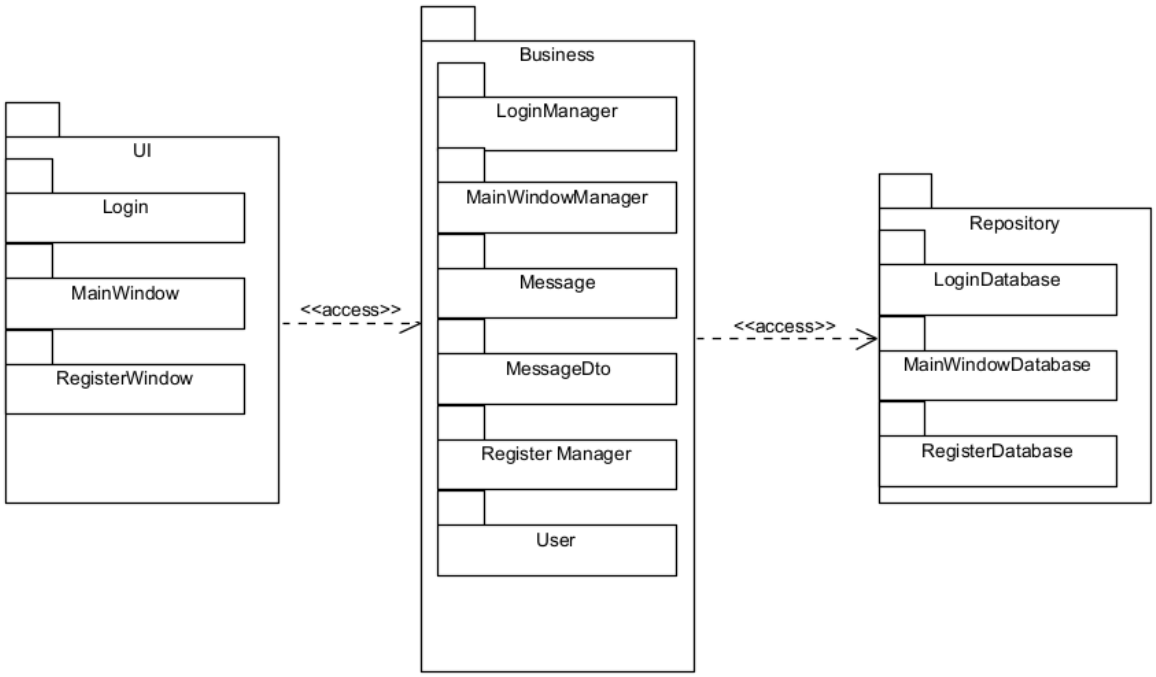
Bib Talk | Chatprogramm

1. Softwarearchitektur

1.1Technische Infrastruktur

Es wird das 3 Schichten Modell angewandt mit den Schichten UI, Repository und Business. In der UI-Schicht ist alles, was Visuell angezeigt wird. In der Business Schicht ist die Logik der z.B. Buttons hinterlegt und in der Repository Schicht sind die Daten abgelegt, die beim Login z.B. übergeben werden.

1.2 Paketdiagramm



1.3 Schnittstellen zu Nachbarsystemen und Beschreibung der Exportschnittstelle

1. Überblick über Nachbarsysteme

- Systemname: ASP.NET Core-Server
- Funktion: Bereitstellung der API für die Chat-App, einschließlich Benutzerverwaltung und Nachrichtenverarbeitung.
- Verantwortlichkeiten:
    - Benutzerverwaltung: Registrierung, Anmeldung, Logout, und Verwaltung von Profilbildern.
    - Nachrichtenverarbeitung: Senden, Abrufen und Löschen von Nachrichten.

