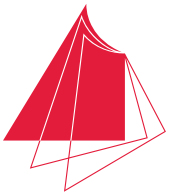


Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Analyse einer Referenzimplementierung des ActivityPub Standards mit Blick auf die Sicherheit

Bachelor Thesis von

Armin Kunkel

an der Fakultät für Informatik und Wirtschaftsinformatik
Fachrichtung Verteilte Systeme (VSYS)

Erstgutachter: Prof. Dr. rer. nat. Christian Zirpins
Betreuer: M. Sc. Robert Schäfer

01. Dezember 2018 – 31. März 2019

Hochschule Karlsruhe Technik und Wirtschaft
Fakultät für Informatik und Wirtschaftsinformatik
Moltkestr. 30
76133 Karlsruhe

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Änderungen entnommen wurde.

Karlsruhe, 20. März 2019

. . . . . .

(Armin Kunkel)

Zusammenfassung

Deutsche Zusammenfassung

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

SWWG Social Web Working Group

W3C World Wide Web Consortium

DSA Digital Signature Algorithm

OWP Open Web Platform

RSA Rivest-Shamir-Adleman

API Application Programming Interface

CLI Command Line Interface

BDD Behaviour Driven Development

TDD Test Driven Development

CPU Central Processing Unit

1. Einleitung

Zentralisierung von Daten und das Vertrauen auf einzelne Instanzen ist heutzutage allgegenwärtig. Im Internet gibt es eine Fülle an sozialen Netzwerken wie Facebook, Twitter, Google+, Instagram oder Pinterest die zumeist das Recht an den Daten, die Sie von ihren Nutzer bekommen, behalten und nicht zuletzt diese Daten auch verkaufen für z. B. Werbezwecke. Für Kriminelle sowie Geheimdienste ist es außerdem leichter an die gesamte Datenbank zu kommen, da bei zentralisierten Netzwerken meist auch eine zentrale Datenbank, bzw. ein zentraler Zugriffspunkt, vorhanden ist. Wenn der Zugriff auf diesen zentralen Punkt erreicht ist kann die ganze Datenbank ausgelesen werden.

Was bedeutet nun dezentral? In der Politik wird unter Dezentralisierung „die Übertragung zentral staatlicher Aufgaben auf subnationale oder subsidiäre Ebene(n) verstanden“[14]. In der Energiewirtschaft spricht man von einer „dezentralen Stromerzeugung“, wenn der Strom an den Stellen wo er verbraucht auch erzeugt wird. Ein Beispiel hierfür wäre ein Wasserkraftwerk, dass den Strom für die umgebenen Dörfer oder Städte liefert[13]. In der Informatik versteht man unter Dezentralisierung das verteilen von u. a. Daten über mehrere unabhängige Server hinweg.

Durch die Verteilung der Aufgaben auf sub-nationale oder subsidiäre Ebene wird die zentral staatliche Ebene entlastet und hat somit mehr Ressourcen für andere Aufgaben. Beim Errichten von Wasserkraftwerken nahe an Dörfern und Städten entfällt der Transport des Stroms über größere Distanzen und somit verringert sich der Verlust beim Transport über das Stromnetz. Dezentralisieren von sozialen Netzwerken bring verschiedene Vorteile mit sich. Ein Vorteil ist die Skalierbarkeit bei dezentralen sozialen Netzwerken. Durch das hinzufügen weiterer Instanzen können dem Netzwerk neue Ressourcen bereitgestellt werden. Ein weiterer Vorteil liegt darin, dass keine zentrale Kontrollinstanz vorhanden ist welche korrumpiert werden kann, sei es durch Kriminelle, wirtschaftliche Interessenvertreter oder staatliche Autoritäten.

Ziel der Arbeit ist es die Implementierung des verteilten Server-zu-Server Protokolls als Prototyp vorzunehmen und die bis dato von der World Wide Web Consortium (W3C) Gemeinschaft empfohlenen Sicherheitsstandards für das ActivityPub Protokoll zu prüfen. Es soll die Frage geklärt werden ob HTTP- und Linked Data Signaturen bestmöglich dafür geeignet sind, die sichere Kommunikation zu gewährleisten anhand eines Vergleichs der beiden Verfahren. Darüber hinaus soll die Arbeit einen Überblick über die Grundlagen des Protokolls geben und wie die relevanten Bestandteile zu verstehen sind. In diesem Zusammenhang ist mit „sicher“ die authentische und manipulationsfreie Übertragung der Server untereinander gemeint.

1.1. Motivation

Bei zentralen sozialen Netzwerken, z.B. Facebook, besitzt die Kontrolle über Daten das Unternehmen. Durch die Dezentralisierung bleiben Daten bei den einzelnen Instanzen des sozialen Netzwerkes. Die Datenbanken bei zentralen Netzwerken können zwar verteilt sein und auch das Netzwerk könnte dezentral angelegt sein, trotzdem gehören die Daten einem Unternehmen. Außerdem können bei zentralen sozialen Netzwerken nicht ohne weiteres Inhalte mit anderen Netzwerken ausgetauscht werden. Mit einem Protokoll wie ActivityPub wird das Verteilen sowie der Zugriff auf Inhalte und der Austausch von Aktivitäten über mehrere soziale Netzwerke hinweg ermöglicht.

ActivityPub besteht aus zwei Teilen. Dem Client-zu-Server Protokoll, auch „Social API“ genannt, und dem förderierten Server-zu-Server Protokoll. Durch die ActivityPub W3C Empfehlung Anfang 2018 und die Social Web Working Group (SWWG) Arbeitsgruppe des W3C sowie weiteren, wird die Verbreitung des Protokolls vorangebracht.

Der Netzwerkeffekt ist ein aus der Volkswirtschaftslehre stammender Begriff. Als Beispiel sei das Telefonnetz genannt. Umso mehr Leute ein Telefon besitzen, umso höher ist der Nutzen für jeden einzelnen Telefon Besitzer[11]. Übertragen auf ActivityPub bedeutet das soviel wie: „Je mehr Netzwerke den Standard implementieren, desto höher ist der Nutzen für ein einzelnes Netzwerk und im Endeffekt für die einzelnen Nutzer“.

Protokolle wie „Diaspora Federation und OStatus“ bieten den Zugriff auf dezentral gespeicherte soziale Inhalte und ermöglichen das veröffentlichen von sozialen Inhalten. Der im März letzten Jahres vom W3C empfohlene Standard namens „ActivityPub“ wurde auf Basis des Wissens im Umgang mit dem OStatus und Pump.io Protokoll entwickelt[6].

Quelle für dezentralen Zugriff

ActivityPub ist wie „OStatus“ ein Protokoll für dezentrale soziale Netzwerke und wird in dieser Bachelor Arbeit untersucht.

2. Verwandte Protokolle

Zu Beginn dieses Kapitels wird das Wort Fediverse etwas näher definiert und erläutert worum es sich dabei handelt. Im Anschluss werden mehrere Protokolle die, sowie auch ActivityPub, zum Fediverse gehören vorgestellt.

Als erstes soll der Begriff „förderiertes Netzwerk“ von Förderation hergeleitet werden. Unter einer Förderation versteht man den Verbund von etwas. Deutschland ist z. B. ein förderierter Bundesstaat, welcher 16 Bundesländern verbindet. Ein Verbund von Netzwerken wird demnach als Netzwerkverbund, oder auch „förderiertes Netzwerk“, bezeichnet.

„Fediverse“ ist ein Kofferwort aus „Federated“ und „Universe“. Historisch gesehen beinhaltete der Begriff nur Microblogging Plattformen die das OStatus Protokoll unterstützen. Mehr zu OStatus **S. 2.1**. Mittlerweile beinhaltet das Fediverse mehr als nur Microblogging Plattformen wie Mastodon[1] und GNU Social[2], sondern auch soziale Netzwerke, sowie „Video hosting“ (PeerTube[3]),„Content publishing“ Plattformen, wie Wordpress, und viele weitere[4].

Im „Fediverse“ dreht sich alles um freie (Open Source) Software anstelle von kommerziellen Produkten. Zudem kann ausgesucht werden welchem Administrator man die Kontrolle über seine Daten geben möchte, anstatt auf eine einzelne Instanz vertrauen zu müssen[4]. Ein weiterer wichtiger Aspekt ist das verbünden von Netzwerken über Protokolle wie OStatus oder Pump.io. Somit kann ein verteiltes soziales Netzwerk durch die Implementierung eines dieser Protokolle zum förderierten sozialen Netzwerk werden.

