The present work was submitted to Lehr- und Forschungsgebiet Informatik 9

Nutzerbasierte Zugriffsverwaltung im Learning-Context-Projekt

Bachelor Thesis

presented by

Kölsch, Alexander

323524

First examiner:	Prof. Dr.-Ing. Ulrik Schroeder
Second examiner:	Prof. Dr.-Ing. Klaus Wehrle

Aachen, 29. September 2015

Erklärung

Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed.

Aachen, 29. September 2015 Alexander Kölsch

Abstract

In this bachelor thesis, an improvement for the interface of the Learning Context Project is developed and implemented. The question i am facing is how to change the authorisation-protocol to OAuth and how to give a user the opportunity to manage his given accesses for the apps. A new authorisation concept for the interface is presented and integrated into the already existing interface. This concept bases on the OAuth 2.0 protocol. Furthermore the user-menu is modified such that the user is able to administrate the apps he uses and change the given accesses. Because of the new implemented access-system, a concept for libraries is developed and implemented with the result that external developers are able to use libraries in Java, PHP and JavaScript. To evaluate the concepts, there are developed some testing programs and testing scripts. These programs and scripts check the function of the interface and the reaction of the interface in case of an error.

Zusammenfassung

Im Rahmen dieser Bachelorarbeit wird eine Verbesserung der bestehenden Schnittstelle für den Zugriff auf die Datenbasis des Learning Context Projects entwickelt und implementiert. Dabei wird sich mit der Fragestellung beschäftigt, wie die bestehende Schnittstelle auf das OAuth-Autorisierungsprotokoll umgestellt werden kann. Außerdem soll dem Benutzer die Möglichkeit gegeben werden, seine Berechtigungen zu verwalten und den Zugriff auf seine Daten über die Schnittstelle für externe Anwendungen zu verweigern. Für die neue Schnittstelle wird ein Konzept zur Autorisierung vorgestellt, welches sich an das OAuth 2.0-Protokoll anlehnt. Dieses wird im Anschluss in die bestehende Schnittstelle integriert. Für den Benutzer wird ein Menü entwickelt, welches ihm die Verwaltung der Berechtigungen für die von ihm genutzten Apps möglich macht. Damit externe Entwickler das neu implementierte System nutzen können, wird ein Konzept für Libraries entwickelt und dieses anhand der Programmiersprachen Java, PHP und JavaScript erläutert und implementiert. Zur Evaluation der vorgestellten Konzepte und Implementierungen werden Testprogramme beziehungsweise Testskripte entwickelt, welche die Funktion der Schnittstelle und die Reaktion auf Fehlerfälle überprüfen sollen.

Inhaltsverzeichnis

1 Einleitung

Überall auf der Welt werden Daten gespeichert. Damit auf diese Daten von externen Anwendungen zugegriffen werden kann, werden Schnittstellen bereitgestellt, die diesen Zugriff ermöglichen. Eine solche Plattform zum Sammeln von Daten ist auch das Learning Context Project[1], das im Zuge der Forschung durch die Learning Technologies Research Group der Informatik an der RWTH Aachen geschaffen wurde. Ziel dieses Projektes ist es, in einer Datenbasis Informationen über eine Person, unter anderem mit Hilfe des Nutzungsverhaltens von (mobilen) Endgeräten, zu sammeln. Die so gesammelten Daten geben dem Nutzer ein Feedback und bieten eine Hilfestellung für ein effektives Lernen. Durch das Mitwirken von externen Entwicklern können Anwendungen/Apps entwickelt werden, mit deren Unterstützung der Nutzer seinen Lernvorgang reflektieren kann.[13]
Damit die externen Anwendungen auf die Datenbasis zugreifen können, wird den Entwicklern eine Schnittstelle bereitgestellt. Für diese Schnittstelle sind Bibliotheken vorhanden, welche durch den Entwickler eingebunden werden können. Die Schnittstelle basiert auf Username/ Password-Authentifizierung. Somit muss der externen Anwendung sowohl der Benutzername, als auch das Passwort bekannt sein. Ein weiteres Problem dieser Schnittstelle ist es, dass dem Benutzer keine Möglichkeit gegeben wird, den Zugriff über die Schnittstelle auf seine Daten zu beschränken oder zu entziehen, sobald er erteilt wurde.
Daher wird im Rahmen dieser Arbeit untersucht, wie eine solche Autorisierung von externen Anwendungen bei sozialen Netzwerken umgesetzt ist. Aus den daraus gewonnenen Informationen und der bestehenden Voruntersuchung durch eine Masterarbeit[2] wird ein Konzept entwickelt, welches die Schnittstelle auf eine Token-basierte Autorisierung umstellt. Im Anschluss an das Konzept wird dieses in die bestehende Schnittstelle des Learning Context Projects implementiert. Zur einfacheren Nutzung durch externe Anwender werden die bestehenden Libraries auf die neue Schnittstelle umgebaut. Zusätzlich zu den Änderungen der Schnittstelle wird dem Benutzer eine Möglichkeit zur Rechteverwaltung gegeben. Dazu wird das Benutzermenü erweitert, damit der Benutzer entscheiden kann, welche Berechtigungen er einer App freigibt. Zusätzlich wird er bestehende Berechtigungen wieder entziehen können und einer App den kompletten Zugriff auf seine Daten verweigern können.

[1] www.learning-context.de

Die vorliegende Arbeit gliedert sich in die folgenden Bereiche: Nachdem in diesem Kapitel eine kurze Einführung ins Thema gegeben wurde, wird im zweiten Kapitel auf die Grundlagen, die für die spätere Implementierung benötigt werden, eingegangen. Dazu gehören das Autorisierungsprotokoll OAuth und die Programmiersprachen Java, PHP und JavaScript. Bei den Programmiersprachen werden Bibliotheken angesprochen, welche in der Implementierung der Libraries benutzt werden.
Das sich anschließende dritte Kapitel legt seinen Fokus auf die Autorisierung für die Schnittstellen von Facebook[2] und Google[3]. Dabei wird neben dem Ablauf der Autorisierung auch auf die Möglichkeiten des Benutzers, die erteilten Berechtigungen zu verwalten, eingegangen.
Im vierten Kapitel wird das Learning Context Project vorgestellt. Dabei wird das Datenmodell erklärt und die sich daran anschließende Schnittstelle behandelt. Bei der Schnittstelle liegt der Fokus auf dem Rechtemanagement, also der Möglichkeit für den Benutzer seine erteilten Berechtigungen einzusehen und zu verwalten, und auf den Ablauf des Zugriffs durch externe Anwendungen über die Schnittstelle. Im letzten Teil des Kapitels wird eine Masterarbeit behandelt, die sich mit einer Umstellung der Schnittstelle auf einen OAuth-basierten Zugriff beschäftigt.
Die im vierten Kapitel angesprochenen Problematiken der Schnittstelle des Learning Context Projects führen dazu, dass im fünften Kapitel ein Konzept zur Rechteverwaltung durch den Benutzer und eine Änderung der Schnittstelle mit einem Umbau auf das OAuth-Protokoll vorgestellt wird. Im weiteren Verlauf des Kapitels werden Libraries vorgestellt, die es einem Entwickler von externen Anwendungen erleichtern, auf die bereits vorgestellte neue Schnittstelle zuzugreifen.
Das sechste Kapitel beschreibt Tests, welche an der Schnittstelle mit Hilfe der Libraries vorgenommen wurden. So wurde die korrekte Funktion des Konzepts und deren Implementation überprüft. Am Ende des Kapitels werden die bei den Tests erhaltenen Ergebnisse vorgestellt.
Das siebte und letzte Kapitel fasst den Inhalt der gesamten Arbeit in einem kurzen Überblick zusammen und gibt einen Ausblick, wo Ansatzpunkt zur Verbesserung des Konzepts liegen könnten.

[2] www.facebook.com
[3] www.google.com

2 Grundlagen

In diesem Kapitel wird auf die für diese Arbeit benötigten Grundlagen genauer eingegangen. Der erste Teil des Kapitels behandelt das Autorisierungsprotokoll OAuth. Dieses ermöglicht dem Besitzer eine geschützte Ressource, die auf einem anderen Server liegt, für einen Dritten freizugeben. Dabei müssen keine Authentifizierungsdaten, wie Benutzername oder Passwort, an Dritte preisgeben werden.
Der zweite Teil des Kapitels beinhaltet die in der Implementierung dieser Arbeit verwendeten Programmiersprachen PHP (serverseitig) und JavaScript (clientseitig) sowie das Betriebssystem Android und dessen Standardprogrammiersprache Java mit ihren speziellen, für Android angepassten, Bibliotheken.

2.1 OAuth

Das Autorisierungsprotokoll OAuth wurde durch eine kleine Gruppe von Webentwicklern[4] entwickelt und im Oktober 2007 vorgestellt. Das Ziel der Gruppe ist es gewesen, eine einfache Lösung für den Zugriff auf geschützte Ressourcen hervorzubringen.
Das Grundproblem bestand darin, dass neben der klassischen Client/Server-Kommunikation vermehrt verteilte oder Cloud-basierte Systeme auftraten. Bei der klassischen Client/Server Kommunikation findet die Kommunikation nur zwischen Client und Server statt, so dass sich ein Client direkt bei dem angesprochenen Server authentifizieren muss. Bei verteilten Systemen kommt es allerdings sehr oft vor, dass die Speicherung der Daten und die Authentifizierung auf unterschiedlichen Systemen stattfinden. OAuth stellt einen Lösungsansatz für eine solche Problematik dar.
Das Grundgerüst von OAuth erweitert das Client/Server-Modell um einen Resource-Owner, der den Zugriff auf serverseitig gespeicherte Informationen verwaltet. So besteht die Möglichkeit, dass eine externe Anwendung unter Erlaubnis des Resource-Owners als autorisierte Anwendung auf die gespeicherten Daten zugreifen darf. Dabei soll verhindert werden, dass der externen Anwendung Daten wie Benutzername und Passwort preisgegeben werden müssen.[4]

Die aktuell (Stand September 2015) gültige Version OAuth 2.0 wurde im Oktober 2012 durch die Internet Engineering Task Force, kurz IETF, unter RFC 6749 veröffentlicht[5]. Die beiden OAuth-Versionen sind untereinander nicht kompatibel, allerdings können in einem gemeinsamen Netzwerk beide Versionen gleichzeitig verwendet werden.

[4] http://oauth.net/core/1.0a/#anchor1
[5] http://tools.ietf.org/html/rfc6749.html#section-1

Die Basiskomponenten des Autorisierungsprotokolls OAuth 2.0, wie in Abbildung 2.1 zu sehen, sind:

Resource-Owner Der Resource-Owner erteilt dem anfragenden Client die Zugriffsberechtigung auf die geschützten Daten. Er muss sich dazu beim Authorization-Server authentifizieren.

Resource-Server Auf dem Resource-Server sind die geschützten Daten, auf die zugegriffen werden soll, gespeichert. Er liefert diese bei Erhalt eines korrekten Access-Tokens zurück.

Authorization-Server Nach erfolgreicher Authentifizierung des Resource-Owners generiert der Authorization-Server das Access-Token und das Refresh-Token. Mittels Refresh-Token kann das Access-Token durch den Authorization-Server erneuert werden.

Client Der Client stellt eine Anfrage, um auf geschützte Daten zugreifen zu können. Bei dem Client kann es sich u.a. um eine Drittanbieteranwendung handeln. Die Art des Client (Desktop-PC, Server, mobiles Endgerät) spielt keine Rolle.

Neben den Basiskomponenten ist in dieser Abbildung der Standardablauf eines Zugriffes nach OAuth dargestellt. Wenn ein Client, wir nehmen in diesem Beispiel an, es handele sich um eine Drittanbieteranwendung, auf geschützte Daten zugreifen möchte, stellt sie eine Anfrage an den entsprechenden Resource-Owner mit der Bitte, den Zugriff auf die Daten zu erhalten. Der Client wendet sich mit der nun erhaltenen Zugriffsberechtigung an den Authorization-Server, wo sich der Resource-Owner authentifizieren muss, damit der Client ein Access-Token und, je nach Implementierung, ein Refresh-Token zurückerhält. Dies geschieht durch die Zugriffsberechtigung, die der Resource-Owner dem Client erteilt hat, oder mittels der Eingabe von Benutzername und Passwort durch den Resource-Owner, sofern es sich beim Resource-Owner um einen Endbenutzer handelt. So ist sichergestellt, dass eine clientseitige Speicherung von Benutzernamen und/oder Passwort nicht benötigt wird. Nach korrekter Verifikation der Zugriffsberechtigung oder der Kombination aus Benutzername und Passwort durch den Authentifizierungsserver liefert dieser an die Anwendung das Access-Token und gegebenenfalls das Refresh-Token zurück. Mit dem Access-Token autorisiert sich der Client beim Resource-Server, um den Zugriff auf die geschützten Daten zu erhalten.

