Funktionsweise und Programmierung eines Webservers mit Fokus auf das http-Protokoll

Gianfranco Kessler

Kantonsschule Zug

Eine technische Maturaarbeit betreut durch

Bernhard Keller

Schuljahr 16/17

Inhalt

Vorwort

Knapp vier Jahre ist es her, seit ich das erste Mal mit den Befehlen des CMD-Terminals in Windows herumgespielt habe. Dies war mein erster richtiger Kontakt mit dem «Programmieren». Obwohl ich damit zum Beispiel erst meinen PC mit einem Klick herunterfahren oder simple Matrizen erstellen konnte, war dies eine interessante Erfahrung für mich. Zwei Jahre später, in meinem 4. Jahr an der Kantonsschule, hatte ich dann das erste Mal richtig Informatik-Unterricht, welcher sich mit dem Programmieren befasste. Wir haben die Programmiersprache Python mithilfe des Python IDLE kennengelernt. Sogleich war ich gepackt vom Programmieren. Leider war die Zeit eher knapp, weshalb wir nicht so weit kamen und ich nur einen kleinen Einblick bekommen habe. Trotzdem war mir schon dann recht klar, dass ich Informatik studieren möchte und als Ergänzungsfach für die 6. Klasse Informatik wählen werde, da ich davon einfach fasziniert war. Auch für meine Maturaarbeit wusste ich schnell, dass ich etwas programmieren möchte.

Ursprünglich wollte ich eine Smartphone-Applikation, präziser ein Spiel programmieren, jedoch wäre es dafür notwendig gewesen, eine ganze neue Programmiersprache zu erlernen und sowohl die App- als auch die Netzwerkseite zu programmieren, was ein unrealistisch grosser Aufwand gewesen wäre. Somit entschliess ich mich zum Thema http-Webserver.

Ich möchte mich hier ausdrücklich bedanken bei allen, die mich während dieser Arbeit unterstützt haben. Mein besonderer Dank gilt:

- Meiner Betreuungsperson Bernhard Keller, welcher mich kompetent und tatkräftig unterstützt hat und mir bei Problemen schnell weiterhelfen konnte.
- Jonas Wyss, welcher mir mit seinem Fachwissen als Informatiklehrling sowohl bei meiner Arbeit, als auch vor allem bei meinem Webserver weiterhelfen konnte.
- Meiner Familie für ihre Unterstützung und Motivierung.

Einleitung

Um das Thema http-Protokoll ausführlich zu erklären, befasst sich diese Arbeit zuerst mit dem sogenannten OSI-Modell, um dem Leser die verschiedenen Schichten und Abläufe während der Benutzung des Internets näher zu bringen. Im Kapitel zum OSI-Modell wird zuerst erklärt, was Server sind und was passiert, wenn man eine Webseite aufruft. Dann werden die einzelnen Schichten näher beschrieben. Der Fokus liegt jedoch auf dem http-Protokoll und behandelt dessen Entstehung, dessen Möglichkeiten und das http-Protokoll in Python im Spezifischen.

Material

Für die Recherche wurde hauptsächlich das Internet benutzt, da dort viel über Python, sowie allgemein Informationen zum http-Protokoll verfügbar sind.

Der Code wurde in Python 3 geschrieben, wofür sowohl Python IDLE Version 3.5 sowie die PyCharm Community Edition in der Version 2016.2.1 benutzt wurde. Die verschiedenen Dateien wurden auf GitHub hochgeladen, wo auch das Arbeitsprotokoll verfügbar ist. Meine Arbeit findet man im Repository unter: https://github.com/Giankess/Maturaarbeit

Hauptteil

Webserver anhand des OSI-Modell

Server sind Computer, welche ihre Daten anderen Computern innerhalb eines Netzwerks zur Verfügung stellen. So ähnlich funktioniert dies auch mit Webservern, mit dem Unterschied, dass die Daten öffentlich zugänglich sind und hier jeder mit Internetanschluss Zugang dazu hat. Die Daten werden als Webseite abgerufen. Jede Webseite hat eine URL (mehr dazu Seite 6, letzter Abschnitt), welche ähnlich wie eine Hausadresse funktioniert. Sie besteht meist aus dem zu verwendenden Protokoll, einer Domain[i], einem Webserver und teilweise aus dem Verzeichnis, in welchem sich die aufzurufenden Dateien befinden. (Aidex)