2. Schnittstellenbeschreibung

- Schnittstellendefinition:
    - Die Chat-App kommuniziert über REST-APIs mit dem ASP.NET Core-Server, der die Anfragen verarbeitet und die entsprechenden Aktionen durchführt.
- Kommunikationsprotokolle:
    - HTTP/HTTPS für REST-APIs.
- Schnittstellenendpunkte:
    - Benutzerverwaltung:
        - Registrierung:
            - URL: http://daddypig.dns.navy:5114/api/users/register
            - HTTP-Methode: POST
            - Beschreibung: Registriert einen neuen Benutzer.
        - Anmeldung:
            - URL: http://daddypig.dns.navy:5114/api/users/login
            - HTTP-Methode: POST
            - Beschreibung: Loggt einen Benutzer ein.
        - Logout:
            - URL: http://daddypig.dns.navy:5114/api/users/logout
            - HTTP-Methode: POST
            - Beschreibung: Loggt einen Benutzer aus.
        - Profilbild hochladen:
            - URL: http://daddypig.dns.navy:5114/api/users/upload-profile-image?username={loggedinUser}
            - HTTP-Methode: POST
            - Beschreibung: Lädt ein Profilbild für einen Benutzer hoch.
    - Nachrichtenverarbeitung:
        - Nachricht senden:
            - URL: http://daddypig.dns.navy:5114/api/messages/send
            - HTTP-Methode: POST
            - Beschreibung: Sendet eine neue Nachricht.

        - Nachrichten abrufen:
            - URL: http://daddypig.dns.navy:5114/api/messages/get
            - HTTP-Methode: GET
            - Beschreibung: Holt alle Nachrichten, die derzeit in der Datenbank gespeichert sind.
        - Nachrichten löschen:
            - URL: http://daddypig.dns.navy:5114/api/messages/clear
            - HTTP-Methode: DELETE
            - Beschreibung: Löscht alle Nachrichten in der Datenbank.
    - Online-Nutzer abfragen:
        - URL: http://daddypig.dns.navy:5114/api/users/online-users
        - HTTP-Methode: GET
        - Beschreibung: Ruft eine Liste aller aktuell eingeloggten Benutzer ab.

3. Datenfluss und -struktur

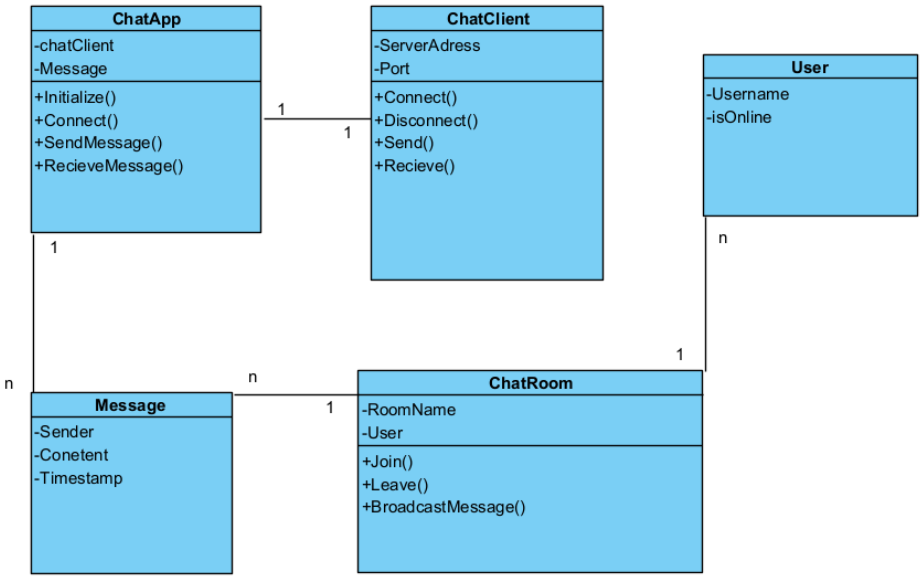
- Dateneingaben und -ausgaben:
    - Benutzerverwaltung:
        - Registrierung: Eingaben umfassen Benutzername, E-Mail-Adresse, Passwort und Geburtsdatum.
        - Anmeldung: Eingaben umfassen E-Mail-Adresse und Passwort.
        - Logout: Keine Eingaben, Token im Header.
        - Profilbild hochladen: Eingaben umfassen das Bilddateiformat.
    - Nachrichtenverarbeitung:
        - Nachricht senden: Eingaben umfassen Sender-ID, Empfänger-ID und Nachrichtentext.
        - Nachrichten abrufen: Keine Eingaben.
        - Nachrichten löschen: Keine Eingaben.
    - Online-Nutzer abfragen: Keine Eingaben; Ausgabe enthält eine Liste der derzeit online befindlichen Benutzer.
- Datenvalidierung:
    - Benutzerverwaltung: Validierung der Eingaben wie Passwortstärke und Format von E-Mail-Adressen; Profilbild-Größe und -Format.
    - Nachrichtenverarbeitung: Validierung der Nachrichteninhalte und -formate.