Das Access-Token besitzt nur eine begrenzte Lebensdauer. Diese Lebensdauer sollte unter dem Zeitraum liegen, für die die Freigabe erteilt wurde. Aus diesem Grund wird vom Authorization-Server nach erfolgreicher Authentifizierung durch den Resource-Owner auch ein Refresh-Token zurückgeliefert. Das Refresh-Token hat die Funktion, beim Authorization-Server ein neues Access-Token anzufragen. Somit kann das Access-Token erneuert werden, ohne dass sich der Resource-Owner erneut beim Authorization-Server authentifizieren muss.
Der Ablauf eines Refreshs des Access-Tokens läuft nach Protokoll folgendermaßen ab: Der Client erhält, wie in Abbildung 2.1 beschrieben die beiden Token und greift normal auf die geschützte Ressource zu. Nach Ablauf des Access-Tokens liefert der Resource-Server einen „Invalid Token Error“ an den Client zurück. Dieser wendet sich nun an den Authorization-Server. Der Authoriaztion-Server generiert ein neues, gültiges Access-Token und optional auch ein neues Refresh-Token und liefert das Ergebnis an den Client zurück. Mittels des neuen Access-Tokens kann der Client nun wieder auf die geschützten Daten zugreifen.[5]
Der Grund für die begrenzte Lebensdauer des Access-Tokens ist, dass das OAuth-Protokoll auch die unverschlüsselte Kommunikation über Protokolle wie HTTP unterstützt. Da bei HTTP das Token bei jeder Anfrage an den Resource-Server in Klartext übertragen wird, liegt ein

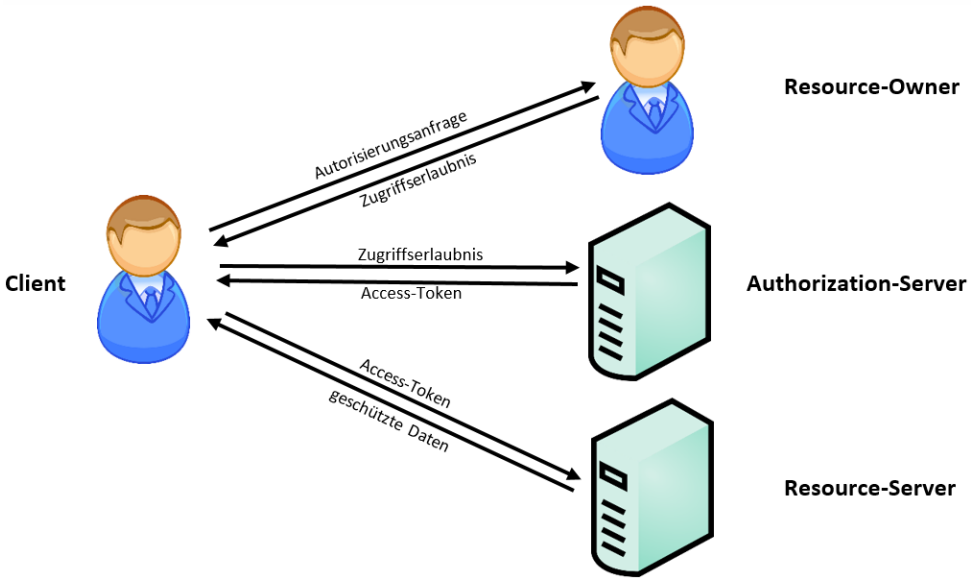

Abbildung 2.1: Ablauf einer Anfrage nach RFC 6749[5]

gewisses Sicherheitsrisiko vor, weil der Datenverkehr mitgeschnitten werden könnte (Man-in-the-Middle-Angriff[6] oder ähnliche). Durch das mitschneiden des Tokens wäre für fremde Personen ein Zugriff auf die geschützten Daten möglich. Ein weiteres Problem des Access-Tokens ist seine begrenzte Länge. Mittels einer groß gewählten Zeitspanne und einer hohen Anzahl an Anfragen könnte das Access-Token erraten werden (Brute-Force). Durch die begrenzte Lebensdauer wird diese Problematik eingeschränkt.

Für die Umsetzung einer OAuth ähnlichen Authentifizierungen, der Authorization-Server und der Resource-Server sind im weiteren Verlauf als eine Einheit zu betrachten (vgl. Kapitel 5), werden im Rahmen dieser Arbeit verschiedene Programmiersprachen eingesetzt, auf die in den folgenden Unterkapiteln weiter eingegangen wird.

2.2 PHP

PHP, was als Akronym für PHP: Hypertext Preprocessor steht, ist eine serverseitige Programmiersprache, die 1997 als Nachfolger von PHP/FI[6] durch Andi Gutmans und Zeev Suraski vorgestellt wurde. PHP/FI wurde 1995 von Rasmus Lerdorf entwickelt und stellte eine Sammlung von Perl-Skripten dar. Die spätere Umsetzung in C und die Ursprünge in Perl führten dazu, dass der Syntax stark an C und Perl angelehnt ist. Ziel der gesamten Entwicklung war eine Programmiersprache für einfache dynamische Webapplikationen. Die heutige Verwendung von PHP bezieht sich auf die drei Hauptgebiete serverseitige Programmierung, Kommandozeilenprogrammierung und Desktopanwendungsprogrammierung. Im Verlaufe dieser Arbeit liegt der Fokus auf der serverseitigen Programmierung.
Ein großer Vorteil von PHP ist, dass die Sprache von Einsteigern leicht zu erlernen ist, aber trotzdem auch professionellen Programmierern einen großen Funktionsumfang liefert. Dieser

[6] http://php.net/manual/phpfi2.php

Vorteil und auch die kostenfreie Verbreitung unter der PHP-License[7] machen PHP zu einer beliebten Programmiersprache in der Webentwicklung[14]. Die serverseitig ausgeführten Skripte können nicht nur HTML-Seiten zurückliefern, sondern PHP unterstützt auch die Ausgabe und/oder serverseitige Speicherung von z.B. PDF-Dateien, Bildern oder auch JSON-Objekten[8]. Besonders JSON-Objekte bieten eine Möglichkeit zum standardisierten Datenaustausch und werden in dieser Arbeit verwendet. Die Möglichkeit von PHP zur Modifikation des Headers einer Antwort bietet dem Programmierer die Chance, selbst in die Antwort des Servers einzugreifen und gezielte Statuscodes als Antwort an den anfragenden Client zu senden.[10]
Die größte Neuerung in PHP 5 gegenüber der vorherigen Version 4 ist die Einführung objektorientierter Programmierungskonzepte, welche auch in Kapitel 5 eine größere Bedeutung bekommt, da diese Programmierart für die PHP-Library verwendet wird.

Eine für diese Arbeit wichtige Bibliothek von PHP, welche seit PHP 4.0.2 standardmäßig in PHP enthalten ist, ist cURL. Die Bibliothek wurde ursprünglich von Daniel Stenberg entwickelt und dient dem Verbindungsaufbau zu Servern mittels verschiedener Protokolle. Die aktuell unterstützten Protokolle sind: HTTP, HTTPS, FTP, gopher, telnet, DICT, FILE und LDAP.
Für diese Arbeit ist besonders die Unterstützung der Protokolle HTTP und HTTPS wichtig, da es sich bei den Anfragen an die Schnittstelle des Learning Context Projects, im Folgenden mit API bezeichnet, und die OAuth-Schnittstelle um HTTP bzw. HTTPS-Requests handelt. Ein weiterer Vorteil von cURL sind die Angabe von Timeouts, so dass zu lange oder unendliche Anfragen nach einer gewissen Zeit abgebrochen werden und das PHP-Skript nicht selbst in einen Timeout läuft. Außerdem unterstützt cURL das Abfragen von HTTP-Statuscodes und dem Übertragen von Inhalten über getrennte Funktionsaufrufe, wodurch die einzelnen HTTP-Statuscodes besser behandelt werden können.[10]
Durch die eben genannten Vorteile von cURL eignet sich diese Bibliothek für die Kommunikation zwischen der PHP-Bibliothek und der API beziehungsweise der OAuth-Schnittstelle des Learning Context Projects.

2.3 Android

Das durch die Open Handset Alliance[9], eine von Google gegründete Gruppe von Unternehmen, für Handys herausgebrachte Open Source Betriebssystem Android, hat sich mittlerweile zu einer Softwareplattform für eine Vielzahl unterschiedlicher Geräte entwickelt. Neben der ursprünglichen Verwendung als Betriebssystem für Smartphones wurde Android durch die stetige Weiterentwicklung eine Softwareplattform für verschiedenste Endgeräte. So dient Android mittlerweile als Betriebssystem für Tablets, Wearables, TVs und Multimediageräten in modernen Autos. Durch die Großzahl an Verwendungsmöglichkeiten wird Android auf mehr als einer Milliarde Endgeräte eingesetzt.[8]
Der Kernel von Android besteht aus einem Multi-User Linux System, so dass jede Applikation (kurz App) in einer eigenen Virtuellen Maschine (VM) läuft. Zur Zugriffskontrolle weist Android jeder App eine eigene User-ID zu, welche nur dem System bekannt ist und worüber der Zugriff auf die hinterlegten Dateien geregelt wird. Da jeder App auch ein eigener Linux-Prozess zugewiesen wird, ist die Standardeinstellung von Android, dass im Falle von Ressourcenmangel

[7] http://www.php.net/license/3_01.txt
[8] https://tools.ietf.org/html/rfc7159
[9] http://www.openhandsetalliance.com

einzelne, nicht mehr benötigte Prozesse durch das Betriebssystem beendet werden können. Dadurch ist sichergestellt, dass Android auf einer Vielzahl von Geräten mit unterschiedlichen Hardwarekomponenten eingesetzt werden kann.[8]

Jede App läuft in einer Sandbox und besteht aus verschiedenen Activities. Jede Activity stellt dabei eine eigene Ansicht inklusive dem dazugehörigen Userinterface dar, das heißt, für jede einzelne Ansicht muss eine eigene Activity oder ein Fragment erstellt werden. Als Beispiel würde eine App mit einer Liste von Musiktiteln und einer Detailansicht für jeden einzelnen Musiktitel aus 2 Activities bestehen. Diese Activities können mittels eines sogenannten Intents Daten austauschen oder auch andere Activities, sogar in anderen Apps, starten. Die für die Entwicklung von Apps verwendete Programmiersprache ist Java, wofür eine offizielle Entwicklungsumgebung[10] mit Emulator bereitgestellt wird.

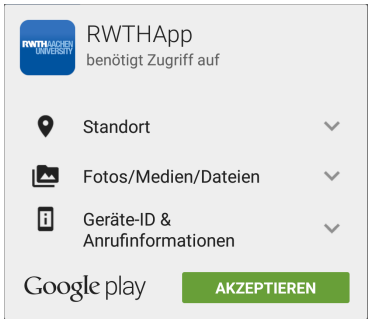

Abbildung 2.2: Berechtigungen der App der RWTH Aachen

Die Berechtigungen einer App werden bei der Installation angezeigt und können nur durch ein Update oder die Installation einer anderen Version der App verändert werden. Bei einer Änderung der Berechtigungen wird der Benutzer jedes Mal mit einem Hinweis darüber informiert und muss dies bestätigen. In Abbildung 2.2 ist exemplarisch die Abfrage der Berechtigungen für die App der RWTH Aachen[11] dargestellt. Diese Berechtigungen umfassen nicht nur den Zugriff auf Geräteinformationen, Dateien oder den Standort, sondern auch Funktionen wie den Kamerazugriff, Internetzugriff oder den Zugriff auf das interne Mikrofon.[8] Somit kann jeder Entwickler genau definieren, welche Freigaben er benötigt und der Benutzer ist immer genau darüber informiert, welche Freigabe eine Applikation benötigt. Android lässt keine Zugriffe zu, die vorher nicht als Berechtigung für die App erteilt worden sind. Somit kann diese Vorgehensweise als Sicherheitsfeature von Android angesehen werden.

Neben den Features von Android, über Intents Activities in anderen Applikation aufzurufen, werden im Verlauf dieser Arbeit auch die von Android bereitgestellten Services, eine Möglichkeit lang laufende Operation in eine Hintergrundprozess zu verschieben, genutzt. Diese Feature von Android kommt vor allem im Rahmen der Anfragen an die API in Form eines asynchronen Tasks zur Anwendung, da eine Anfrage das Interface der App blockiert hätte und es für den Benutzer und das System so ausgesehen hätte, als wäre die Applikation eingefroren.

2.4 JavaScript

Die dritte Programmiersprache, die in der späteren Implementierung verwendet wird, ist JavaScript. JavaScript ist eine 1995 von Netscape und Sun Microsystems publizierte clientseitige, objektorientierte Skriptsprache, deren Grundsyntax stark an Java angelehnt ist. War die ursprüngliche Verwendung von JavaScript noch die Einbettung in HTML-Seiten, um diesen

[10] http://developer.android.com/sdk/index.html
[11] https://play.google.com/store/apps/details?id=de.rwth_aachen.rz.rwthapp

Dynamik zu verleihen, so besitzt JavaScript heutzutage eine Vielzahl an Anwendungsmöglichkeiten, wie z.B. Node.js[12], welches für serverseitige Netzwerkapplikationen verwendet werden kann.
Im Kontext dieser Arbeit wird JavaScript in seiner ursprünglichen Form benutzt. Der Vorteil dieser Verwendung ist, dass HTML-Seiten dynamisch, ohne neu geladen werden zu müssen, verändert werden können. Außerdem kann mit Hilfe von JavaScript direkt auf das Nutzerfeedback reagiert werden, ohne eine Verbindung zu einem Webserver aufbauen zu müssen. Der letzte hier genannter Vorteil von JavaScript ist, dass Daten vor dem Absenden zum Server bearbeitet oder vom Server in einer asynchronen Anfrage angeforderte Daten an den entsprechenden Stellen eingefügt werden können.[3]

Nachdem in diesem Kapitel die Grundlagen für die spätere Arbeit genauer erläutert wurden, behandelt das nächste Kapitel die Umsetzung der Autorisierung bei den Schnittstellen von Facebook und Google. Dabei spielt vor allem das im ersten Unterkapitel behandelte OAuth eine entscheidende Rolle, da auch auf die Technik eingegangen werden soll, die von den jeweiligen Firmen verwendet wird.

[12] https://nodejs.org/en/about/

3 Autorisierung bei anderen Projekten

Nachdem im vorherigen Kapitel die Grundlagen für die spätere Arbeit genauer erläutert wurden, behandelt dieses Kapitel die Umsetzung der Autorisierung bei den Schnittstellen von Facebook und Google. Dabei spielt vor allem das im ersten Unterkapitel behandelte OAuth eine entscheidende Rolle, da auch auf die Technik eingegangen werden soll, die von den jeweiligen Firmen verwendet wird. Von den zwei Konzernen Facebook und Google werden Libraries angeboten, mit deren Hilfe sich Benutzer in externen Anwendungen anmelden können, um diese externen Anwendungen auf ihre gespeicherten Daten zugreifen zu lassen. In diesem Kapitel wird ein genauer Blick auf die Autorisierung über diese Libraries geworfen. Dabei liegt der Fokus auf den verwendeten Abläufen und Protokollen.

