Conversphere

Feil Lukas
l.feil@oth-aw.de
OTH Amberg-Weiden
Bad Kötzting, Deutschland

Markus Fleischmann
m.fleischmann2@oth-aw.de
OTH Amberg-Weiden
Wernberg-Köblitz, Deutschland

Stefan Reger
s.reger@oth-aw.de
OTH Amberg-Weiden
Schwarzenfeld, Deutschland

Lukas Rupp
l.rupp@oth-aw.de
OTH Amberg-Weiden
Amberg, Deutschland

Sorel Tahata Djoumsi
s.tahata-djoumsi@oth-aw.de
OTH Amberg-Weiden
Regensburg, Deutschland

Abstract

Web-Anwendung - Angular - TypeScript - MongoDB - Node.js - Socket.io - REST

1 Mission statement

Conversphere ist eine Web Chat-Anwendung, die es Benutzern ermöglicht, miteinander über Textnachrichten zu kommunizieren. Dadurch wird ein sicherer Raum geschaffen, in welchem man seine Gedanken und Ideen teilen kann und gleichzeitig sozial distanziert bleibt. Benötigt wird dafür lediglich ein Gerät mit Webbrowser und ein Internetzugang.

1.1 Kontextabgrenzung

Die Anwendung wird im Rahmen einer Studienarbeit an der Oth Amberg-Weiden erstellt. Die Anwendung ist zu Lern- und Testzwecken gedacht und wird nicht kommerziell genutzt. Es soll die Möglichkeit geboten werden die Anwendung in Zukunft bei persönlichem Interesse weiterzuentwickeln.

1.2 Qualitätsziele

Gute Benutzbarkeit:.

- Conversphere ist für Benutzer aller Altersklassen intuitiv und ohne Erklärung bedienbar.

Hohe Zuverlässigkeit:.

- Das System steht den Spielern jederzeit zur Verfügung und stellt Chaträume live dar.

Gute Wartbarkeit :.

- Conversphere ist ohne großen Administrationsaufwand betreibbar und leicht um zusätzliche Features erweiterbar.
- Die Funktionen der Anwendung sind durch Tests geprüft und funktionieren wie erwartet.

1.3 Rahmenbedingungen

Zeitlicher Rahmen:.

- Das Projekt ist innerhalb von fünf Kalenderwochen abzuschließen.

Technischer Anspruch:.

- Das Projekt muss mindestens einen der Themenbereiche Web-Anwendung oder Cloud-Computing abdecken.
- Die Anwendung soll so einfach gehalten werden, dass sie im gegebenen Zeitraum umgesetzt werden kann.

2 Umsetzung

2.1 Architekturentscheidung

Bei der Umsetzung von Conversphere haben wir uns für eine Web-Anwendung entschieden, um eine breite Zugänglichkeit über verschiedene Geräte und Plattformen hinweg zu gewährleisten. Dabei setzen wir die leistungsfähige und moderne Programmiersprache TypeScript ein, die uns eine einfache und typsichere Entwicklung ermöglicht.

Um die Entwicklung der Benutzeroberfläche zu vereinfachen und eine hohe Benutzerfreundlichkeit zu gewährleisten, verwenden wir das Angular Framework. Angular bietet uns eine strukturierte und komponentenbasierte Architektur, die es uns ermöglicht, die Anwendung in logisch getrennte Teile aufzugliedern und somit den Entwicklungsprozess zu vereinfachen.

Für die Datenverwaltung setzen wir die MongoDB Datenbank ein. MongoDB ist eine dokumentenorientierte Datenbank, die uns Flexibilität und Skalierbarkeit bietet. Durch die Integration der MongoDB in unsere Anwen-

dung können wir Daten effizient speichern und abrufen, um eine reibungslose Kommunikation zwischen den Benutzern zu gewährleisten.

Bei der serverseitigen Entwicklung nutzen wir das leistungsstarke Node.js Framework. Node.js ermöglicht uns die Entwicklung von serverseitigem Code in JavaScript und bietet eine effiziente und skalierbare Plattform für die Ausführung unserer Web-Anwendung.

Zusätzlich verwenden wir das Express.js Framework, das auf Node.js aufbaut und uns eine einfache und effektive Art der Entwicklung von Web-Anwendungen ermöglicht. Express.js bietet uns eine Vielzahl von Funktionen und Erweiterungen, die uns helfen, einen robusten und sicheren Webserver für unsere Anwendung bereitzustellen.

