SE-Praktikum
Torben Könke
TeamNahme
Version: **0.2**

Aufgabenblatt 5

1. Projektvorbereitung

Kurze, aussagefähige Darstellung des Projekts

Es soll ein Chatprogramm auf Basis des *XMPP-Protokolls*[1] entwickelt werden. Wir haben uns für das XMPP-Protokoll entschieden, da dieses im Gegensatz zu vielen anderen Chat-Protokollen (ICQ, AIM usw.) quelloffen ist und entwickelt wird und keinen Lizenzgebühren oder anderen Restriktionen unterliegt. Außerdem bietet das XMPP-Protokoll neben der grundlegenden Chat-Funktionalität verschiedene Erweiterungen, die optionalerweise implementiert werden können, je nach dem wie gut das Projekt voranschreitet; Hierzu gehören u.a. Kontaktlisten, Dateiübertragung, Videoübertragung und VOIP[2].

Liste der Anwendungsfälle

- Senden und Empfang einer Nachricht.
- Setzen einer Statusnachricht bei einem speziellen Anbieter, welcher in der Statusnachricht verschiedene andere Informationen kodiert.
- Senden und Empfang während einer verschlüsselten Sitzung.

Begründung der Machbarkeit

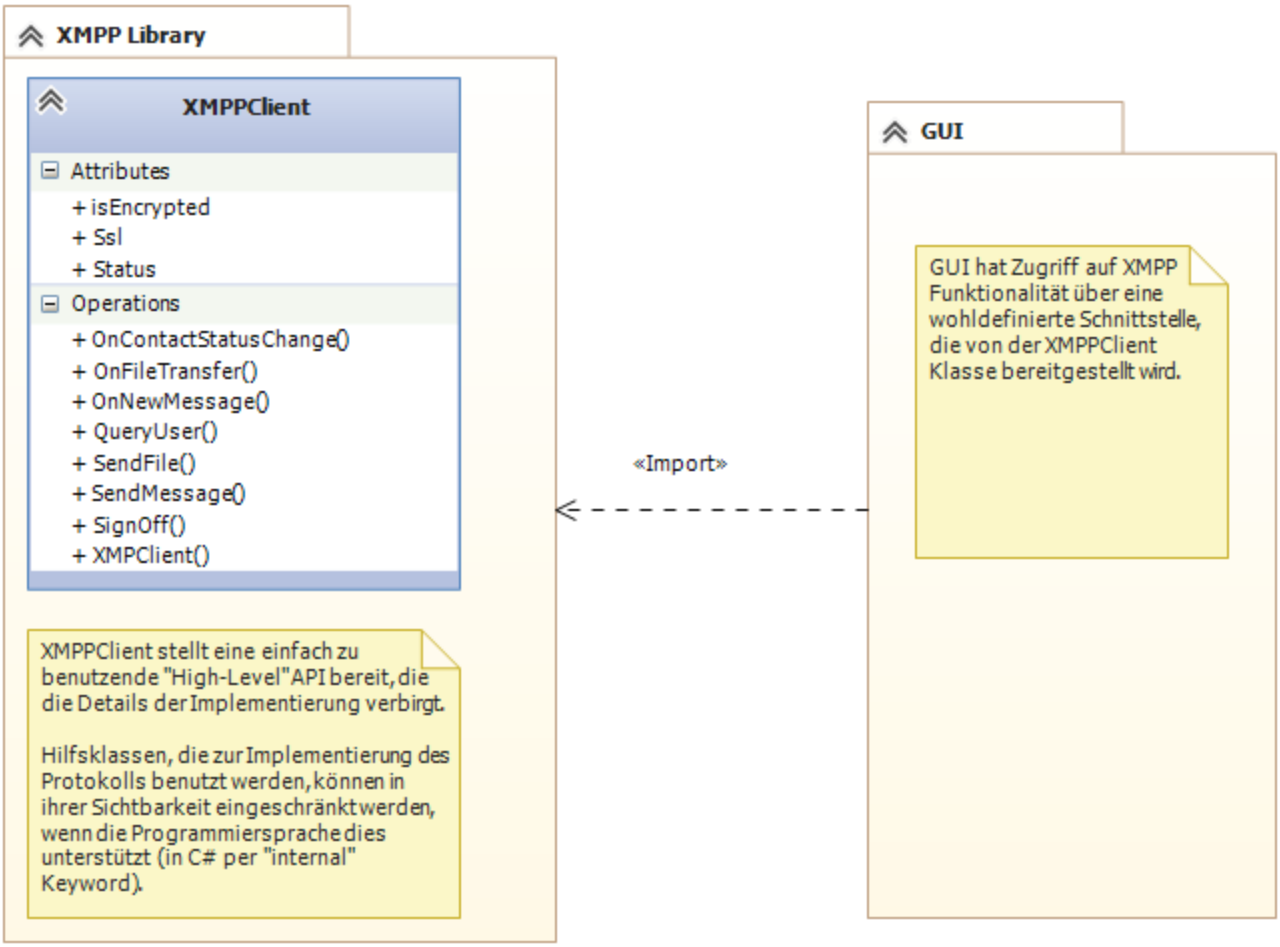
Da das XMPP-Protokoll quelloffen ist, ist es komplett über RFC (*Request for comments*) spezifiziert. Es müssen also keine Protokoll über Reverse Engineering entschlüsselt werden oder ähnliches (wie dies beim *OSCAR* Protokoll von ICQ der Fall ist). Dies vereinfacht die Entwicklung natürlich enorm.

