Anwendungsentwicklung WI20 – Fallstudie

Problemstellung

Entwerfen und Implementieren Sie ein minimales Chat-System. Es besteht aus einem Chat-Server und einer variablen Anzahl von Chat-Clients.

Szenario

Ein Client meldet sich zunächst beim Chat-System an. Nach erfolgreicher Anmeldung

- kann abgefragt werden, welche anderen Clients derzeit im System angemeldet sind,
- können Nachrichten an angemeldete Clients versendet bzw. von diesen empfangen werden.

Randbedingungen

Alle übertragenen Daten sind Textnachrichten ("Strings"). Die Länge einer Nachricht ist auf 126 Zeichen begrenzt.

Die Anmeldung ist nur für eine Sitzung gültig. Es wird ein beliebiger Name angegeben, der allerdings im System eindeutig sein muss. Ein Login (d. h. die Eingabe eines Passworts) ist nicht erforderlich.

Die persistente Speicherung von Daten im System ist nicht erforderlich:

- Nachrichten an einen Client gehen mit dessen Abmeldung aus dem System verloren.
- Es werden keine Zugangsdaten gespeichert.

Die Implementierung erfolgt in Python. Es ist unter Windows 10 oder Linux (Ubuntu 22.04) lauffähig.

Funktionen (Client)

Der Client greift auf die zentrale Verwaltungskomponente für den Chat (den "Chat-Server") zu.

Der Client kann sich mit einem beliebigen, aber eindeutigen Namen anmelden. Falls der gewählte Name nicht eindeutig ist, wird die Anmeldung abgelehnt und die Eingabe eines neuen Namens angefordert.

Es kann eine Liste der am System angemeldeten Clients abgerufen werden.

Es können Nachrichten an andere Clients, die im System angemeldet sind, versendet werden. Gruppennachrichten sind nicht Teil der Funktionalität, d. h. es wird immer eine Nachricht an genau einen Client versendet.

Es können Nachrichten von allen Clients, die im System angemeldet sind, empfangen werden.

Nachrichten an einen nicht im System angemeldeten Client gehen verloren. Es gibt dazu keine Rückmeldung an den Sender.

Die Auslieferung einer Nachricht an den Empfänger erfolgt unmittelbar (d. h. so schnell wie möglich). Es muss nicht geprüft werden, ob die Nachricht übermittelt werden konnte.

Funktionen (Verwaltung)

Die zentrale Verwaltungskomponente dient zur Anmeldung eines Clients am System. Sie prüft bei der Anmeldung, ob der gewählte Name (s. o.) eindeutig ist.

Typischerweise muss die Verwaltungskomponente auch andere Aufgaben übernehmen (bspw. Routing von Nachrichten an Clients). Diese Aufgaben unterscheiden sich aber je nach gewählter Architektur der Anwendung.

Die Verwaltungskomponente benötigt keine Benutzerschnittstelle.

Dokumentation

Erstellen Sie entsprechend der oben genannten Anforderungen den Use-Case für das System und dokumentieren Sie die nicht-funktionalen Anforderungen.

Beschreiben und skizzieren Sie die Architektur und das Design Ihrer Anwendung in Textform und mit Hilfe von Diagrammen:

- Welche Komponenten gibt es im System und wie arbeiten diese zusammen um die Funktionalität des Programms zu implementieren?
- Welche Typen von Nachrichten werden in Ihrem System versendet und wie erfolgt die Verarbeitung der Antworten auf diese Nachrichten?

Hinsichtlich der Ausführung der Diagramme gibt es keine Vorgaben. Bspw. können Sie zur Illustration UML-Diagramme einsetzen.

Umfang der Dokumentation 6-10 Seiten.

Implementierung

Die Anwendung soll nicht als Web-Anwendung implementiert werden: Der Server soll „selbst gebaut" sein. Der Client kann über ein Kommandozeilen-Interface oder eine GUI bedient werden.

Grundsätzlich dürfen Zusatzbibliotheken verwendet werden, allerdings ist der Einsatz von Bibliotheken, die bereits ein Chat-System implementieren, nicht zulässig.

Die Anwendung muss lokal (Client und Server auf dem selben Rechner) aber auch verteilt (Client und Server auf unterschiedlichen Rechnern) lauffähig sein.

Beispiel: Start des Servers

```
===== Start Chat Server =====
Starting server on 127.0.0.1:1900
Server running
```

Der Server kann (aber muss nicht) Informationen ausgeben. Er kann bei Bedarf mit einer Tastenkombination beendet werden. Siehe auch das Code-Beispiel in `echo_server.py` und

https://docs.python.org/3/library/exceptions.html#KeyboardInterrupt

Beispiel: Start und Schnittstelle des Clients

```
python chat_client.py 127.0.0.1
Choose client ID: Kirk
Client Kirk registered
====== Minimal Chat System ======
1: Send message
2: Check incoming messages
3: Quit
Your selection:
```

Beim Start des Clients wird die IP-Adresse des Servers angegeben. Dies kann, wie hier und im Code-Beispiel `echo_client.py` direkt beim Aufruf als Kommandozeilen-Argument angegeben oder direkt nach dem Start interaktiv eingegeben werden.

Vom Client empfangene Nachrichten müssen nicht sofort angezeigt werden, sondern können im Hintergrund gesammelt und auf Anforderung (hier Menüpunkt `2`) ausgegeben werden:

```
Your selection: 2
Spock: Hello, Spock speaking
====== Minimal Chat System ======
1: Send message
2: Check incoming messages
3: Quit
Your selection:
```

Beim Versenden einer Nachricht werden Empfänger und die Nachricht angegeben, bspw. so:

```
Your selection: 1
Send message to: Spock
Your message: Hello, Kirk speaking
====== Minimal Chat System ======
1: Send message
2: Check incoming messages
3: Quit
Your selection:
```

Beim Beenden des Clients wird dieser vom Server abgemeldet. Noch nicht abgerufene Meldungen werden ausgegeben.

```
Your selection: 3
Old messages:
Spock: To the bridge please
Closing down Session actor
Goodbye
```

Administratives

Die Arbeit wird in Gruppen von je 3 Personen erstellt (Ausnahmen ggf. mit dem Dozenten absprechen).

In der Dokumentation muss aufgeführt sein, wer welche Aufgabe durchgeführt hat.

Abgabe bis zum 17.03.2023 in Moodle als Dokumentation im PDF-Format und eine ZIP-Datei mit Source-Code.

Bei Unklarheiten zu den Anforderungen: Fragen Sie den Kunden (d. h. den Dozenten).

Bewertung

Dokumentation und Implementierung mit gleichem Gewicht in die Bewertung ein.

Kriterien:

- Implementierung der geforderten Features
- Klarheit von Dokumentation und Code. Aussagekraft der Diagramme.

Es wird grundsätzlich eine Gruppennote vergeben, wobei je nach Aufgabenverteilung (s. o.) innerhalb der Gruppe noch differenziert werden kann.

Literatur

Zur ersten Orientierung:

- Beispiel in Moodle
- https://docs.python.org/3/howto/sockets.html
- https://docs.python.org/3/library/socket.html#socket-objects
- https://de.wikipedia.org/wiki/Socket_(Software)