

Peer-to-Peer Chatanwendung mit GUN.js und Angular

PROJEKTARBEIT Verteilte Systeme

des Studienganges Angewandte Informatik

an der Dualen Hochschule Baden-Württemberg Mannheim

von

Lev Likhanov, Rafael Simon, Jan-Luca Wolf und Mehmet Yavuz

Abgabedatum

22.12.2023

Bearbeitungszeitraum 02.10.2023 – 22.12.2023

Kurs TINF21AI2

Matrikelnummern:
6088902 – Jan-Luca Wolf
4568447 – Rafael Simon
4064127 – Mehmet Yavuz
– Lev Likhanov

Inhaltsverzeichnis

Inhaltsverzeichnis	I
Abbildungsverzeichnis	I
Abkürzungsverzeichnis	II
1 Einleitung	1
2 Grundlagen	1
2.1 „Peer-to-Peer“-Systeme	1
2.2 GUN.js und Graphdatenbanken	5
2.3 Angular	7
3 Umsetzung	8
3.1 Development- und Deploymentworkflow	8
3.2 Webanwendung	10
4 Ergebnis	11
Quellenverzeichnis	13

Abbildungsverzeichnis

Abbildung 1: P2P-Netzwerkarchitekturen	2
Abbildung 2: Beispiel-Graphdatenbank	6
Abbildung 3: Workflow	8
Abbildung 4: Architektur	9
Abbildung 5: Login-Fenster	11
Abbildung 6: Chatanwendung	12

Abkürzungsverzeichnis

Peer-to-Peer.....	P2P
Distributed Hash Tables.....	DHT
Web Real-Time Communication.....	WebRTC
Continuous Integration.....	CI
Continuous Deployment.....	CD

1 Einleitung

Die vorliegende Projektarbeit beschäftigt sich mit „Peer-to-Peer“-Systemen. Im Besonderen wird die Erstellung einer „Peer-to-Peer“-Chatanwendung unter Verwendung von GUN.js und Angular beschrieben.

2 Grundlagen

In diesem Kapitel werden die technischen Grundlagen von „Peer-to-Peer“-Systemen vorgestellt. Des Weiteren folgen die technischen Grundlagen der verwendeten Tools GUN.js sowie Angular.

2.1 „Peer-to-Peer“-Systeme

„Peer-to-Peer“ Systeme, oft auch als „P2P“-Systeme bezeichnet, bestehen aus mehreren einzelnen „Peers“. Dabei bezeichnet das englische Wort „Peer“ einen Ebenbürtigen oder Gleichrangigen. Somit stellt ein „Peer-to-Peer“-System oder ein „Peer-to-Peer“-Netzwerk ein Netzwerk aus gleichrangigen Instanzen dar. Es gibt keinen zentralen Teilnehmer, der anderen Teilnehmern übergeordnet ist. „Peer-to-Peer“-Systeme sind dezentral aufgebaut und organisieren sich selbst. Anders als beim Client-Server-Netzwerk, gibt es in „Peer-to-Peer“-Netzwerken keine Unterscheidung in Client- oder Server-Rollen. Die einzelnen Peers können sowohl die Server-Rolle (Dienstanbieter) als auch die Client-Rolle (Dienstnutzer) übernehmen. [vgl. Peter Mahlmann, Christian Schindelbauer, Seiten: 6f]

Neben den ursprünglich definierten dezentralen „P2P“-Systemen, haben sich im Laufe der Zeit auch zentralisierte „P2P“-Systeme und hybride „P2P“-Systeme entwickelt. In Abbildung 1: P2P-Netzwerkarchitekturen ist der Aufbau der verschiedenen Netzwerkarchitekturen dargestellt. **Dezentrale „P2P“-Systeme** bestehen wie oben erwähnt nur aus gleichrangigen Peers. Sie besitzen eine hohe Robustheit gegen Ausfälle einzelner Peers, da die Aufgaben durch andere Peers übernommen werden

können. Nachteile von dezentralen „P2P“-Systemen sind jedoch die hohe Komplexität, eine schwierige Verwaltung und eine eingeschränkte Leistung.

