Protokoll für die Aufgabe_2

A1 Einarbeitungsphase

wget Aufruf im Linux Terminal:

```
navruz@DESKTOP-706DH16:/mnt/c/Users/navru$ wget scimbe.de/_index.html
--2021-11-27 16:16:39--  http://scimbe.de/_index.html
Resolving scimbe.de (scimbe.de)... 81.169.145.86, 2a01:238:20a:202:1086::
Connecting to scimbe.de (scimbe.de)|81.169.145.86|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12675 (12K) [text/html]
Saving to: '_index.html'

_index.html                 100%[=============================================>]  12.38K  --.-KB/s    in 0.001s

2021-11-27 16:16:39 (16.7 MB/s) - '_index.html' saved [12675/12675]
```

Aufzeichnung im Wireshark:

No.	Time	Source	Destination	Protocol	Lengt	Info
	6.279715	192.168.178.20	81.169.145.86	TCP	74	57334 → 80 [SYN] Seq=0 Win=64240 Le
	6.306411	81.169.145.86	192.168.178.20	TCP	74	80 → 57334 [SYN, ACK] Seq=0 Ack=1 W
	6.306746	192.168.178.20	81.169.145.86	TCP	66	57334 → 80 [ACK] Seq=1 Ack=1 Win=64
→	6.306886	192.168.178.20	81.169.145.86	HTTP	213	GET /_index.html HTTP/1.1
	6.334310	81.169.145.86	192.168.178.20	TCP	66	80 → 57334 [ACK] Seq=1 Ack=148 Win=
	6.336450	81.169.145.86	192.168.178.20	TCP	14…	80 → 57334 [ACK] Seq=1 Ack=148 Win=
	6.336693	192.168.178.20	81.169.145.86	TCP	66	57334 → 80 [ACK] Seq=148 Ack=1401 W
	6.338817	81.169.145.86	192.168.178.20	TCP	14…	80 → 57334 [ACK] Seq=1401 Ack=148 W
	6.338817	81.169.145.86	192.168.178.20	TCP	14…	80 → 57334 [ACK] Seq=2801 Ack=148 W
	6.338817	81.169.145.86	192.168.178.20	TCP	14…	80 → 57334 [ACK] Seq=4201 Ack=148 W
	6.338817	81.169.145.86	192.168.178.20	TCP	14…	80 → 57334 [ACK] Seq=5601 Ack=148 W
	6.338817	81.169.145.86	192.168.178.20	TCP	14…	80 → 57334 [ACK] Seq=7001 Ack=148 W
	6.338817	81.169.145.86	192.168.178.20	TCP	14…	80 → 57334 [ACK] Seq=8401 Ack=148 W
	6.338817	81.169.145.86	192.168.178.20	TCP	14…	80 → 57334 [ACK] Seq=9801 Ack=148 W
	6.338817	81.169.145.86	192.168.178.20	TCP	14…	80 → 57334 [ACK] Seq=11201 Ack=148
←	6.338817	81.169.145.86	192.168.178.20	HTTP	427	HTTP/1.1 200 OK (text/html)
	6.338982	192.168.178.20	81.169.145.86	TCP	66	57334 → 80 [ACK] Seq=148 Ack=8401 W
	6.339016	192.168.178.20	81.169.145.86	TCP	66	57334 → 80 [ACK] Seq=148 Ack=11201
	6.339082	192.168.178.20	81.169.145.86	TCP	66	57334 → 80 [ACK] Seq=148 Ack=12962

```
▾ Hypertext Transfer Protocol
  ▾ GET /_index.html HTTP/1.1\r\n
    ▾ [Expert Info (Chat/Sequence): GET /_index.html HTTP/1.1\r\n]
        [GET /_index.html HTTP/1.1\r\n]
        [Severity level: Chat]
        [Group: Sequence]
      Request Method: GET
      Request URI: /_index.html
      Request Version: HTTP/1.1
    User-Agent: Wget/1.20.3 (linux-gnu)\r\n
    Accept: */*\r\n
    Accept-Encoding: identity\r\n
    Host: scimbe.de\r\n
    Connection: Keep-Alive\r\n
    \r\n
    [Full request URI: http://scimbe.de/_index.html]
    [HTTP request 1/1]
    [Response in frame: 41]
```

HTTP Header beinhalten Informationen, mit denen Server und Client wichtige zusätzliche Informationen übermittelt bekommen. Es gibt verschiedene sogenannte Header Felder. Die Wichtigsten werden kurz erläutert.

- **Content-Type:** Beinhaltet die Information, welcher Media Typ vorliegt.
- **Range:** Spezifiziert einen Teilbereich des angeforderten Inhalts. Also welcher Teil der gesamten Nachricht angefordert wird.
- **Content-Range:** Wird nur gemeinsam mit einer Teilnachricht versendet. Sagt aus, wo sich dieser Teil in der gesamten Nachricht befindet.
- **Content-Language:** Spezifiziert die natürliche Sprache.
- **Content-Location:** Beinhaltet die direkte URL für den Zugriff der Ressource.
- **Date:** Sagt aus, an welchem Tag und zu welcher Uhrzeit die Nachricht entstanden ist. Es gibt nur wenige Ausnahmen, bei denen kein solches Date Feld benötigt wird, z.B. wenn es nicht möglich ist durch einen Server Error.

