**Anwendungsentwicklung - Chat**

|  |
| --- |
| Portfolio Anwendungsentwicklung  Studiengang 2018  Kurs C |
| Fakultät Wirtschaft  Studiengang Wirtschaftsinformatik |
| Duale Hochschule Baden-Württemberg  Villingen-Schwenningen |

|  |  |  |
| --- | --- | --- |
|  | | |
| Bearbeiter:  David Bährens  Dominik Frank  Julian Gommlich  Leon Haase |  | Betreuender Dozent:  Prof. Dr. Wolfgang Funk |

**Inhaltsverzeichnis**

**Seite**

[Abkürzungsverzeichnis III](#_Toc67496466)

[Abbildungsverzeichnis IV](#_Toc67496467)

[Tabellenverzeichnis V](#_Toc67496468)

[Anhangsverzeichnis VI](#_Toc67496469)

[1 Einführung & Ziele 1](#_Toc67496470)

[2 Qualitätsziele für die Architektur 1](#_Toc67496471)

[2.1 Fachliche Perspektive 2](#_Toc67496472)

[2.2 Technische Perspektive 3](#_Toc67496473)

[2.2.1 Bausteinschicht 3](#_Toc67496474)

[2.2.2 Laufzeitenschicht 4](#_Toc67496475)

[2.2.3 Verteilungsschicht 5](#_Toc67496476)

[2.3 Klassendiagramm 6](#_Toc67496477)

[2.4 Datenbank (in-memory) – ERM 7](#_Toc67496478)

[2.5 Architekturentscheidungen 7](#_Toc67496479)

[3 Technische Schulden 8](#_Toc67496480)

[4 Kritische Codestellen 8](#_Toc67496481)

[5 Projekttabelle 9](#_Toc67496482)

[Anhang 10](#_Toc67496483)

[Erklärung 13](#_Toc67496484)

Abkürzungsverzeichnis

API Application programming interface

DB Datenbank

ERM Entity-Relationship-Modell

REST Representational State Transfer

Abbildungsverzeichnis

**Seite**

[Abb. 1: Use Case Diagramm Chat-Anwendung 3](#_Toc67496485)

[Abb. 2: Bausteinansicht 4](#_Toc67496486)

[Abb. 3: Laufzeitschicht 5](#_Toc67496487)

[Abb. 4: Verteilungsschicht 6](#_Toc67496488)

[Abb. 5: Klassen 6](#_Toc67496489)

[Abb. 6: ERM 7](#_Toc67496490)

Tabellenverzeichnis

**Seite**

Tab. 1: Qualitätsziele der Architektur 2

Tab. 2: Zuständigkeitstabelle 9

Anhangsverzeichnis

**Seite**

[Anhang I: Technisch einwandfreies ERM 10](#_Toc67496493)

[Anhang II: Adobe XD - Anmeldemaske 10](#_Toc67496494)

[Anhang III: Adobe XD – Hauptmenü Chatroom 11](#_Toc67496495)

[Anhang IV: Adobe XD – Hauptmenü Chatroom 11](#_Toc67496496)

[Anhang V: Adobe XD – Hauptmenü Aktive Teilnehmer 12](#_Toc67496497)

[Anhang VI: Adobe XD – Hauptmenü Abmelden 12](#_Toc67496498)

1. Einführung & Ziele

Im Rahmen des 5. Semesters ist die Aufgabe im Modul „Anwendungsentwicklung“ einen kleinen Chatraum zu programmieren. In diesem soll es möglich sein, dass zwei Personen mit einander kommunizieren können. Die spezifisch geforderten Funktionalitäten des Chats sind, das mehrere Clients in einem lokalen Netzwerk oder mit einer lokalen Maschine miteinander kommunizieren.

In dieser Dokumentation wird erörtert, welche Komponenten gibt es in einem solchen Server-Client orientierten System und wie Arbeiten diese zusammen, um die Funktionalitäten des Programms erfolgreich zu implementieren. Zusätzlich soll die Frage geklärt werden, welche Typen von Nachrichten im System versendet werden und wie die Verarbeitung der Antworten auf diese Nachrichten erfolgen soll. In Kapitel zwei werden die Qualitätsziele für die Architektur festgelegt, bevor Kapitel drei die technische Umsetzung beschreibt. Eine kurze Rezension dieser Gruppenarbeit wird diese Dokumentation abschließen.