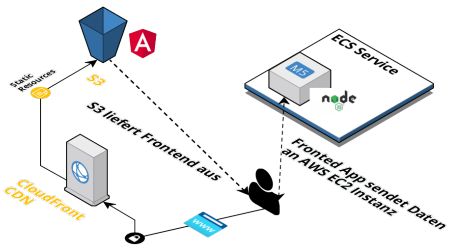
Damit die Conversphere-Webanwendung Echtzeitkommunikation unterstützen kann, integrieren wir das Socket.io Framework. Socket.io ermöglicht es uns, Websockets zu implementieren und eine zuverlässige bidirektionale Echtzeitkommunikation zwischen den Benutzern herzustellen. Dadurch können die Benutzer in Echtzeit Nachrichten austauschen und interaktiv miteinander kommunizieren. Zusätzlich zu der Kommunikation über Socket.io wurde auch noch eine REST-Schnittstelle verwendet, um Anfragen an das Backend zu senden.

Durch die geschickte Kombination dieser Technologien und Frameworks schaffen wir eine leistungsfähige und benutzerfreundliche Webanwendung, die es den Benutzern ermöglicht, sicher und effizient miteinander zu kommunizieren. Die Umsetzung in TypeScript, die Verwendung von Angular, MongoDB, Node.js, Express.js und Socket.io ermöglichen es uns, eine hochwertige und skalierbare Lösung zu entwickeln, die den Anforderungen der Conversphere-Webanwendung gerecht wird.

Um einen detaillierten Einblick in die Funktionalität des Express-Servers zu erhalten, werden nachfolgend die REST-Schnittstelle und die Websocket-Endpoints kurz beschrieben.

Die App wird über AWS S3 bereitgestellt. Der Zugriff auf AWS S3 wird durch Cloudfront Weltweit mit niedriger Latenz ermöglicht.

Interagiert der Benutzer mit der bereitgestellten Web App so greift diese auf eine AWS EC2 Instanz zu in welcher der Docker Container läuft. Die EC2 Instanz wird durch einen ECS Service verwaltet. Der Docker Container enthält den Node.js Server.



Deployment in AWS

2.2 Beschreibung der REST-Schnittstelle

Die REST-Schnittstelle dient zur Bereitstellung einer Kommunikationsschiene für die Anfragen, welche nicht für die Funktionalität des Chat-Services benötigt werden, beispielsweise das Erstellen von Räumen. Die Schnittstelle stellt 3 Endpunkte bereit:

GET /api/rooms:.

Unter diesem Endpunkt kann eine List mit den aktuellen Räumen abgerufen werden.

POST /api/createRoom.

Über diesen Endpunkt kann ein neuer Raum mit einem neuen Namen erstellt werden.

POST /api/join.

Über diesen Endpunkt kann einem bestehenden Raum beigetreten werden. Als Response wird das User-Objekt gesendet, mit welchem der Aufrufer im Chat-Room identifiziert wird. Mit diesem kann anschließend die Socket.io-Verbindung initiiert werden.

2.3 Beschreibung der Socket.io-Schnittstelle

Diese Schnittstelle dient zur bidirektionalen Echtzeitkommunikation zwischem dem Server und der Webseite. Folgende Endpunkte sind für diese Schnittstelle definiert:

ws/rooms/{roomId}/messages.

Dieser Kanal wird verwendet, um über neue Nachrichten im Chat-Room zu informieren. Das Backend hört dabei auf Nachrichten mit der Topic „sendNewMessage“. Über diese kann das Frontend neue Nachrichten vom Nutzer an das Backend senden. Dort wird diese Nachricht verarbeitet und an die anderen Nutzer weitergesendet. Das Weitersenden der Nachrichten geschieht über Nachrichten mit der Topic „receiveNewMessage“. Bei diesen Nachrichten wird die anzuzeigende Nachricht mit der entsprechenden Sichtbarkeit an jeden einzelnen Nutzer in dem Raum gesendet.

ws/rooms/{roomId}/users.