3.1 Facebook

Für den Zugriff einer externen Anwendung auf die durch Facebook gespeicherten Daten, bietet Facebook eine Schnittstelle (Graph-API) an. Der Zugriff auf die Graph-API[13] erfolgt über eine tokenbasierte Zugriffskontrolle. Für den Zugriff durch die Anwendung muss sich ein Benutzer zuerst in seinem Facebook-Konto anmelden und der Anwendung, wie in Abbildung 3.1 zu sehen ist, die einzelnen Freigaben (auch Permissions genannt) im Ganzen erteilen. Dabei stehen den Entwicklern über 30 verschiedene Permissions zur Verfügung. Sobald eine Anwendung auf mehr als das öffentliche Profil, die Freundesliste und die Mail-Adresse zugreifen kann, muss die Anwendung von Facebook geprüft werden. Facebook unterscheidet zwischen benötigten Permissions und optionalen Permissions. Die optionalen Permissions können durch den Benutzer verweigert werden. Auch die Sichtbarkeit der Anwendung im eigenen Profil kann durch den Benutzer verändert werden.
Nach erfolgreicher Autorisierung der Anwendung durch den Benutzer bekommt die Anwendung, abhängig von der Anmeldeart, ein kurzlebiges oder ein langlebiges Access-Token zurückgeliefert, wobei die kurzlebigen Access-Tokens bei einer Anmeldung via Web-Login erzeugt werden. Im Falle eines abgelaufenen Access-Tokens kann dieses erneuert werden. Ein Benutzer muss sich zur Erneuerung eines Tokens nicht erneut anmelden, da er die Anwendung bereits vorher schon einmal autorisiert hat.
Eine Änderung der erteilten Permissions und die Rücknahme einer Autorisierung einer Anwendung kann über die Kontoeinstellungen[14] bei Facebook vorgenommen werden. Für das vereinfachte Einbinden in Anwendungen liefert Facebook für Android, iOS und JavaScript eigene Bibliotheken. Falls ein Entwickler eine andere Sprache verwenden möchte, besteht die

[13] https://developers.facebook.com/docs/graph-api
[14] https://www.facebook.com/settings?tab=applications

Möglichkeit, den Login-Vorgang in einer eigenen Sprache nachzubauen. Eine Anleitung befindet sich in der Facebook-Dokumentation[15].[11]

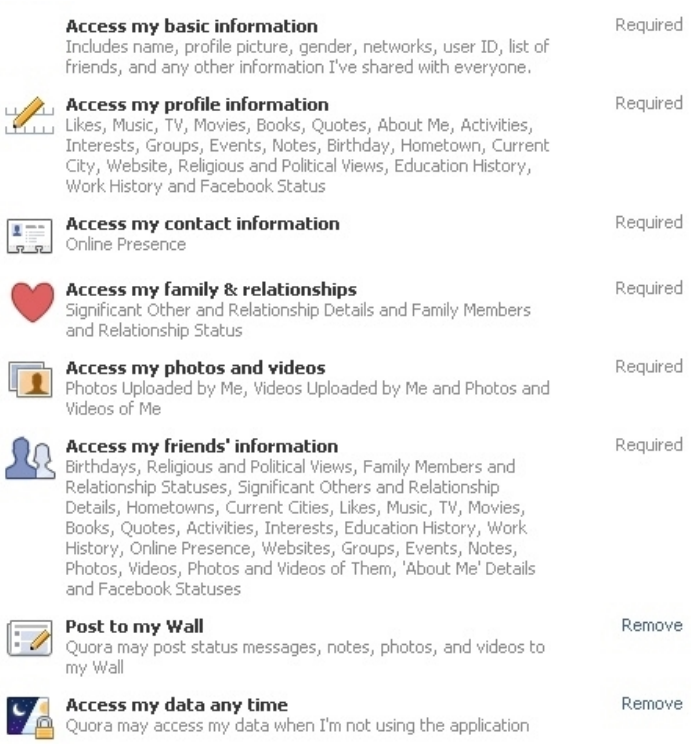

Abbildung 3.1: Freigabe Berechtigungen Facebook

3.2 Google

Für den Zugriff einer externen Anwendung auf die durch Google gespeicherten Daten bietet Google eine Schnittstelle nach OAuth 2.0 an. Um dieses Schnittstelle nutzen zu können, muss jede Anwendung bei Google eine OAuth 2.0 Client ID beantragen. Die Autorisierung erfolgt, nachdem sich ein Benutzer über sein Google+-Konto angemeldet und die für die App benötigten Freigaben erteilt hat. In Abbildung 3.2 ist dargestellt, wie eine solche Berechtigungsanfrage aussieht. Die Schnittstelle von Google bietet dem Benutzer die Möglichkeit auszuwählen, wie die durch die Anwendung veröffentlichten Daten angezeigt sein sollen und für welche Kreise diese sichtbar sind. Die Option, nur einen Teil der angeforderten Berechtigungen freizugeben, wird dem Benutzer nicht gegeben. Dem Entwickler der Anwendung bietet Google die Möglichkeit, die Berechtigungen einer Anwendung nachträglich zu ändern. In einem solchen Fall wird dem Benutzer erneut eine Anfrage der Berechtigungen angezeigt, jedoch nur für die neu angefragten Berechtigungen.[12]
Für den Fall, dass der Benutzer Berechtigungen ändern möchte, bietet Google zwei verschiedene Ansätze an. Der erste Ansatz ist, dass der Benutzer die Sichtbarkeit für die durch die Anwendung veröffentlichten Daten nachträglich ändert[16]. Der zweite Ansatz ist es, den Zugriff

[15] https://developers.facebook.com/docs/facebook-login/manually-build-a-login-flow
[16] https://plus.google.com/apps

auf die Daten zu sperren, indem der Anwendung die erteilte Autorisierung entzogen wird. Dies geschieht über das Google+-Dashboard[17].
Entwicklern externer Anwendungen werden durch Google Libraries für die verschiedenen Programmiersprachen und Betriebssysteme Android, iOS, C# / .NET, Go, Java, JavaScript, PHP, Python und Ruby bereitgestellt.[18]

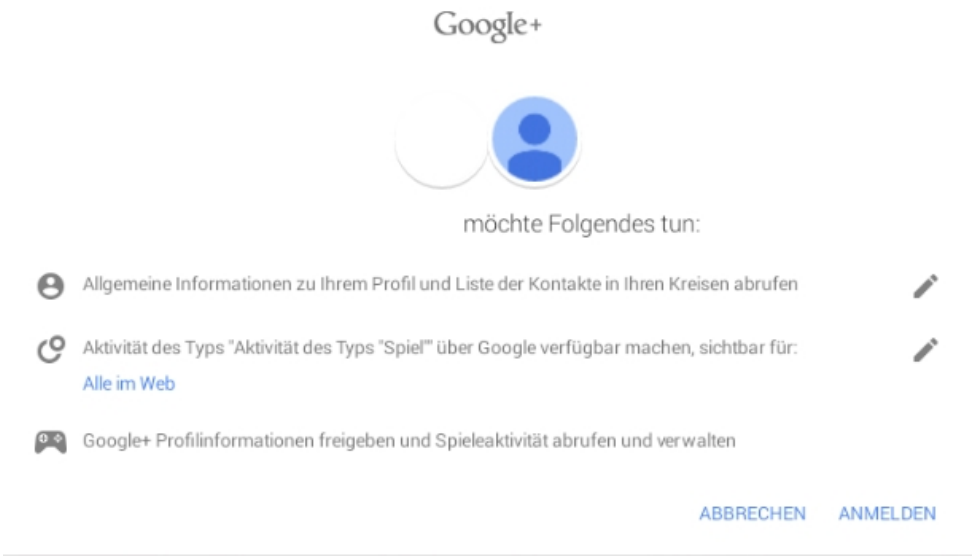

Abbildung 3.2: Freigabe Berechtigungen Google+

In diesem Kapitel wurde der Ablauf eines Zugriffs einer externen Anwendung auf die Schnittstellen von Facebook und Google mit dem Fokus auf die Autorisierung betrachtet. Auffällig ist dabei, dass dem Benutzer wenige Möglichkeiten zur Auswahl der Berechtigungen an die Hand gegeben werden. Er kann bei beiden Schnittstellen nicht, beziehungsweise bei Facebook nur teilweise, die Freigabe einzelner Berechtigungen ablehnen. Bei Google kann er nur alle angefragten Berechtigungen erteilen oder keine. Somit besteht bei beiden Schnittstellen ein Verbesserungspotential.
Das nächste Kapitel beschäftigt sich mit dem Learning Context Project. Dabei wird auch auf die API des Projektes eingegangen.

[17] https://security.google.com/settings/security/permissions
[18] https://developers.google.com/+/quickstart/

4 Learning Context Project

Dieses Kapitel beschreibt das Learning Context Project, in dessen Kontext diese Arbeit stattfindet. Das erste Unterkapitel geht auf das Projekt selbst ein und erläutert den Aufbau des Datenmodells und die Art der Sammlung von Daten sowie die Ziele des Projekts genauer. Auch wird ein Blick auf die bisherige Möglichkeit der Benutzer geworfen, ihre Daten zu verwalten und die Zugriffe der einzelnen Apps einzuschränken.
Im weiteren Verlauf wird anschließend auf die API des Learning Context Projects genauer eingegangen. Dabei liegt der Fokus auf der Kommunikation zwischen Client und API, dem Aufbau der einzelnen Pakete und der Authentifizierung des Benutzers während des Zugriffs einer externen App über die API.
Der letzte Teil dieses Kapitels behandelt eine Arbeit von Elyas Esnaashari[19], die das Ziel hatte, den Zugriff auf das Learning Context Project, auch mit Hinblick auf Sicherheit, genauer zu untersuchen.

4.1 Das Projekt

Das Learning Context Project wurde im Rahmen von PRiME (Professional Reflective Mobile Personal Learning Environment)[20] ins Leben gerufen, um eine zentrale Datenbasis für die Erfassung von nutzerspezifischen Daten zu liefern. Das Ziel des Projekts ist es, dem Benutzer ein Hilfsmittel zu geben, sein Lernverhalten und den daraus resultierenden Tagesablauf zu analysieren, reflektieren und zu verbessern. Das Sammeln und die Verarbeitung beziehungsweise die Visualisierung der Daten soll über externe Anwendungen geschehen, welche über eine Schnittstelle auf die Datenbasis zugreifen können. Der Zugriff soll dabei unter hoher Datensicherheit erfolgen. Daher wurde auf einen Serverstandort in Deutschland geachtet und versucht, bei der Übertragung der Daten, wie im nächsten Unterkapitel genauer beschrieben, nur die nötigsten Informationen übertragen zu müssen.[13]
Die Anwendungen (hier Apps genannt) lassen sich in 3 Bereiche aufteilen: Collector[21], Visualisierer[22] und Analysierer[23]. Ein Collector sammelt die Daten über das Nutzerverhalten und die Umgebung und leitet die gesammelten Daten über die Schnittstelle an die Datenbasis weiter. Dazu kann, auch bedingt durch die Tatsache, dass mobile Endgeräte zu ständigen Begleitern geworden sind, auf eine Vielzahl an verschiedenen Sensoren zurückgegriffen werden. Die ent-

[19] http://www.learning-context.de/upload/files/publications/ElyasEsnaashari.pdf
[20] http://prime.rwth-aachen.de
[21] http://www.learning-context.de/text/3/Collectors
[22] http://www.learning-context.de/text/4/Visualizers
[23] http://www.learning-context.de/text/13/Analyzers

sprechenden Sensoren werden in die Kategorien Bio-Sensoren, Environmental-Sensoren und Activity-Sensoren eingeteilt. Beispiele für gesammelte Daten sind der Aufenthaltsort des Benutzers oder die Umgebungsgeräusche als äußere Faktoren, aber auch Informationen über das Nutzerverhalten wie die Nutzung bestimmter Apps oder die Nutzung des Gerätes generell. Die Visualisierer stellen die Daten aufbereitet dar, wohingegen die Analysierer genauer auf die Daten eingehen und Zusammenhänge erstellen.[7][13]

Die so gesammelten Informationen werden in dem in Abbildung 4.1 dargestellten Kontextmodell gespeichert. Damit eine Vielzahl von Anwendungsfeldern und Szenarien abgedeckt werden kann, wurde das Modell so allgemein wie möglich gehalten. Die zentrale Rolle dieses Modells ist der User. Jedem User kann eine Vielzahl von Events zugeordnet werden, wie zum Beispiel die Änderung des Standorts. Dabei ist jede einzelne Änderung der Position ein Event, dem verschiedene Entities zugeordnet werden, die in einer eigenen Tabelle gespeichert werden. Neben diesen Informationen werden ebenfalls Daten wie der Timestamp oder die dazugehörige App, welche die Daten gesammelt hat, gespeichert. All diese Informationen können über die im nachfolgenden Kapitel beschriebene API gespeichert und verändert werden.[7]

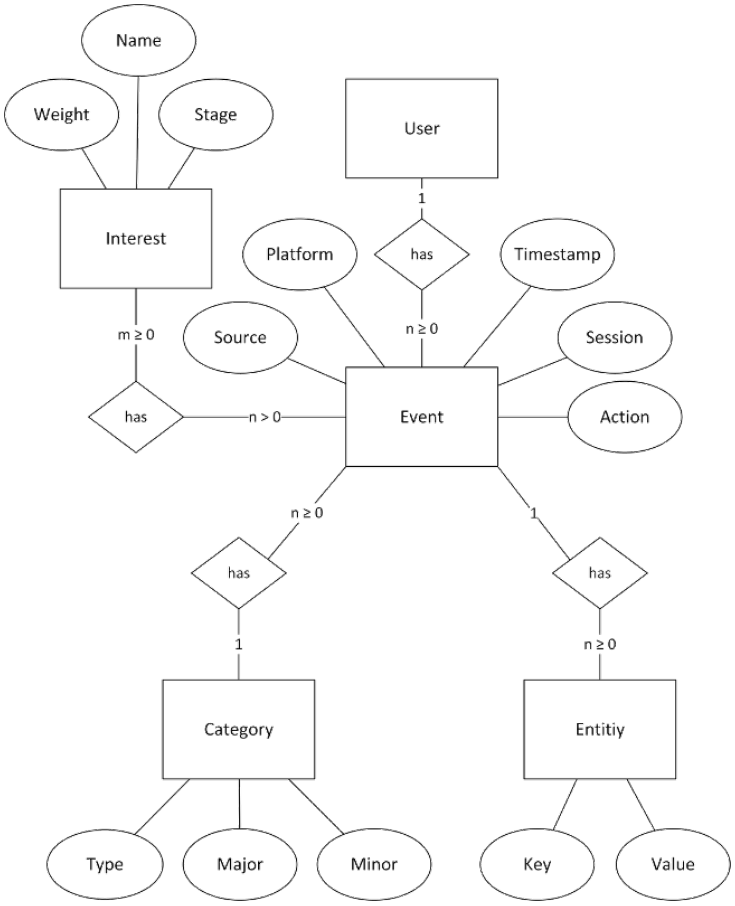

Abbildung 4.1: Das Datenmodell der vierten Version des Learning Context Projects[24]

[24] http://www.learning-context.de/text/1/Project

4.2 API

Damit auf die im vorherigen Kapitel beschriebene Datenbasis zugegriffen werden kann, stellt das Learning Context Project eine Schnittstelle (englisch Application Programming Interface, kurz API) zur Verfügung. Der Zugriff auf die API erfolgt als RESTful-Service über HTTP-GET beziehungsweise HTTP-POST-Anfragen. Mit der neuen Version 4 der API werden auch HTTP-PUT und HTTP-DELETE unterstützt. Die Daten werden als JSON-Objekt bei der Anfrage übertragen. Die einzelnen Anfragen und eventuelle Filterfunktionen sind in der API-Dokumentation[25] nachzulesen.[7]