1. Qualitätsziele für die Architektur

Im Rahmen der Ausarbeitung haben sich vier wesentliche Qualitätsziele für die Architektur herausgestellt. Die Benutzbarkeit, die Chatanwendung sollte sowohl von mehreren Clients auf einer lokalen Maschine als auch von mehreren Clients in einem lokalen Netzwerk nutzbar sein. Die Lösung hierfür ist eine Client-Server Architektur, die keine monolithische Anwendung ist. Das zweite Qualitätsziel, ist eine möglichst hohe Leistungseffizienz des Programmes zu erreichen. Die Auslieferung einer Nachricht an den Empfänger soll sofort erfolgen. Die Latenzzeit darf höchstens eine Sekunde betragen. Als Lösungsansatz wurde hier das minimieren der Verarbeitung der Nachrichten festgelegt. Zusätzlich wird die Latenz getestet, um sicher in der Anforderungszeit zu liegen. Ein weiteres Ziel ist die Schnittstellenspezifikation zwischen dem Backend und dem Frontend. Die Anwendung wird mit einer RESTfulAPI entwickelt, das bedeutet Client zu Server basiert. Zudem wird ein WebSocket mit der Abfolge Client zu Server zu Client implementiert, um einen reibungslosen Ablauf zu gewährleisten. Das vierte Qualitätsziel ist die zuverlässige Ansprache des Servers und die Datenintegrität. Die Nachrichten müssen trotz der asynchronen Kommunikationsorganisationsform zuverlässig versendet und empfangen werden. Um dies sicherzustellen werden die Schnittstellen zum Server getestet, die Nachrichten dürfen hierbei nicht verloren gehen. Es wird also persistiert und getestet. Tabelle 1 Qualitätsziele der Architektur fasst die oben beschriebenen Ziele und Lösungen nochmals zusammen.

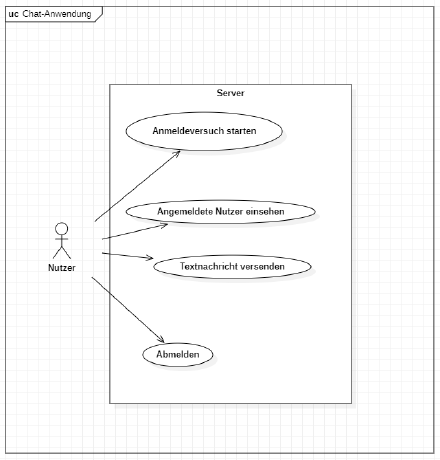
Tab. 1: Qualitätsziele der Architektur



* 1. Fachliche Perspektive

Um eine fachliche Perspektive auf das Chatprogramm zu erzeugen, zeigt Abbildung 1 ein Use Case Diagramm. Use Case Diagramme stellen und beschreiben das Systemverhalten aus Anwendersicht. Dadurch wird ein Verständnis der fachlichen Anforderungen durch die Entwickler erzeugt. Das Use Diagramm Chat-Anwendung zeigt, das ein Nutzer mit dem Server interagiert. Dabei versucht der Nutzer einen Anmeldeversuch zu starten. Ist der Anmeldeversuch erfolgreich, kann er alle aktuell angemeldete Nutzer sehen und hat die Möglichkeit, eine Textnachricht zu versenden. Nach Befriedigung seines Bedürfnisses, sich mit anderen Nutzern auszutauschen, kann der Nutzer sich wieder abmelden.

Abb. 1: Use Case Diagramm Chat-Anwendung



Quelle: Eigene Darstellung

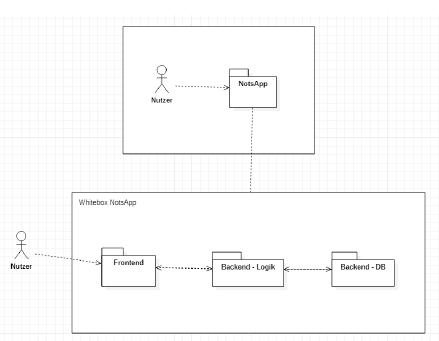
* 1. Technische Perspektive

Im Folgenden wird die Chat-Anwendung aus der technische Perspektive dargestellt.

