. ; Fielding, R. ; Frystyk, H.: *Hypertext Transfer Protocol – HTTP/1.0*. RFC 1945 (Informational). `http://www.ietf.org/rfc/rfc1945.txt`. Version: Mai 1996 (Request for Comments)

[BP10] Boyd, Ryan ; Primmer, David: *OpenID-based single sign on and OAuth data access for Google Apps*. `http://www.google.com/events/io/2010/sessions/untangling-auth-enterprise.html`. Version: 2010

[FGM+99] Fielding, R. ; Gettys, J. ; Mogul, J. ; Frystyk, H. ; Masinter, L. ; Leach, P. ; Berners-Lee, T.: *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616 (Draft Standard). `http://www.ietf.org/rfc/rfc2616.txt`. Version: Juni 1999 (Request for Comments). – Updated by RFCs 2817, 5785, 6266, 6585

[HBH07] Hardt, D. ; Bufu, J. ; Hoyt, J.: *OpenID Attribute Exchange 1.0 - Final*. `http://openid.net/specs/openid-attribute-exchange-1_0.html`. Version: 2007

[HDR06] Hoyt, J. ; Daugherty, J. ; Recordon, D.: *OpenID Simple Registration Extension 1.0*. `http://openid.net/specs/openid-simple-registration-extension-1_0.html`. Version: 2006

[HL10] Hammer-Lahav, E.: *The OAuth 1.0 Protocol*. RFC 5849 (Informational). `http://www.ietf.org/rfc/rfc5849.txt`. Version: April 2010 (Request for Comments)

[Jos03] Josefsson, S.: *The Base16, Base32, and Base64 Data Encodings*. RFC 3548 (Informational). `http://www.ietf.org/rfc/rfc3548.txt`. Version: Juli 2003 (Request for Comments). – Obsoleted by RFC 4648

[KBC97] Krawczyk, H. ; Bellare, M. ; Canetti, R.: *HMAC: Keyed-Hashing for Message Authentication*. RFC 2104 (Informational). `http://www.ietf.org/rfc/rfc2104.txt`. Version: Februar 1997 (Request for Comments). – Updated by RFC 6151

[Res99] Rescorla, E.: *Diffie-Hellman Key Agreement Method*. RFC 2631 (Proposed Standard). `http://www.ietf.org/rfc/rfc2631.txt`. Version: Juni 1999 (Request for Comments)

[Res00] Rescorla, E.: *HTTP Over TLS*. RFC 2818 (Informational). `http://www.ietf.org/rfc/rfc2818.txt`. Version: Mai 2000 (Request for Comments). – Updated by RFC 5785

[Ser06] Sermersheim, J.: *Lightweight Directory Access Protocol (LDAP): The Protocol*. RFC 4511 (Proposed Standard). `http://www.ietf.org/rfc/rfc4511.txt`. Version: Juni 2006 (Request for Comments)

[spe07] specs@openid.net: *OpenID Authentication 2.0 - Final*. `http://openid.net/specs/openid-authentication-2_0.html`. Version: 2007