Die bei einem Aufruf übertragenden Daten sind in Tabelle 4.1 dargestellt. Jede Anfrage über dieselbe App unterscheidet sich beim selben Benutzer nur im Feld „data“. Dieses variiert, je nach Schnittstelle, die angesprochen werden soll. Die ID und das AppSecret einer App, welches für den Hash-Wert benötigt wird, können als Entwickler über die Website des Projektes[26] beantragt werden. Das Nonce wird bei jedem Aufruf neu erzeugt und serverseitig in einer Datenbank gespeichert, um doppelte Aufrufe zu verhindern. Bei der doppelten Verwendung ein und desselben Nonce liefert der Server eine Fehlermeldung zurück. Der Benutzername und das Passwort müssen clientseitig durch die App gespeichert werden, da sie bei jeder Anfrage benötigt werden und der Benutzer diese sonst jedes Mal erneut eingeben müsste. Dies stellt ein Sicherheitsrisiko dar, da der Benutzer seine Daten einer Drittanbieteranwendung preisgeben muss, welche nicht überprüft wurde. Zudem kann er den Zugriff einer App auf seine gespeicherten Daten nur noch durch eine Änderung seiner Zugangsdaten verhindern. Um diese Probleme zu lösen, wird die API im Rahmen dieser Arbeit auf OAuth-Autorisierung umgestellt.

data	Die zu übertragenden Daten als JSON-Object codiert
nonce	Eine randomisierte Zeichenkette der Länge 41-59 Charakter
aid	Die AppID der App, die die Schnittstelle anspricht
user	Der Benutzername des Users, für den die Daten geändert werden sollen.
hash	Ein SHA1-Hash über die Werte: data, AppID, Username, nonce, appSecret und das gehashte Passwort

Tabelle 4.1: Frame der übertragenen Daten

Um den Entwicklern von Apps den Zugriff auf die API zu vereinfachen, werden durch das Learning Context Project mehrere Libraries für die Programmiersprachen Java(Android), PHP, JavaScript und Objective-C bereitgestellt. Die Libraries enthalten die Klassen ContextData, Entity und Event. Entity und Event stehen dabei für Entity und Event aus dem ContextModel (vergleiche Kapitel 4.1) und werden nicht explizit für die Kommunikation benötigt. Die beiden Klassen Entity und Event sind eine Hilfestellung zur Speicherung der Events und Entities. Die Klasse Event bietet die Möglichkeit, aus den im Objekt gespeicherten Daten ein JSON-Objekt für eine Anfrage zu generieren. Die Klasse ContextData ist für den Verbindungsaufbau und die Abwicklung der HTTP-GET/POST-Anfragen zuständig. Die Klasse erzeugt das für die Anfrage benötigte „Nonce“, generiert alle verwenden Hashes und führt über die Methoden get() beziehungsweise post() die entsprechenden Anfragen aus. Die Methoden get() und post() liefern

[25] http://docs.learning-context.de/
[26] www.learning-context.de

als Rückgabewert immer den übertragenen Content und können nicht auf HTTP-Statuscodes reagieren. Ein Beispiel für die Verwendung der PHP-Library für Version 4 der API. Zuerst muss ein Objekt vom Typ ContextData erstellt werden:

```
$cd = new ContextData("
            http://api.learning-context.de",
            4,
            "test",
            "8e67bb26b358e2ed20fe552ed6fb832f397a507d",
            29,
            "a3N7UJefAVinYLiJh4eRAaaPFpuPth0yMKvbBpAUsL6tkB2H2X"
);
```

Die Daten des Beispiels gelten für einen Benutzer mit dem Benutzernamen „test“ und dem Passwort „superuser“, welcher mittels einer App mit der ID „29“ und dem AppSecret „a3N7UJefAVinYLiJh4eRAaaPFpuPth0yMKvbBpAUsL6tkB2H2X“ über die API auf die Datenbasis zugreifen will. Mit Hilfe des gerade erzeugten Objekts können nun die Anfragen erfolgen. Ein GET-Request für die Schnittstelle „user“ sieht folgendermaßen aus:

```
echo $cd->get("user", "{}");
```

Die GET-Anfrage liefert als Ergebnis JSON-codiert „"result": 1“ falls dieser aktuelle Benutzer existiert. Weitere Beispiele dazu sind in der Dokumentation zur API auf der Website[27] des Projekts zu finden.

Die Konzepte für eine API des Learning Context Projects wurden im Rahmen einer Masterarbeit von Elyas Esnaashari untersucht. Auf diese Masterarbeit wird im folgenden Unterkapitel genauer eingegangen werden.

4.3 Bisherige Arbeiten

Im Kontext des in den vorherigen Unterkapiteln vorgestellten Learning Context Projects wurden mehrere Arbeiten verfasst. Dabei befasst sich die Masterarbeit „Users’ Decisions about the Security of Mobile Applications” von Elyas Esnaashari[2] unter anderem mit der Schnittstelle des Projektes. Elyas Esnaashari untersucht in seiner Masterarbeit wie sich Datenschutz und Sicherheit einer App in der Nutzung dieser App wiederspiegeln. Dazu geht er im weiteren Verlauf seiner Arbeit auf das Autorisierungsprotokoll OAuth ein. Ein Teil seiner Implementierung versucht, OAuth in das Learning Context Project zu integrieren. Dazu veröffentlicht er ein Konzept, wie OAuth in die API des Learning Context Projects eingebaut werden kann. Außerdem geht er auch auf die Problematik einer unverschlüsselten HTTP-Verbindung ein und stellt einen Lösungsansatz dafür vor. Seine Idee ist es, das Access-Token nicht als Plain-Text zu übertragen, sondern nur seinen Hash, und das Access-Token zusammen mit den Daten und einem „Nonce“ als gesonderten Hash zu übertragen. Somit kann ein Angreifer, der den Datenverkehr mitliest, keine Rückschlüsse auf das ursprüngliche Token ziehen.[2]

Im weiteren Verlauf seiner Arbeit baut er im Benutzermenü auf der Website des Projekts die Funktion „revoke Access“ ein. Diese hat die Funktion, dass ein Benutzer einer App den Zugriff auf die gespeicherten Daten mittels Löschen der Tokens wieder entziehen kann.[2] Da das

[27] http://docs.learning-context.de

OAuth Protokoll nur in Form von Datenbanktabellen in das Learning Context Project integriert wird, hat „revoke Access“ keine Funktion, da es von der alten API nicht unterstützt wird. Die API Version 3 basiert noch auf die Username/Password-Authentifizierung und unterstützt keine Token. Somit wird das Konzept nur exemplarisch an Beispielen dargelegt und nicht endgültig in das Learning Context Project integriert. Außerdem werden keine Libraries für die Nutzung der Schnittstelle nach dem neu vorgestellten Konzept implementiert und für Entwickler bereitgestellt, da die Schnittstelle noch nicht umgebaut ist.
Des Weiteren sieht sein Konzept für den Benutzer keine Möglichkeit vor, einer App nur teilweise die Berechtigungen zu entziehen oder nachzuschauen, welche Berechtigungen von der App überhaupt angefordert werden. Somit hätte der Benutzer nur die Möglichkeit, der App vollständig den Zugriff zu entziehen. Bei der Freigabe der Berechtigungen sieht sein Konzept für den Benutzer keine Möglichkeit vor, nur einen Teil der von der App angeforderten Berechtigungen freizugeben. Der Benutzer kann der App entweder alle angeforderten Berechtigungen erteilen oder keine.

Die im vorherigen Abschnitt genannten Probleme, dass die API noch immer auf der Username/Password-Authentifizierung basiert und dem Nutzer keine Möglichkeit zur Verwaltung der Berechtigungen gegeben wird, haben dazu geführt, dass im Rahmen dieser Arbeit ein Konzept erarbeitet wird, wie diese Probleme gelöst werden können. Das Lösungskonzept und die damit verbundenen Implementierung werden im nächsten Kapitel behandelt.

5 Design und Implementierung

Im vorherigen Kapitel wurde das Learning Context Project inklusive der bereitgestellten Schnittstelle vorgestellt. Diese Schnittstelle setzt noch auf eine Username/Password-Authentifizierung. Daher wird im Rahmen dieser Arbeit ein Konzept erarbeitet, die Schnittstelle auf eine Token-basierte Autorisierung umzustellen. Das nachfolgende Kapitel behandelt das Konzept und die Implementierung. Zusätzlich wird ein Konzept vorgestellt, wie der Benutzer die erteilten Berechtigungen im Benutzermenü bearbeiten kann. Am Ende des Kapitels werden Libraries für die Programmiersprachen Java, PHP und JavaScript behandelt. Die Libraries basieren auf den durch das Learning Context Project bereitgestellten Bibliotheken und werde an die veränderte Schnittstelle angepasst.

5.1 Rechtemanagement

Die im vorangegangenen Kapitel dargelegte Problematik, dass ein Benutzer die freigegebenen Berechtigungen nicht nachträglich ändern kann, hat dazu geführt, dass im Rahmen dieser Arbeit das Benutzermenü des Learning Context Projects umgebaut wird. Des Weiteren wird auf die Problematik eingegangen, dass ein Benutzer nachträglich nicht sehen kann, welche Berechtigungen eine App benötigt.

Damit die einzelnen Berechtigungen einem Benutzer zugeordnet werden können, muss in der Datenbank eine neue Tabelle „sources_scopes_users“ angelegt werden, die eine Verbindung zwischen der App (source), der Berechtigung (scope) und dem Benutzer (user) herstellt. Die drei Spalten können dabei als Schlüssel, auch mit „Key“ bezeichnet, angesehen werden. Die Verwendung als Key bedeutet, dass eine Kombination aus App, Berechtigung und Benutzer nur ein einziges Mal in der Datenbanktabelle existieren kann. In Tabelle 5.1 ist der Aufbau der Datenbanktabelle dargestellt. Die einzelnen Spalten bestehen aus der jeweiligen ID der App, des Benutzers und der Berechtigung.

source_id	scope_id	user_id
int(11)	int(11)	int(11)

Tabelle 5.1: Struktur der Datenbanktabelle sources_scopes_users

Die Abbildung 5.1 zeigt das hinzugefügte Formularfeld. Die Darstellungsform wurde so gewählt, um die Benutzung für den Anwender so intuitiv wie möglich zu gestalten. Daher wurden die angezeigten Daten nach Apps gruppiert. Für jede App werden die ihr zugeordneten Be-

rechtigungen aufgelistet. Die alternative Darstellungsform einer Tabelle, mit den Apps als Zeilen und den Berechtigungen als Spalten, wurde verworfen, da eine Tabelle bei einer Vielzahl an verwendeten Berechtigungen an Übersichtlichkeit verloren hätte. Die im nächsten Unterkapitel genannten Berechtigungen bilden nur den aktuellen Stand des Kontext-Modells ab. Das Kontext-Modell soll in Zukunft jedoch noch erweitert werden.
Die Auflistung der Berechtigungen für eine App kann über die Pfeile neben dem Namen der App geöffnet oder geschlossen werden. Dieses sogenannte Folding wird mit Hilfe der freien JavaScript-Library jQuery[28] realisiert. Neben der Funktion des Foldings wird jQuery für das Ein- und Ausblenden des Mülleimericons eingesetzt.
Die beiden Pfeile für das Folding und das Mülleimericon werden aus Gründen der Wiedererkennung von gewohnten Symbolen durch den Benutzer mit Hilfe der freien[29] Iconsammlung „Material icons“[30] eingebunden.

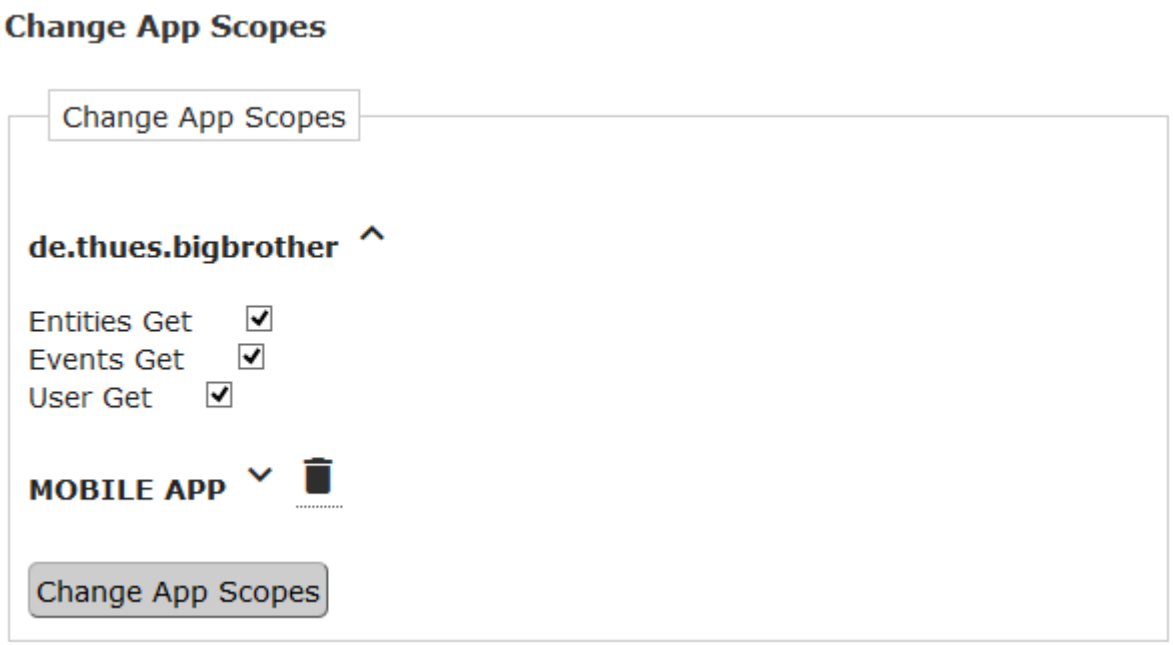

Abbildung 5.1: Das Formularfeld zur Verwaltung der Berechtigungen

Die Benutzung des Formularfeldes wurde ebenfalls so gestaltet, dass sie von einem Anwender intuitiv benutzt werden kann. Durch die angesprochene Verwendung von bekannten Icons wird dem Benutzer ermöglicht, ohne Problem die Funktion der einzelnen Komponenten zu erkennen.
Die Checkboxen der einzelnen Apps können unabhängig voneinander aktiviert und deaktiviert werden. Auch das Auf- und Zuklappen einzelner Bereiche ändert an den vorgenommenen Änderungen nichts. Die vorgenommenen Änderungen aller Apps werden erst durch einen Klick auf den Button „Change App Scopes“ an die Datenbank übertragen. Während des Speichervorgangs der Berechtigungen werden in der Datenbank alle Einträge mit der passenden Kombina-

[28] https://jquery.com/
[29] https://google.github.io/material-design-icons/#licensing
[30] https://google.github.io/material-design-icons/

tion aus „user_id“ und „source_id“ gelöscht und anhand der übertragenen Daten neu eingefügt. Dies ist problemlos möglich, da es keinen inkrementellen Primary-Key gibt, sondern sich der Key, wie im vorherigen Abschnitt beschrieben, aus den drei Spalten source_id, scope_id und user_id zusammensetzt.
Im Falle, dass ein Benutzer für eine App alle Checkboxen deaktiviert hat, er also der App keine Berechtigung mehr erteilen möchte, erscheint ein Hinweis in Form eines Popups, wie in Abbildung 5.2 dargestellt. Das Popup weist auf die Tatsache hin, dass die App im Falle einer Speicherung nicht mehr für den Benutzer sichtbar wäre und die Berechtigungen neu über die OAuth-Schnittstelle gesetzt werden müssten. Der Hintergrund zu dieser Abfrage ist, dass eine App, der keine Freigaben erteilt werden, durch die Datenbankabfrage nicht mehr dem Benutzer zugeordnet ist. Der Benutzer würde die App nach dem Speichervorgang nicht mehr angezeigt bekommen.
Über das Mülleimericon werden alle Berechtigungen, die der Benutzer der App erteilt hat, aus dem System gelöscht. Das Löschen geschieht über eine Datenbankabfrage an die Tabelle „sources_scopes_users“ mittels Filterung über „user_id“ und „source_id“ und ist nicht umkehrbar.