4. Fehlerbehandlung und Logging

- Fehlerarten:
    - Benutzerverwaltung: Fehler bei der Registrierung, Anmeldung oder beim Hochladen des Profilbildes
- Logging:
    - Serverseitiges Logging: Protokollierung von API-Zugriffen, Fehlern und Interaktionen mit der Datenbank in der Console.

2. Paketbeschreibung

2.1 Entwurfsklassendiagramm



2.1.1 Beschreibe Klasse ChatApp

Diese Klasse repräsentiert die Hauptanwendung für den Chat. Sie hat folgende Attribute und Methoden:

- **Attribute:**
  - chatClient: Verweis auf ein Objekt der Klasse ChatClient.
  - Message: Verweis auf ein Objekt der Klasse Message.
- **Methoden:**
  - Initialize(): Initialisiert die Chat-Anwendung.
  - Connect(): Stellt eine Verbindung zum Chat-Server her.
  - SendMessage(): Sendet eine Nachricht.
  - ReceiveMessage(): Empfängt eine Nachricht.

2.1.2 Beschreibe Klasse ChatClient

Diese Klasse repräsentiert den Client, der die Verbindung zum Server herstellt.

- **Attribute:**
  - ServerAddress: Die Adresse des Servers, zu dem die Verbindung hergestellt wird.
  - Port: Der Port, der für die Verbindung verwendet wird.
- **Methoden:**
  - Connect(): Verbindet den Client mit dem Server.
  - Disconnect(): Trennt die Verbindung zum Server.
  - Send(): Sendet Daten an den Server.
  - Receive(): Empfängt Daten vom Server.

2.1.3 Beschreibe Klasse User

Diese Klasse repräsentiert einen Benutzer im Chat-System.

- **Attribute:**
  - Username: Der Benutzername des Users.
  - isOnline: Ein boolescher Wert, der anzeigt, ob der Benutzer online ist.
- **Methoden:**
  - Es sind keine Methoden explizit aufgeführt.

2.1.4 Beschreibe Klasse ChatRoom

Diese Klasse repräsentiert einen Chatraum, in dem Benutzer miteinander kommunizieren können.

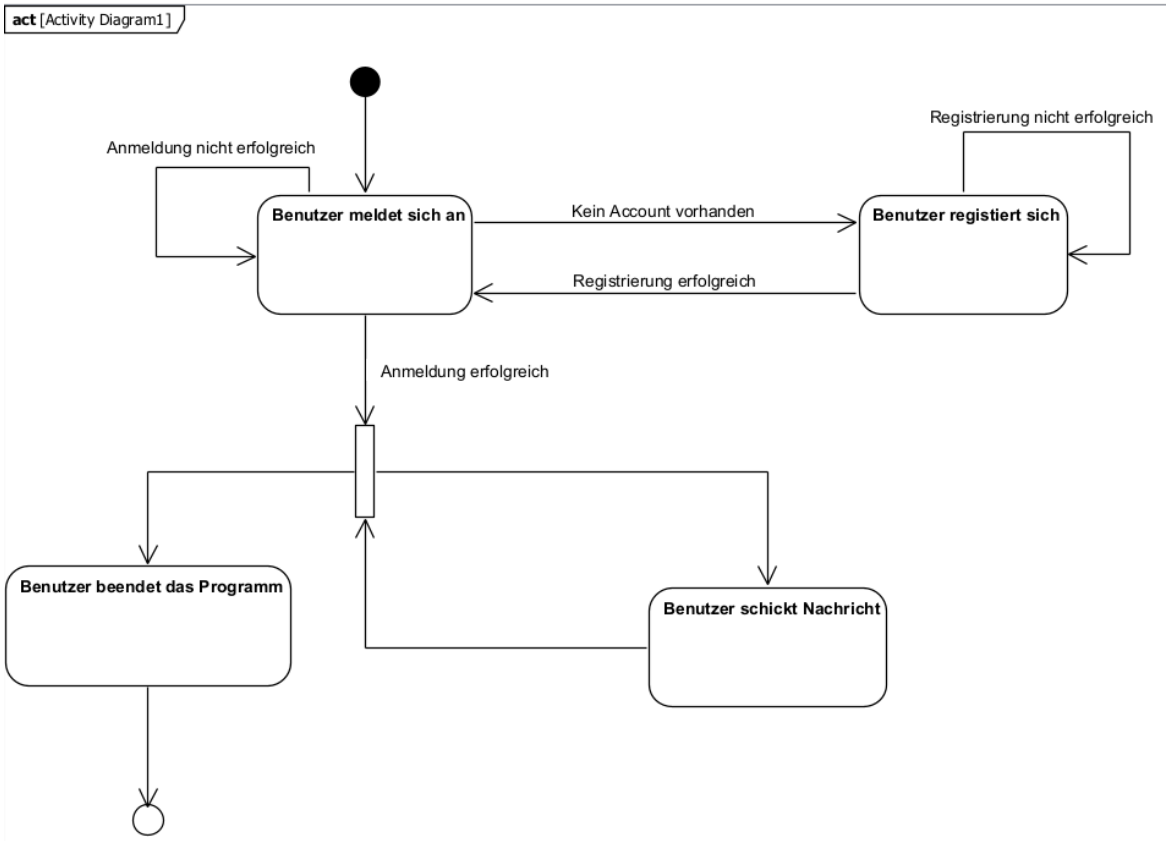
- **Attribute:**
  - RoomName: Der Name des Chatraums.
  - User: Verweis auf Benutzerobjekte, die dem Chatraum zugeordnet sind.
- **Methoden:**
  - Join(): Ermöglicht einem Benutzer, dem Chatraum beizutreten.
  - Leave(): Ermöglicht einem Benutzer, den Chatraum zu verlassen.
  - BroadcastMessage(): Sendet eine Nachricht an alle Benutzer im Chatraum.