Wie funktioniert nun das Aufrufen einer Webseite genau? Das Ganze kann mit dem sogenannten Open Systems Interconnection (OSI)-Modell dargestellt werden. Das Problem bei der Kommunikation von verschiedenen Programmen ist, dass sie verschiedene Sprachen (Protokolle) verwenden. Nun müssen diese verschiedenen Protokolle irgendwie übersetzt werden. Der ganze Vorgang vom Abrufen einer Webseite wird beim OSI-Modell in sieben Schichten unterteilt: Anwendung, Darstellung, Sitzung, Transport, Vermittlung, Sicherung und Bitübertragung. (Wikibooks)

Anwendungsschicht

Wie der Name schon sagt, besteht diese Schicht aus der Anwendung (z.B. Google Chrome), welche der Benutzer gerade benutzt. Sie stellt Funktionen zur Verfügung wie Internetkommunikation über das Hypertext Transfer Protocol (http; siehe Seite 6 ff.) oder Secure Shell (SSH)[ii] und regelt die Datenein- und ausgabe. (Wikibooks; Mitchell)

Darstellungsschicht

Die Darstellungsschicht übersetzt systemabhängige Zeichen in eine unabhängige Form, sodass sie von verschiedenen Systemen verstanden werden können. Sie ermöglicht durch Übersetzen die Kommunikation mit den unteren Schichten. (Wikibooks; Mitchell)

Sitzungsschicht

Damit Systeme miteinander kommunizieren können, muss eine Sitzung aufgebaut und instand gehalten werden. Dies regelt die Sitzungsschicht. Wenn die Kommunikation vorbei ist, wird die Sitzung wieder beendet. Dies ist notwendig, damit Anfragen an den Zielserver gesendet werden können und dieser antworten kann. Hier bietet http (mehr dazu ab Seite 6) den Dienst an, dass wenn ein Client einen Uniform Ressource Locator, kurz URL (siehe Seite 6, letzter Abschnitt) eingibt, Daten vom Zielserver geholt und an den Client übermittelt werden. (Wikibooks; Mitchell)

Transportschicht

Die Transportschicht baut eine logische Ende-zu-Ende-Verbindung auf. Das Transfer-Control-Protocol (TCP) nutzt hier die Eigenschaft des Internetprotokolls, einzelne Datenpakete zwischen Systemen mithilfe der Internet Protocol (IP)-Adresse zu versenden. Die Transportschicht wandelt die Datenpakete in Teilpakete um und sorgt für die richtige Zusammensetzung beim Zielsystem. (Wikibooks; Mitchell)

Vermittlungsschicht

Die Datenpakete gelangen nicht direkt zum Ziel, sie haben verschiedene «Zwischenstationen». Die Vermittlungsschicht sucht sogenannte «Netzknoten» und baut eine Verbindung auf und danach wieder ab, sie steuert also die Route, welche die Datenpakete nehmen. Diese Schicht läuft meist über einen Router. (Wikibooks; Mitchell)

Sicherungsschicht

Die Sicherungsschicht besteht aus zwei Unterschichten, nämlich der Medium Access Control (MAC)-Schicht und der Logical Link Control (LIC)-Schicht. Sie sorgt für den zuverlässigen Austausch von Datenpaketen zwischen den Systemen. (Wikibooks; Mitchell)

Bitübertragungsschicht

Die Bitübertragungsschicht ist für die Übertragung der Bitströme verantwortlich. Sie definiert die elektrische, mechanische und funktionale Schnittstelle zum Übertragungsmedium (z.B. Metall, Glas oder Luft). (Wikibooks; Mitchell)

Beispiel

Was passiert nun also, wenn man im Webbrowser eine Webseite aufrufen will? Man sucht zum Beispiel mittels Google Chrome nach «youtube.com», dies ist die Anwendungsschicht. In der Darstellungsschicht wird die Anfrage nun also umgewandelt. Die Sitzungsschicht baut eine Sitzung auf, die Transportschicht baut eine logische Ende-zu-Ende-Verbindung mithilfe der IP-Adresse, welche jeder Computer besitzt, (in diesem Falle 208.65.153.238) auf. Die Anfrage wird anschliessend an den Server vermittelt, die Vermittlungsschicht routet die Datenpakete zum nächsten Netzknoten. Die Sicherungsschicht fügt die Sender- und Empfängeradresse hinzu und teilt die Pakete in mehrere Frames, sogenannte Teilpakete, ein. Diese werden dann in der Bitübertragungsschicht physisch an den Youtube-Server gesendet. Dort werden die Datenpakete geholt und passieren dann dieselben Vorgänge in umgekehrter Reihenfolge nochmals, bis die Webseite in Google Chrome erscheint.

http-Protokoll

Was sind Protokolle?

