Chat mit Bot

In dieser Aufgabe soll ein Chatprogramm mit integrierter Bot-Funktion entwickelt werden. Die Teilnehmenden sollen in der Lage sein, miteinander zu chatten, während der Bot zufällige technische Fragen in den Chat stellt, die von den Anwendenden beantwortet werden können. Die Programmiersprache ist frei wählbar und als UI kann ein Web-Frontend oder die Konsole verwendet werden. Die Kommunikation zwischen den Clients kann mit verschiedenen Technologien wie Web-Sockets, Queues (RabbitMQ), MQTT oder einer anderen Producer/Consumer-Möglichkeit erfolgen. Das Projekt besteht aus den Teilen Planung, Umsetzung, technischer Dokumentation und Präsentation. Jede*r Teamteilnehmende wird eine Teilaufgabe des Projekts übernehmen, diese umsetzen und das Ergebnis am Ende präsentieren.

Projektablauf

1. **Planung**: In der Planungsphase sollten die Auszubildenden das Projekt in Teilaufgaben unterteilen und jedem Teammitglied eine Aufgabe zuweisen. Mögliche Teilaufgaben könnten die Entwicklung des Chat-UI, die Implementierung der Bot-Funktion, die Einrichtung der Kommunikation zwischen den Clients und die Erstellung der technischen Dokumentation sein.
2. **Umsetzung**: In der Umsetzungsphase sollten die Auszubildenden ihre zugewiesenen Teilaufgaben umsetzen. Dabei können sie die Programmiersprache und die Technologien (z.B. Web-Sockets, Queues, MQTT) frei wählen. Wichtig ist, dass die verschiedenen Komponenten des Projekts am Ende zusammenarbeiten.
3. **Dokumentation**: Die technische Dokumentation sollte alle wichtigen Informationen über das Projekt enthalten, wie z.B. die verwendeten Technologien, die Architektur des Systems und die Funktionsweise der verschiedenen Komponenten.
4. **Präsentation**: In der Präsentationsphase sollten die Auszubildenden das Ergebnis ihres Projekts vorstellen und erklären, wie sie ihre Teilaufgaben umgesetzt haben.

Hinweise für die Umsetzung

Python

1. **Backend**: Für das Backend des Chatprogramms können Sie ein Python-Webframework wie Flask oder Django verwenden. Diese Frameworks bieten eine einfache Möglichkeit,

eine REST-API zu erstellen, über die Clients Nachrichten senden und empfangen können.

2. **Kommunikation zwischen Clients**: Für die Kommunikation zwischen den Clients können Sie verschiedene Technologien wie WebSockets, Queues (z.B. RabbitMQ) oder MQTT verwenden. Es gibt mehrere Python-Bibliotheken, die die Verwendung dieser Technologien vereinfachen, wie z.B. die `websockets`-Bibliothek für WebSockets, `pika` für RabbitMQ oder `paho-mqtt` für MQTT.

3. **Bot-Funktion**: Für die Bot-Funktion können Sie eine separate Python-Anwendung erstellen, die sich als Client mit dem Chatprogramm verbindet und zufällige technische Fragen in den Chat stellt. Die Fragen können entweder vordefiniert sein oder dynamisch aus einer Datenbank oder einem anderen Datenquellen generiert werden.

4. **Frontend**: Für das Frontend des Chatprogramms können Sie entweder ein Web-Frontend oder eine Konsolenanwendung erstellen. Ein Web-Frontend kann mit HTML, CSS und JavaScript erstellt werden, während eine Konsolenanwendung mit einer Python-Bibliothek wie `curses` oder `PyInquirer` erstellt werden kann.

Java

1. **Backend**: Für das Backend des Chatprogramms können Sie ein Java-Webframework wie Spring oder JavaServer Faces (JSF) verwenden. Diese Frameworks bieten eine einfache Möglichkeit, eine REST-API zu erstellen, über die Clients Nachrichten senden und empfangen können.

2. **Kommunikation zwischen Clients**: Für die Kommunikation zwischen den Clients können Sie verschiedene Technologien wie WebSockets, Queues (z.B. RabbitMQ) oder MQTT verwenden. Es gibt mehrere Java-Bibliotheken, die die Verwendung dieser Technologien vereinfachen, wie z.B. die `javax.websocket`-API für WebSockets, `RabbitMQ Java Client` für RabbitMQ oder `Eclipse Paho` für MQTT.

3. **Bot-Funktion**: Für die Bot-Funktion können Sie eine separate Java-Anwendung erstellen, die sich als Client mit dem Chatprogramm verbindet und zufällige technische Fragen in den Chat stellt. Die Fragen können entweder vordefiniert sein oder dynamisch aus einer Datenbank oder einem anderen Datenquellen generiert werden.

4. **Frontend**: Für das Frontend des Chatprogramms können Sie entweder ein Web-Frontend oder eine Konsolenanwendung erstellen. Ein Web-Frontend kann mit HTML, CSS und JavaScript erstellt werden, während eine Konsolenanwendung mit einer Java-Bibliothek wie `Lanterna` oder `JLine` erstellt werden kann.

Node.js (JavaScript)

1. **Backend**: Für das Backend des Chatprogramms können Sie ein Node.js-Webframework wie Express oder Koa verwenden. Diese Frameworks bieten eine einfache Möglichkeit, eine REST-API zu erstellen, über die Clients Nachrichten senden und empfangen können.

2. **Kommunikation zwischen Clients**: Für die Kommunikation zwischen den Clients können Sie verschiedene Technologien wie WebSockets, Queues (z.B. RabbitMQ) oder MQTT verwenden. Es gibt mehrere Node.js-Bibliotheken, die die Verwendung dieser Technologien vereinfachen, wie z.B. die `ws` -Bibliothek für WebSockets, `amqplib` für RabbitMQ oder `mqtt` für MQTT.

3. **Bot-Funktion**: Für die Bot-Funktion können Sie eine separate Node.js-Anwendung erstellen, die sich als Client mit dem Chatprogramm verbindet und zufällige technische Fragen in den Chat stellt. Die Fragen können entweder vordefiniert sein oder dynamisch aus einer Datenbank oder einem anderen Datenquellen generiert werden.

4. **Frontend**: Für das Frontend des Chatprogramms können Sie entweder ein Web-Frontend oder eine Konsolenanwendung erstellen. Ein Web-Frontend kann mit HTML, CSS und JavaScript erstellt werden, während eine Konsolenanwendung mit einer Node.js-Bibliothek wie `Inquirer.js` oder `Vorpal` erstellt werden kann.