* + 1. Bausteinschicht

Wie in den vorherigen Kapiteln herausgearbeitet, besteht das Programm aus mehreren Bausteinen, die im Rahmen einer Microservice-Architektur miteinander verknüpft werden. Im Falle der Chat-Anwendung wird ein Frontend für die Nutzer eingerichtet und ein Backend für die logische Weiterleitung der Nachrichten. Im Frontend wird das Design der Oberfläche umgesetzt. Außerdem werden sämtliche Funktionen implementiert, die dazu dienen, die Usability der Website zu erhöhen. Im Backend werden die aktuell im Chat geschriebenen Daten gespeichert, die allerdings nach Beendigung der Anwendung wieder gelöscht werden. Abbildung 2 verdeutlicht die Trennung der einzelnen Bausteine.

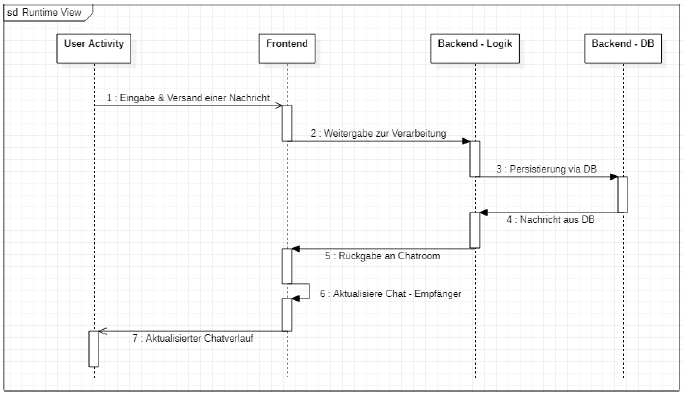
Abb. 2: Bausteinansicht



* + 1. Laufzeitenschicht

Eine Laufzeitenschicht beschreibt Abläufe, die durch eine Interaktion mit dem Nutzer angestoßen werden. Abbildung 3 verdeutlicht den Hauptprozess ausgehenden vom Nutzer. Ein Chatraum wird durch eine User Aktivität mit der Eingabe und dem Versand einer Nachricht gestartet. Anschließend gibt das Frontend die Daten an das Backend weiter. Dieses nimmt anschließend das Persistieren der Daten via Datenbank vor. Im vierten Schritt wird die Nachricht aus der Datenbank zurückgesendet. Nun kann das Backend die Nachricht an den Chatraum zurückgeben. Das Frontend aktualisiert, sofern notwendig, den Chatraum. Der aktualisierte Chatverlauf wird dem Nutzer anzeigt. Die Anwendung hat weitergehend, den Anmeldeprozess und den Abmeldeprozesse. Diese Prozesse sind Nebenprozesse und werden deshalb nicht visualisiert dargestellt.

Abb. 3: Laufzeitschicht



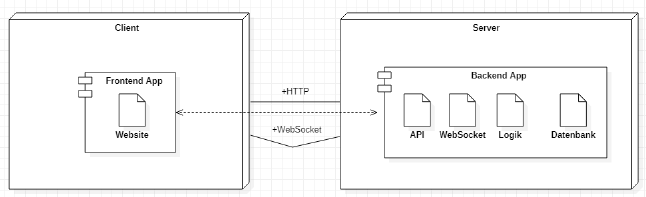
* + 1. Verteilungsschicht

Die Verteilungsschicht zeigt einen technischen Blick auf die Konzeptionierte Anwendung. Es werden die Bausteine unter dem Aspekt ihrer zugrundeliegenden Technologien sowie deren Deployment betrachtet.

Das Web-Frontend des Chatraums wird unter Anwendung des Angular 2-Frameworks entwickelt. Es werden die Skriptsprachen HTML und CSS eingesetzt, um die die Benutzeroberfläche zu strukturieren und zu gestalten.

Zur Implementierung des Backends wurde die Programmiersprache Java mit dem Spring-Boot-Framework eingesetzt. Konzeptionell lässt sich dieser Baustein in zwei Teilbereiche gliedern. Die Schnittstelle und die Geschäftslogik. Die Schnittstelle für die Kommunikation des Clients wird mittels RESTfulAPI (HTTP) umgesetzt. Während die Geschäftslogik mit dem WebSocket implementiert ist. Die Datenbank ein in-memory Anwendung, d. h. die Daten werden nach der Session wieder gelöscht. In Abbildung 4 werden die Verteilungsschichten visualisiert verdeutlicht.