Ein Protokoll ist einfach gesagt, wie eine Sprache. So hat jedes Protokoll seine eigenen Regeln, wie die Grammatik, ohne welche verschiedene Systeme nicht miteinander kommunizieren können. Benutzen beide Systeme die gleiche Sprache und die gleichen Regeln, so können sie miteinander kommunizieren. (Menzerath)

Das http-Protokoll

1989, kurz nachdem das Internet erfunden wurde, hatte der Brite Tim Berners-Lee der Europäischen Organisation für Kernforschung (CERN)[iii] die Idee eines weltweiten Netzwerkes vorgeschlagen. Innerhalb dieses sollte jeder Computer Daten anbieten und anfordern können, damit Wissenschaftler von überall auf der Welt miteinander kommunizieren können. Dafür mussten die verschiedenen Computer jedoch eine gemeinsame Sprache sprechen. Dazu entwickelte Berners-Lee bis Ende 1990 das http-Protokoll. Nebenbei hat er noch die Hypertext Markup Language (HTML)[iv] und die URL entwickelt und so die Basis für seinen Webserver und -browser geschaffen. (Webfoundation; Franke)

Im Laufe der Jahre wurden neue Versionen herausgebracht. Angefangen hat das Ganze 1991 mit HTTP/0.9. Dieses ermöglichte das Senden einer Anfrage für eine Datei, einer sogenannten GET-Request. Da diese Anfrage nur aus einer Zeile bestand, wurde diese Version oft auch das One-Line-Protocol genannt. Die Response bestand nur aus einem Body mit dem Inhalt der Datei. 1996 wurde dann das HTTP/1.0 als Request for Comment (RFC)[v] veröffentlicht. Die grösste Veränderung war, dass hier viel mehr Daten über Client und Server übertragen werden können. Eine Request besteht seither aus der Methode, dem Header und einem Body. Neu gab es nun neben GET auch die Methoden HEAD und POST. Zusätzlich war es jetzt auch möglich, eine Authentifizierung zu verlangen, damit nicht jeder auf alles zugreifen konnte. 1999 wurde dann HTTP/1.1 publiziert. Diese neue Version brachte neue Methoden wie DELETE, OPTIONS oder PUT, eine verbesserte Authentifizierung, das Ermöglichen von persistenten, daher offenbleibenden Verbindungen und vieles mehr. (O'Reilly; Franke)

Es ist klar, dass das Internet niemandem gehören kann, doch wer kontrolliert das http-Protokoll? Das HTTP/1.0 wurde ja als RFC 2616 veröffentlicht. Diese RFC's werden von der sogenannten Internet Society[vi]kontrolliert. Zusätzlich sind aber auch noch die Internet Engineering Task Force[vii](IETF) und das Internet Architecture Board[viii](IAB) als Unterorganisationen der Internet Society beteiligt, welche das Ganze auch steuern. (Strickland)

Das http-Protokoll ist ein sogenanntes Übertragungsprotokoll. Im Normalfall sendet ein Browser eine Anfrage an den http-Server und bekommt eine Antwort, danach wird die Verbindung beendet.