Das Team hat (zum. teilweise) Erfahrung mit der Implementierung von RFC-Standards (IMAPv4, POP3, SASL) und Netzwerkprogrammierung. Das Projekt lässt sich zudem gut unterteilen, die GUI kann komplett von der Implementierung des XMPP-Protokolls getrennt werden, so daß die

[1] Extensible Messaging and Presence Protocol (XMPP): Core, IETF, RFC 6120,
XMPP: Instant Messaging and Presence, IETF, RFC 6121
[2] XMPP Protocol Extension XEP-0166: Jingle, XMPP Standards Foundation, Draft 1.1

Teammitglieder je nach persönlichem Interesse und Fähigkeiten an unterschiedlichen Teilen des Projekts arbeiten können. Die Implementierung des XMPP-Protokolls kann hierbei als eine Bibliothek erfolgen, die eine API bereitstellt, welche dann von der grafischen Oberfläche aufgerufen werden kann.

Eine erste und sehr grobe Übersicht über die grundlegende Architektur der Implementierung könnte wie folgt ausehen:



Erläuterung des Diagramms:

Die Klasse XMPPClient kapselt die grundlegende XMPP-Funktionalität und stellt eine sinnvolle Schnittstelle zur Verfügung, über die die verschiedenen Funktionen in Anspruch genommen werden können. Attribute wie *isEncrypted* und *Ssl* geben darüber Aufschluss, ob die Sitzung verschlüsselt per *SASL*[3] verschlüsselt ist, oder ob die Verbindung über *SSL*[4] gesichert ist.

[3] Simple Authentication and Security Layer, Network Working Group, RFC 4422

Die Methoden der Klasse sollten möglichst selbsterklärend gehalten werden, so kann eine Chat-Nachricht über die Methode *SendMessage* gesendet werden (XMPP: *Stanza Exchange*[5]), ein Dateitransfer über *SendFile* initiiert werden. In dem obigen Modell stellen die Methoden, die mit "*On*" beginnen, Events dar. Im Falle von Javascript lassen sich diese einfach mit Hilfe des Observer *Beobachtungsmusters*[6] implementieren. Die GUI würde sich also bei diesen Events anmelden und darüber informiert werden, wenn ein Event auftritt und kann entsprechend reagieren.

[4] The Secure Sockets Layer (SSL) Protocol, IETF, RFC 6101
[5] Extensible Messaging and Presence Protocol (XMPP): Core, IETF, RFC 6120, Section 9.1.4
[6] siehe http://en.wikipedia.org/wiki/Observer_pattern