2.1.5 Beschreibe Klasse Message

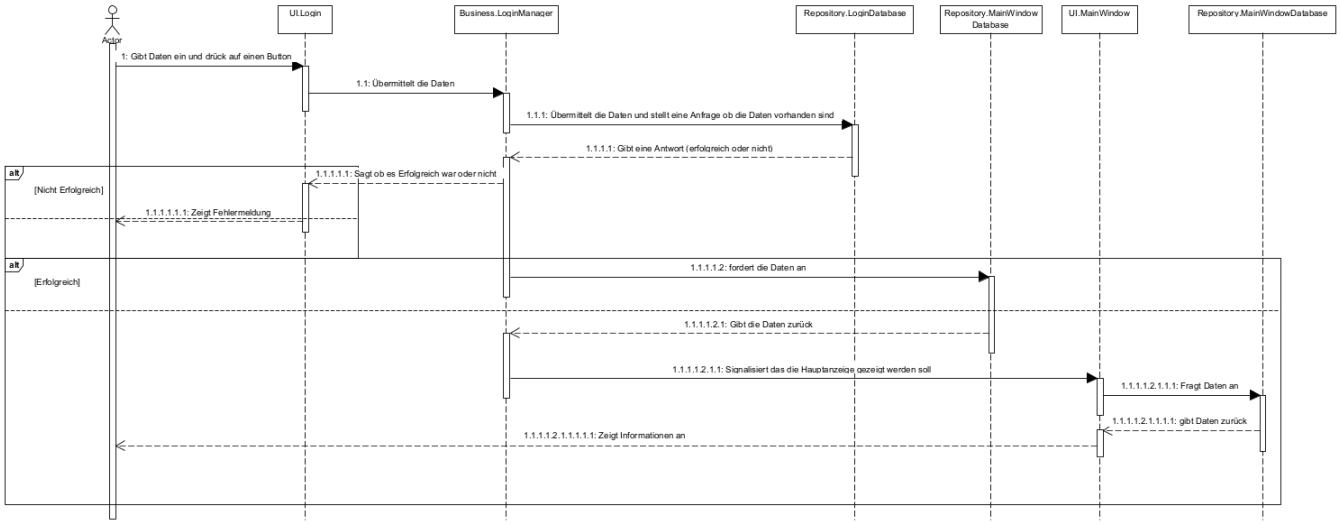
Diese Klasse repräsentiert eine Nachricht im Chat-System.

- **Attribute:**
  - Sender: Der Absender der Nachricht (ein User-Objekt).
  - Content: Der Inhalt der Nachricht.
  - Timestamp: Der Zeitstempel, wann die Nachricht gesendet wurde.
- **Methoden:**
  - Es sind keine Methoden explizit aufgeführt.