Bevor eine solche Anfrage aber überhaupt möglich ist, muss der Client zuerst den Server finden und die zu schickenden Daten angeben. Dies übernimmt die URL, welche gleich auch noch das Protokoll, welches verwendet werden soll, angibt. Eine URL ist meist wie folgt aufgebaut: http://www.domain.topleveldomain:Port/pfad/Dateiname wobei der Port optional ist. Das http ist das zu verwendende Protokoll, welches der Browser jedoch meist selbst einfügt. www.domain.topleveldomain ist der Servername, welchen man auch durch die IP ersetzen könnte, da domain.de nur ein im Domain Name System (DNS) eingetragener Name für die betreffende IP ist und welcher von Nameservern übersetzt werden kann. Der Port gehört zum verwendeten Protokoll, bei http z.B. ist es standardmässig der Port 80. Dieser kann jedoch individuell verändert werden. Standardmässig sind auf einem Computer die ersten 1024 Ports festgelegt. Der Pfad gibt an, in welchem Verzeichnis die abzurufende Datei ist, sofern nicht das Stammverzeichnis abgerufen werden soll. (Menzerath; Cyon)

Damit Daten gesendet werden können, muss zuerst eine Anfrage geschickt werden, eine sogenannte http-Request. Sie besteht aus der Methode[ix] (meist GET oder POST) und dem Request-Header. Ein Beispiel für einen http-Request-Header wäre folgendes:

«
```
GET /hello.txt HTTP/1.1
User-Agent: curl/7.16.3 libcurl/7.16.3 OpenSSL/0.9.7l zlib/1.2.3
Host: www.example.com
Accept-Language: en, mi
```
» (Reschke, 2017)
Die erste Zeile benennt die Methode, welche verwendet wird, die angeforderte Datei und die verwendete Version des Protokolls. Die zweite Zeile ist da, damit die Antworten auf den Browser zugeschnitten sind. Die Dritte wählt den Host aus, bei welchem die Datei geholt werden soll, falls der Server mehrere Domains verwaltet. Die letzte Zeile gibt an, welche Sprachen (und evtl. Formate) akzeptiert werden. (Menzerath)
Auf die Anfrage sendet der http-Server dann eine Antwort zurück, welche aus einem dreistelligen Status-Code über die Verfügbarkeit und einer Klartextmeldung besteht.

Ein Beispiel für einen Response-Header wäre:

«
```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
ETag: "34aa387-d-1568eb00"
Accept-Ranges: bytes
Content-Length: 51
Vary: Accept-Encoding
Content-Type: text/plain

Hello World! My payload includes a trailing CRLF.
```
» (Reschke, 2017)
Bei der Response beschreibt die erste Zeile die Protokollversion sowie einen Statuscode (mehr zum Statuscode im nächsten Abschnitt). In der zweiten Zeile findet man das Datum, in der dritten den verwendeten Server. Darunter findet man noch Angaben zum Inhalt, zum Beispiel die Länge in Bytes, den Typ des Inhalts, etc. Ausser der ersten Zeile haben all diese Angaben weder in der Request noch in der Response eine bestimmte Reihenfolge.

Wie bereits erwähnt, findet man im Response-Header verschiedene Statuscodes. Ein Statuscode gibt an, ob die Anfrage erfolgreich war oder ob ein Fehler vorliegt und wenn, welcher. Verschiedene wichtige Statuscodes sind folgende:

<u>2XX Erfolgreiche Operation</u>

200 OK: Die Anfrage wurde erfolgreich bearbeitet.

<u>3XX Umleitung</u>

301 Moved Permanently: Die benutze Adresse ist nicht mehr gültig. Die neue wird unten im Location-Feld angegeben.

307 Temporary Redirect: Die benutzte Adresse ist vorübergehend nicht verfügbar. Die vorübergehende wird im Location-Feld angegeben.

4XX Client-Fehler

400 Bad Request: Die Anfrage war fehlerhaft.

403 Forbidden: Kein berechtigter Zugriff

404 Not Found: Die angeforderte Ressource konnte nicht gefunden werden.

5XX Server-Fehler

500 Internal Server Error: Unerwarteter Server-Fehler.

504 Gateway Time-out: Keine Antwort innerhalb der festgelegten Zeitspanne.

(iana.org; W3.org)

Was aber ist, wenn bestimmte Daten nicht für alle bestimmt sind? http hat dafür das Basic-http-Authentication-Schema, welches einen Benutzernamen und ein Passwort verwendet. Wenn dies der Fall ist, kommt bei der http-Response der Status-Code 401. Der Browser öffnet dann ein Fenster, wo man aufgefordert wird, den Benutzernamen und das Passwort einzugeben, die dann an den Server gesendet werden. Falls die Eingaben stimmen, wird Zugriff freigegeben.