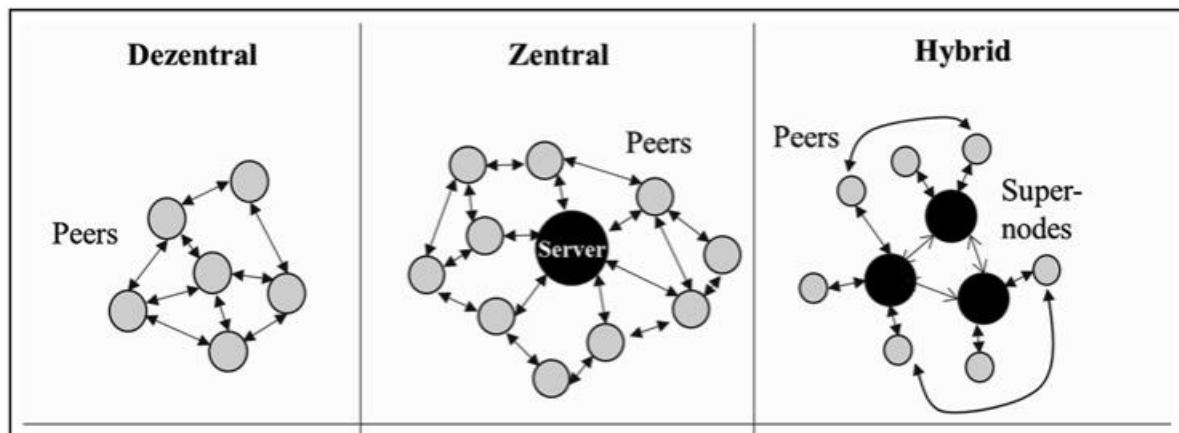


Abbildung 1: P2P-Netzwerkarchitekturen¹

Zentralisierte „P2P“-Systeme bestehen aus vielen Peers sowie einem zentralisierten Peer. Dieser zentralisierte Peer übernimmt die Verwaltung des Systems, wie beispielsweise die Pflege von Adressdaten der verschiedenen Peers sowie eine Pflege von Dateilisten. Dadurch kann eine gezielte und schnelle Suche nach Daten im Netzwerk erfolgen. Der eigentliche Datenaustausch erfolgt jedoch nach wie vor zwischen den einzelnen Peers. Der zentralisierte Peer stellt einen Engpass im Netzwerk dar. Falls notwendig kann diese Rolle jedoch auch an einen beliebigen anderen Peer übertragen werden. [vgl. **R.Clement, D.Schreiber, P.Bossauer, S.Pakusch** Seiten: 307f]

Einen Sonderfall stellen **hybride „P2P“-Systeme** dar. Sie bestehen aus vielen Peers und einigen wenigen „Super-Peers“. Super-Peers sind besonders leistungsstarke Peers im Netzwerk. Sie sind zwar immer noch dezentral angeordnet, teilen sich jedoch die Verwaltungsaufgaben des zuvor beschriebenen zentralisierten Peers. Des Weiteren werden die Daten vorrangig auf Super-Peers abgelegt, um die Abfragekomplexität zu reduzieren und die Suchgeschwindigkeit zu erhöhen. [vgl. **R.Clement, D.Schreiber, P.Bossauer, S.Pakusch** Seiten: 307f]

¹ Nach **R.Clement, D.Schreiber, P.Bossauer, S.Pakusch** Seite: 308

Bei „Peer-to-Peer“-Systemen kann grundsätzlich zwischen strukturierten und unstrukturierten „P2P“-Systemen unterschieden werden. Bei **Unstrukturierten „P2P“-Systeme** sind die Verbindungen zwischen den einzelnen Peers unbekannt. Die einzelnen Peers besitzen keine Informationen über die Wege zwischen den Quellen und dem Zielort. Ebenso ist unbekannt welcher Peer die notwendigen Daten besitzt. Daher stellen die einzelnen Peers Suchanfragen, mit denen sie die benötigten Daten anfragen. Diesen Suchanfragen garantieren keinen Sucherfolg und erzeugen eine große Last im Netzwerk. [vgl. Konrad Miller, Seite: 4]

Im Folgenden werden verschiedene Suchmethoden in unstrukturierten „P2P“-Systemen vorgestellt. Bei der **Flooding-Methode** wird die Suchanfrage an alle Teilnehmer im Netzwerk gesendet. Durch die parallele Breitensuche ist die Methode robust gegen Topologieänderungen und Nachrichtenverlust. Nachteilig bei dieser Methode ist jedoch das enorm hohe Nachrichtenaufkommen im Netzwerk. [vgl. Konrad Miller, Seite: 5]