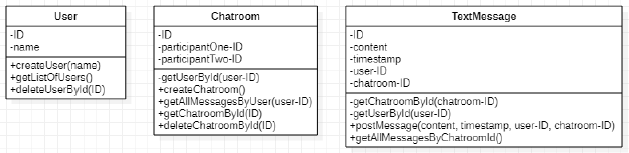
Abb. 4: Verteilungsschicht



* 1. Klassendiagramm

Das Klassendiagramm stellt die Klassen, die Vererbungsstruktur und die Beziehung zwischen Klassen dar. Abbildung 5 zeigt die Klassen User, Chatroom und TextMessage. Alle drei Klassen besitzen den Primärschlüssel ID. Die User kommunizieren via Chatroom miteinander. Alle Attribute sind vom Typ private, damit diese nur aus der Klasse/via Getter aufgerufen werden können. Public-Methoden werden mit der REST-API-Schnittstelle angesprochen werden. Private-Methoden hingegen sind solche, welche im BCE-Patter der Controller zur Erstellung des Models verwendet werden, d. h. Entitäten benötigen.

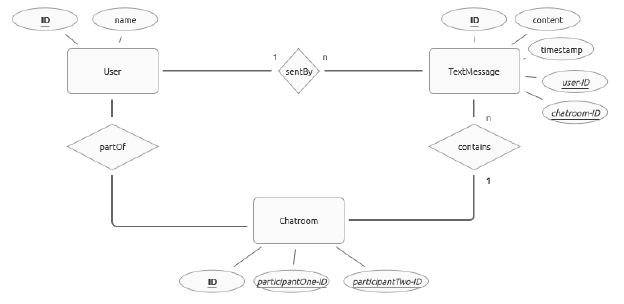
Abb. 5: Klassen



* 1. Datenbank (in-memory) – ERM

Im Backend ist eine in-memory Datenbank initialisiert, d. h. die Daten sind nach einer Session wieder gelöscht. Aus dem technisch einwandfreien ERM ergeben sich vier Tabelle. Wie in Kapitel 2.3 die Tabellen User, Chatroom und TextMessage. Außerdem ergibt sich eine Tabelle aus der n zu m Beziehung des Users mit dem Chatroom (Anhang eins). Um eine Anforderungsadäquatere und performanceverbesserte Anwendung zu entwickeln, muss auf diese Tabelle verzichtet werden. Abbildung 6 zeigt das zum Einsatz kommende ERM, damit das Leistungsziel der Latenzzeit kleiner einer Sekunde umgesetzt werden kann. Die Beziehung User zu TextMessage, ist eine eins zu n Beziehung. Das bedeutet, ein User kann beliebig viele Nachrichten senden, eine Nachricht ist genau einem User zuzuordnen. Die Kardinalität zwischen der TextMessage und dem Chatroom ist ein n zu eins. In einem Chatroom sind beliebig viele Nachrichten enthalten, eine Nachricht kann immer einem Chatroom zugeordnet werden.

Abb. 6: ERM



* 1. Architekturentscheidungen

Auf Grundlage der zuvor ausgearbeiteten Aspekte werden in diesem Abschnitt die Architekturentscheidungen zusammengefasst. Die ERM Beziehungen zwischen dem Chatroom und dem User wurde aufgrund der Performance Verbesserung nicht mit der technisch einwandfreien Lösung aufgelöst. Es kommt eine Memory DB zum Einsatz, das bedeutet nach Abmeldung des Users sind alle Nachrichten wieder gelöscht. Die Nutzer selbst kommunizieren in Chatrooms, die sich mit dem Websocket verbinden. Ein Websocket besitzt im Gegenteil zu einer gewöhnlichen API den Vorteil, dass an sämtliche Clients eine Nachricht versendet werden kann. Das Frontend sortiert den Chat und entscheidet über die Aktualisierung, damit muss das Backend nicht jedes Mal den ganzen neuen Chatverlauf senden. Dadurch erreicht die Anwendung eine höhere Geschwindigkeit.

1. Technische Schulden

Um ein umfassendes Bild über das entwickelte System zu geben, müssen die technischen Schuleden erfasst werden.

* Die Nachrichten im Frontend, bleiben bei der Latenz ggf. nicht in der richtigen Reihenfolge.