Das aktuelle HTTP verfügt jedoch noch über viele andere Funktionen, wie Message Syntax, Routing, Caching, etc., welche ich hier jedoch nicht weiter behandeln werde.

Das http-Protokoll im OSI-Modell

Wo kommt das http-Protokoll eigentlich vor in den Schichten des OSI-Modells? Die Bitübertragungsschicht läuft über Kabel, Lan oder Licht, die Sicherungsschicht über das Ethernet, die Vermittlungsschicht über das Internet Protocol, die Transportschicht über TCP. Die Kommunikation- und Darstellungsschichten laufen nun über http und die Anwendungsschicht über das Internet. Nun wird klar, dass http dem TCP und IP übergeordnet ist (siehe Tabelle 1). http verarbeitet die Daten der beiden Systeme und benutzt TCP für die Verbindung und IP für die Adressierung.

(Elektronik Kompendium; Menzerath)

Tabelle 1: Protokolle im OSI-Modell

	OSI	TCP/IP
7	Anwendungsschicht	Anwendungen (FTP, SMTP, http, etc)
6	Darstellungsschicht	
5	Sitzungsschicht	
4	Transportschicht	TCP (Host to Host)
3	Vermittlungsschicht	IP
2	Sicherungsschicht	Netzwerkzugriff (Normalerweise Ethernet)
1	Bitübertragungsschicht	

Das http-server-Modul in Python

In Python kann man mithilfe des Socketserver[x]- und des http-Moduls relativ einfach einen simplen Webserver programmieren, welcher GET- und POST-Requests erhalten und bearbeiten kann. Welche Rolle spielt das http-Modul dabei?

Das http-Modul übernimmt die Rolle des Handlers (Bearbeiter), welcher eingehende Requests analysiert und dann beantwortet. Dieser prüft, welche Methode er verwenden muss, er schaut, welche Protokollversion verwendet wird, was angefordert wird und ob es verfügbar ist. Er liest die Host-Adresse, prüft für welchen User-Agent er die Antwort optimieren muss und welche Sprachen akzeptiert werden. Dann schreibt er eine Request, welche sagt, ob die Anfrage erfolgreich war oder nicht und sie je nachdem anschliessend beantwortet mit Infos zur angeforderten Datei.

Die Kommunikation durch das http-Protokoll verläuft kodiert. Die Daten kommen kodiert als Bytestring an und werden dann mit UTF-8[xi] dekodiert. Wenn sie zurückgesendet werden muss das mitgesendete File im Body je nachdem mit einer anderen Kodierung kodiert werden.

Webserver in Python

Als Ausgangslage wurde ein Standard-http-Webserver-Skript verwendet.

«

```
import http.server
import socketserver

PORT = 8000

Handler = http.server.SimpleHTTPRequestHandler

httpd = socketserver.TCPServer(("", PORT), Handler)

print("serving at port", PORT)
httpd.serve_forever()
```

» (Python Software Foundation, 2017)

Dieses Programm hostet einen Server unter localhost:8000. Alle Requests, welche eingehen, werden hier automatisch vom SimpleHTTPRequestHandler bearbeitet. In dieser Arbeit wird der http-Handler durch einen eigenen ersetzt. Dazu muss man wissen, was ein Handler alles beinhalten muss und welche Prozesse wie ablaufen. Zuerst kommt die Request als Bytestring im Webserver an. Diese muss mit UTF-8, der am weitesten verbreiteten Unicode-Kodierung, decodiert werden. Dann wird sie geparsed, also in Teile zerlegt, damit der Handler die einzelnen Elemente der Request verarbeiten kann. Anschliessend wird die Anfrage bearbeitet und eine Response zusammengestellt, die muss jedoch auch wieder verschlüsselt werden. Der Responseheader wird jeweils erneut mit UTF-8 codiert, der Body hängt von der Art der Datei, welche mitgesendet wird, ab.

Nun muss man sich Gedanken machen, was ein Webserver alles braucht. Die Maindatei wie im Beispiel oben kann manübernehmen und muss nur ein wenig abgeändert werden. Dann braucht es einen Requestparser, eine Klasse, die testet welche Methode verwendet wird und eine Responsebuilderklasse. Man soll nun also einen Requestparser erstellen, welcher die Zeilen der Request auseinandernimmt und dann ein Dictionary mit Headerfeld (z.B. Host) und Inhalt (z.B. www.google.com) für den Inhalt des Headers erstellt und sofern vorhanden, den Body speichert.