Dieser Kanal wird verwendet, um Nutzer bezogene Nachrichten zwischen dem Frontend und dem Backend auszutauschen. Das Backend kann über eine Nachricht mit der Topic „userInformation“ eine aktuelle Liste aller Nutzer im Chatraum und deren akuteller Position an alle Nutzer im Raum zu senden. Dies wird beispielsweise gemacht, wenn ein neuer Nutzer den Raum betreten hat oder sich ein Status eines Nutzers im Raum verändert hat. Damit das Frontend das Backend über eine Änderung des Nutzer informieren kann, sind dafür zwei Topics definiert: „leaveRoom“ und „positionUpdate“. Mittels „leaveRoom“ kann ein Nutzer einen akutellen Chatraum verlassen. „positionUpdate“ informiert das Backend über einen Positionswechsel des Nutzers. In beiden Fällen wird die akutelle Nutzerinformation über die Topic „userInformation“ an alle Nutzer im Raum verteilt.

3 Herausforderungen während der Entwicklung

Während der Entwicklung der Conversphere-Webanwendung traten verschiedene Herausforderungen auf, die es zu bewältigen galt. Eine dieser Herausforderungen bestand darin, die Position eines Mausklicks im Browser zu erfassen und die Spieler entsprechend im Spielfeld zu platzieren. Die genaue Erfassung der Mausposition und die korrekte Umsetzung im Spielfeld erforderten eine Umrechnung der ausgelesenen Positionen im Verhältins zum Spielfeld. Dadurch muss zusätzlich die Größe des Browserfensters erfasst werden. Außerdem muss die Position der Spieler im Spielfeld in Abhängigkeit von der Größe des Browserfensters angepasst werden und bei einer Änderung der Fenstergröße neu berechnet werden.

Die Umsetztung des Chats war zunächst nicht wie gewünscht möglich. Nachdem wir uns gegen long polling entschieden haben, wurde die Kommunikation zwischen den Benutzern über Websockets realisiert. Websockets ermöglichen eine Echtzeitkommunikation zwischen den Benutzern und sind entscheidend für die sofortige Übertragung von Nachrichten im Chat. Es erforderte ein gründliches Verständnis der Websocketimplementierung von Socket.io und eine effiziente Integration in die Anwendung, um eine zuverlässige und stabile Kommunikation zu gewährleisten.

Das Durchführen von Angular-Komponententests gestaltete sich ebenfalls als Herausforderung. Da wir die aktuellsten Features des Angular Frameworks in Version 16 verwenden, konnten wir nur sehr wenig Dokumentation dazu finden. Am schwierigsten stellte sich das Erstellen der Testkonfiguration dar. Angular verwendet die unabhängige Testumgebung Karma mit dem Test Framework Jasmine im Hintergrund. Aus diesem Grund gibt es auf der offiziellen Homepage von Angular „Angular.io“, nur sehr wenig Dokumentation, wie diese korrekt zu verwenden sind. Es ist notwendig Erkenntnisse auf der Dokumentation von Karma bzw. Jasmine und Anuglar zu verbinden um eine Testumgebung zu initiieren. Nachdem wir die Testkonfiguration erstellt hatten, konnten wir die Tests jedoch erfolgreich durchführen.

Eine weitere Herausforderung bestand darin, die MongoDB-Datenbank anzusprechen und effizient mit ihr zu interagieren. Die Integration der

Datenbank erforderte das Schreiben geeigneter Abfragen und die Beherrschung der Datenbank-APIs. Die korrekte Handhabung von Datenzugriff und -manipulation, sowie die Gewährleistung einer effizienten Datenbankperformance waren entscheidend für eine reibungslose Funktion der Anwendung.

4 Ausblick

Aufgrund zeitlicher Beschränkungen wird die Anwendung derzeit nicht auf AWS gehostet, sondern lokal als Docker-Container betrieben. Es ist jedoch geplant, den Docker-Container in Zukunft auf dem AWS Elastic Container Service zu hosten. Darüber hinaus planen wir in naher Zukunft die Implementierung eines Schiebereglers zur Einstellung der Lautstärke, um die Reichweite von Nachrichten anzupassen. Abhängig von der Lautstärke wird der Radius für den Empfang oder Versand von Nachrichten entsprechend vergrößert oder verkleinert. Auch der Code für eine automatische Farbverteilung der anderen Spieler oder sogar eigener Farbwahl ist bereits vorhanden, aus zeitlichen Gründen hat dieser es jedoch nicht in die bisherige Implementierung geschafft. Des Weiteren möchten wir den „Drunk-Modus“ einführen, bei welchem Buchstaben zufällig vertauscht werden, um einen spielerischen Aspekt in unseren Chatraum zu integrieren. Denkbar für die Anwendung ist auch, eine tokenbasierte Benutzerauthentifizierung zu erstellen und diese dann in unserer Datenbank abzuspeichern, um ein Benutzerprofil erstellen zu können.