(weitere noch ausstehend)

1. Kritische Codestellen

(noch ausstehend)

1. Projekttabelle

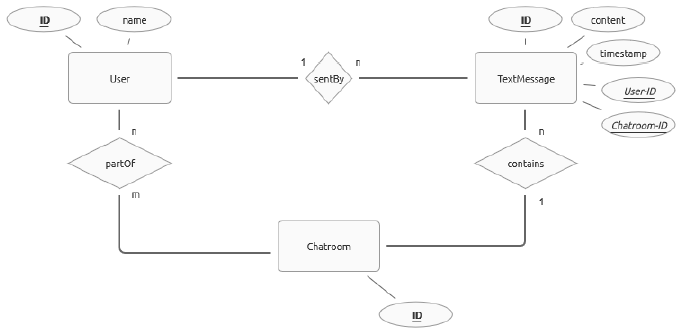
Tabelle 7 zeigt die Zuständigkeiten innerhalb der Projektumsetzung. Der Fettdruck weißt jeweils die Hauptverantwortlichkeit für diese Aufgabe aus. Es gilt herauszustellen, dass die komplette Gruppe durch regelmäßige Teamabsprachen immer auf dem aktuellen Stand des Projektfortschritts war.

Tab. 2: Zuständigkeitstabelle

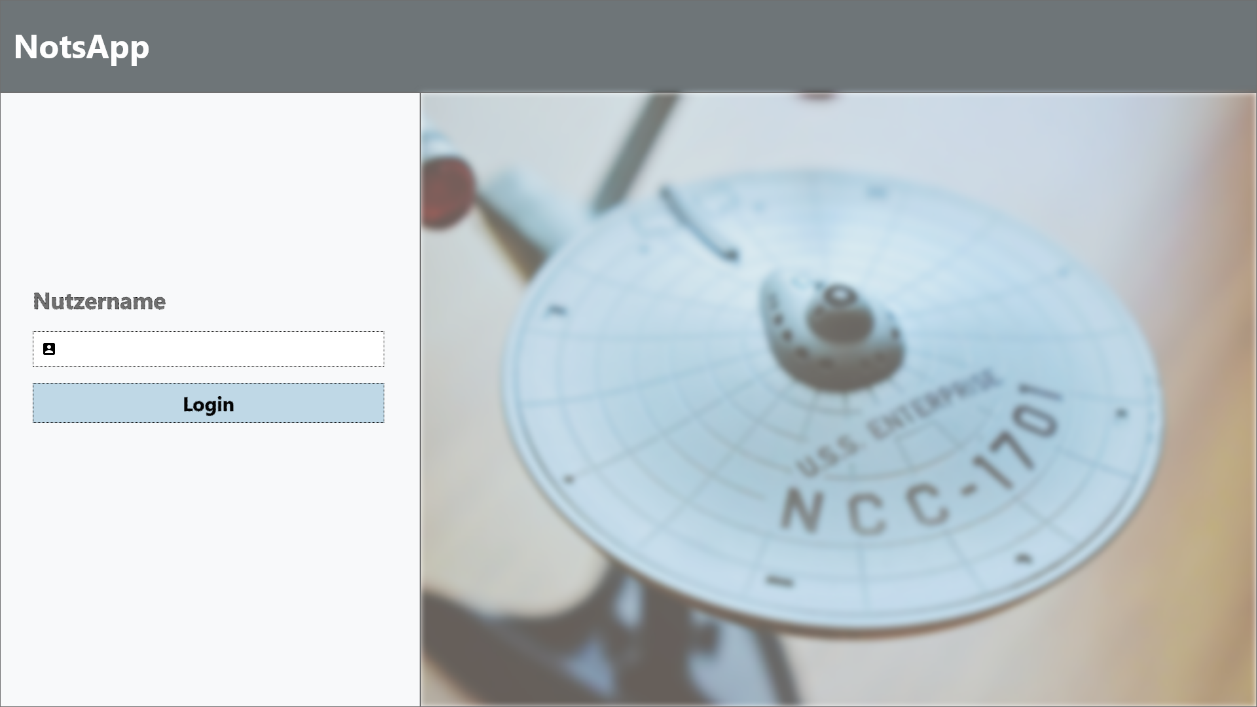


Anhang

Anhang I: Technisch einwandfreies ERM



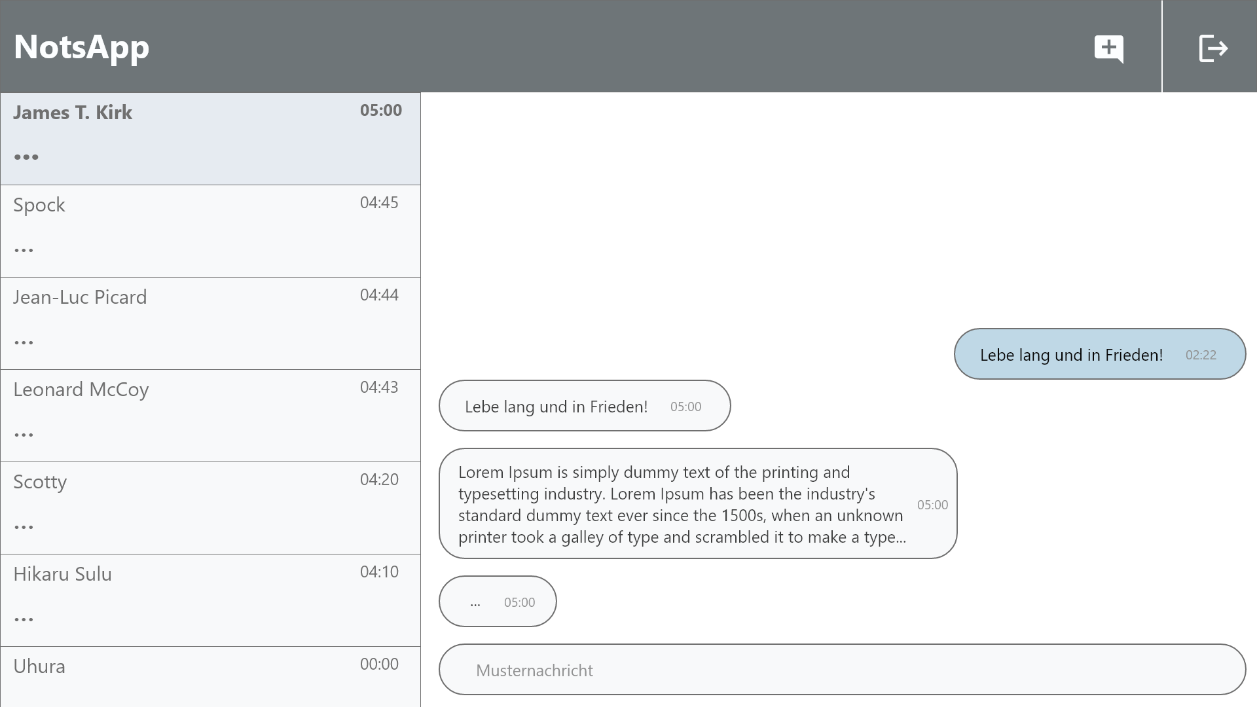
Anhang II: Adobe XD - Anmeldemaske



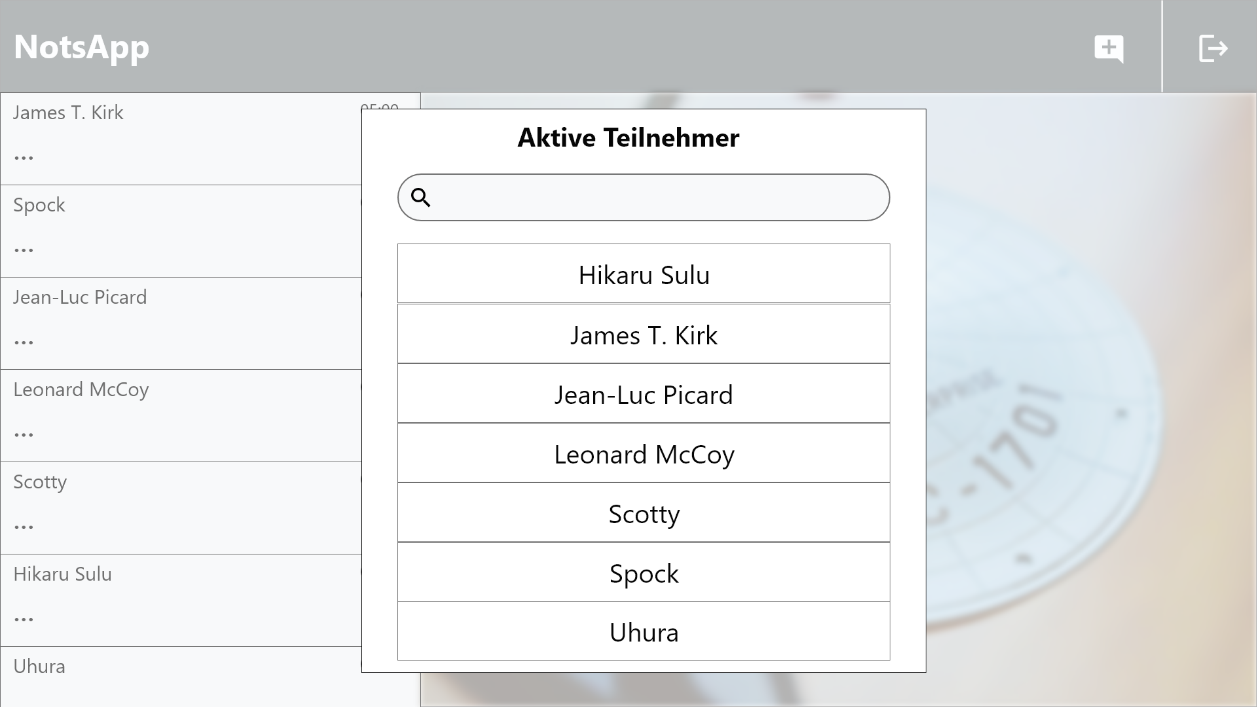
Anhang III: Adobe XD – Hauptmenü Chatroom



