MODUL 295

Projekt von Jan G.

Inhalt

Informieren

Link zum GitHub Projekt:

https://github.com/lunar-noon/project-chatbox

Grundidee anhand von Elevator Pitch:

In der heutigen Welt wird es immer wichtiger mit der Zeit zu gehen. Viele Jugendliche führen ihre Konversationen online. Viele Menschen wünschen sich eine sichere und zuverlässige App, in welcher man miteinander kommunizieren kann.

Diese grossen und wichtigen Wünsche werden mit diesem Produkt Vergangenheit.

Ab heute wird niemand jemals mehr eine andere Chat-App verwenden wollen.

Wir bieten eine Webseite, in der man mit Freunden, Kollegen, Mitarbeitern und Verwandten von überall aus der Welt schreiben kann. Man kann sogar den gesamten Chatverlauf anzeigen lassen. Es war noch nie so leicht mit anderen online zu schreiben.

So kannst du mit deinen Kollegen wann auch immer spass haben und dich mit anderen verbinden.

User Stories:

User Story 1:

Ich als Benutzer will mit meinen Kollegen chatten können. Mein Benutzername soll bei mir und meinen Kollegen ersichtlich sein.

Akzeptanzkriterien:

- Der Benutzername sollte beim Chatpartner ersichtlich sein.
- Der Benutzername darf nicht leer oder länger wie 20 Zeichen lang sein.

User Story 2:

Als Benutzer lege ich grossen Wert auf die Sicherheit meiner Daten. Ich will somit mit meinen Arbeitskollegen in einem privaten Chat schreiben, ohne dass andere diese einsehen können.

Akzeptanzkriterien:

- Der Chat soll nur von den beiden Mitgliedern ersichtlich sein.
- Die Nachrichten dürfen nicht leer oder länger wie 500 Zeichen gross sein.
- Neben den Nachrichten soll die Erstellzeit der Nachricht angezeigt werden.

User Story 3:

Als Benutzer möchte meine alten Nachrichten, sowie die alten Nachrichten meines Kollegen einsehen können.

Akzeptanzkriterien:

- Ältere Nachrichten soll man einsehen können und im Chat angezeigt werden.

User Story 4:

Als Benutzer möchte ich meine Nachrichten bearbeiten können.

Akzeptanzkriterien:

- Eigene Nachrichten soll man bearbeiten und speichern können.
- Die bearbeiteten Nachrichten dürfen nicht leer oder länger wie 500 Zeichen gross sein.
- Diese Änderung soll auch beim Chatpartner übernommen werden.

User Story 5:

Als Benutzer will ich unerwünschte Nachrichten löschen können.

Akzeptanzkriterien:

- Eigene Nachrichten soll man löschen können.
- Die Nachrichten sollen beim Verlauf sowie dem Chatpartner nicht mehr ersichtlich sein.

Planen

Arbeitsplan:

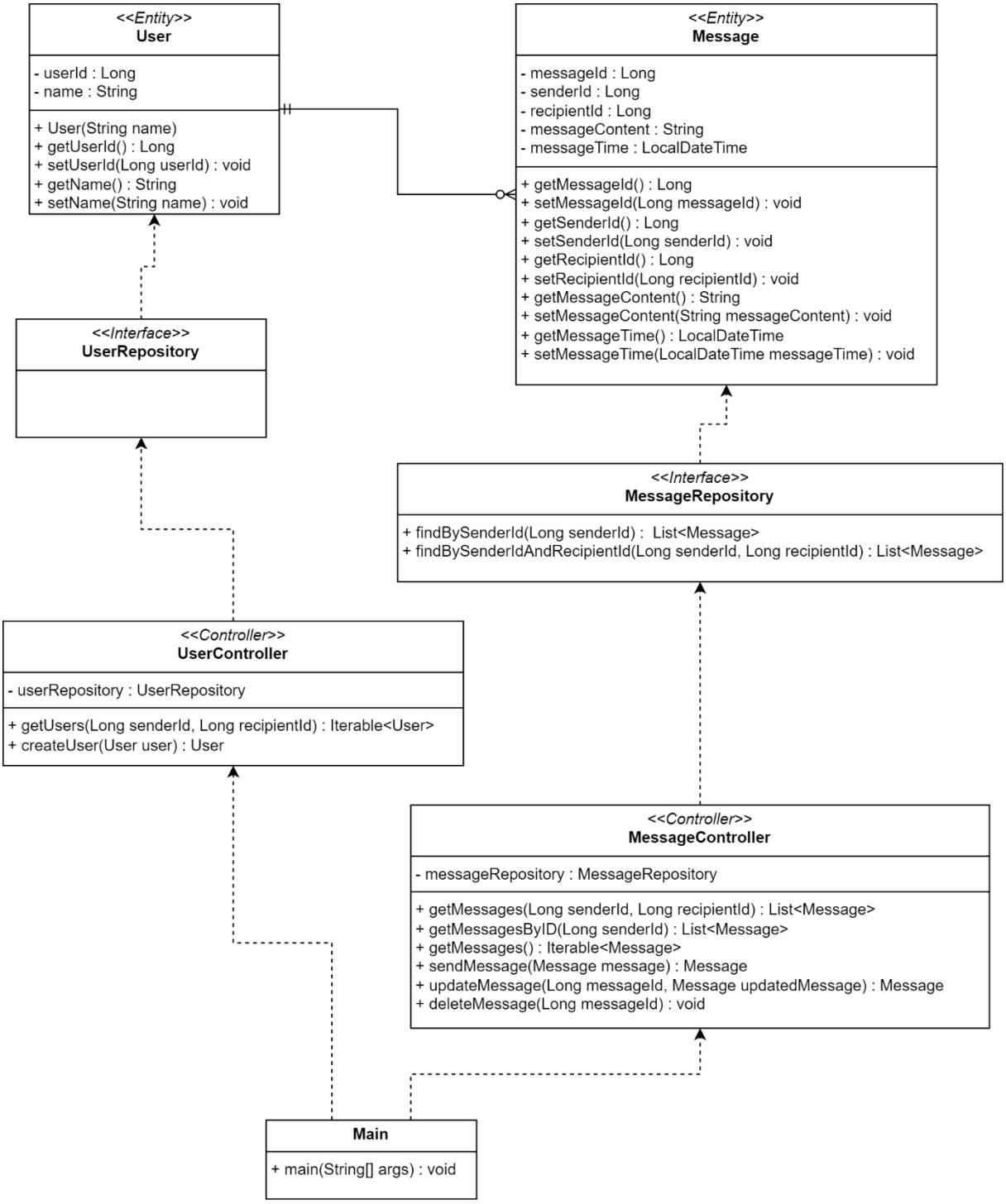
Hier ist die Arbeitsplanung mit den Zeitangaben und den Aufgabengrössen ersichtlich.

- **Informieren** **2h**
  - Grundidee 1h
  - User Stories 1h
- **Planen** **3h 20min**
  - Arbeitsplan 20min
  - Klassendiagramm 2h
  - Testplan 1h
- **Realisieren** **9h 20min**
  - Umsetzung des Projekts 8h
  - REST-Schnittstellen 1h
  - Installationsanleitung 20min
- **Kontrollieren** **4h**
  - Durchführung Testing 3h
  - Testprotokoll 1h
- **Auswerten** **50min**
  - Hilfestellung 20min
  - Reflexion 30min

Zeitaufwand insgesammt: 21h 30min

Klassendiagramm

Die Hintergrundfarbe ist jetzt grau damit man es leichter in diesem Dokument sehen kann. Auf der Webseite selbst ist die Hintergrundfarbe weiss.