Laut dem Fediverse Netzwerkreport 2018, welcher online zur Verfügung steht, hat sich die Gesamtanzahl der erreichbaren Instanzen von 2.756 auf 4.340 erhöht. Dies entspricht einem Wachstum von 58%. Die Nutzeranzahl ist von 1.786.036 auf 2.474.601 gestiegen (39%). Weitere Details sind im Netzwerk Report nachzulesen. Es ist hinzuzufügen das der Netzwerkreport unvollständig ist und Fehler beinhalten kann[5].

2.1. OStatus

Der OStatus Standard ist eine Sammlung von verschiedenen Protokollen wie Atom, Activity Streams, WebFinger und weiteren. Ausgelegt ist der Standard auf das empfangen

[1] `https://mastodon.social/about`
[2] `https://gnu.io/social/`
[3] `https://joinpeertube.org/en/`

und versenden von Status updates für förderierte Microblogging Dienste wie Mastodon und Friendica. Das Zusammenspiel mehrerer Protokolle ermöglicht den Austausch von Status updates in fast Echtzeit.

2.2. Diaspora

Diaspora ist ein freier Web Server mit einer integrierten Implementierung eines verteilten sozialen Netzwerkes. In diesem Netzwerk wird ein Knoten als Pod bezeichnet. Die einzelnen Pods sind das, was das Diaspora Netzwerk ausmacht. Durch die integrierte Funktionalität zum sozialen Netzwerken kann auf dem Web Server aufsetzend eine eigene Applikation entwickelt werden.

2.3. Pump.io

Der ActivityPub Standard wurde mit der gesammelten Erfahrung im Umgang mit dem Pump.io Protokoll entwickelt.

Welche Erfahrungen wurden von wem gesammelt

3. Grundlagen zur sicheren Umsetzung dezentraler sozialer Netzwerke

Allgemeine Grundlagen zu Sozialen Netzwerken/Social Media und deren Sicherheitsaspekte

Welche Klassen von Anwendungen und Implementierungen gibt es allgemein bei sozialen Netzwerken? Einordnen von ActivityPub!!

Dieses Kapitel gibt einen Überblick über allgemeine Grundlagen von sozialen Netzwerken. Es wird grob erläutert für was soziale Netzwerke gedacht sind; Der Unterschied zwischen zentralen, verteilen, dezentralen und förderierten sozialen Netzwerken herausgestellt sowie auf Sicherheitsaspekte eingegangen.

Im Anschluss wird ein Kryptographie Unterkapitel eingeführt in dem die benötigten Verfahren für die sichere Umsetzung erläutert werden. Begonnen wird mit einer kurzen Einführung in Kryptographie. Darauf folgend wird kurz das RSA Verfahren sowie eine allgemeine Beschreibung von Signatur Algorithmen vorgenommen. HTTP Signaturen werden ausführlicher beschrieben, da diese bei der Implementierung des Prototypen verwendet wurden.

Darauf folgend wird der ActivityPub Standard eingeführt sowie eingeordnet, Bestandteile des Protokolls beschrieben, die Funktionsweise der Client-zu-Server sowie Server-zu-Server Kommunikation und zugehörige Standards kurz erläutert. Des weiteren wird auf die Authentifizierung und Datenintegrität bei ActivityPub eingegangen um damit folgende Fragen zu klären:

- Wie authentifiziert sich ein Benutzer gegenüber dem Server?
- Wie stellt man sicher, dass die übertragenen Daten unverändert angekommen sind?

3.1. Allgemeine Grundlagen sozialer Netzwerke

In den Sozialwissenschaften versteht man unter einem sozialen Netzwerk mehrere Personen die miteinander wechselwirkend auf soziale weise interagieren. Die Informatik bildet soziale Netzwerke als Plattformen zum Aufbau und der Pflege von Beziehungen ab. Weiter gefasst können auch Mikroblogging-Dienste, Chat Software, Voice Chat-Programme u. s. w. zu sozialen Netzwerken gezählt werden, da auch die Möglichkeit geboten wird sozial miteinander zu interagieren[15]. Im Allgemeinen besteht ein soziales Netzwerk aus Softwaretechnischer Sicht aus 3 Komponenten, dies ist aber keine Prämisse. Der Nutzer benutzt das Netzwerk durch eine meist grafische Schnittstelle. Um Zugriff auf das Netzwerk zu erlangen benötigt der Nutzer ein ***Frontend***. Dies kann in grafischer Form oder

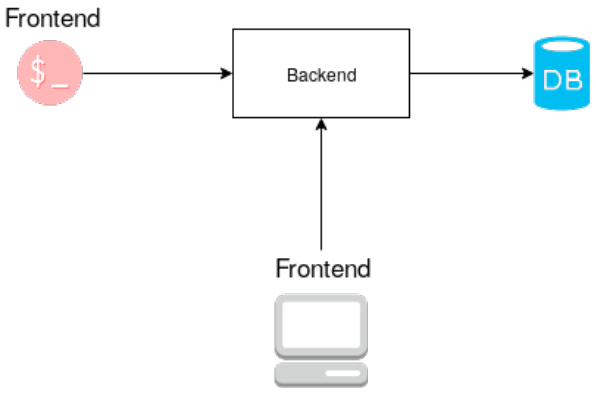


Abbildung 3.1.: 3 Tier Architektur

als Command Line Interface (CLI) zur Verfügung stehen. Über das Frontend kann der Nutzer sich nun mit seinen Anmeldeinformationen am ***Backend*** anmelden und somit meist eine Sitzung eröffnen. Steht ein Web Interface zur Verfügung, werden Sitzungen oft über Clientseitige Cookie Speicherung aufrecht erhalten. Das Backend hat die Aufgabe eingehende Nutzeranfragen entsprechend zu beantworten und, falls autorisiert, die gewünschten Aufgaben auszuführen. Bei größeren sozialen Netzwerken fällt eine große Menge generierter Daten an die verwaltet werden müssen. Dafür wird meistens eine ***Datenbank*** herangezogen die zumeist auch repliziert betrieben wird. An sich kann ein zentrales Netzwerk auch über den Einsatz von virtuellen Maschinen und Lastenverteilung verteilt oder dezentral sein. Beispielsweise könnte ein zentrales soziales Netzwerk Anfragen länderspezifisch an zuständige Instanzen des Netzwerks weiterleiten. Auch ist es Möglich die Daten länderspezifisch zu speichern und somit den Nutzern aus Deutschland andere Inhalte anzubieten als Nutzern aus der Schweiz.

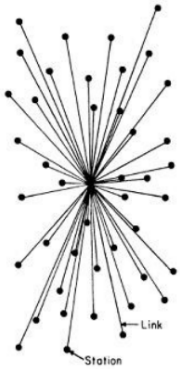
3.1.1. Unterschiedliche Klassen sozialer Netzwerke

Zentrale soziale Netzwerke haben den Nachteil eines „Single point of Failures“. Fällt dieser Knoten aus, bricht das ganze Netzwerk zusammen. Zudem sind sie kaum Skalierbar. Ein Vorteil eines zentralen Netzwerks ist die schnelle Bereitstellbarkeit. Obwohl aus Sicht der Daten die meisten sozialen Netzwerke zentral sind, kann ihre Architektur intern sowohl verteilt als auch dezentral sein.

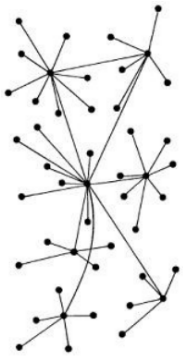
In der obigen Abbildung ist ein zentrales Netzwerk gezeigt welches aus einem zentralen Knoten (Peer) und verschiedenen Blättern (Inhalten) besteht. Fällt der zentrale Knoten aus, kann auf die von diesem Netzwerk bereitgestellten Inhalte nicht mehr zugegriffen werden.

Bei einem *dezentralen sozialen Netzwerk* verhält sich das anders. Fällt ein Peer aus, kann auf die Inhalte aller weiterer Peers weiterhin zugegriffen werden. Zudem sind die Inhalte bei dezentralen Netzwerken auf die Peers verteilt anstatt das alle auf ein einer Instanz residieren.

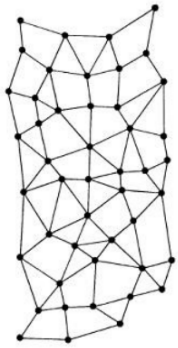
Verteilte soziale Netzwerke verfügen zusätzlich über eine Middleware Schicht über die einzelne Peers kommunizieren können. Dadurch ist unter anderen eine leichte Skalierbarkeit gegeben, da eine weitere Instanz lediglich gestartet werden muss.



