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

- Was wollen wir tun?

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, Räumen beitreten und verlassen, sowie meinen Anzeigenamen ändern 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

- Hohe Verfügbarkeit: Als Nutzer, Systemadministrator und Betreuer möchte ich, dass das System weiterhin funktioniert, auch wenn ein Backend ausfällt, ohne das der Nutzer eine Unterbrechung mitbekommt. Nach dem Ausfall eines Backends soll die Verbindung automatish von einem anderen Backend übernommen werden ohne das Nachrichten verloren gehen.
- Zuverlässigkeit: Als Nutzer und Systemadministrator möchte ich, dass Nachrichten zugestellt werden, solange Redis und mindestens ein Backend verfügbar sind, so dass Nachrichten nicht verloren gehen, auch wenn Teile des Systems ausfallen. Die Konsistenz der Daten soll durch eine Datenbank sichergestellt werden, sodass keine Nachrichten des Chatverlaufs fehlen und dieser damit eindeutig und zuverlässig ist.
- Skalierbarkeit: Als Entwickler und Betreuer möchte ich, dass das System horizontal skalierbar ist, damit es auch unter hoher Last performant bleibt. Es sollen mehrere Redis Caches und Backends verwendet werden.

2.2 Systemkomponenten

3 Umsetzung

4 Reflektion

Anhang