Danach wird die Methode getestet. Je nach Methode wird dann ein entsprechender Responseheader mithilfe des Dictionarys erstellt und das File bearbeitet oder gesendet. Der im Rahmen dieser Arbeit erstellte Handler kann nur PUT und GET bearbeiten. Falls nun also eine andere Methode eingeht oder das File bei GET nicht vorhanden ist, wird eine Error-response mit entsprechendem Statuscode zurück gesendet. Das Programm ist so aufgebaut, das man die Maindatei startet. Diese hostet dann einen Webserver und sobald eine Request eingeht, wird diese erfasst. Zunächst startet die Maindatei den Requestparser, welcher die Request bearbeitet. Dann wird die Testmethode aufgerufen. Die Methode aus der Request wird getestet und die Testmethode ruft den Responsebau auf, welcher die jeweilige Response zusammenstellt. Danach wird die Response zurück an die Maindatei gegeben, welche diese dann an den Empfänger zurückschickt. Wie bei Webservern üblich, wird, falls bei der GET-Methode kein Pfad angegeben ist, die index.html-Datei aufgerufen. In dieser kann man dann auf die anderen Dateien im Verzeichnis mit einem Klick zugreifen.

Schlussteil

Schwierigkeiten und Erkenntnisse bei der Programmierung

Der vermutlich schwierigste Teil meines Projekts war das Programmieren des Requestparsers. Zuerst habe ich ihn so umgesetzt, dass er Elemente aus einzelnen, bestimmten Zeilen als Variablen speichert. Jedoch war mir bis dahin noch nicht klar, dass die Zeilen, ausser der ersten, keine bestimmte Reihenfolge haben müssen. Auch habe ich nicht beachtet, dass die Anzahl Zeilen nicht definiert war, ein möglicher Body nicht berücksichtigt wurde und die Anzahl Zeilen nicht vordefiniert war. Daraufhin habe ich zuerst versucht, die Zeilen von 2 bis zum Ende mit einem Loop in einem Dictionary zu speichern. Doch auch hier war der Body und allgemein Zeilen ohne Doppelpunkt nicht berücksichtigt. Schlussendlich habe ich alles in eine Schleife eingebaut, welche jede Zeile einzeln auf Doppelpunkt oder http im Inhalt testet und diese dann, je nachdem unterschiedlich parst. Falls ein http vorkommt, ist es die erste Zeile und muss in Methode, Version und Pfad zerlegt werden, welche dann als Variablen abgespeichert werden. Für den Fall, dass ein Doppelpunkt vorkommt, ist es eine Zeile aus dem Header und kann ins Dictionary abgespeichert werden. Falls die Zeile weder Doppelpunkt noch http enthält, ist es eine Zeile des Body und wird deshalb dem String «Body» hinzugefügt.

Ein anderes Problem tauchte beim Responsebuilder auf. Ich habe zuerst jede Response einzeln zusammengefügt. Da dies sehr unübersichtlich ist, habe ich eine Vorlage mit mehreren Lücken gemacht, welche dann an die jeweilige Methode angepasst formatiert werden kann.

Fazit

Im Verlaufe meiner Maturaarbeit konnte ich mir viel neues Wissen aneignen. Sowohl Wissen über die Funktionsweise des Internets und das http-Protokoll, als auch Erfahrung im fortgeschrittenen Umgang mit Python und der objektorientierten Programmierung. Da die meisten Programmier-sprachen ähnlich aufgebaut sind, wird mir dies beim Erlernen weiterer Programmiersprachen viel helfen.

Eine weitere wichtige Erkenntnis ist, dass ich eventuell früher mit dem Programmieren hätte beginnen sollen, da dort immer wieder neue Fehler auftreten können, mit welchen man nicht rechnet und welche viel Zeit kosten können. Deshalb sollte man immer eine zeitliche Reserve einplanen. Eine Optimierung wäre, wenn der Handler noch andere Dateien als nur Textdateien kodieren könnte. Momentan ist es nur möglich, Textdateien wie zum Beispiel HTML mit UTF-8 zu kodieren und zurückzusenden.