(a) Zentrales soziale Netzwerk



(b) Dezentrales soziale Netzwerk



(c) Verteilte soziale Netzwerk

Quelle: (Zentrale, dezentrale, verteilte Systeme)

Abbildung 3.2.: Klassen sozialer Netzwerke

Um die Inhalte verschiedener Netzwerke zu verbinden können mehrere soziale Netzwerke zu einem großen verbunden oder „förderiert“ werden. Dies kann über Protokolle und Standards wie OStatus und ActivityPub geschehen oder durch Netzwerkbrücken, im Sinne von Transformatoren, realisiert werden.

3.1.2. Sicherheitsaspekte sozialer Netzwerke

Zu den Sicherheitsaspekten, die ein soziales Netzwerk umsetzen sollte, gehören:

- Die Authentisierung der Nutzer gegenüber dem Server
- Das Sicherstellen der Datenintegrität
- Schutz der Datenbank vor unbefugtem Zugriff
- Filterung (Nutzer bekommt nur das zu sehen, was er sehen darf)

Um die einzelnen Nutzer gegenüber dem Server zu authentisieren, können Technologien wie OAuth[1] oder JSON Web Token[2] verwendet werden. Dadurch ist sichergestellt, das

[1] siehe `https://tools.ietf.org/html/rfc6749`
[2] siehe `https://tools.ietf.org/html/rfc7519`

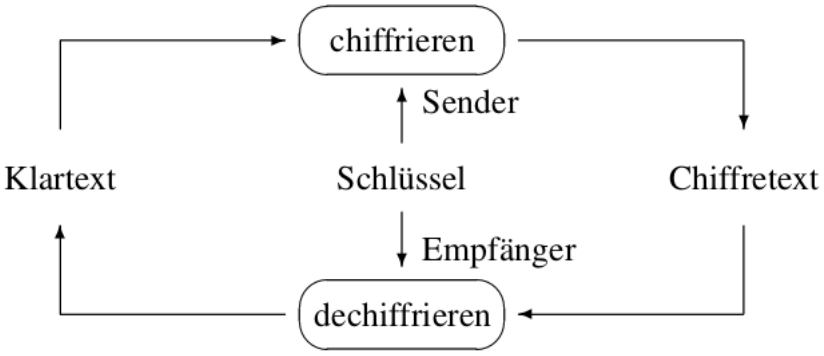
nur derjenige mit korrekten Anmeldedaten das Profil benutzen kann. Außerdem weiß der Server dadurch wie er die Inhalte Filtern muss um dem Nutzer das für ihn erlaubte anzeigen zu können.

Das Sicherstellen der manipulationsfreien Übertragung von Inhalten ist sowohl bei der Client zu Server, als auch bei der Server zu Server Kommunikation zu Empfehlen. Bei beiden Kommunikationsszenarien werden die Daten über das Internet übertragen und sind somit potentiell manipulierbar. Sichergestellt werden kann dies z. B. über HTTP-Signaturen s. 3.2.3.

Ein weiterer wichtiger Sicherheitsaspekt ist das Schützen der Datenbank vor unbefugtem Zugriff. Bei einem sozialen Netzwerk kann man die Datenbank durchaus als das Herzstück des Netzwerkes bezeichnen, da in dieser, gerade bei „sozialen Netzwerken“ eine riesige Menge an Daten liegen. Schaffen es Angreifer auf diese Zugriff zu bekommen und unbemerkt zu bleiben, kann die Datenbank in Ruhe analysiert oder womöglich die Inhalte auf einen eigenen Server transferiert werden. Durch das isolieren der Datenbank, restriktiven oder keinen direkten Zugriff durch Nutzer oder eine Mehrfaktor Authentisierung kann die Sicherheit der Datenbank erhöht werden.

3.2. Kryptographie

„Unter dem Begriff Kryptographie ist die Wissenschaft vom geheimen Schreiben zu verstehen“[3]. Man spricht von symmetrischer Verschlüsselung wenn eine Nachricht im Klartext vom Sender mit einem geheimen Schlüssel, welcher beiden Parteien bekannt ist, verschlüsselt und vom Empfänger, mit demselben Schlüssel, entschlüsselt wird[4].



Quelle: (Wätjen, 2018, S.1)

Abbildung 3.3.: Symmetrische Ver- und Entschlüsselung

Von asymmetrischer Verschlüsselung ist die Rede, wenn die Teilnehmer anstatt einen gemeinsamen Schlüssel zu haben, der im vor hinein ausgetauscht werden muss, jeder ein Schlüsselpaar, bestehend aus öffentlichem und privatem Schlüssel, besitzt.

[3]Dietmar Wätjen, 2018, S.1
[4]Vgl. Dietmar Wätjen, 2018, S.1

3.2.1. RSA

Das Rivest-Shamir-Adleman (RSA) Verfahren ist ein solches asymmetrisches Verschlüsselungsverfahren welches von den drei namens gebenden Mathematikern 1977 entwickelt wurde. Das RSA Verfahren ist wohl das am häufigsten verwendete Public-Key-Kryptosystem.

Beispiel 1. Die Gesprächspartner Alice und Bob, welche beide ein Schlüsselpaar besitzen, wollen miteinander kommunizieren. Alice verschlüsselt einen Nachrichtentext mit dem öffentlichen Schlüssel von Bob und sendet die verschlüsselte Nachricht an Bob. Dieser kann seinerseits mit seinem privaten Schlüssel die Nachricht entschlüsseln[5].

3.2.2. Signaturen

Für die Sicherstellung der Authentizität können sogenannte Signaturen verwendet werden. Das oben kurz erläuterte RSA Verfahren kann nicht nur zum Ver- und Entschlüsseln von Nachrichten benutzt werden, sondern auch zum signieren.

Dabei wird statt des öffentlichen Schlüssels, der private Schlüssel, zusammen mit einer Hashfunktion, benutzt um eine Signatur zu erzeugen. Diese kann dann mit dem öffentlichen Schlüssel des zugehörigen privaten Schlüssels verifiziert werden.

Beispiel 2. Bob möchte eine Nachricht an Alice schicken und sichergehen, dass diese auf dem Weg nicht verändert wurde. Er verwendet seinen privaten Schlüssel und wendet diesen auf eine Nachricht an um eine Signatur zu erzeugen. Beides übermittelt er an Alice. Mit dem öffentlichen Schlüssel von Bob kann die fehlerfreie Übertragung der Nachricht verifiziert werden.

Eine Hashfunktion ist eine Einwegfunktion welche auch zur Signierung verwendet werden kann. Bei solch einer Funktion wird ein Eingangswert auf eine kryptische Zeichenfolge abgebildet. Dies wird sehr oft bei Passwörtern verwendet um diese nicht Umkerbar aufzubewahren.

3.2.3. HTTP Signaturen

Eine *HTTP Signatur* wird verwendet um die Authentizität sicherzustellen. Diese wird als Wert einer „Signature“ Kopfzeile eingetragen und besteht aus mehreren Teilen:

- ***keyId***=”https://example.org/activitypub/users/lea#main-key”
- ***algorithm***=”rsa-md4”
- ***headers***=”(request-target) date host content-type”
- ***signature***=”DHeEH0Okmtf1ec/lbM1/F5FiLVfQfbWuoFf9t/TzNZiZ7ak”

Über die *keyId*, was eine Referenz auf einen Schlüssel darstellt, kann der öffentliche Schlüssel angefragt werden. Dies wird beim verifizieren einer Signatur benötigt um die übertragenen Daten auf ihre Authentizität hin zu prüfen.

[5]Vgl. Dietmar Wätjen, 2018, S.73 f.

Welcher Hashing-Algorithmus bei der Erstellung verwendet wurde, kann über das *algorithm* Feld der Signatur nachgeschlagen werden. Zudem muss der Algorithmus bei Erstellung in dieses Feld zum Nachschlagen eingetragen werden.