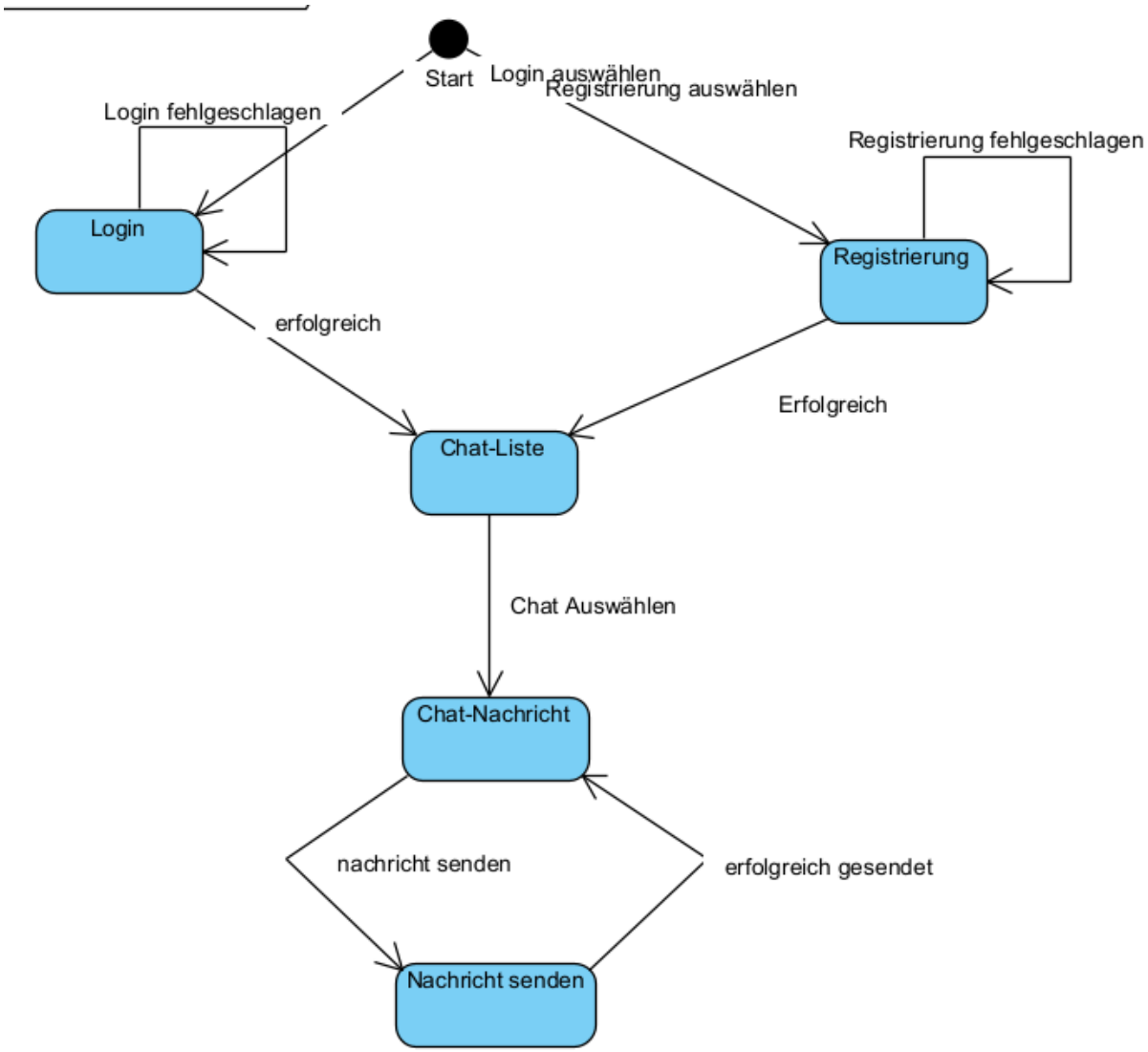
2.2 Aktivitätsdiagramm



2.3 Sequenzdiagramm



2.4 Zustandsdiagramm



3. Datenstrukturen

3.1 Datenbankentwurf

3.1.1 ER-Modell

3.1.2 Tabellenbeschreibung

1. Users-Tabelle

Die **Users**-Tabelle enthält die grundlegenden Informationen über registrierte Benutzer der Anwendung. Jede Zeile in dieser Tabelle repräsentiert einen einzelnen Benutzer.