Schlussendlich hat mich diese Arbeit in meinem Entscheid Informatik zu studieren nur bestärkt und war eine wertvolle Erfahrung.

Literaturverzeichnis

Aidex. (29. 1 2017). *Aidex.de*. Von https://www.aidex.de/software/webserver/was-ist-ein-webserver.html abgerufen

Cyon. (29. 1 2017). *Cyon.ch*. Von Cyon.ch: https://www.cyon.ch/support/a/was-ist-dns-und-was-ist-ein-nameserver abgerufen

Elektronik <kompendium. (29. 1 2017). *elektronik-kompendium.de*. Von http://www.elektronik-kompendium.de/sites/net/0902231.htm abgerufen

Franke, J., Jaschkowski, J., & Miller, A. (29. 1 2017). *Goessner*. Von http://goessner.net/download/learn/mwt/ws2005/presentations/HTTP.pdf abgerufen

iana.org. (29. 1 2017). *iana.org*. Von http://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml abgerufen

Menzerath, M. (29. 1 2017). *Menzerath*. Von Menzerath: https://menzerath.eu/artikel/wie-funktioniert-das-http-protokoll/ abgerufen

Mitchell, B. (23. Januar 2017). *Lifewire*. Von https://www.lifewire.com/layers-of-the-osi-model-illustrated-818017 abgerufen

O'Reilly. (29. 1 2017). *High Performance Browser Networking*. Von https://hpbn.co/brief-history-of-http/ abgerufen

Python Software Foundation. (29. 1 2017). *doc.python.org*. Von https://docs.python.org/3.4/library/http.server.html abgerufen

Reschke, R. F. (28. 1 2017). *Tools.ietf.org*. Von https://tools.ietf.org abgerufen

Scholz, R. (29. 1 2017). *Netzwerke.com*. Von http://www.netzwerke.com/OSI-Schichten-Modell.htm abgerufen

Strickland, J. (29. 1 2017). *How Stuff Works*. Von http://computer.howstuffworks.com/internet/basics/who-owns-internet3.htm abgerufen

W3.org. (29. 1 2017). *W3.org*. Von https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html abgerufen

Webfoundation. (29. 1 2017). *Webfoundation.org*. Von http://webfoundation.org/about/vision/history-of-the-web/ abgerufen

Wikibooks. (29. 1 2017). *Wikibooks*. Von Wikibooks: https://de.wikibooks.org/wiki/Netzwerktechnik:_OSI abgerufen

Wikipedia. (4. 11 2016). *Wikipedia HTTP Statuscode*. Von https://de.wikipedia.org/wiki/HTTP-Statuscode abgerufen

Glossar

[i] Eine Domain ist ein Teilbereich im hierarchischen Domain Name System(DNS)
[ii] SSH oder Secure Shell ist ein Netzwerkprotokoll für verschlüsselte Netzwerkkommunikation.
[iii] Das CERN ist die Abkürzung für Conseil européen pour la recherche nucléaire. Es hat den Sitz in der Schweiz.
[iv] HTML ist die sogenannte Hypertext Markup Language. Mit HTML kann man Dokumente unter anderem im WWW strukturiert darstellen.
[v] RFC ist die Abkürzung für Request for Comment, welches eine Art der Publikation für das IETF ist.
[vi] Die Internet Society ist für Pflege und Weiterentwicklung des Internets zuständig. Sie besteht aus Einzelpersonen sowie mehreren anderen Unterorganisationen wie z.B. IETF.
[vii] Die Internet Engineering Task Force arbeitet an der technischen Weiterentwicklung des Internets.
[viii] Das Internet Architecture Board ist ein Komitee, welches den architekturellen Überblick über die Aktivitäten der IETF wahrt und die Internet Society unterstützt.
[ix] Eine Methode in einer Request definiert, welche Aktion ausgeführt wird. Je nachdem wird eine Datei erstellt, bearbeitet, gelöscht oder einfach gesendet.
[x] Das Socketserver-Modul wird verwendet, um Netzwerk-Server zu erstellen.
[xi] UTF-8 ist die Abkürzung für 8-Bit Universal Character Set Transformation Format. Es ist die am weitesten verbreitete Kodierung für Unicode-Zeichen.