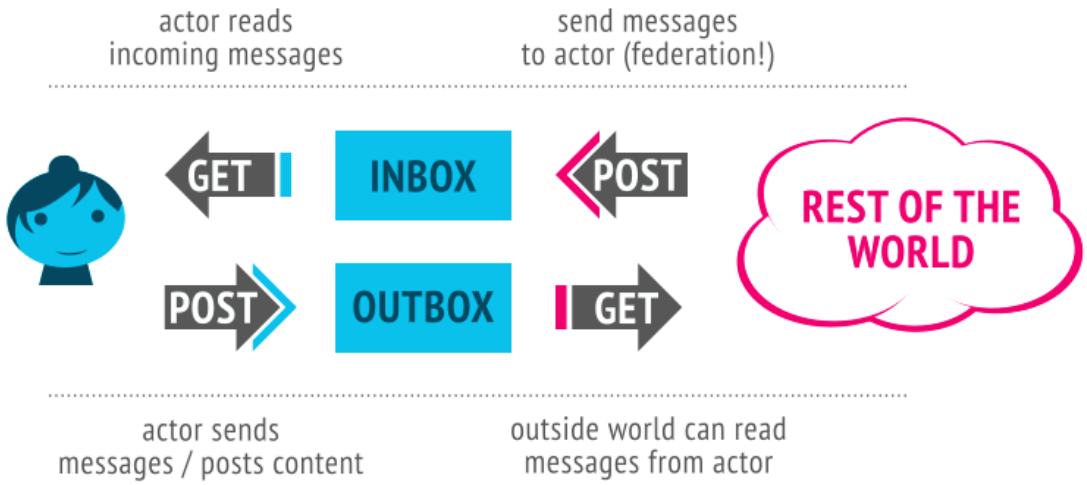
Um die eigentliche Signatur zu erzeugen werden die in *headers* angegebenen Kopfzeilen der HTTP Anfrage verwendet. Somit kann eingeschränkt werden welche Metadaten in die Signierung einfließen.

Die bei der Signierung mit gegebenem Hashing-Algorithmus und Kopfzeilen erzeugte Signatur wird in das *signature* Feld der HTTP Signatur eingetragen sowie die HTTP Signatur an sich als „Signature“ Kopfzeile der HTTP Anfrage gesetzt.[3]

3.3. ActivityPub Standard

Der ActivityPub Standard wurde am 23 Januar 2018 von der W3C empfohlen[6] und von einer Arbeitsgruppe des W3C, der SWWG[1, 2], entwickelt. Diese Gruppe war vom 21. Juli 2014 bis zum 13 Februar 2018 aktiv[1] und entwickelte unter anderem ActivityPub, ActivityStreams Core[10] und ActivityStreams Vocab[9]. Die SWWG war eine Arbeitsgruppe des W3C mit dem Ziel neue Protokolle, Vokabulare und Application Programming Interface (API)’s zu definieren für den Zugriff auf soziale Inhalte der sogenannten Open Web Platform (OWP)[12].

ActivityPub definiert zwei Protokollschichten, sowie Konzepte, Sammlungen und Interaktionen für dezentrale soziale Netzwerke. Eine Protokollschicht ist das Client-zu-Server Protokoll (Social API), um Clients den Zugriff auf die neusten an sie gesendeten Inhalte zu ermöglichen sowie zum entgegennehmen von Anfragen die vom Client abgesetzt wurden[6].



Quelle: ActivityPub 2018 - Overview

Abbildung 3.4.: Schnittstellen des ActivityPub Protokolls

Die zweite Protokollschicht besteht aus dem förderierten Server-zu-Server Protokoll (Federation Protocol), welches den einzelnen Instanzen von dezentralen sozialen Netz-

werken den Austausch von Inhalten untereinander gestattet. ActivityPub setzt auf bereits bestehende Empfehlungen des W3C auf, welche teilweise auch von der SWWG entwickelt wurden wie z. B. ActivityStreams Core und ActivityStreams Vocab[6]. Die zwei Protokollschichten können unabhängig voneinander implementiert werden.

Auch andere Technologien wie JSON Linked Data werden verwendet um die Erweiterbarkeit zu gewährleisten. Über neue Ontologien und Vokabulare können weitere syntaktische Definitionen und semantische Beschreibungen zu den bestehenden hinzugefügt werden[6]. Diese Vokabulare können im Kontext des JSON Linked Data Objektes, angegeben werden. Bei ActivityPub wird das ActivityStreams 2.0 Vokabular verwendet welches durch ActivityStreams Vocab erweitert wird.

3.3.1. Bestandteile des Protokolls

Die Hauptbestandteile des ActivityPub Standards sind die folgenden:

- Aktoren
- Objekte
- Sammlungen
- Aktivitäten

In ActivityPub werden Benutzer als „Aktoren“(actors) dargestellt. Diese können nicht nur Personen, sondern auch Applikationen, Organisationen, Gruppen und Services sein[10]. Jedes Aktoren Objekt muss eine „Inbox“ und „Outbox“, welche geordnete Sammlungen sein müssen, sowie eine ID und ein Typ besitzen[6]. Die ID muss global einzigartig sein. Dies kann garantiert werden durch eine Domänen und Protokoll bezogene URI oder IRI wie z. B. „https://example.org/users/alice“oder „https://example.org/users/álìcê“. Der Typ eines Aktor (z. B. ”type”: ”Person”) kann variieren zwischen den fünf oben genannten.

3.3.2. Zugehörige Standards und Komponenten

ActivityPub benutzt die ActivityStreams Daten Syntax und das Vokabular. Zusätzlich kann ein weiteres Sicherheitsvokabular[6] benutzt werden um Definitionen zum Bereitstellen eines öffentlichen Schlüssels, Signaturen sowie Verschlüsselten Inhalten u.v.m. zu haben. Am 22 April 2016 hat die „W3C Community Group“ einen Entwurfsbericht herausgebracht. Durch diesen wird neue Syntax und Semantik definiert um Internet basierten Applikationen das Verschlüsseln, Entschlüsseln sowie digitale Signieren und Verifizieren von verlinkten Daten (Linked Data) zu ermöglichen. Es enthält auch Vokabeln für die Erstellung und Verwaltung einer dezentralen Public-Key-Infrastruktur über das Internet[8]. Ein Anwendungsfall ist das holen des öffentlichen Schlüssels eines Nutzers, über dessen Aktoren Objekt, um eine von Nutzer gesendete Nachricht zu verifizieren.

[6]Eine Ontologie die Sicherheitsaspekte definiert wie öffentliche Schlüssel, Signaturen u.v.m.

```
{
  "@context": [
    "https://www.w3.org/ns/activitystreams",
    "https://w3id.org/security/v1"
  ],
  "id":"https://human-connection.org/activitypub/users/macu",
  "type":"Person",
  "following":"https://human-connection.org/activitypub/users/macu/following",
  "followers":"https://human-connection.org/activitypub/users/macu/followers",
  "inbox":"https://human-connection.org/activitypub/users/macu/inbox",
  "outbox":"https://human-connection.org/activitypub/users/macu/outbox",
  "preferredUsername":"macu",
  "name":"Marco Curosa",
  "summary":"This is my Profile!",
  "url":"https://human-connection.org/activitypub/@macu",
  "publicKey":{
    "id":"https://human-connection.org/activitypub/users/macu#main-key",
    "owner":"https://human-connection.org/activitypub/users/macu",
    "publicKeyPem":"-----BEGIN PUBLIC KEY-----\
        nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAvGOWsb/tTRCd5BcZOZYa\
        nqQ6os0vMJmOeg9kNvB5KXZhZwiwjTbXf7VEshUoDnoYUAF1jFnhfP5ChODVq9r9X\nS+
        Wx3o65vjdhjXTrZrVHicTFoB4RaC9sCEHfTinR8ywewzo7VrEVQ2KnbeXzzf+v\na+
        Dr2h1qkwNv9E9QJ5Mq6RfCKdoFzOFhPqdFOKmgCPuaUOnzgirKoLkwJz0Qe8ky\
        neFDNT4uQKoFPmjXt3tl5ZMlR/0FEuk9Vn0l+rrLXqNUrsq7AINoY3rZf97d/CMDy\
        nChNbN/6N9B1xFN+AvQSAoLWjbD0Zrkm3vy6/dnyDcgXQmrpR0selWQvV/PgbeSgP\
        nhQIDAQAB\n-----END PUBLIC KEY-----\n"
  },
  "tag":[],
  "attachment":[],
  "endpoints":{
   "sharedInbox":"https://human-connection.org/activitypub/inbox"
  }
}
```

Abbildung 3.5.: Beispiel Aktoren Objekt

„ActivityStreams 2.0“ beinhaltet Modelle für Aktoren, Aktivitäten, Intransitiven Aktivitäten, Objekte, Links, Sammlungen, Natürliche Sprachwerte (Strings) und für Internationalisierung. Das Kernvokabular von ActivityStreams 2.0 wird durch ActivityStreams Vocab erweitert. Dazu gehören verschiedene Aktivitätstypen wie z.B. „Accept“,„Add“,„Remove“,„Delete“ und „Create“[7], um Aktorentypen wie „Person“, „Application“ und „Group“[8] sowie um verschiedenste Objekttypen wie „Article“, „Event“, „Note“ und „Relationship“[9].