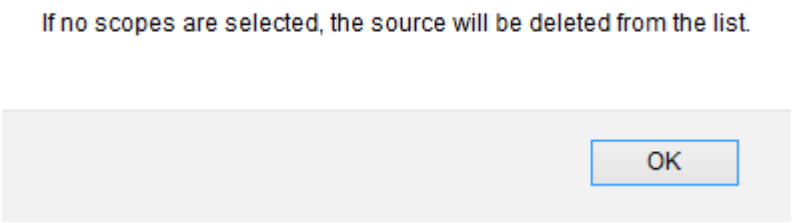

Abbildung 5.2: Hinweis, wenn alle Berechtigungen entzogen wurden

Zusätzlich zum Hinzufügen des Formularfeldes zur Änderung der Berechtigungen wurde die bestehende Funktion „revoke Access“ überarbeitet. Die Funktion stellte alle vorhanden Access-Tokens dar. Da diese Darstellung bei einer großen Anzahl an Access-Tokens für eine App unübersichtlich geworden wäre, wird eine Gruppierung der Tokens nach der jeweiligen App in der Datenbankabfrage vorgenommen. Durch den Button „revoke Access“ werden alle Refresh- und Access-Tokens aus der Datenbank gelöscht, die für die entsprechende App diesem Benutzer zugeordnet wurden.

Die in diesem Kapitel behandelte Rechteverwaltung ermöglicht es dem Benutzer, die Berechtigungen für jede einzelne App gezielt festzulegen. Außerdem besteht die Möglichkeit, den Zugriff komplett zu entziehen. Das Entziehen des Zugriffs kann über 2 verschiedene Arten vorgenommen werden. Die erste Variante ist das Löschen der App im Formularfeld. Die andere Variante ist, über den Button „revoke Access“ die Access- und Refresh-Token zu löschen. Somit kann durch die App nicht mehr auf die im nächsten Kapitel beschriebene Schnittstelle zugegriffen werden.

5.2 API

Die API des Learning Context Projects besteht aktuell aus 15 verschiedenen Schnittstellen und wurde bereits in Kapitel 4.2 genauer erläutert. Um die im vorherigen Kapitel beschriebene Rechteverwaltung nutzen zu können, wurden für 14 Schnittstellen Berechtigungen festgelegt. Die festgelegten Berechtigungen können der Tabelle 5.2 entnommen werden. Jeder Berechtigung wurde eine ID zugewiesen, auf welche im weiteren Verlauf dieses Kapitels noch einmal eingegangen wird. Die Schnittstelle „User POST“ benötigt keine eigene Berechtigung, da sie zum Erstellen eines neuen Benutzers genutzt wird. Da der Benutzer im Vorfeld der Anfrage in der Datenbank nicht existiert, könnte keine Abfrage bezüglich erteilter Berechtigungen vorgenommen werden.

ID	Name
1	Entities Get
2	Event Delete
3	Events Delete
4	Events Get
5	Events Post
6	Interest Put
7	Interests Delete
8	Interests Get
9	Interests History Delete
10	Interests History Get
11	Interests Merge Put
12	Interests Orphans Get
13	User Delete
14	User Get

Tabelle 5.2: Liste der Berechtigungen

Das Datenframe der API (vgl. Tabelle 4.1) war auf eine Authentifizierung über Benutzername und Passwort ausgelegt. Daher wurde das Frame für die neue Schnittstelle angepasst. Die Tabelle 5.3 zeigt das geänderte Frame. Dabei wird, wie bei der ursprünglichen Implementierung, auf gehashte Werte gesetzt, weil die Daten bei einer Verwendung von HTTP unverschlüsselt übertragen werden. Daher wurde Username nicht gegen das Access-Token, sondern gegen das gehashte Access-Token ausgetauscht. Hier ist ein Unterschied zu OAuth (vergleiche Kapitel 2.1), wo das Access-Token in Plain-Text übertragen wird. Auch der mitgesendete Hash hat sich verändert. Im geänderten Frame werden für den Hash, im Vergleich zum ursprünglichen Frame, das Access-Token und das gehashte Access-Token zur Generierung verwendet.

Die Implementierung des Servers wurde so abgeändert, dass der Server die neuen Datenframes verarbeiten kann. Dazu wurde die Datei „check_key.php“ angepasst. Die Schwierigkeit

data	Die zu übertragenden Daten als JSON-Object codiert.
nonce	Eine randomisierte Zeichenkette der Länge 41-59 Charakter.
aid	Die AppID der App, die die Schnittstelle anspricht.
token_h	Das mittels SHA-1 gehashte Access-Token.
hash	Ein SHA-1-Hash über die Werte: data, AppID, gehashtes Access-Token, nonce, appSecret und das Access-Token.

Tabelle 5.3: Das neue Frame der übertragenen Daten

lag darin, die ursprüngliche Funktionalität von „check_key.php“ nicht zu beeinträchtigen. Dazu wurde die Datei so angepasst, dass die „user_id“ über das Access-Token aus der Datenbank abgefragt wird und „check_key.php“ auch ohne den übertragenen Benutzernamen ein Array der Benutzerinformationen erstellen kann. Für den Zugriff auf die neuen Tabellen wurde die Hilfsfunktion für Datenbankaufrufe (DbV4.php)[1] entsprechend angepasst. Damit eine Überprüfung der Berechtigung der angefragten Schnittstelle stattfinden kann, wurde den jeweiligen Schnittstellen die aus Tabelle 5.2 zu entnehmenden IDs hinzugefügt.
Die Schnittstelle liefert als Antwort auf eine Anfrage neben dem Content einen angepassten HTTP-Statuscode zurück. Die ursprünglich definierten Statuscodes wurden überarbeitet und sind der Tabelle 5.4 zu entnehmen.

Statuscode	Erklärung
200	Die Anfrage war erfolgreich.
203	Die Berechtigungen für die App haben sich geändert.
401	Ungültiges oder abgelaufenes Access-Token.
403	Fehlerhaft übertragene Daten oder App hat keine Berechtigung für diese Schnittstelle.
500	Datenbankfehler, der Benutzer wurde nicht gefunden, obwohl ein gültiges Access-Token vorlag.

Tabelle 5.4: Statuscodes einer Antwort

Die Autorisierung einer App erfolgt über einen OAuth ähnlichen Anmeldevorgang. Der Vorgang ist ähnlich zu OAuth, da der Resource-Server und der Authorization-Server keine getrennten Systeme sind. Zudem erteilt der Benutzer der App im Voraus keine Zugriffserlaubnis, da dem System die angefragten Berechtigungen bereits bekannt sind.
Über den Aufruf der Seite `learning-context.de/oauth/login` mit dem Datenframe aus Tabelle 5.5 wird dem Benutzer die Anmeldeseite, in Abbildung 5.3 zu sehen, angezeigt. Nachdem er sich dort authentifiziert hat, wird er auf die Seite zur Auswahl der Berechtigungen weitergeleitet (siehe Abbildung 5.4). Dies geschieht nur, falls der Benutzer die App vorher noch nicht autorisiert oder im Benutzermenü gelöscht hatte. Auf der Seite sieht der Benutzer die durch die App angefragten Berechtigungen. Der Benutzer wählt die entsprechenden Berechtigungen, die er der App erteilen möchte, aus und erteilt über „Give access“ der App den Zugriff auf die ausgewählten Berechtigungen. Im Anschluss daran werden der App über eine GET-Anfrage an die Redirect-URL das Refresh-Token und das Access-Token übermittelt. Die Lebensdauer eines

Access-Tokens beträgt 30 Minuten, während ein Refresh-Token nach einem Monat abläuft und erneuert werden muss.
Zur Erneuerung der Tokens wird die Adresse `learning-context.de/oauth/refresh_token` mittels einer GET-Anfrage aufgerufen. Über die GET-Anfrage werden die AppID und ein SHA-512-Hash, bestehend aus AppID, AppSecret und Refresh-Token, übermittelt. Im Falle eines erfolgreichen Erneuerns des Access-Tokens wird dieses als JSON-Objekt ({"token":"String des Access-Tokens"}) zurückgeliefert. Auch für diese Schnittstelle wurden HTTP-Statuscodes definiert. Im Falle einer falschen AppID liefert der Server einen 400-Fehler. Falls das Refresh-Token abgelaufen sein sollte, wird vom Server der HTTP-Statuscode „401 Unauthorized“ zurückgeliefert.

id	Die AppID der App, die die Schnittstelle anspricht
url	Die Redirect-Url
hash	Ein SHA-512-Hash über die Werte: AppID, AppSecret und Redirect-URL

Tabelle 5.5: Das neue Frame der übertragenen Daten

Abbildung 5.3: OAuth Anmeldeseite

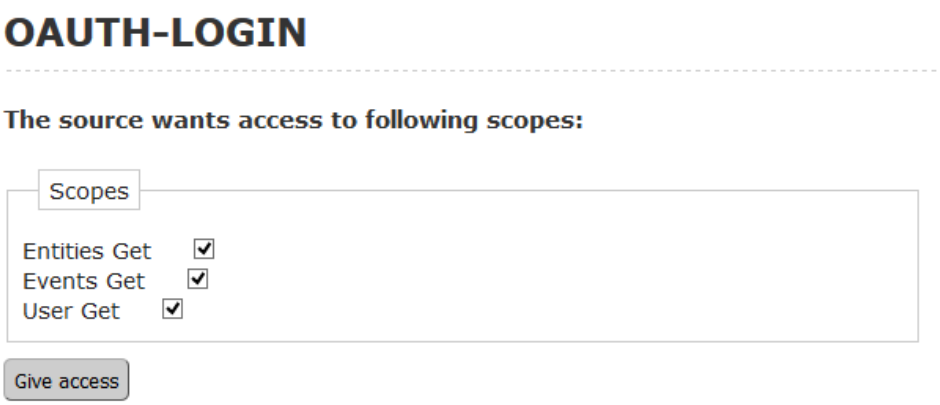

Abbildung 5.4: Seite zur Auswahl der freigegebenen Berechtigungen

In diesem Kapitel wurden die Änderungen der API beschrieben. Dabei wurde sowohl auf die Änderungen der Datenframes, als auch auf die serverseitigen Änderungen eingegangen. Die übermittelten Werte von Benutzername und gehashtem Passwort wurden durch ein gehashtes Token ersetzt. Außerdem wurde die Implementierung von OAuth genauer erläutert. Im nächsten Kapitel werden Libraries behandelt, die dem Entwickler als Hilfestellung dienen, um diese beschriebene Schnittstelle anzusprechen.

5.3 Libraries

Damit ein Entwickler die im vorherigen Kapitel behandelte Schnittstelle ansprechen kann, wurden exemplarisch Libraries für die verschiedenen Programmiersprachen Java (Android), PHP und JavaScript entwickelt und implementiert. Diese sollen dem Entwickler helfen, besser auf Fehler zu reagieren oder eine Unterstützung bei der Aktualisierung der Tokens liefern. Die Grundlage dieser Libraries stellen die durch das Learning Context Projects bereitgestellten Libraries dar, welche auf eine Token-basierte Kommunikation umprogrammiert wurden. Ziel der Libraries ist es, dem Entwickler einen Großteil der Kommunikation mit der Schnittstelle abzunehmen. Das bedeutet, dass sich die Library eigenständig um das Verwalten und Aktualisieren der Tokens kümmert und in einem gewissen Grad eigenständig auf die zurückgelieferten HTTP-Statuscodes reagiert. Somit muss der Entwickler nur noch ein Speichern des Refresh-Tokens gewährleisten, damit dieses nicht bei jedem Neustart der Anwendung erzeugt werden muss.