Testplan:

Testplan für manuelle Eingabe:

Test-ID	Eingabe	Erwartung	Ergebnis	Massnahme
1	Keinen Benutzernamen eintragen.	Fehlermeldung: "Username cannot be empty"		
2	Leere Nachricht absenden.	Fehlermeldung: "Message cannot be empty"		
3	Korrekter Benutzername eintragen.	Rückmeldung der erfolgreich übermittelten Daten.		
4	Fehlerfreie Nachricht absenden.	Rückmeldung der erfolgreich übermittelten Daten.		
5	Zu langer Benutzername eintragen.	Fehlermeldung: "The Username must be between 3 and 20 characters long!"		
6	Zu lange Nachricht absenden.	Fehlermeldung: "The Message must be under 500 characters long!"		
7	http://localhost:8080/messages/1 aufrufen.	Nachrichten von dem Benutzer mit der ID 1 werden ausgegeben.		

Realisieren

REST-Schnittstellen:

Wie ist der Datentyp aufgebaut?

Die Nachrichten werden in einzelnen Objekten gespeichert, welche über die senderId und recipientId mit der userId verbunden sind.

Jedes Objekt erhält eine id, um wieder gefunden werden zu können.

Um mit der Datenbank zu interagieren, wird das alles jeweils in einen String umgewandelt und in Form von JSON übermittelt.

Ein Beispiel eines Strings der übertragen wird:

{ "**senderId**":"1", "**recipientId**":"2", "**messageContent**":"Message Text" }

Post:

Die Nachricht, welche der Nutzer eingegeben hat, wird in einem String gespeichert. Die Post Methode greift auf den Link zu und sendet den String als JSON-Datei dazu. Die Daten werden dann in der Datenbank eingetragen.

Get:

Die Methode greift auf den angegebenen Link zu und zeigt alle Nachrichten an. Man kann zusätzlich nach dem Sender und dem Empfänger sortieren. Die JSON-Datei wird ausgegeben und in einen String umgewandelt.

Put:

Die Methode greift über eine Kombination von Link und ID auf den Datensatz zu. Dieser Datensatz wird dann mit der neuen Nachricht überschrieben.

Delete:

Die Methode greift auf den Link mit der angegebenen ID zu und löscht die Nachricht.

Installationsanleitung:

Voraussetzungen:

- Docker oder Docker Desktop sollte installiert und eingerichtet sein.
- Maven und Java sollten installiert und eingerichtet sein.

Die einzelnen Schritte:

1. Beim GitHub Repository das Projekt herunterladen oder klonen: GitHub project-chatbox
2. Das Terminal im "project-chatbox\database" Ordner öffnen und den Befehl: "docker compose up -d" ausführen. Die Datenbank wird somit direkt erstellt.
3. Das Terminal im "project-chatbox\backend" Ordner öffnen und den Befehl: "mvn spring-boot:run" ausführen. Dadurch wird das Projekt ausgeführt.
4. Man kann nun auf das Projekt unter dem Link "http://localhost:8080/" zugreifen.
5. Wenn man auf die REST-API Dokumentation zugreifen will, kann man dies unter "http://localhost:8080/v3/api-docs".

Kontrollieren

Testprotokoll:

Testprotokoll für manuelle Eingabe:

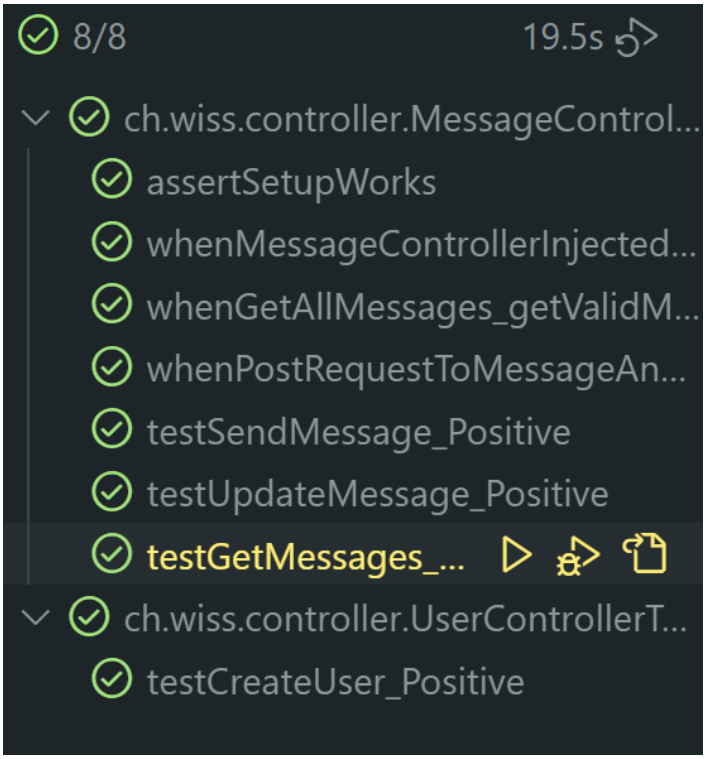
Test-ID	Eingabe	Erwartung	Ergebnis	Massnahme
1	Keinen Benutzernamen eintragen.	Fehlermeldung: "Username cannot be empty"	Fehlermeldung wird korrekt angezeigt.	Keine Massnamen nötig.
2	Leere Nachricht absenden.	Fehlermeldung: "Message cannot be empty"	Fehlermeldung wird korrekt angezeigt.	Keine Massnamen nötig.
3	Korrekter Benutzername eintragen.	Rückmeldung der erfolgreich übermittelten Daten.	Benutzername wurde korrekt übertragen.	Keine Massnamen nötig.
4	Fehlerfreie Nachricht absenden.	Rückmeldung der erfolgreich übermittelten Daten.	Nachricht wurde korrekt übertragen.	Keine Massnamen nötig.
5	Zu langer Benutzername eintragen.	Fehlermeldung: "The Username must be between 3 and 20 characters long!"	Fehlermeldung wird korrekt angezeigt.	Keine Massnamen nötig.
6	Zu lange Nachricht absenden.	Fehlermeldung: "The Message must be under 500 characters long!"	Fehlermeldung wird korrekt angezeigt.	Keine Massnamen nötig.
7	http://localhost:8080/messages/1 aufrufen	Nachrichten von dem Benutzer mit der ID 1 werden ausgegeben.	Die Nachrichten des Benutzers mit der ID 1 werden angezeigt.	Keine Massnamen nötig.

Die Tests haben alle wie erwartet funktioniert. Es sind keine weiteren Massnahmen mehr notwendig.

Testprotokoll für automatisierte Tests:

Man sieht anhand des Screenshots, dass die Tests erfolgreich durchgeführt wurden:

Ich habe die Ausgaben getestet, wenn man fehlerhaften Inhalt absendet, sowie weitere Tests, welche die Funktion und den Aufbau testen.



Auswerten

Hilfestellung:

Bei dem Projekt wurde teilweise ChatGPT als Hilfestellung eingesetzt.

Damit die Verbindung zwischen der Datenbank und der API einwandfrei funktionieren kann wurde ChatGPT für die application.properties Datei verwendet.

Ich habe bei den models habe ich ChatGPT als Hilfe benutzt, da die Daten zuerst nicht in der Datenbank gespeichert wurden.

Bei dem MessageController habe ich ChatGPT als Unterstützung verwendet, damit das PutMapping funktioniert.

Ich habe ChatGPT als Hilfe bei dem Klassendiagramm benutzt, da ich zuerst nicht genau wusste, wie ich dies Strukturieren musste.

Reflexion:

Beim Umsetzen des Projekts hatte ich am Anfang Schwierigkeiten.

Ich habe einen Chat erstellt, in welchem man mit WebSockets miteinander kommunizieren kann.

Ich hatte einige Probleme mit der Datenbank, in welcher die Daten nicht vollständig übernommen wurden. Ich konnte diese Probleme jedoch beseitigen.

Ich hatte auch weitere Fehler bei dem Verbinden der Tabellen. Ich habe diese wegen dem Zeitlimit nicht mehr fertigstellen können. Die Beziehungen wurden jedoch aufgezeichnet.

Ich habe die Zeit unterschätzt und kam zum Schluss unter grossen Zeitdruck.

Die REST-API konnte ich fast ohne Probleme implementieren. Nachdem ich die Methoden alle fertig hatte, hat das meiste bereits funktioniert.