JSON Linked Data ist eine Erweiterung des JSON Formates um verlinkte Daten zu Repräsentieren. JSON an sich, ist ein Format welches im Web häufig Anwendung findet um Daten auszutauschen. Im Kern sind ActivityStreams 2.0 auch JSON Linked Data Objekte. Der ActivityStreams 2.0 Kontext definiert verschiedene Klassen und Eigenschaften, von denen nicht alle benutzt werden. Typische Klassen sind „Activity“, „Link“ und „OrderedCollection“. Ein Beispiel Notiz, sowie Artikel ActivityStreams 2.0 Objekt sieht wie folgt aus:

[7] activity-types https://www.w3.org/TR/activitystreams-vocabulary/
[8] actor-types https://www.w3.org/TR/activitystreams-vocabulary/
[9] object-types https://www.w3.org/TR/activitystreams-vocabulary/

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "type": "Note",
  "id": "http://example.org/note/124",
  "summary": "A note by Sally",
  "content": "Everything is OK here."
}
```

Abbildung 3.6.: Beispiel Notiz Objekt

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "type": "Article",
  "id": "http://example.org/note/124",
  "summary": "A article by Sally",
  "content": "Lorem ipsum dolor sit amet. Lorem ipsum dolor ist amet.",
  "published": "2019-03-14T21:25:40.081Z"
}
```

Abbildung 3.7.: Beispiel Artikel Objekt

3.3.3. Authentisierung und Datenintegrität

Für die Authentisierung und zum sichern der Datenintegrität definiert der Standard keine Mechanismen. Es gibt allerdings „Best Practices“ für die Umsetzung dieser Anforderungen.

Zum einen werden bei der Client-zu-Server Authentisierung „OAuth 2.0 Bearer Tokens“ benutzt, zum anderen auf der Server Seite „HTTP“ oder „Linked Data Signatures“zur Sicherstellung der Datenintegrität.

Bei„OAuth 2.0 Bearer Tokens“ handelt es sich um eine Methode um auf geschützte Ressourcen zugreifen zu können[7]. ActivityPub nutzt diese für jegliche Interaktionen mit dem Server.

„Die Datenintegrität umfasst Maßnahmen damit geschützte Daten während der Verarbeitung oder Übertragung nicht durch unautorisierte Personen entfernt oder verändert werden können. Sie stellt die Konsistenz, die Richtigkeit und Vertrauenswürdigkeit der Daten während deren gesamten Lebensdauer sicher und sorgt dafür, dass die relevanten Daten eines Datenstroms rekonstruierbar sind“[**data-integrity**].

Um sicherzustellen das HTTP Anfragen beim Transport nicht verändert wurden, können HTTP Signaturen verwendet werden. Diesen verwenden einen kryptografischen Algorithmus um aus ausgewählten Kopfzeilen einer HTTP Anfrage einen kryptischen Zeichenfolge zu generieren. Auf der Empfängerseite kann die Zeichenfolge mit mitgelieferten und auch nachschlagbaren Information verifiziert werden.

Wenn ein Objekt nicht nur vom Client zum Server gesendet, sondern auch zwischen Servern untereinander weitergeleitet werden soll wird zum Sicherstellen der Datenintegrität ein anderes Verfahren benötigt als HTTP Signaturen. Die „Best Practices“ empfehlen für solche Fälle „Linked Data Signatures“. Der größte Unterschied zwischen HTTP Signaturen und „Linked Data Signatures“ besteht darin, welche Daten zum Erstellen der Signatur verwendet werden. Bei HTTP Signaturen sind es die Kopfzeilen. Mit „Linked Da-

ta Signatures“ kann auch das Objekt selbst, also der Payload einer HTTP Anfrage, anstatt nur die Kopfzeilen, zum signieren verwendet werden.

4. Entwurf einer Lösung für sichere Server-zu-Server Interaktion mit ActivityPub

Zu Beginn dieses Kapitels ist die Frage zu stellen, wie die Architektur sicher gestaltet wird.

Eine Trennung der ActivityPub Komponenten in einen eigenen Service würde für eine isolierbare Ausführung sorgen. Es

4.1. Entwurfsentscheidung

Für die nachfolgende Architektur wurde sich in dieser Arbeit entschieden, da sie einen Modularen Charakter hat. Dies ist gegeben durch die Kapselung des Services in eine Express Middleware. Sowohl das alleinstehende betreiben des Services ist möglich als auch die Integration in einen bestehen Express Server.

4.2. Technische Architektur

Der technischen Architektur liegt das Express Middleware Framework zugrunde. Der ActivityPub Service wird als Express Middleware, sowie als allein stehender Express Web Server implementiert. Bei der letzteren Variante wird anstatt die Middleware in ein bestehenden Express Server zu integrieren, ein eigener Express Server gestartet.
In der oben gezeigten Abbildung wurde der ActivityPub Service in einen bestehenden Express Server einer GraphQL API integriert.

Benutzer der API kommunizieren mit dem Web Server über das HTTP Protokoll. Der Web Server leitet die Anfragen dann an den entsprechenden Router weiter, welcher wiederum den Anfrage Inhalt transformiert und „Handler“ Funktionen des ActivityPub Service mit entsprechenden Parametern aufruft.

Die obige Abbildung zeigt eine detailliertere Variante des ActivityPub Service Diagramms aus Abb. 4.1.
Es wird außerdem ersichtlich, dass, um eine maßgeschneiderte Version für ein weiteres Projekt zu erstellen, das Interface IDataSource implementiert werden muss.
Sonst wird bei der Architektur weitestgehend darauf Wert gelegt, dass die meisten ActivityPub konformen Inhalte „on the fly“generiert werden können. Damit wird versucht

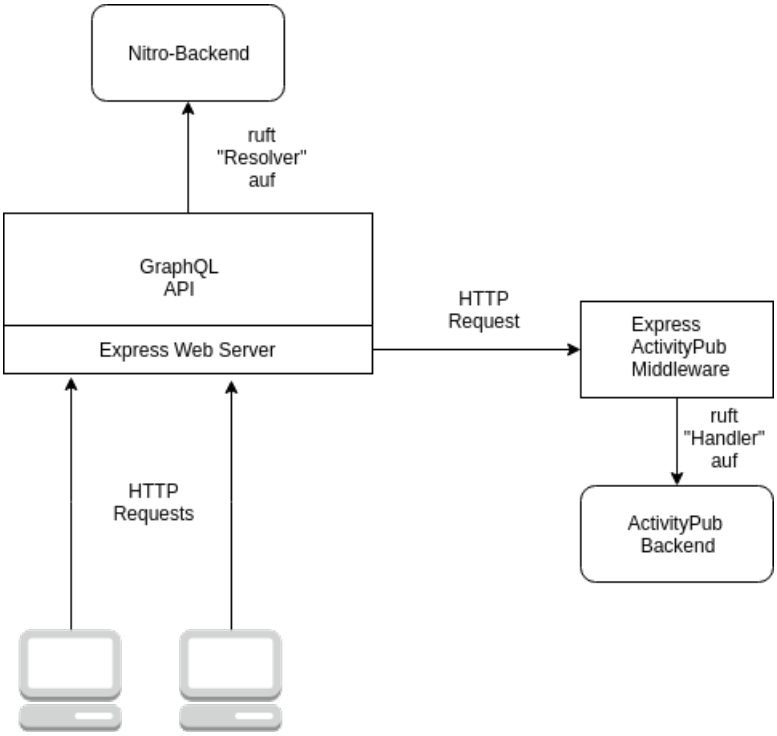


Abbildung 4.1.: Technische Architektur ActivityPub

für weitere Projekte die auf diesem Prototypen aufbauen wollen einen einfachen Einstieg zu bieten um andere Datenbanken o.ä. anbinden zu können.