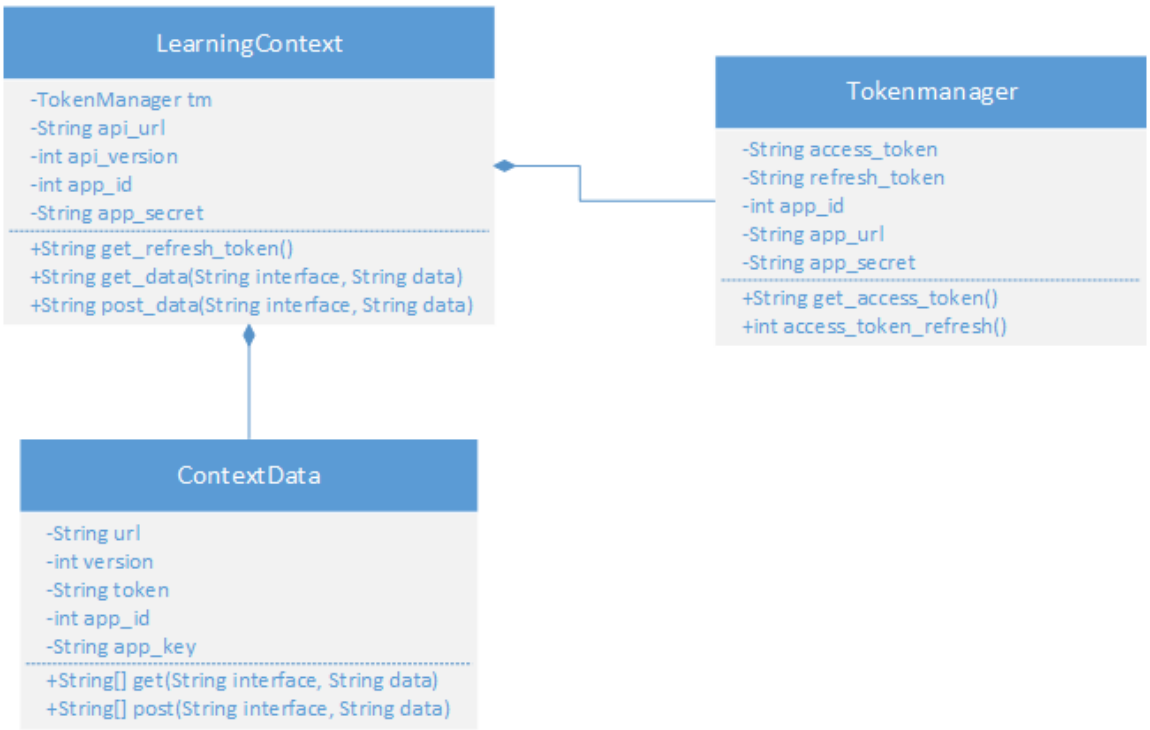

Abbildung 5.5: Klassendiagramm der Library

Das Klassendiagramm in Abbildung 5.5 zeigt das Grundkonzept des Aufbaus einer Library. Jede Library besteht aus den folgenden drei Klassen: LearningContext, TokenManager und

ContextData. Die Klassen Event und Entity sind ebenfalls in der Library vorhanden, wurden aber im Rahmen dieser Arbeit nicht geändert und in ihrem Ursprungszustand belassen:

LearningContext ist die Hauptklasse der Library. Sie muss vom Entwickler eingebunden werden, damit er die Library nutzen kann.

ContextData ist für die Kommunikation mit der API zuständig. Sie führt die GET-, DELETE-, PUT- und POST-Anfragen durch.

TokenManager bündelt die Handhabung des Access-Tokens und des Refresh-Tokens. Die Klasse speichert die beiden Tokens zwischen und kümmert sich um die Erneuerung des Access-Tokens. Falls das Refresh-Token nicht gültig oder kein Token vorhanden sein sollte, instanziiert der TokenManager die Weiterleitung zum Anmeldeformular auf der Learning-Context-Seite.

Event erzeugt ein Objekt für ein Event. Jedes Objekt vom Typ Event beinhaltet ein Array oder eine Liste vom Typ Entity.

Entity speichert einen Key-Value-Eintrag.

Der Aufbau der Klasse ist von der jeweiligen Programmiersprache abhängig, da nicht in allen Programmiersprachen die benötigten Funktionen mitgeliefert werden, sondern teilweise implementiert werden mussten. Das Klassendiagramm liefert einen groben Überblick über das Konzept der Library.

Der Ablauf einer GET-Anfrage mit Hilfe der Library ist als Sequenzdiagramm in Abbildung 5.6 dargestellt. Dabei werden die folgenden Schritte durch die Library abgearbeitet:

1. Die externe Anwendung ruft den Konstruktor der Klasse Learning Context auf.
2. LearningContext erzeugt ein Objekt vom Typ TokenManager.
3. Falls an TokenManager keine Tokens übergeben wurden, leitet dieser den Benutzer an die Anmeldeseite des Learning Context Projects weiter. Ansonsten fragt er ein Access-Token bei der Schnittstelle an.
4. Die externe Anwendung ruft get_data() von LearningContext auf.
5. LearningContext fragt das Access-Token bei dem TokenManager an.
6. LearningContext startet den Konstruktor von ContextData.
7. LearningContext ruft get() von ContextData auf.
8. ContextData stellt die Anfrage an die Schnittstelle und liefert die Antwort zurück.
9. Wenn der Statuscode!=401 ist, liefert LearningContext den erhaltenen Content an Anwendung zurück. Ansonsten ruft LearningContext refresh_access_token() des TokenManagers zur Erneuerung des Access-Tokens auf.
10. LearningContext fragt das Access-Token bei dem TokenManager an.
11. LearningContext startet den Konstruktor von ContextData.
12. LearningContext ruft get() von ContextData auf.
13. ContextData stellt die Anfrage an die Schnittstelle und liefert die Antwort zurück.
14. LearningContext liefert den erhaltenen Content unabhängig vom Statuscode an Anwendung zurück.

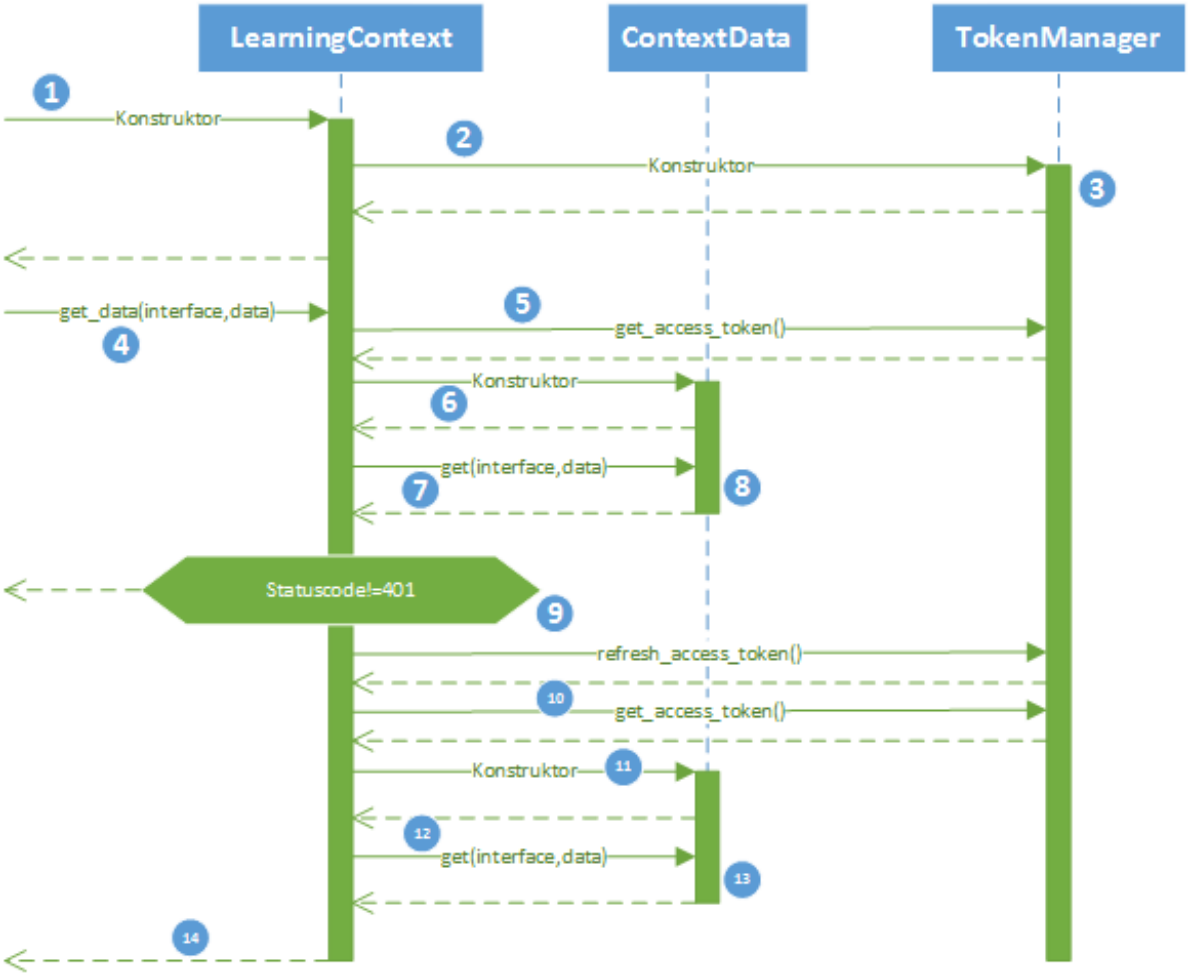

Abbildung 5.6: Sequenzdiagramm der Library

Der beschriebene Ablauf ist bei allen Libraries, unabhängig von der Programmiersprache gleich. Auf die implementierten Libraries wird in den folgenden Unterkapiteln genauer eingegangen. Dabei wird die Verwendung der jeweiligen Library anhand von Codezeilen dargestellt. In Kapitel 6 wird die Funktion der Libraries getestet und der Aufbau anhand von realen Beispielen gezeigt.

5.3.1 Android

Diese Unterkapitel behandelt die Umsetzung der im vorherigen Kapitel vorgestellten Library für die Programmiersprache Java auf dem Betriebssystem Android. Die neu implementierte Library benötigt als minimale Anforderung Android 4.4 (KitKat)[31] beziehungsweise die Android-API 19 und die API-Version 4 des Learning Context Projects. Außerdem baut die neue Library auf der durch das Learning Context Project bereitgestellten Library auf.

[31] http://www.android.com/versions/kit-kat-4-4/

Damit die Library mit der im vorherigen Kapitel beschriebenen Schnittstelle kommunizieren kann, wurden die Klassen LearningContext und TokenManager hinzugefügt. Zur vereinfachten Handhabung für einen Entwickler einer externen App wurde die Library so aufgebaut, dass nur die Klasse LearningContext inkludiert werden muss. Dies wird im späteren Verlauf dieses Kapitels genauer erläutert.
Die alte Library setzte zur Kommunikation auf die Android-Bibliothek HTTPClient. Diese ist mittlerweile veraltet[32], so dass die Kommunikation auf die aktuelle Bibliothek HttpURLConnection[33] umgebaut wurde. Die Bibliothek HttpURLConnection unterstützt ebenfalls Timeouts und liefert die HTTP-Statuscodes zurück. Außerdem wurde die Funktionalität der asynchronen Tasks verändert. Die asynchronen Tasks zur Schnittstellenkommunikation werden nun in der Klasse LearningContext, anstatt in der Klasse ContextData aufgerufen. Durch die Verwendung von asynchronen Tasks ist sichergestellt, dass die Anwendung nicht durch eine Anfrage an die Schnittstelle einfriert (vergleiche Kapitel 2.3).

Die Arbeit mit der Library soll in diesem Abschnitt exemplarisch dargestellt werden. Damit ein Benutzer die Library nutzen kann, muss er die Klasse LearningContext in die Java-Klasse seiner Activity, in der er die Schnittstelle nutzen möchte, einbinden.

```
1 import LearningContext.LearningContext;
```

Die Berechtigungen, die die App mindestens benötigt, um die Library einbinden zu können, sind der Zugriff auf die Internetverbindung und das Abfragen des Netzwerkstatus. Daher müssen in der AndroidManifest.xml folgende Einträge hinzugefügt werden, um die Library vollständig nutzen zu können:

```
1 <uses-permission android:name="android.permission.INTERNET" />
2 <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"
    />
```

Damit der Browser den Benutzer wieder an die App zurück leiten kann, müssen zusätzlich noch folgende Zeilen durch den Entwickler in der AndroidManifest.xml eingetragen werden:

```
1 <intent-filter>
2     <data android:scheme="learning-context" />
3     <action android:name="android.intent.action.VIEW" />
4     <category android:name="android.intent.category.BROWSABLE" />
5     <category android:name="android.intent.category.DEFAULT" />
6 </intent-filter>
```

Nun legt der Entwickler in der Klasse ein Attribut vom Typ LearningContext an und weist diesem in der Funktion onCreate ein neues Objekt zu. Für die Werte von API-Version, AppID, AppSecret und Activity müssen die entsprechen Konfigurationen und erhaltenen Daten eingefügt werden. Da die alte API-Version auf einer anderen Art von Authentifizierung basierte, ist die Minimalanforderung Version 4. Die Angabe eines Refresh-Tokens ist optional.

[32] http://developer.android.com/preview/behavior-changes.html#behavior-apache-http-client
[33] http://developer.android.com/reference/java/net/HttpURLConnection.html

```
lc = new LearningContext(
       "http://api.learning-context.de",
       API-Version,
       AppID,
       "AppSecret",
       "learning-context://lc",
       Refresh-Token,
       "Activity".this);
```

Falls kein Refresh-Token übergeben wurde, leitet der TokenManager mittels Aufruf des Browsers des Smartphones automatisch auf die Anmeldeseite des Learning Context Projekts weiter.

```
Intent browserIntent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
context.startActivity(browserIntent);
```

Nach erfolgreicher Authentifizierung durch den Benutzer und Freigabe der angefragten Berechtigungen, leitet der Server wieder an die Redirect-URL (hier die App) zurück. Durch das vorherige Anlegen des Intentfilters öffnet sich die App. Das Auslesen des übertragenen Refresh-Tokens muss durch den Benutzer vorgenommen werden.

```
String rt = "";
try {
  Uri uri = getIntent().getData();
    if (uri.getQueryParameter("rt") != "") {
        rt = uri.getQueryParameter("rt");
    }
} catch (NullPointerException e) {
    Log.d("Main","Refresh-Token error");
}
```

Wenn das so erhaltene Refresh-Token bei der Erzeugung des LearningContext-Objekts übergeben wird, fragt der TokenManager in einem asynchronen Task selbstständig ein Access-Token an. Das durch den TokenManager temporär gespeicherte Refresh-Token kann mittels des Aufrufes

```
lc.get_refresh_token();
```

ausgelesen werden. Eine Speicherung und Übermittlung des Refresh-Tokens beim Erzeugen des LearningContext-Objekts ist sinnvoll, da der Benutzer ansonsten bei jedem Aufruf der App erneut zur Anmeldeseite weitergeleitet werden würde.