- **Id** (INTEGER, NOT NULL, PRIMARY KEY, AUTOINCREMENT):
  Eindeutiger Identifikator für jeden Benutzer. Dieser Wert wird automatisch inkrementiert und dient als Primärschlüssel der Tabelle.
- **Username** (TEXT, NOT NULL):
  Eindeutiger Benutzername für jeden registrierten Benutzer. Dieser Wert wird verwendet, um Benutzer im System anzusprechen und für die Anmeldung.
- **Password** (TEXT, NOT NULL):
  Gespeichertes Passwort für die Benutzeranmeldung, das sicher verschlüsselt werden sollte.
- **Email** (TEXT, NOT NULL):
  Eindeutige E-Mail-Adresse des Benutzers, die auch zur Kommunikation und gegebenenfalls für die Anmeldung verwendet werden kann.
- **Birthday** (TEXT, NOT NULL):
  Geburtsdatum des Benutzers im Textformat. Dieses Feld kann für altersbezogene Berechnungen oder Personalisierungen genutzt werden.
- **IsOnline** (TEXT, NOT NULL, DEFAULT 0):
  Statusfeld, das angibt, ob der Benutzer derzeit online ist. Der Standardwert ist 0 (offline), und 1 bedeutet, dass der Benutzer aktiv ist.
- **ProfileImagePath** (TEXT, optional):
  Pfad zum Profilbild des Benutzers. Dieses Feld ist optional und kann genutzt werden, um den Speicherort des Bildes zu definieren.

---

2. Messages-Tabelle

Die **Messages**-Tabelle speichert alle Nachrichten, die von den Benutzern im System gesendet werden. Jede Nachricht wird eindeutig identifiziert und enthält den Inhalt sowie Metadaten wie Zeitstempel und den Absender.

- **Id** (INTEGER, NOT NULL, PRIMARY KEY, AUTOINCREMENT):
  Eindeutiger Identifikator für jede Nachricht. Dieser Wert wird automatisch inkrementiert und dient als Primärschlüssel der Tabelle.
- **Username** (TEXT, NOT NULL):
  Benutzername des Absenders der Nachricht. Dieses Feld stellt die Verbindung zur **Users**-Tabelle her, da es den Absender der Nachricht beschreibt.
- **Text** (TEXT, NOT NULL):
  Der Inhalt der Nachricht. Dieses Feld enthält den eigentlichen Nachrichtentext, der von den Benutzern verfasst wurde.
- **Timestamp** (TEXT, NOT NULL):
  Zeitstempel, der den Zeitpunkt angibt, zu dem die Nachricht gesendet wurde.

3.1.3 SQL-Abfragen

MessagesController

1. **SendMessage**
   - **Was sie macht:** Fügt eine neue Nachricht in die Messages-Tabelle der Datenbank ein.
2. **GetMessages**
   - **Was sie macht:** Ruft alle Nachrichten aus der Messages-Tabelle ab und sortiert sie nach dem Zeitstempel.
3. **ClearMessages**
   - **Was sie macht:** Löscht alle Nachrichten aus der Messages-Tabelle.

UsersController

1. **Login**
   - **Was sie macht:** Überprüft die Anmeldedaten eines Benutzers und setzt den Status des Benutzers auf online, wenn die Anmeldung erfolgreich ist.
2. **Register**
   - **Was sie macht:** Registriert einen neuen Benutzer, indem sie überprüft, ob der Benutzername bereits existiert, und dann den neuen Benutzer in die Datenbank einfügt.
3. **Logout**

- **Was sie macht:** Setzt den Status eines Benutzers auf offline.

4. **GetOnlineUsers**
    - **Was sie macht:** Ruft alle Benutzer ab, die aktuell online sind.
5. **UploadProfileImage**
    - **Was sie macht:** Aktualisiert das Profilbild eines Benutzers, indem es das hochgeladene Bild speichert und den Bildpfad in der Datenbank aktualisiert.

3.2 Dateibeschreibung

3.2.1 Einsatzzweck

Der Einsatzzweck von Bib Talk liegt bei einer einfachen Chat Anwendung für jeden Benutzer der einen Desktop PC besitzt.

3.2.2 Dateiaufbau

Als Design Pattern wurde das 3 Schichten Modell genommen also MVC(Model, View und Controller). Programmiert wird alles in Visual Studio im .Net Framework.