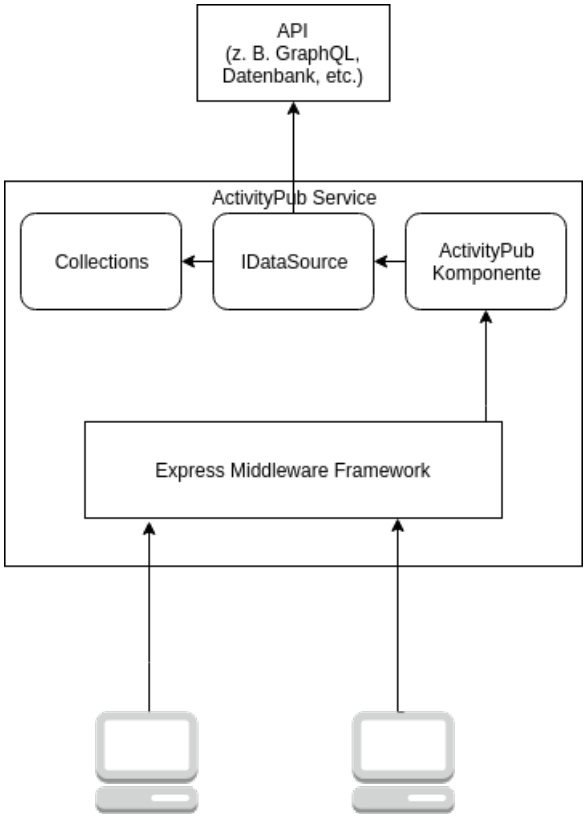


Abbildung 4.2.: Technische Architektur als allein stehender Server

5. Implementierung eines ActivityPub Prototyps

Die IDataSource Implementierung wird für das Unternehmen angefertigt in der die Bachelor Thesis bearbeitet wird. Im Falle man möchte den Service mit einer anderen Datenquelle versorgen, kann das IDataSource Interface implementiert und so angepasst werden, dass eine andere Datenbank Verwendung findet.

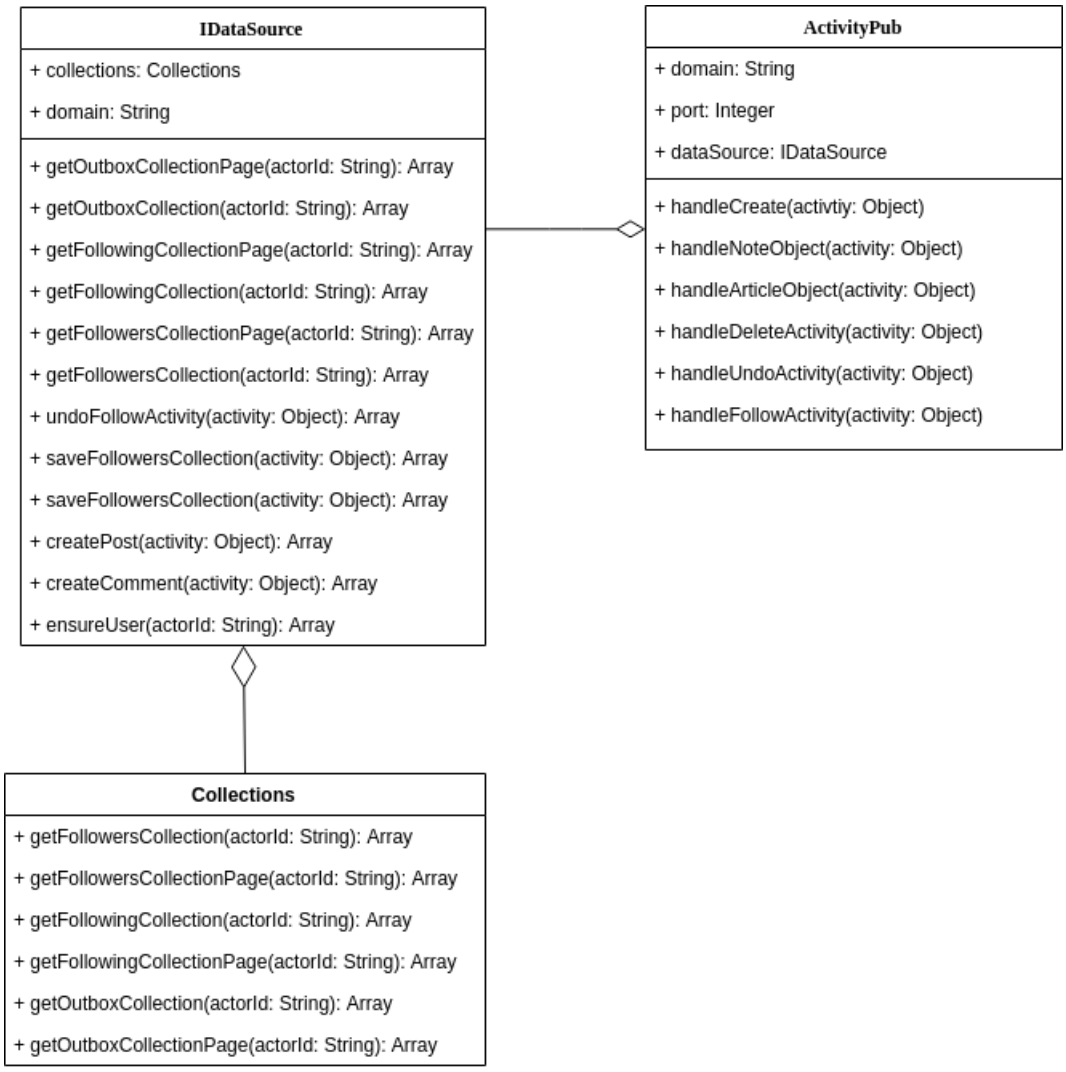


Abbildung 5.1.: Hauptkomponenten des förderierten Servers

Obig abgebildetes Klassendiagramm zeigt die Hauptkomponenten des ActivityPub Services und die enthaltenen Methoden.
In der Collections Klasse sind Weiterleitungen zu entsprechenden Methoden der Datenquelle; Sie dient ausschließlich der Lesbarkeit für den Entwickler.
Bei der ActivityPub Klasse handelt es sich den Eingangspunkt der Express Router. Die entsprechenden Anfrage-Handler wandeln die Anfrage in einen Service Aufruf um. Diese Klasse enthält nicht nur Handler für eingehende, sondern auch Methoden zum senden von Anfragen.

Um eine möglichst Modulare Implementierung zu gewährleisten wurden alle Datenbankspezifischen Methoden in das IDataSource Interface ausgelagert. Durch die Implementierung dieses Interfaces können Aktivitäten ActivityPub konform empfangen werden. Das senden wiederum muss jedes Backend selbst implementieren. Dafür stehen in der ActivityPub Klasse Methoden zum senden von Aktivitäten bereit. Durch einfaches importieren der ActivityPub Klasse erhält man eine Instanz. Darüber sind die Methoden verfügbar.

Es ist auch möglich andere Datenquellen als die in der Bachelor Arbeit verwendete GraphQL Schnittstelle zu nutzen, wie z. B. MySQL oder MongoDB.

Getestet wird die Implementierung mit dem Cucumber Test-Framework, welches ein Werkzeug zum ausüben von Behaviour Driven Development (BDD) ist. Die Tests selbst werden im Rahmen von Cucumber als sogenannte „Feature"'s bezeichnet und haben die Dateiendung .feature. Geschrieben sind die Tests in der Gherkin Sprache welche eine Sammlung von Syntax ist um diese auf entsprechende Aktionen abzubilden. Es wurden bei der Gherkin Syntax extra leicht verständliche Schlüsselworte gewählt um die Tests auch für Benutzer, Produktbesitzer und weitere lesbar zu halten. Somit muss nur die Gherkin Sprache bekannt sein um die Features schreiben zu können. Das Verständnis beim lesen ergibt sich aus der Verwendung natürlicher Sprachkonstrukte als Schlüsselwörter.

Unter Test Driven Development (TDD) versteht man eine Vorgehensweise beim Entwickeln von Software. Dabei werden zuerst fehlschlagende Tests geschrieben, um diese im zweiten Schritt durch eine Implementierung der Komponenten zur erfolgreichen Ausführung zu bringen. Im dritten Schritt werden die Tests angepasst und der Zyklus wiederholt sich.

Bei der Softwareentwicklung mit dem BDD Ansatz wird im Grunde genommen Wert auf eine hohe Kommunikation zwischen allen beteiligten im Entwicklungsprozess gelegt. Werkzeuge wie Cucumber fördern dabei die Lesbarkeit von Softwaretests und somit auch die Kommunikation zwischen Programmierern und weiteren. Zudem ist es für Neueinsteiger leichter einen Überblick über die Funktionalität des Systems zu bekommen.