A2 HTTP Clientanwendung

In dieser Teilaufgabe dreht sich alles um eine einfache HTTP-Clientanwendung. Es werden verschiedene Funktionen unterstützt. Die Anwendung kann eine Ressource von einem Webserver mit einem GET Request anzufordern. Außerdem wird auch darauf gesorgt, dass nach dem Ausführen wieder aufgeräumt wird und Speicherbereiche wieder freigegeben werden.

Zu Veranschaulichung wurde ein Klassendiagramm erstellt, welches eine Übersicht über die Struktur der Applikation darstellt.

Connection:

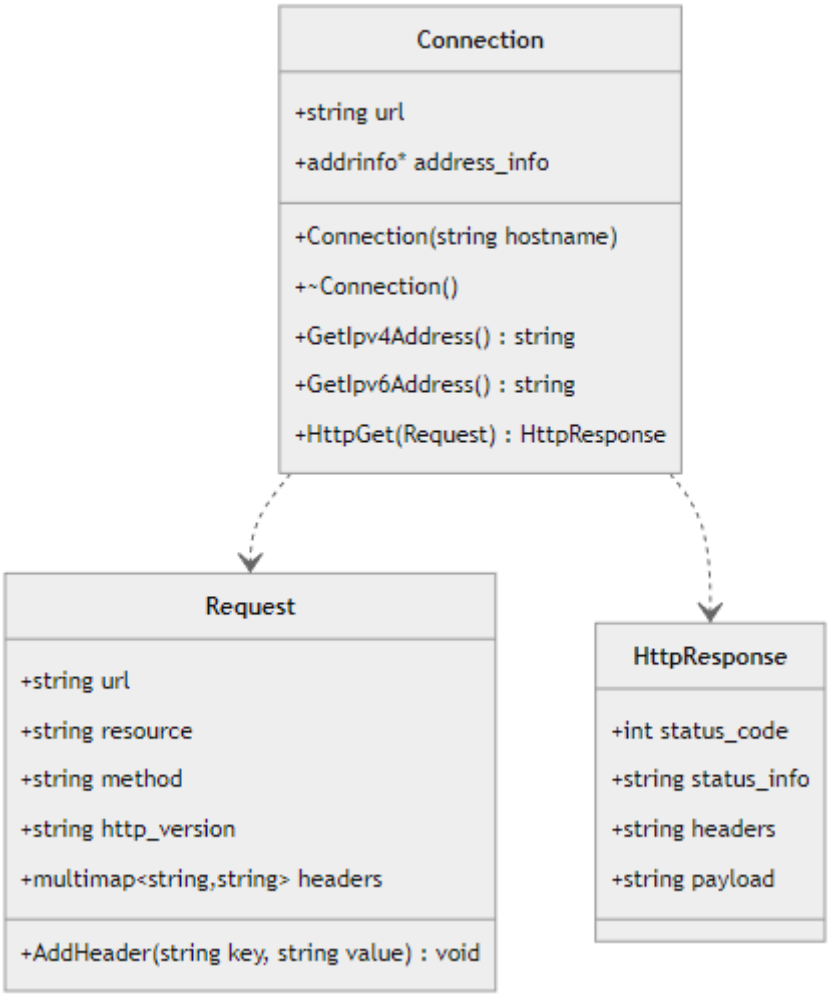
- Initialisierung einer Verbindung mit einer URL
- Destruktor zum automatisierten Schließen einer Verbindung
- Abfragen von IP-Informationen des Hostnamens über DNS
- Durchführen eines HTTP-GET-Requests

Request:

- Klasse zur Initialisierung eines HTTP-Requests
- Setzen gewünschter Header-Felder

HTTP-Response:

- Speichern empfangener HTTP-Response in separaten Datenstrukturen
- Statuscode und textuelle Informationen zum HTTP-Status
- Header-Informationen
- Payload: empfangene Nutzdaten


Connection
+string url
+addrinfo* address_info
+Connection(string hostname)
+~Connection()
+GetIpv4Address() : string
+GetIpv6Address() : string
+HttpGet(Request) : HttpResponse
Request
+string url
+string resource
+string method
+string http_version
+multimap<string,string> headers
+AddHeader(string key, string value) : void
HttpResponse
+int status_code
+string status_info
+string headers
+string payload

Um den Ablauf eines GET-Request besser zu verstehen und nachvollziehen zu können bietet sich ein Sequenzdiagramm an.

1. Aufbau der Verbindung über TCP
2. Senden des HTTP-GET-Requests
3. Empfang der HTTP-GET-Response
4. Schließen der Verbindung über TCP