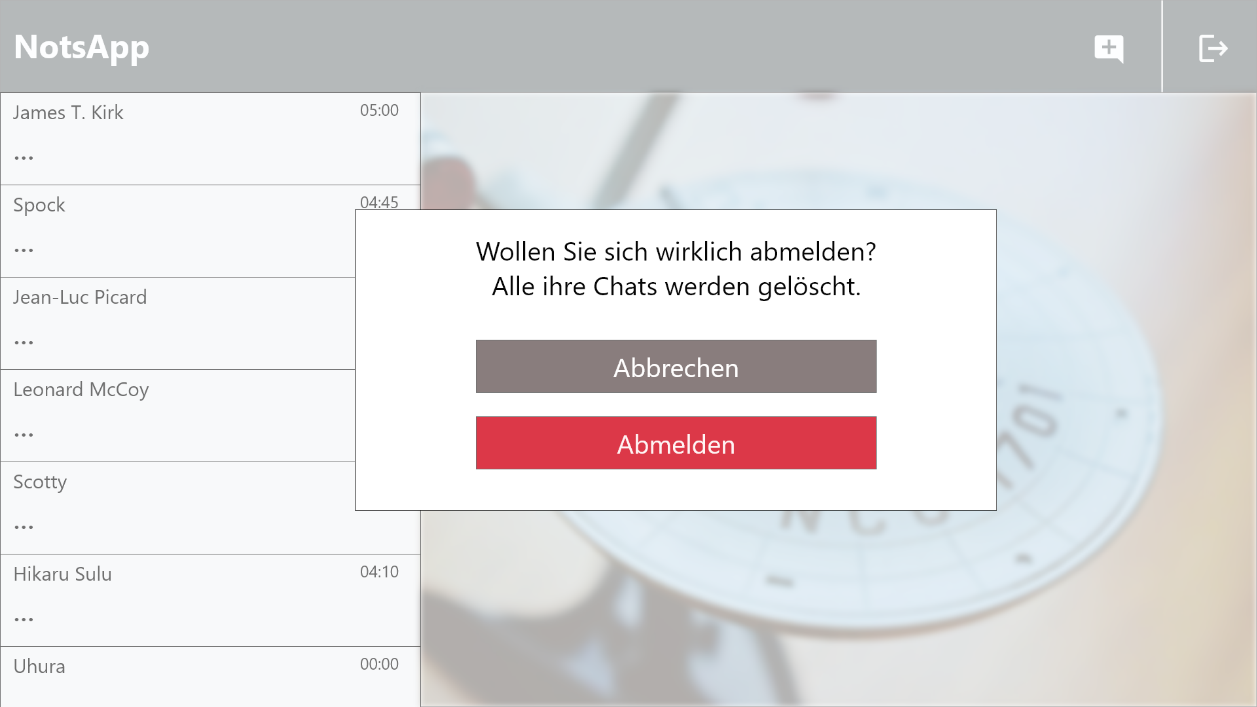
Anhang IV: Adobe XD – Hauptmenü Chatroom