5.1. Server-zu-Server Protokoll

Es wird eine Teilmenge der ActivityPub Funktionalität implementiert, die angepasst ist auf die Anforderungen der Firma in welcher diese Arbeit angefertigt wird. Dabei handelt es sich um folgende Aktivitäten des Standards: *Create*, *Update*, *Delete*, *Follow*, *Undo*, *Accept*, *Reject*, *Like*, *Dislike*.

Die ersten drei Aktivitäten werden zum Manipulieren von Objekten wie z. B. einem Artikel benötigt. *Follow* und *Undo* sind zum folgen eines Nutzers so wie zum rückgängig machen. Wenn einem Nutzer gefolgt wurde, wird eine *Accept* Aktivität an die *Inbox* des Folgenden gesendet mit der *Follow* Aktivität als Objekt.

Seitens ActivityStreams werden folgende Objekte die Objekte *Article* und *Note* verwendet.

Der Funktionsumfang des förderierten Servers beschränkt sich auf den folgenden:

- Empfangen von ***Article*** und ***Note*** Objekten am Geteilten- und Nutzernachrichteneingang (sharedInbox, Inbox)
- Empfangen von ***Like*** und ***Follow*** Aktivitäten am Geteilten- und Nutzernachrichteneingang
- Empfangen von ***Undo*** und ***Delete*** Aktivitäten für ***Articles*** und ***Notes*** am Geteilten- und Nutzernachrichteneingang
- Bereitstellen eines ***Webfinger*** und ***Aktoren*** Objektes
- Bereitstellen der ***Followers***, ***Following*** und ***Outbox*** Sammlungen

Artikel und Notizen werden bei der in dieser Arbeit angefertigten Implementierung gleich behandelt. Beim empfangen einer Create Aktivität die eine Notiz oder Artikel als Objekt Attribut hat, wird der Artikel über die IDataSource Implementierung erstellt und entsprechende Metadaten zum rekonstruieren der ID's gespeichert. Somit können empfangene Posts, welche an die Öffentlichkeit gerichtet sind, im Netzwerk angezeigt werden.

Wie wird nun sichergestellt, dass ein Nutzer dazu berechtigt ist
Eingehende HTTP POST Anfragen werden auf eine vorhandene Signatur Kopfzeile geprüft. Wird diese nicht gefunden, wird die Anfrage verworfen. Ist sie vorhanden, wird die Signatur geprüft indem das Aktoren Objekt über die zugehörige *keyId* angefragt wird und die Signatur gegen den öffentlichen Schlüssel geprüft wird. Dies stellt sicher, das die Inhalte der Nachricht beim Transport nicht verändert wurde. Eine Authentifizierung des Clients wird nicht benötigt, da es sich um eine öffentliche Schnittstelle handelt.

Acceptance Testing mit Cucumber und der Gherkin Language

6. Evaluation

6.1. Anwendungsbeispiel

6.2. Performanzmessungen

Es wurde die Performanz der Erstellung von Signaturen, sowie der Verifikation solcher gemessen. Insgesamt wurden 2 Messungen mit je 10 Testdurchläufen pro Hashfunktion durchgeführt. Die erste Messung wurde auf einem Notebook mit eingebauter Intel 2-Kern Central Processing Unit (CPU) der Serie Core i5 und einer Taktfrequenz von 2,4 GH durchgeführt[1]. Bei der zweiten Messung kam eine 16-Kern AMD CPU aus der „Ryzen“ Serie mit der Bezeichnung „Threadripper 2950X“ und einer Taktfrequenz von 3,5 GH zum Einsatz.

Die Performanzmessungen wurden in der Node.js Laufzeitumgebung mit der Version 10.15.3 durchgeführt unter Zuhilfenahme der internen process.hrtime() Funktion. Diese wird vor und nach dem Erstellen sowie Verifizieren aufgerufen. Die Rückgabe des zweiten Aufrufs nach der jeweiligen Funktion enthält die Differenz zum ersten Aufruf in Nanosekunden.

	RSA-MD4	RSA-MD5	RSA-SHA256	RSA-SHA512
Testlauf 1	24.254 ms	24.123 ms	24.171 ms	24.173 ms
Testlauf 2	22.833 ms	23.006 ms	23.626 ms	22.948 ms
Testlauf 3	27.186 ms	23.113 ms	22.873 ms	22.781 ms
Testlauf 4	23.630 ms	24.256 ms	22.597 ms	22.667 ms
Testlauf 5	25.012 ms	24.027 ms	24.343 ms	30.278 ms
Testlauf 6	30.221 ms	23.032 ms	22.827 ms	22.768 ms
Testlauf 7	27.050 ms	29.950 ms	22.629 ms	22.774 ms
Testlauf 8	23.630 ms	24.282 ms	24.625 ms	25.495 ms
Testlauf 9	23.151 ms	22.958 ms	22.854 ms	26.816 ms
Testlauf 10	22.654 ms	24.067 ms	23.637 ms	23.579 ms
Arithmetisches Mittel	25,122 ms	24,281 ms	23,418 ms	24,427 ms

Tabelle 6.1.: Testergebnisse der ersten Messung (Signatur Erstellung)

Beim betrachten der arithmetischen Mittel fällt auf, dass die Messwerte sehr nahe beieinander liegen. Die SHA256 Hashfunktion schneidet dabei mit einer Funktionslaufzeit von

[1] `https://ark.intel.com/content/www/de/de/ark/products/47341/intel-core-i5-520m-processor-3m-cache-2-40-ghz.html`

23,418 ms am Besten ab. Auf Platz zwei liegt die MD5 Funktion mit 24,281 ms gefolgt von SHA512 mit 24,427 ms. Das Schlusslicht bildet MD4 mit einer Laufzeit von 25,122 ms. Aus Sicht der Performanz wird bei der ersten Messung somit die SHA256 Hashfunktion zur Signierung empfohlen.

	RSA-MD4	RSA-MD5	RSA-SHA256	RSA-SHA512
Testlauf 1	107,177 ms	128.993 ms	121.406 ms	114.669 ms
Testlauf 2	43,660 ms	51.178 ms	35.524 ms	39.633 ms
Testlauf 3	32.101 ms	32.691 ms	30.461 ms	32.293 ms
Testlauf 4	36.096 ms	30.279 ms	32.018 ms	43.842 ms
Testlauf 5	31.869 ms	34.882 ms	30.877 ms	26.555 ms
Testlauf 6	39.218 ms	67.507 ms	38.734 ms	31.330 ms
Testlauf 7	40.996 ms	39.646 ms	45.433 ms	27.290 ms
Testlauf 8	25.977 ms	39.302 ms	26.778 ms	53.478 ms
Testlauf 9	23.641 ms	37.833 ms	30.485 ms	36.284 ms
Testlauf 10	31.097 ms	52.725 ms	28.943 ms	28.634 ms
Arithmetisches Mittel	41,183 ms	51,503 ms	42,065 ms	43,400 ms

Tabelle 6.2.: Testergebnisse der ersten Messung (Signatur Verifikation)

Ausreißer beschreiben

Werden die Arithmetischen Mittel der Verifikation betrachtet ist eine Zeitspanne von 10 ms Sekunden zu Beobachten in welcher sich alle Messwerte befinden. Dabei ist der MD4 Algorithmus mit 41,183 ms auf Position eins, während sich MD5 mit 51,503 ms auf der letzten Position befindet. Die beiden Funktionen SHA256 und SHA512 liegen mit Laufzeiten von 42,065 und 43,400 ms näher an der MD4 Funktion. Zudem fällt auf, dass bei der Verifikation die ersten Messwerte stark abweichen von allen weiteren. Das ist dadurch zu erklären, dass zuerst eine HTTP Anfrage gesendet werden muss um den benötigten öffentlichen Schlüssel zu erhalten. Jegliche weitere Anfragen werden aus dem Cache bedient und sind somit um einen groben Faktor von 2 schneller als die erste Anfrage.

Anders als bei der ersten Messung liegt bei Betrachtung der arithmetischen Mittel keine Zeitspanne vor, sondern die Messwerte liegen sehr nahe beieinander. Dabei ist die SHA512 Hashfunktion am schnellsten mit 6,911 ms, dicht gefolgt von der SHA256 Funktion mit einer Laufzeit von 6,917 ms. Auf Platz drei ist die MD4 Funktion mit 7,088 ms und auf Platz vier MD5 mit 7,128 ms.