Ein Aufruf der Schnittstellen via HTTP-GET, HTTP-PUT, HTTP-DELETE und HTTP-Post ist über den Aufruf der entsprechenden Funktion des vom Typ LearningContext erzeugten Objektes möglich. Da die Anfragen in einem asynchronen Task ausgeführt werden, müssen Listener erstellt werden, damit die Antworten dargestellt werden können. Zur korrekten Funktion müssen die Platzhalter „Schnittstelle“ gegen die anzusprechende Schnittstelle und „Daten“ gegen das zu übertragende JSON-Objekt ausgetauscht werden.

```
lc.get("Schnittstelle", 'Daten'));
lc.post("Schnittstelle", 'Daten'));
lc.put("Schnittstelle", 'Daten'));
lc.delete("Schnittstelle", 'Daten'));
```

Durch die in diesem Unterkapitel beschriebenen Funktionen zum Aufruf der Schnittstelle, wird die Handhabung der Schnittstelle in einem Android-Projekt erleichtert. Im nächsten Unterkapitel wird auf die Implementierung der Library in PHP eingegangen.

5.3.2 PHP

Die in diesem Unterkapitel beschriebene Implementierung von PHP basiert auf die PHP-Library des Learning Context Projects. Die neu implementierte Library benötigt als minimale Anforderung PHP 5.3 und die API-Version 4 des Learning Context Projects. Damit die Library mit der im vorherigen Kapitel beschriebenen Schnittstelle kommunizieren kann, wurden die Klassen LearningContext und TokenManager hinzugefügt. Zur vereinfachten Handhabung für einen Entwickler einer externen App wurde die Library so aufgebaut, dass nur die Klasse LearningContext inkludiert werden muss. Dies wird im späteren Verlauf dieses Kapitels genauer erläutert.
Da die alte Library keine HTTP-Statuscodes verarbeiten kann, wurde die Kommunikation auf die PHP-eigene Library cURL umgebaut. cURL liefert, wie in Kapitel 2.2 beschrieben, im Gegensatz zu dem vorher verwendeten file_get_contents() die http-Statuscodes zurück. Dadurch ist es für die Library möglich, auf einen Teil der ankommenden Statuscodes eigenständig zu reagieren.
Zusätzlich zu den genannten Änderungen wurden die übertragenen Frames auf die neue Version der Schnittstelle angepasst. Im Gegensatz zur im vorherigen Unterkapitel beschriebenen Library für Android, ist die Kommunikation zur Schnittstelle nicht in einen asynchronen Task ausgelagert. Diese Tatsache führt dazu, dass ein Blockieren der Kommunikationsschnittstelle zu einem Blockieren des gesamten PHP-Skripts führen würde. Daher wurde in cURL ein Timeout für die maximale Dauer einer Verbindung gesetzt.

Die Arbeit mit der Library soll in diesem Abschnitt exemplarisch dargestellt werden. Damit ein Benutzer die Library nutzen kann, muss er die Klasse LearningContext in sein Projekt einbinden.

```
1 require_once 'learning-context/LearningContext.php';
```

Nun erzeugt der Entwickler ein neues Objekt vom Typ LearningContext. Für die Werte von API-Version, AppID, AppSecret und Redirect-URL müssen die entsprechen Konfigurationen und erhaltenen Daten eingefügt werden. Da die alte API-Version auf einer anderen Art von Authentifizierung basierte, ist die Minimalanforderung zur Nutzung der Library Version 4. Die Angabe eines Refresh-Tokens ist optional.

```
1 $lc = new LearningContext(
2     'http://api.learning-context.de',
3     'API-Version',
4     'AppID',
5     'AppSecret',
6     'Redirect-URL',
7     'Refresh-Token'
8 );
```

Falls kein Refresh-Token übergeben wurde, leitet der TokenManager mittels einer Header-Modifikation automatisch auf die Anmeldeseite des Learning Context Projekts weiter.

```
header('Location: http://www-dev.learning-context.de/oauth/login?
    id='.$this->app_id.'&
    url='.$this->app_url.'&
    hash='.hash('sha512',$this->app_id.$this->app_secret.$this->app_url)
);
```

Nach erfolgreicher Authentifizierung durch den Benutzer und Freigabe der angefragten Berechtigungen, leitet der Server wieder an die Redirect-URL zurück. Das Auslesen des übertragenen Refresh-Tokens übernimmt hierbei der TokenManager. Der TokenManager fragt nun selbstständig ein Access-Token an. Das durch den TokenManager temporär gespeicherte Refresh-Token kann mittels des Aufrufes

```
$lc->get_refresh_token();
```

ausgelesen werden. Eine Speicherung und Übermittlung des Refresh-Tokens beim Erzeugen des LearningContext-Objekts ist sinnvoll, da der Benutzer ansonsten bei jedem Aufruf der Seite erneut zur Anmeldeseite weitergeleitet werden würde.

Ein Aufruf der Schnittstellen via HTTP-GET, HTTP-DELETE, HTTP-PUT und HTTP-POST ist über den Aufruf der entsprechenden Funktion des vom Typ LearningContext erzeugten Objektes möglich. Die zurückgelieferten Ergebnisse werden mittels print_r() angezeigt. Zur korrekten Funktion müssen „Schnittstelle“ gegen die anzusprechende Schnittstelle und „Daten“ gegen das zu übertragende JSON-Objekt ausgetauscht werden.

```
print_r ($lc->get_data("Schnittstelle", 'Daten'));
print_r ($lc->post_data("Schnittstelle", 'Daten'));
print_r ($lc->put_data("Schnittstelle", 'Daten'));
print_r ($lc->delete_data("Schnittstelle", 'Daten'));
```

Durch die in diesem Unterkapitel beschriebene Funktionen zum Aufruf der Schnittstelle, wird die Handhabung der Schnittstelle in einem PHP-Projekt erleichtert. Im nächsten Unterkapitel wird die Implementierung der Library in JavaScript genauer erläutert.

5.3.3 JavaScript

Dieses Unterkapitel beschreibt die Implementierung der Library für die Programmiersprache JavaScript. Die Library baut auf die durch das Learning Context Project bereitgestellte Bibliothek für JavaScript auf. Für die Nutzung der Library wird lediglich ein aktueller Webbrowser und die API-Version 4 des Learning Context Projects vorausgesetzt.

Damit die Library mit der im vorherigen Kapitel beschriebenen Schnittstelle kommunizieren kann, wurden die Klassen LearningContext und TokenManager hinzugefügt. Zur vereinfachten Handhabung für einen Entwickler in einer Website wurde die Library so aufgebaut, dass der Entwickler nur ein Objekt vom Typ LearningContext ansprechen muss. Dies wird im späteren Verlauf dieses Kapitels genauer erläutert.
Zur Kommunikation mit dem Server wird die Library XMLHttpRequest verwendet. Diese Library ist für die Kommunikation via HTTP gedacht. Die HTTP-Anfrage erfolgt im Gegensatz zu Android in einem synchronen Ablauf. Daher kann ein Blockieren der Anfrage zu einem Blockieren des gesamten Skripts führen.

Die Arbeit mit der Library soll in diesem Abschnitt exemplarisch dargestellt werden. Damit ein Benutzer die Library nutzen kann, muss er die JavaScript-Dateien in sein Projekt einbinden. Zusätzlich wird für die Generieung des SHA-512-Hashes die externe Library CryptoJS[34] eingebunden.

```
<script src="http://crypto-js.googlecode.com/svn/tags/3.1.2/build/
    rollups/sha512.js"></script>
<script src="scripts/LearningContext.js" type="text/javascript"></script
    >
<script src="scripts/LearningContextUtil.js" type="text/javascript"></
    script>
<script src="scripts/TokenManager.js" type="text/javascript"></script>
<script src="scripts/ContextData.js" type="text/javascript"></script>
<script src="scripts/Entity.js" type="text/javascript"></script>
<script src="scripts/Event.js" type="text/javascript"></script>
```

Nun erzeugt der Entwickler ein neues Objekt vom Typ LearningContext. Für die Werte von API-Version, AppID, AppSecret und Redirect-URL müssen die entsprechen Konfigurationen und erhaltenen Daten eingefügt werden. Da die alte API-Version auf einer anderen Art von Authentifizierung basierte, ist die Minimalanforderung zur Nutzung der Library Version 4. Die Angabe des Refresh-Tokens ist im Vergleich zu PHP nötig, da der TokenManager das Refresh-Token nicht eigenständig auslesen kann. Die angegegebene Funktion[9] ermöglicht das Auslesen des übergebenen Refresh-Tokens durch Zugriff auf die übermittelten GET-Parameter.

```
(function(){
  var s = window.location.search.substring(1).split('&');
  if(!s.length) return;
  window.$_GET = {};
  for(var i  = 0; i < s.length; i++) {
    var parts = s[i].split('=');
    window.$_GET[unescape(parts[0])] = unescape(parts[1]);
  }
})()

var lc = new LearningContext("http://learning-context.de",
               4,
               AppID,
               AppSecret,
               Redirect-URL,
               $_GET['rt']);
```

Falls kein Refresh-Token übergeben wurde, leitet der TokenManager mittels einer Location-Modifikation automatisch auf die Anmeldeseite des Learning Context Projekts weiter.

```
window.location = 'http://www-dev.learning-context.de/oauth/login?
       id='+this.app_id+'&
       url='+this.app_url+'&
       hash='+CryptoJS.SHA512(this.app_id+this.app_secret+this.app_url);
```

[34] https://code.google.com/p/crypto-js/

Nach erfolgreicher Authentifizierung durch den Benutzer und Freigabe der angefragten Berechtigungen leitet der Server wieder an die Redirect-URL zurück. Der TokenManager fragt nun selbstständig ein Access-Token an.
Ein Aufruf der Schnittstellen via HTTP-GET, HTTP-DELETE, HTTP-PUT und HTTP-POST ist über den Aufruf der entsprechenden Funktion des vom Typ LearningContext erzeugten Objektes möglich. Die zurückgelieferten Ergebnisse werden mittels console.log() in der Konsole angezeigt. Zur korrekten Funktion müssen „Schnittstelle“ gegen die anzusprechende Schnittstelle und „Daten“ gegen das zu übertragende JSON-Objekt ausgetauscht werden.

```
1 console.log(lc.get("Schnittstelle", 'Daten'));
2 console.log(lc.post("Schnittstelle", 'Daten'));
3 console.log(lc.put("Schnittstelle", 'Daten'));
4 console.log(lc.delete("Schnittstelle", 'Daten'));
```

Durch die in diesem Unterkapitel beschriebene Funktionen zum Aufruf der Schnittstelle wird die Handhabung der Schnittstelle in einem JavaScript-Projekt erleichtert.
Die in diesem Kapitel exemplarisch vorgestellten Libraries wurden auf ihre korrekte Funktionsweise getestet. Die Vorgehensweise bei den Tests und deren Ergebnisse werden im nächsten Kapitel genauer beschrieben und erläutert.

6 Evaluation

Im vorherigen Kapitel wurden Libraries für die Kommunikation mit der API des Learning Context Projects vorgestellt. Um die ordnungsgemäße Funktion der vorgestellten Libraries zu garantieren, wurden im Rahmen dieser Arbeit verschiedene Tests mit den Libraries durchgeführt, die in diesem Kapitel genauer erläutert werden.

6.1 Testaufbau

Für die Tests wurden aus den 15 Schnittstellen der API zwei Schnittstellen ausgewählt. Damit ein großes Spektrum abgebildet werden kann, wurden „user_GET“ und „user_POST“ verwendet. Die Schnittstelle „user_GET“ liefert ein positives Resultat zurück, falls der aktuelle Benutzer existiert. Daher kann mit dieser Schnittstelle die ordnungsgemäße Funktion des Access-Tokens überprüft werden. Außerdem können durch gezielt falsche Anfragen an diese Schnittstelle die Fehlercodes abgerufen werden. Die Schnittstelle „user_POST“ wurde ausgewählt, da diese Schnittstelle die einzige Schnittstelle ohne Autorisierung ist. So konnte überprüft werden, ob die API auch noch mit diesem Spezialfall funktioniert.
Da Android ein eigenes Betriebssystem ist, wurde auf verschiedene Emulatoren zurückgegriffen, um die korrekte Funktion der Library zu testen. Dabei kamen der durch das Android Studio[35] bereitgestellte Emulator und BlueStacks[36], ein freier Emulator für Android, zum Einsatz. Mit der Hilfe des Emulators von Android Studio wurde ein Nexus 5 mit der Android-Version 5.1 emuliert. Neben den Tests mit den beiden Emulatoren wurde auch ein Test mit einem Smartphone durchgeführt. Dabei wurde ein HTC One M7 mit der Android-Version 5.0 eingesetzt.
Bei PHP handelt es sich um eine serverseitige Programmiersprache. Daher musste zum Testen ein Server aufgesetzt werden, welcher PHP unterstützt. Dazu bot sich die freie Software XAMPP[37] an. XAMPP erzeugt einen lokalen Apache-Server und unterstützt MySQL, PHP und Perl. Somit können die Testskripte auf dem erzeugten Server über einen Webbrowser aufgerufen werden.
Für die Tests von JavaScript wurde neben einem Webbrowser keine weitere Software eingesetzt. Der Grund dafür ist, dass JavaScript eine clientseitige Sprache ist und eingebettet in eine HTML-Seite im Webbrowser ausgeführt werden kann.

Nachdem in diesem Kapitel der Testaufbau für die einzelnen Libraries beschrieben wurde, wird im nächsten Unterkapitel auf die Implementation der Test genauer eingegangen.

[35] https://developer.android.com/sdk/
[36] http://www.bluestacks.com/
[37] https://www.apachefriends.org/de/

6.2 Testprogramme

Zum Testen der ausgewählten Schnittstellen wurden Testskripte beziehungsweise Testapplikationen für die einzelnen Programmiersprachen und Systeme (Android, PHP und JavaScript) geschrieben.