Anhang V: Adobe XD – Hauptmenü Aktive Teilnehmer



Anhang VI: Adobe XD – Hauptmenü Abmelden

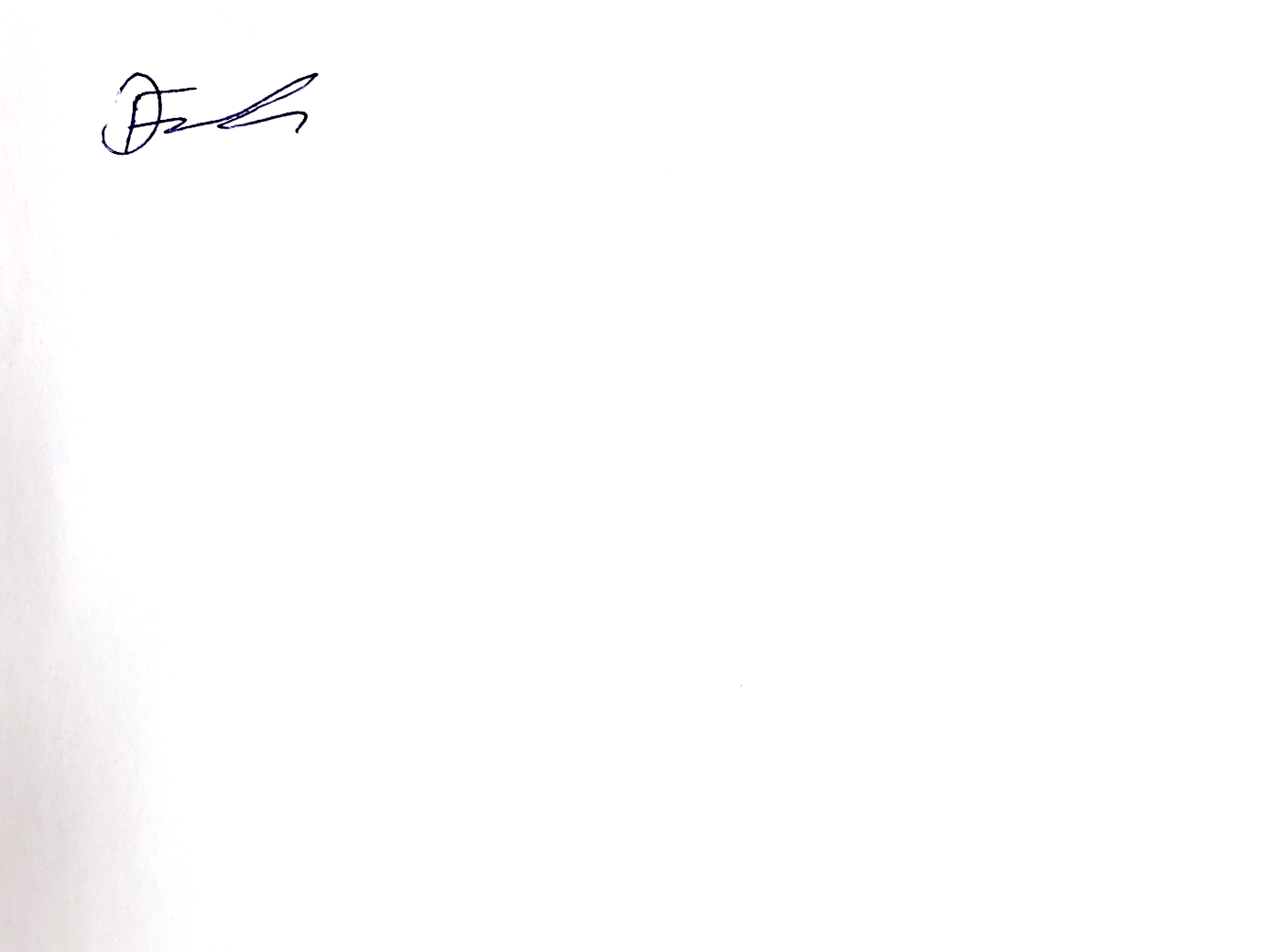


Erklärung

Wir versichern hiermit, dass wir unser Portfolio mit dem Thema „Anwendungsentwicklung - Chat“ selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben.

---------------------------------------

Ort



--------------------------------------- -------------------------------------

Datum -------------------------------------

-------------------------------------

-------------------------------------

-------------------------------------

Unterschrift