

Dokumentation

Matthias Vonend Jan Grübener Patrick Mischka
Michael Angermeier Troy Keßler Aaron Schweig

25. März 2020

Inhaltsverzeichnis

1	Basisanforderungen	1
2	Erweiterungen	2
2.1	Grafische Oberfläche bei den Nutzern	2
2.2	Verwendung von Emojis	2
2.3	Mehrere Chatverläufe pro Nutzer	2
2.4	Persistentes Speichern der Chatverläufe	2
2.5	Verschlüsselte serverseitige Speicherung der Chats	2
2.6	Gruppenchats	2
2.7	Verschlüsselte Übertragung der Chat-Nachrichten	2
3	Anhang	3

1 Basisanforderungen

Test

Die Anwendung soll //TODO

Aus diesem Grund wird die Anwendung von mehreren Gleichwertigen Servern bedient. Jeder dieser sog. Nodes hat dabei eine vollständige Kopie sämtlicher dem System bekannten Daten. Eine Veränderung des Datenbestandes muss dem entsprechend an alle anderen Nodes weiter gegeben.

Jeder Client kann sich zu einer Node verbinden. Möchte dieser eine Nachricht an einen weiteren Client senden, schickt er diese an seine verbundene Node und überlässt die Zustellung dieser. Da alle Nachrichten aus Persistenzgründen an alle Nodes verteilt werden müssen, brauchen die Nodes keine Information über die Clients anderer Nodes. Im Falle einer solchen Anforderung (z. B. Abfrage ob ein anderer Nutzer aktiv ist) könnte ein Protokoll ähnlich zu Routing-Tabellen implementiert werden. Empfängt eine Node eine Nachricht, egal ob von einem Client oder von einer anderen Node, überprüft diese ob die Nachricht für einen ihr bekannten Client bestimmt war und sendet diese gegebenenfalls an diesen.



Abbildung 1: Architecture

2 Erweiterungen

- 2.1 Grafische Oberfläche bei den Nutzern
- 2.2 Verwendung von Emojis
- 2.3 Mehrere Chatverläufe pro Nutzer
- 2.4 Persistentes Speichern der Chatverläufe
- 2.5 Verschlüsselte serverseitige Speicherung der Chats
- 2.6 Gruppenchats
- 2.7 Verschlüsselte Übertragung der Chat-Nachrichten

3 Anhang

```

40  @Override
    public void run() {
        Thread.currentThread().setName("Connection Handler");
        while (!this.context.isCloseRequested().get()) {
            System.out.println("Handling");
            try {
                var object = inputStream.readObject();
                var packet = (Packet) object;
                this.handlePacket(packet);
            } catch (IOException | ClassNotFoundException e) {
                break;
            }
50  }
        try {
            this.client.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
55  }
    }

    private void handlePacket(final Packet packet) throws IOException
    {
60  for (var listener : this.context.getListeners()) {
            try {
                var methods = listener.getClass().getDeclaredMethods();
                for (var method : methods) {
                    if (method.getName().equals("next") && !method.
isSynthetic()) {
65  var packetType = method.getParameters()[0];
                            if (packetType.getType().isAssignableFrom(packet.
getClass())) {
                                var retu = (Packet) method.invoke(listener, packet,
this.context, this);
                                this.pushTo(retu);
70  }
                            }
                    }
                }
            } catch (SecurityException | IllegalArgumentException |
IllegalAccessException
                | InvocationTargetException e) {
75  e.printStackTrace();
            }
        }
    }

```

../src/main/java/vs/chat/server/ConnectionHandler.java

Test