Da Android ein Betriebssystem ist, musste eine App geschrieben werden. In der Implementierung der App wird das LearningContext-Objekt mit den folgenden Parametern in der onCreate-Methode der Main-Activity instanziiert. Zusätzlich wurde die Activity auf die Listener des LearningContext-Objekts registriert:

```
lc = new LearningContext(
       "http://api.learning-context.de",
       4,
       1,
       "",
       "learning-context://lc",
       rt,
       MainActivity.this);

lc.registerGETListener(this);
lc.registerPOSTListener(this);
```

Da es sich bei dem AppSecret um ein real existierendes AppSecret handelt, wurde dieses aus dem Beispiel entfernt. Die Varbiable „rt“ für das Refresh-Token wurde, wie im Codebeispiel 5.3.1 dargestellt, instanziiert und verwendet.
Für die Ausgabe der Antworten der GET- und Post-Anfragen wurden Textfelder angelegt, welche über einen Listener befüllt werden.

```
@Override
public void onGETResult(String result) {
    final TextView textView = (TextView) findViewById(R.id.textView);
    textView.setText(result);
}

@Override
public void onPOSTResult(String result) {
    final TextView textView = (TextView) findViewById(R.id.textView2);
    textView.setText(result);
}
```

Eine Anfrage an die Schnittstelle wird durch das Drücken eines Buttons ausgelöst. Dazu wurde ein onClickListener auf einen vorher gesetzten Button registriert. Bei einem Klick auf den Button wird eine GET-Anfrage an die Schnittstelle „user“ gesendet, um zu überprüfen ob der Benutzer existiert. Zudem wird über die Schnittstelle „user“ mittels einer POST-Anfrage versucht, einen Benutzer mit dem Namen „Max Mustermann“, dem Passwort „secret“ und der Mail-Adresse „max@mustermann.de“ anzulegen.

```
final Button button = (Button) findViewById(R.id.button);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        lc.get("user", "{}");
        lc.post("user", "{\"name\": \"Max Mustermann\",  \"pass\": \"
            secret\",  \"email\": \"max@mustermann.de\"}");
    }
});
```

Bei dem Testskript für PHP wurde, im Gegensatz zu Android, zuerst überprüft, ob ein Refresh-Token gespeichert wurde. Dies ist notwendig, da bei einem erneuten Laden des Skripts alle in den Variablen gespeicherten Daten weg wären. Bei Android ist das nicht der Fall, da dort der Aufruf nicht über einen Neustart der App, sondern über eine Nutzerinteraktion ausgelöst wird. Im Anschluss wird das LearningContext-Objekt erzeugt. Dabei ist wieder aus den im vorherigen Abschnitt genannten Gründen kein Wert für das AppSecret eingetragen. Die Redirect-URL beinhaltet einen GET-Parameter, welcher für die Funktion des Beispiels nicht benötigt wird. Der eingetragene GET-Parameter soll lediglich testen, ob das Übermitteln von GET-Parametern über die Redirect-URL von der API unterstützt wird. Im Anschluss daran wird das Refresh-Token abgefragt und im Falle einer Änderung gespeichert.

```
session_start();
if(isset($_SESSION["rt"])) {
  $rt = $_SESSION["rt"];
} else {
  $rt = '';
}
require_once 'learning-context/LearningContext.php';
$lc = new LearningContext('http://api.learning-context.de',
                '4',
                '1',
                '',
                'http://localhost/test/index.php?user_id=1',
                $rt);
if((!isset($_SESSION["rt"])) || ($_SESSION["rt"]!=$lc->get_refresh_token
    ())) {
    $_SESSION["rt"] = $lc->get_refresh_token();
}
```

Für eine Anfrage an die Schnittstelle werden die entsprechenden Funktionen get_data() und post_data() aufgerufen und mittels print_r() ausgegeben. Die Abfrage an die Schnittstelle wird automatisch beim Aufruf des Skripts ausgeführt. Es ist keine Interaktion von Seiten des Anwenders nötig.

```
print_r ($lc->get_data("user", '{}'));
echo '<br/>';
print_r ($lc->post_data("user", '{"name":"Hans Wurst",
"pass": "secret",
"email": "hans@wurst.de"}'));
```

Für das Testskript von JavaScript wird keine spezielle Software benötigt, da JavaScript in eine HTML-Seite eingebettet werden kann. Nachdem die benötigten Dateien, wie in Kapitel 5.3.3 beschrieben, inkludiert wurden, werden die GET-Parameter mittels einer Funktion ausgelesen

und für die weitere Verwendung vorbereitet. Im Anschluss daran wird das LearningContext-Objekt erzeugt. Dabei ist wieder aus den in den vorherigen Abschnitten genannten Gründen kein Wert für das AppSecret eingetragen. Die Redirect-URL beinhaltet einen GET-Parameter, welcher für die Funktion des Beispiels nicht benötigt wird. Der eingetragenen GET-Parameter soll nur lediglich testen, ob das Übermitteln von GET-Parametern über die Redirect-URL von der API unterstützt wird. Im Anschluss daran wird das Refresh-Token abgefragt und im Falle einer Änderung gespeichert. Die Erzeugung des LearningContext-Objektes wird folgendermaßen vorgenommen:

```
1 var lc = new LearningContext("http://api.learning-context.de",
2                4,
3                1,
4                '',
5                'http://localhost/test/index.html?user_id=1',
6                $_GET['rt']);
```

Für eine Anfrage an die Schnittstelle werden die entsprechenden Funktionen get() und post() aufgerufen und mittels console.log() auf der Konsole ausgegeben. Die Abfrage an die Schnittstelle wird automatisch beim Aufruf des Skripts ausgeführt. Es ist keine Interaktion von Seiten des Anwenders nötig. Beim Laden des Skripts wird eine GET-Anfrage an die Schnittstelle „user“ gesendet, um zu überprüfen, ob der Benutzer existiert. Zudem wird über die Schnittstelle „user“ mittels einer POST-Anfrage versucht, einen Benutzer mit dem Namen „Max Mustermann“, dem Passwort „secret“ und der Mail-Adresse „max@mustermann.de“ anzulegen.

```
1 var result01 = lc.get("user", '{}');
2 console.log(result01);
3 var result02 = lc.post("user", '{ "name": "Max Mustermann",  "pass": "
    secret",  "email": "max@mustermann.de"}');
4 console.log(result02);
```

Nachdem in diesem Unterkapitel die Testskripte und Testprogramme für die einzelnen Programmiersprachen vorgestellt wurden, behandelt das nächste Unterkapitel die Ergebnisse der vorgenommenen Tests.

6.3 Ergebnis

Die im vorherigen Kapitel beschriebenen Testprogramme und Testsskripte wurden ausgeführt und die erhaltenen Ergebnisse analysiert. Das Ergebnis dieser Tests ist, wie in den Abbildungen 6.1 und 6.2 zu sehen ist, dass die Libraries ordnungsgemäß arbeiten. Abbildung 6.1 zeigt dabei den erstmaligen Aufruf von „user_GET“ beziehungsweise „user_POST“. Als Antwort wird bei beiden „"result":1“ zurückgeliefert. In Abbildung 6.2 ist die Antwort eines erneuten Aufrufs der beiden Schnittstellen mit denselben Informationen dargestellt. Dabei liefert „user_GET“ immer noch das erwartet Ergebnis. Bei „user_POST“ hat sich die Antwort geändert, da der Benutzer bereits vorhanden ist. Somit werden Fehlermeldungen erfolgreich zurückgeliefert.

Da für die beiden Schnittstellen korrekte Werte zurückgeliefert werden, funktioniert die Sicherheitsüberprüfung der API mit der Umstellung auf eine Token-basierte Autorisierung. Bei den Tests mit falschen AppIDs, AppSecrets oder einem falschen Hash wurden die entsprechenden Statuscodes (siehe Tabelle 5.4) zurückgeliefert.

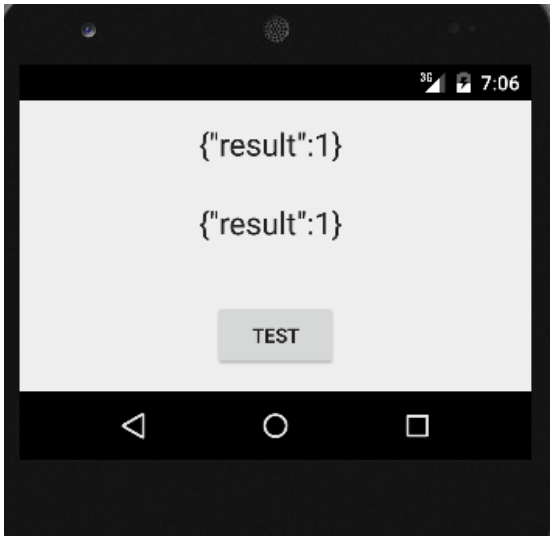

Abbildung 6.1: Antworten auf die Anfragen an die Schnittstelle

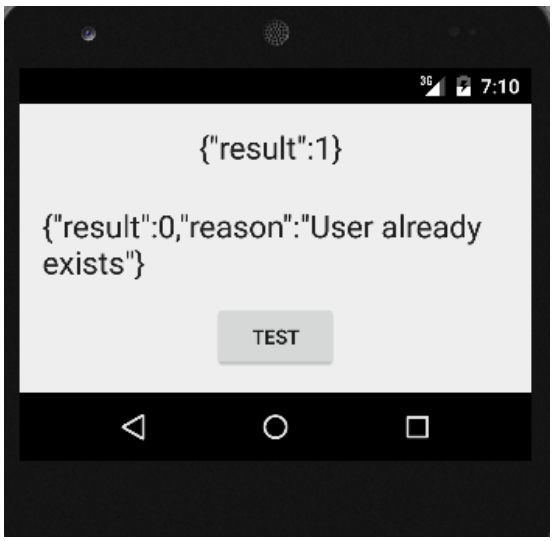

Abbildung 6.2: Antworten nach erneutem Ausführen der Anfragen

Durch die Tests wurde überprüft, ob die Libraries und die API ordnungsgemäß funktionieren. Das Ergebnis ist, dass die API und die Libraries für die einzelnen Programmiersprachen sich wie im Konzept vorgesehen verhalten und die erwarteten Ergebnisse zurückliefern. Somit ist die korrekte Implementation und Funktion der Libraries und der API bewiesen.

7 Fazit

Im Rahmen dieser Bachelorarbeit wurde für das Learning Context Project eine Rechteverwaltung geschaffen, die es dem Benutzer ermöglicht, seine den Apps erteilten Berechtigungen zu verwalten. Dabei wurde ihm die Möglichkeit gegeben, die Freigabe für eine Berechtigung zu entziehen, aber auch nachträglich zu erteilen. Er hat nun die Option, einer App den Zugriff auf seine Daten komplett zu verweigern. Damit wurde die Problematik behoben, dass ein Benutzer einer App keine Freigaben entziehen konnte.
Als weiterer Teil dieser Arbeit wurde die Schnittstelle des Learning Context Projects von einer Username/Password-Authentifizierung auf eine Token-basierte Autorisierung umgebaut. Dazu wurden die APIs von Google und Facebook untersucht und eine Masterarbeit herangezogen, welche das Thema angeschnitten hat. Darauf aufbauend wurde das OAuth-Protokoll angepasst und in die API implementiert. Dafür wurden die übertragenen Datenframes geändert und die jeweiligen, zurückgelieferten HTTP-Statuscodes angepasst. Somit muss ein Benutzer nun einer App nicht mehr seinen Benutzernamen und das Passwort preisgeben, sondern er kann die App über die neu integrierte Anmeldeseite autorisieren und die entsprechenden, durch die App angeforderten Berechtigungen freigeben.
Im Anschluss daran wurde ein Konzept für den Kommunikationsaufbau entwickelt und in die vorhanden Libraries implementiert. Dabei wurde nicht nur das neue System integriert, sondern auch veraltete Bibliotheken aus den Libraries entfernt und durch aktuelle Bibliotheken ausgetauscht. Den Entwicklern von externen Anwendungen stehen nun Libraries in den Programmiersprachen Java, PHP und JavaScript für die aktuellste Version der Schnittstelle zur Verfügung.
Die abgeänderte Schnittstelle und die implementierten Libraries wurden im letzten Schritt dieser Arbeit auf ihre Funktion getestet. Dazu wurden Testprogramme beziehungsweise Testskripte geschrieben und überprüft, ob bei Anfragen die erwartete Antwort zurückgeliefert wird. Durch diese Tests konnte die korrekte Funktion der Schnittstelle und der Libraries nachgewiesen werden.
Zusammenfassend kann festgestellt werden, dass die in der Einleitung definierten Ziele der Arbeit erreicht wurden. Dem Benutzer wurde die Möglichkeit gegeben, seine erteilten Berechtigungen zu verwalten und die alte Schnittstelle wurde auf eine Token-basierte Autorisierung umgebaut. Zudem wurden die vorhandenen Libraries erweitert und an die neue Schnittstelle angepasst. Die Verwendung der Libraries wurde anhand von Codezeilen und in Form von Beispielen für die Tests erläutert.

Im Rahmen dieser Arbeit konnten nicht alle aufgetretenen Probleme gelöst werden. So sind die Berechtigungen für den Benutzer schwer zu verstehen, da sie sich an den vorhandenen Schnittstellen orientieren. Hier könnte überlegt werden, die vorhanden Berechtigungen ver-

ständlich abzuändern oder die Schnittstellen anzupassen.
Ein weiteres Problem, was im Rahmen dieser Arbeit aufgetaucht ist, ist, dass die Website des Learning Context Projects nicht für mobile Endgeräte optimiert ist. Da der Hauptfokus auf dem Sammeln von Daten über (mobile) Endgeräte liegt, besteht hier die Möglichkeit für Verbesserungen. Diese Tatsache kommt vor allem bei der Authentifizierung des Benutzers auf der Anmeldeseite bei Verwendung der Android-Library zum Tragen, da diese Seite in die Oberfläche des Learning Context Projects integriert wurde.

Tabellenverzeichnis

Abbildungsverzeichnis

Literatur

[1] Roman Brandt. *Towards a Lifelong Learner Modeling Framework*. Master Thesis. 2015.

[2] Elyas Esnaashari. *Users' Decisions about the Security of Mobile Applications*. Master Thesis. 2014.

[3] Danny Goodman. *JavaScript bible*. John Wiley & Sons, 2007.

[4] Eran Hammer-Lahav. „RFC 5849: The oauth 1.0 protocol". In: *Internet Engineering Task Force (IETF)* 54 (2010), S. 1–39.

[5] Dick Hardt. *RFC 6749: The OAuth 2.0 Authorization Framework*. 2012.

[6] Alberto Ornaghi und Marco Valleri. „Man in the middle attacks". In: Presented on Blackhat Conference, 2003.

[7] Hendrik Thüs u. a. „Kontexterfassung,-modellierung und–auswertung in Lernumgebungen". In: *DeLFI 2014-Die 12. e-Learning Fachtagung Informatik* (2014).

[8] `http://developer.android.com/`. [Zugriff am 17.09.1015].

[9] `http://javascript.jstruebig.de/javascript/59`. [Zugriff am 17.09.1015].

[10] `http://php.net/`. [Zugriff am 17.09.1015].

[11] `https://developers.facebook.com/docs/facebook-login/`. [Zugriff am 17.09.1015].

[12] `https://developers.google.com/+/web/api/rest/oauth`. [Zugriff am 17.09.1015].

[13] `http://www.learning-context.de/`. [Zugriff am 17.09.1015].

[14] `http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html`. [Zugriff am 17.09.1015].