Einführung in die Programmierung

Prof. Dr. Peter Thiemann
Marius Weidner, Hannes Saffrich
Lukas Kleinert, Timpe Hörig

Universität Freiburg
Institut für Informatik
Wintersemester 2023

Weihnachtsblatt (Blatt 10)

Abgabe: Montag, 08.01.2024, 9:00 Uhr morgens

Sie alle kennen Social Networks, doch wie funktionieren diese eigentlich?

Ein Social Network besteht in der Regel aus zwei Teilen: Aus einem zentralen `Server`, auf dem alle Daten gespeichert sind und aus mehreren `Client`s. In diesem Übungsblatt schreibt jeder von Ihnen einen solchen `Client`, den `Server` stellen wir zur Verfügung[1]. Ein `Client` kann Daten an den `Server` schicken und Daten vom `Server` anfragen, die dieser dann an den `Client` schickt.

Für dieses Übungsblatt benötigen Sie die in der Vorlesung vorgestellte `requests`-Library. Installieren Sie diese via `python3.12 -m pip install requests`[2].

Aufgabe 10.1 (Yeets; 4 Punkte; Datei: `client.py`)

(a) Datenklasse Yeet, 1 Punkt
Schreiben Sie eine Datenklasse `Yeet`, die eine Gleitkommazahl `date`, einen Inhalt `content` als String, einen Autor `author` als String und eine ganze Zahl `id` als Attribute besitzt.
Ein Yeet beschreibt eine Nachricht im Netzwerk. `date` ist eine Zahl, die die Sekunden seit Donnerstag, dem 1. Januar 1970, 00:00 Uhr UTC zählt[3], `id` ist eine Ganzzahl, mit der die Nachricht auf dem Server eindeutig identifiziert werden kann.

(b) Timestamp zu String, 1 Punkt
Schreiben Sie eine Methode `timestamp_to_string`, die einen String zurück gibt, der das Datum und die Uhrzeit des `Yeets` anzeigt. Verwenden Sie hierfür die Klasse `datetime` aus dem Modul `datetime`, um einen Timestamp zu einem String zu konvertieren.
Mit `datetime.fromtimestamp(timestamp, UTC)` können Sie eine neue Instanz der Klasse erzeugen. Benutzen Sie dann die auf `datetime` definiert Methode `strftime`, um einen (nach ihrem Ermessen) schönen String mithilfe von Format Codes[4] zu erzeugen.

---

[1] Den Python Code des Servers finden Sie hier: https://github.com/Mari-W/yee-social-network (Wer einen Fehler findet bekommt einen Schokoriegel von uns :)

[2] Für Mac und Windows (ohne WSL) kann der Befehlt abweichen.

[3] bekannt als Unixzeit: https://de.wikipedia.org/wiki/Unixzeit

[4] https://docs.python.org/3/library/datetime.html#strftime-and-strptime-format-codes

(c) Dictionary zu Yeet, 2 Punkt
Schreiben Sie eine Funktion `yeet_from_dict`, die ein Dictionary `yeet_as_dict` als Argument nimmt und ein `Yeet` zurückgibt, falls alle Attribute der Datenklasse `Yeet` mit dem richtigen Typ in `yeet_as_dict` vorkommen. Ist dies nicht der Fall, soll eine Exception (`KeyError`) erzeugt werden. Verwenden Sie hierfür Pattern Matching. Diese Funktion wird später hilfreich, wenn Sie einen `Yeet` als Dictionary vom Server zurückbekommen.

Aufgabe 10.2 (Authentifizierung; 3 Punkte; Datei: `client.py`)

In der vorgegebenen Datei `prelude.py` findet sich eine bereits definierte Datenklasse `Session`. Diese kümmert sich um die Kommunikation mit dem Social Network. Zu Begin des Programms muss eine neue Instanz `session` der Klasse `Session` erzeugt werden. Einmal erstellt, versucht diese sich automatisch zu authentifizieren. Klappt dies nicht, wird das Attribut `authenticated` auf `False` gesetzt. Dann muss die Session mittels der `login`-Methode authentifiziert werden. Die Methode `login` gibt gerade dann `True` zurück, wenn der Loginversuch gültig war. Einmal eingeloggt, kann die Instanz `session` in den späteren Aufgabenteilen benutzt werden, um mit dem Server zu interagieren.

Schreiben Sie eine Funktion `authenticate`, die ein Argument `session` vom Typ `Session` als Argument nimmt. Falls die Session bereits authentifiziert wurde, soll nichts passieren. Ansonsten, soll die Benutzerin nach zwei Eingaben gefragt werden: `username` und `password`. Damit das Passwort bei der Eingabe nicht als Klartext zu sehen ist, sollen Sie hier die Funktion `getpass` aus dem Modul `getpass` anstelle von `input` benutzen. Mit diesen beiden Eingaben soll versucht werden, sich in die Session einzuloggen. Dies können Sie mit der Methode `login` von Session machen. Schlägt der Login fehl, soll die Benutzerin solange zu neuen Eingaben aufgefordert werden, bis der Login erfolgreich war. Informieren Sie die Benutzerin stets mit `print`s über den Status des Anmeldevorgangs. Anmelden können Sie sich mit Ihrem RZ-Account (wie bei Git).

