
# WIP: Verteilte Systeme Projekt - TuiTalk

**Studienarbeit**

für die Vorlesung

**Verteilte Systeme**

des Studiengangs Informatik

an der Dualen Hochschule Baden-Württemberg Heidenheim

von

**Behr Tobias, Schönig Marc, Seidl Anian**

September 20XX

| | |
|---|---|
| **Bearbeitungszeitraum** | 8 Wochen |
| **Matrikelnummer, Kurs** | WIP, INF2023AI |
| **Gutachter** | |

# Erklärung

Wir versichern hiermit, dass wir unsere Studienarbeit mit dem Thema: *WIP: Verteilte Systeme Projekt - TuiTalk* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben. Wir versichern zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Heidenheim, September 20XX

---

Behr Tobias, Schönig Marc, Seidl Anian

## Abstract

# Inhaltsverzeichnis

# Abkürzungsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Listings

# 1 Einleitung

Kommunikation erfolgt heutzutage über eine Vielzahl von Plattformen. Bekannte Anwendungen wie WhatsApp, Discord oder Signal bieten bereits umfangreiche Möglichkeiten zum Austausch von Nachrichten. Das Ziel dieses Projekts ist die Entwicklung einer einheitlichen Lösung, die sowohl über das Terminal als auch über den Browser genutzt werden kann. Hierfür wird ein Backend implementiert, das durch eine Terminal-Anwendung und eine Weboberfläche ergänzt wird. Ein wesentlicher Unterschied zu bestehenden Lösungen besteht darin, dass die Nutzung ohne vorherige Registrierung möglich ist.

# 2 Architektur

## 2.1 Anforderungen

Im Folgenden werden anhand der aufgelisteten internen und externen Stakeholder die funktionalen und nicht-funktionalen Anforderungen aufgeführt.

Interne Stakeholder

- Entwickler
- Betreuer
- Systemadministrator

Externe Stakeholder

- Nutzer

### 2.1.1 Funktionale Anforderungen

- Client Kommunikation: Als Nutzer und Entwickler möchte ich mich über WebSockets mit dem System verbinden können, wobei ich zwischen einem CLI- und einem WASM-Client entscheiden möchte, damit ich meine präferierte Umgebung verwenden kann. Ich möchte Nachrichten in Echtzeit senden und empfangen, sowie Räumen beitreten und verlassen können. Darüber hinaus soll es möglich sein, vergangene Nachrichten laden zu können, damit ich den gesamten Chatverlauf einsehen kann.
- Loadbalancer: Als Nutzer und Systemadministrator möchte ich, dass ein Loadbalancer die WebSocket Verbindungen gleichmäßig auf die vorhandenen Backends verteilt, damit eine optimale Performance erreicht werden kann. Wenn ein Backend ausfällt, sollen die bestehenden Verbindungen nahtlos vom Loadbalancer zu anderen Backends umgeleitet werden, sodass ich beim Benutzen keine Unterbrechung bemerke.

- Backend Services: Als Entwickler möchte ich, dass die Client Verbindungen vom Backend verwaltet werden und Nutzer spezifische Buffer verwendet werden, um Nachrichten effizient und zuverlässig verarbeiten zu können. Nachrichten sollen über Redis verteilt werden, damit alle Backendinstanzen synchron sind und in einer Datenbank gespeichert werden, um Chatverläufe abrufen zu können. Das System soll Redis Publish/Subscribe im Cluster unterstützen, damit es Skalierbar und Ausfallsicher ist.

### 2.1.2 Nichtfunktionale Anforderungen

- Skalierbarkeit
- Hohe Verfügbarkeit
- Fehlertoleranz
- Zuverlässigkeit
-

## 2.2 Systemkomponenten

# 3 Umsetzung

# 4 Reflektion

# Anhang