Wie auch in der ersten Messung sehen wir bei der Verifikation, dass die beginnenden Messwerte einen deutlichen Unterschied zu weiteren aufweisen. Der Grund ist derselbe wie bei der ersten Messung. Betrachten wir die arithmetischen Mittel, sehen wir Messwerte innerhalb einer Zeitspanne von ungefähr 3 ms in der sich die Messwerte bewegen. Die MD5 Hashfunktion schneidet hierbei am Besten ab mit einer Zeit von 17,900 ms, ge-

	RSA-MD4	RSA-MD5	RSA-SHA256	RSA-SHA512
Testlauf 1	7.152 ms	7.137 ms	8.178 ms	7.002 ms
Testlauf 2	6.955 ms	7.134 ms	6.714 ms	7.048 ms
Testlauf 3	6.910 ms	6.762 ms	6.703 ms	6.860 ms
Testlauf 4	7.044 ms	6.800 ms	6.827 ms	6.982 ms
Testlauf 5	6.935 ms	7.831 ms	6.854 ms	6.958 ms
Testlauf 6	7.191 ms	7.697 ms	6.873 ms	7.050 ms
Testlauf 7	7.195 ms	7.108 ms	6.728 ms	6.840 ms
Testlauf 8	7.237 ms	6.864 ms	6.743 ms	6.722 ms
Testlauf 9	7.136 ms	7.016 ms	6.740 ms	6.834 ms
Testlauf 10	7.128 ms	6.939 ms	6.811 ms	6.819 ms
Arithmetisches Mittel	7,088 ms	7,128 ms	6,917 ms	6,911 ms

Tabelle 6.3.: Testergebnisse der zweiten Messung (Signatur Erstellung)

	RSA-MD4	RSA-MD5	RSA-SHA256	RSA-SHA512
Testlauf 1	45.639 ms	50.422 ms	53.230 ms	50.732 ms
Testlauf 2	16.919 ms	18.139 ms	15.841 ms	16.793 ms
Testlauf 3	19.744 ms	12.443 ms	15.318 ms	16.947 ms
Testlauf 4	18.103 ms	14.628 ms	15.803 ms	13.946 ms
Testlauf 5	16.982 ms	13.784 ms	13.126 ms	12.239 ms
Testlauf 6	18.338 ms	14.137 ms	24.549 ms	19.943 ms
Testlauf 7	14.566 ms	11.247 ms	11.772 ms	12.191 ms
Testlauf 8	17.689 ms	13.369 ms	12.277 ms	16.769 ms
Testlauf 9	13.462 ms	14.808 ms	11.173 ms	14.314 ms
Testlauf 10	15.422 ms	16.029 ms	12.069 ms	12.693 ms
Arithmetisches Mittel	18,144 ms	17,900 ms	18,515 ms	18,656 ms

Tabelle 6.4.: Testergebnisse der zweiten Messung (Signatur Verifikation)

folgt von der MD4 Funktion mit 18,144 ms. Die Schlusslichter bilden hier SHA256 mit 18,515 ms und SHA512 mit 18,656 ms.

Wird bei der Erstellung und Verifikation auf die Laufzeit Wert gelegt, gelten folgende Empfehlungen:

- Bei der Betrachtung von Messung eins der Signatur Erstellung kann die SHA256 Funktion und von Messung zwei SHA512 empfohlen werden.
- Wird die Signatur Verifikation betrachtet, kann die MD4 Funktion bei der ersten Messung und MD5 bei der zweiten empfohlen werden.

Aus Sicht der Sicherheit sei allerdings gesagt, dass die Funktionen MD4 und MD5 als unsicher gelten. Da SHA-1 auch die MD4 Funktion verwendet und diese bereits als unsicher

gilt, ist somit auch SHA-1 nicht sicher[2]. Wird also Wert auf kollisionsfrei erzeugte Hashes gelegt, sollte zulasten der Laufzeit vorzugsweise die SHA256 oder SHA512 Funktion Verwendung finden[3].

6.3. Diskussion von Vor- und Nachteilen der Lösung

[2]Vgl. Dietmar Wätjen, 2018, S. 103
[3]Vgl. Bundesamt für Sicherheit in der Informationstechnik, 2019, S. 40, Tabelle 4.1

7. Ausblick

Literatur

[1] W3 Consortium. *Social Web Working Group*. Date accessed: 12.12.2018. 2018. URL: https://www.w3.org/Social/WG.

[2] W3 Consortium. *W3C launches push for social web application interoperability*. Date accessed: 21.07.2014. 2014. URL: https://www.w3.org/blog/news/archives/3958.

[3] W3C Consortium. *HTTP Signatures*. Date accesed: 14.03.2019. 2014. URL: https://tools.ietf.org/id/draft-cavage-http-signatures-01.html.

[4] Fediverse Gemeinschaft. *Fediverse*. Date accessed: 17.01.2019. 2019. URL: https://fediverse.party/en/fediverse/.

[5] Fediverse Gemeinschaft. *Fediverse Network Report 2018*. Date accessed: 17.01.2019. 2019. URL: https://fediverse.network/reports/2018.

[6] Christopher Lemmer Webber u. a. *Der ActivityPub Standard*. Date accessed: 12.12.2018. 2018. URL: https://www.w3.org/TR/activitypub/.

[7] D. Hardt M. Jones. *RFC 6750 - OAuth 2.0 Authorization Framework: Bearer Tokens Usage*. Date accessed: 24.01.2019. 2019. URL: https://tools.ietf.org/html/rfc6750.

[8] Inc. Manu Sporny Digital Bazaar. *The Security Vocabulary*. Date accessed: 22.04.2016. 2016. URL: https://web-payments.org/vocabs/security.

[9] James M. Snell und Evan Prodromou. *Activity Vocabulary*. Date accessed: 23.05.2017. 2017. URL: https://www.w3.org/TR/activitystreams-vocabulary/.

[10] James M. Snell und Evan Prodromou. *ActivityStreams 2.0*. Date accessed: 23.05.2017. 2017. URL: https://www.w3.org/TR/activitystreams-core/.

[11] Christian Stobitzer. *Netzwerkeffekt*. Date accessed: 29.12.2018. 1986. URL: http://www.wirtschafts-lehre.de/netzwerkeffekte.html.

[12] W3C. *Social Web Working Group Charter*. Date accessed: 17.12.2018. URL: https://www.w3.org/2013/socialweb/social-wg-charter.

[13] Wikipedia. *Dezentrale Stromerzeugung*. Date accessed: 06.11.2018. 2018. URL: https://de.wikipedia.org/wiki/Dezentrale_Stromerzeugung.

[14] Wikipedia. *Dezentralisierung (Politik)*. Date accessed: 10.08.2018. 2018. URL: https://de.wikipedia.org/wiki/Dezentralisierung_(Politik).

[15] Wikipedia. *Soziales Netzwerk (Soziologie)*. Date accessed: 14.03.2019. 2018. URL: https://de.wikipedia.org/wiki/Soziales_Netzwerk_(Soziologie).

A. Anhang

A.1. First Appendix Section

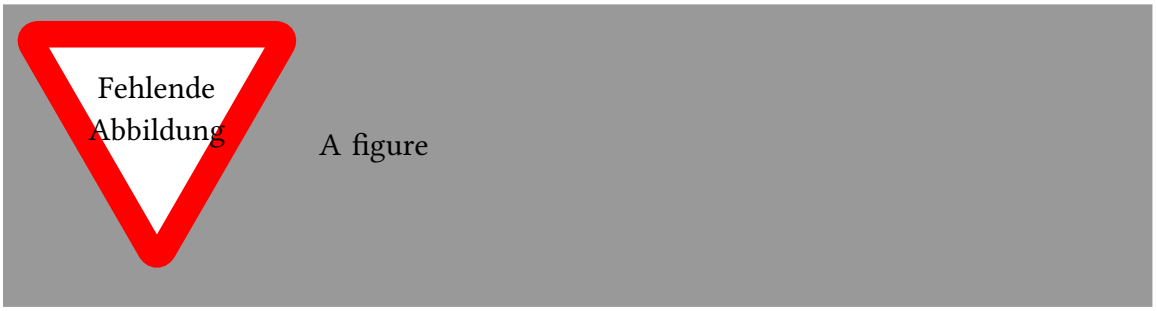


Abbildung A.1.: A figure

...