Aufgabe 10.3 (Serverinteraktion; 0 Punkte; Datei: `client.py`)

Die Klasse `Session` stellt zudem die zwei Methoden `get` und `post`[5] bereit, um mit dem Server zu interagieren. Die Methode `get` nimmt einen Pfad (String) als Argument, der bestimmt, was vom Server angefragt werden soll. Entsprechend des Pfads gibt die Methode eine Antwort zurück. Die Methode `post` nimmt einen Pfad und ein passendes Dictionary, um dieses an den Server zu senden. Der Server gibt hier keine Daten zurück. Eine Auflistung der möglichen Pfade finden Sie unter https://courses.laurel.informatik.uni-freiburg.de/yee.
Dieses Webinterface erlaubt es auch, die Pfade auszuprobieren (benutzen Sie dafür den '`Try it out`' Button). Es wird zum Beispiel angezeigt, wie die von `get` zurückgegebenen Daten aussehen, oder welche Form die Daten für `post` beim jeweiligen

[5] Diese Methoden verwenden intern `get` und `post` der `requests`-Library. Die Server-URL wird von der Klasse `Session` angehängt, sodass nur der Restpfad (beispielsweise `/yee/users/all`) angegeben werden muss.

Pfad haben müssen. Spielen Sie vorab ein wenig mit der Funktionalität des Servers um ein Gefühl für das Verhalten zu bekommen!

Aufgabe 10.4 (Command Line Interface; 12 Punkte; Datei: `client.py`)

Bei der Ausführung von Python-Skripten können externe Argumente angegeben werden, indem man diese beim Aufruf über das Terminal mit Leerzeichen getrennt hinter dem Dateinamen anhängt[6]:

```
python3.12 ./client.py arg1 arg2 arg 3
```

Auf diese externen Argumente kann über die Variable `argv` (eine Liste von Strings) aus dem Modul `sys` zugegriffen werden. In dem Beispiel oben ist `argv` gerade:

```
["./client.py", "arg1", "arg2", "arg", "3"]
```

Stellen Sie zu Beginn des Programms mithilfe Ihrer `authenticate` Funktion sicher, dass die `session` authentifiziert ist. Fangen Sie dann die unten folgenden externen Argumente ab und implementieren Sie die jeweilige Funktionalität in einer gleichnamigen Funktion. Sie dürfen natürlich auch beliebige Hilfsfunktionen dafür anlegen.

- `yeet` (1 Punkt): Fordert die Benutzerin auf, mit `input` einen Text einzugeben, und diesen zu yeeten[7].
- `like {id}` (1 Punkt): Setzt einen Like auf den Yeet mit der ID `{id}`
- `unlike {id}` (1 Punkt): Entfernt einen selbst gesetzten Like vom Yeet mit der ID `{id}`
- `details {id}` (1 Punkte): Zeigt an, wer den Yeet mit ID `{id}` geliked hat.
- `follow {user}` (1 Punkt): Folgt der Benutzerin `{user}`
- `unfollow {user}` (1 Punkt): Entfolgt der Benutzerin `{user}`
- `latest {amount}` (1 Punkte): Zeigt die letzten `{amount}` Yeets mit der Anzahl von Likes an.
- `profile` (2 Punkte): Zeigt das eigene Profil an (Yeets, Follower und Benutzerinnen, denen gefolgt wird)
- `feed` (3 Punkte): Zeigt einige (zum Beispiel die beiden letzten) Yeets aller Personen denen Sie folgen, inklusive der Anzahl der Likes an.[8]

Beispiel:
Der Programmaufruf `python3.12 ./client like 5` soll die Funktion `like(5)` aufrufen, die einen Like auf dem Yeet mit der ID 5 setzt.

---

[6] Dieser Befehl kann auf Mac und Windows (ohne WSL) abweichen.

[7] https://c.tenor.com/w96rgWmGS7gAAAAC/tenor.gif; Verbildlichung des Hochladevorgangs

[8] Sie können auch versuchen, die Yeets zeitlich zu sortieren und davon die neusten auszugeben. Seien Sie gerne kreativ.

Aufgabe 10.5 (Soziale Interaktionen; 1 Punkte; Datei: `client.py`)

Jetzt muss das Netzwerk nur noch gefüllt werden. Verfassen Sie mindestens 3 Yeets, folgen Sie mindestens 2 anderen Usern und liken Sie mindestens 3 andere Yeets!

Aufgabe 10.6 (Extra; 0 Punkte; Datei: `client.py`)

Erweitern Sie ihr Command Line Interface um beliebige weitere externe Argumente. Hier ein paar Vorschläge:

- `profile {user}`: Zeigt das Profil der Benutzerin `{user}` an.
- `search`: Fragt die Benutzerin nach einem Text und sucht nach Yeets, die diesen Text beinhalten.
- `search {user}`: Durchsucht die Yeets der Benutzerin `{user}`.
- `date {day}`: Zeigt alle Yeets an, die am Tag `{day}` erstellt wurden.

Wenn Sie besonders motiviert sind, können Sie beispielsweise auch ein Argument `tui` hinzufügen, das mit der `textualize`[9]-Library ein Terminal-User-Interface[10] bereitstellt. Wenn Ihr Programm zusätzliche Libraries verwendet, merken Sie dies bitte in der `NOTES.md` an.

Sie können sich hier beliebig austoben. :) Die besten/coolsten/schönsten Abgaben werden in der Vorlesung gezeigt.

Aufgabe 10.7 (Erfahrungen; 0 Punkte; Datei: `NOTES.md`)

Notieren Sie Ihre Erfahrungen mit diesem Übungsblatt (benötigter Zeitaufwand, Probleme, Bezug zur Vorlesung, Interessantes, etc.).

Editieren Sie hierzu die Datei `NOTES.md` im Abgabeordner dieses Übungsblattes auf unserer Webplatform. Halten Sie sich an das dort vorgegebene Format, da wir den Zeitbedarf mit einem Python-Skript automatisch statistisch auswerten. Die Zeitangabe `6.5 h` steht dabei für 6 Stunden 30 Minuten.

[9] https://textual.textualize.io/
[10] https://en.wikipedia.org/wiki/Text-based_user_interface