Bericht – Stefan Englmeier

Im Sinne der Individualaufgabe des Wahlpflichtfaches *Single Page Anwendungen mit TypeScript und Angular* möchte ich im folgenden unser Team-Projekt näher beschreiben.

Wir haben uns als Team für den Chat-Client entschieden, da kein weiteres Backend vorhanden, dass unseren Client mit Daten hätte versorgen können. Für die Versionierung und Quellcodeverwaltung entschieden wir uns für *git*, da wir alle hier bereits Vorkenntnisse hatten. Gehostet wurde dies auf Alex Tarasovs privatem Server und als Backup auf *github.com*. Da ich mich im Praxissemester schon ausgiebig mit Angular 2 auseinandergesetzt hatte, nahm ich es mir zur Aufgabe, die grobe Projekt- und Komponentenstruktur aufzubauen. Auf oberster Ebene steht die Root-Komponente, die sich hier in *app.component.ts* befindet. Im *AppModule* definieren wir, welche Komponente *gebootstrapped* wird. In der *main.ts* Datei deklarieren wir unser initial gestartetes Modul. Das Template der *AppComponent* enthält nur das Router-Outlet. In diese werden, je nach URL, die entsprechenden Komponenten *gerendert*. Für das Routing war Alex Tarasov zuständig. Unter der *AppComponent* stehen die *LoginComponent*, *RegistryComponent*, *Greeting* und *ChatComponent*. In der *ChatComponent* befinden sich drei weitere Komponenten. Die *ChatRoomDisplayComponent*, die die Liste der vorhandenen Chat-Rooms enthält. Die *ChatMessageHistory* mit dem Nachrichtenverlauf, dem Inputfeld und Sendebutton. Und schließlich die *Chat-Participants*, welche die Teilnehmer des aktiven Chat-Rooms aufzeigen. Was die Services betrifft, haben wir uns für einen Authentication und Websocket Service entschieden. Der Erste verwaltet klassische Authentifizierungsfunktionalitäten wie Log-in/out, Namen und Password ändern. Der Letztere stellt die Websocket Verbindung des Clients zum *Backend* her, sendet Events an den Server und hört auf eingehende Events. Die Klasse *WebsocketService* enthält zusätzlich diverse Subjekte und

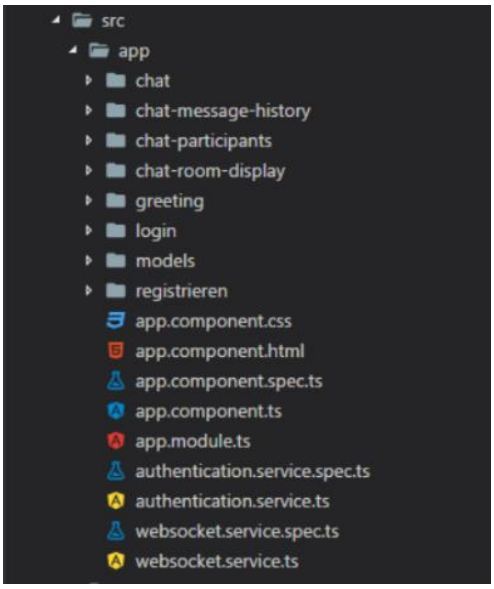

Abbildung 1 Projektstruktur

```
@Injectable()
export class WebsocketService {

  loginFailed : Subject<boolean> = new Subject<boolean>()
  loginFailedObservable: Observable<boolean> = this.loginFailed.asObservable()

  loginSuccess : Subject<any> = new Subject<any>()
  loginSuccessObservable: Observable<any> = this.loginSuccess.asObservable()
```

Abbildung 2 Ausschnitt WebsocketService Klassenvariablen

Observables. Diese stammen von der *Libary RxJs*, welche standardmäßig von Angular 2 verwendet wird. Die Subjekte dienen als *EventEmitter*. Komponenten sind nun in der Lage bei der Initialisierung sich auf die *Observables* zu *subscriben*. Dabei können mehrere *Callback* Funktionen übergeben werden. Alex arbeitete mit Skizzen und Implementierungen am UI, speziell der *CSS* und das

Layouting. Konzentrierten sich Benjamin Klam und ich auf die Business-Logik der einzelnen Komponenten und Services. Die *ChatComponent* dient als zentrale Sammelstelle der Daten. Die Komponente hatte mehrere *Subscriptions* zu handlen und informierte mit *Angulars Core* Feature *Input* die darunterliegenden Komponenten über Änderungen. Dazu gehören Updates zu Daten an beigetretenen Chat-Rooms wie eingehende Nachrichten oder Teilnehmer.

Abschließend möchte ich sagen, dass wir sowohl selbstorganisiert als auch als Gruppe gearbeitet haben. Alle waren beschäftigt und jeder war stets seiner Aufgabe bewusst.