Bei der **Random-Walk-Methode** wird die Suchanfrage über zufällige Peers durch das Netzwerk geleitet. Bei dieser Methode wird eine sequentielle Tiefensuche angewandt. Dadurch können die Anzahl der Nachrichten im Netzwerk und damit auch die Netzwerkauslastung deutlich gesenkt werden. Problematisch bei dieser Methode ist jedoch die große Latenz, da es lange dauern kann, bis die Suchanfrage den Peer mit den notwendigen Informationen erreicht. Zudem liegt die Erfolgswahrscheinlichkeit dieser Suchmethode nur bei etwa 80%, da nicht garantiert werden kann, dass die Anfrage das Ziel überhaupt erreicht. [vgl. Konrad Miller, Seite: 5]

Eine weitere Alternative stellt die **Super-Peer-Methode** dar. Sie wird im Zusammenhang mit hybriden „P2P“-Systemen eingesetzt. Hierbei ist jeder Peer mit einem sogenannten „Super-Peer“ verbunden. Im Netzwerk befinden sich mehrere Super-Peers, welche eng miteinander verbunden sind. Die Daten und Informationen werden ausschließlich auf den Super-Peers abgelegt. Dadurch muss die Suchanfrage nicht an alle Peers in Netzwerk gesendet werden, sondern es ist ausreichend die Suchanfrage nur über die Super-Peers durchzuführen. [vgl. Konrad Miller, Seite: 5]

Bei größeren „P2P“-Netzwerken stoßen jedoch alle Methoden an ihre Grenzen. Daher werden in solchen Fällen meist **strukturierte „P2P“-Systeme** eingesetzt. Bei strukturierten „P2P“-Systemen ist die Topologie des Netzwerks bekannt. Dadurch ist

mithilfe von „Distributed Hash Tables“ (DHT) ein gezieltes und schnelles Suchen nach Daten möglich. Außerdem kann dadurch garantiert werden, dass die Daten gefunden werden, solange sie jedenfalls im Netzwerk verfügbar sind. Bei strukturierten „P2P“-Systemen muss jedoch ein gewisser Aufwand betrieben werden, um die Struktur erst einmal aufzubauen und anschließend dauerhaft aktuell zu halten. [vgl. Konrad Miller, Seiten: 6f, 23]

„P2P“-Systeme zeichnen sich durch folgende Vorteile aus.

- Skalierbarkeit: Dem Netzwerk können ohne großen Aufwand weitere Teilnehmer beitreten. Dadurch kann die Leistungsfähigkeit des Gesamtsystems verbessert werden
- Sicherheit: In dezentralen Systemen gibt es keinen Hauptserver, der Ziel eines Angriffs werden könnte oder der bei einem Ausfall zum Stillstand des Netzwerks. Ein Ausfall eines einzelnen Peers hat meist nur geringe Auswirkungen.
- Flexibilität: Je nach Situation können flexibel Teilnehmer hinzugefügt oder entfernt werden. [vgl. IONOS SE]

Dem gegenüber stehen folgende Nachteile:

- Aufwand: In großen Systemen ist die Verwaltung vergleichsweise aufwändig.
- Abhängigkeiten: Bei einzelnen Veränderungen an einzelnen Rechnern ist meist das ganze System betroffen.
- Rechtliche Probleme: „P2P“-Netzwerke werden vielfach für die illegale Verbreitung urheberrechtlich geschützter Inhalte genutzt. Eine Verfolgung dieser Taten erweist sich in „P2P“-Netzen als kompliziert. [vgl. IONOS SE]

Um Teil eines „P2P“-System zu werden, muss meist spezielle Software auf dem Rechner installiert werden. Aktuell werden „P2P“-Systeme vor allem in den Bereichen Filesharing, Messenger/Chats, Distributed Computing, Blockchains und in internen Netzen eingesetzt. [vgl. IONOS SE]

2.2 GUN.js und Graphdatenbanken

„Gun.js“ ist eine dezentrale Graphdatenbank. Um Gun.js besser verstehen zu können werden nun im Folgenden zunächst die Grundlagen von Graphdatenbanken erläutert.

In **Graphdatenbanken** werden die Daten in Eigenschaftsgraphen gespeichert, welche in Form von Knoten und Kanten vorliegen. Die einzelnen Knoten und Kanten werden durch Knoten- und Kantentypen näher spezifiziert. Die Daten zu einem Knoten- oder Kantentyp werden in Form von Attribut-Wert-Paaren gespeichert. Dabei werden alle Knoten sowie alle Kanten als eigener Datensatz gespeichert, wodurch Graphdatenbanken sehr schnell und effizient sind. Die Datenmanipulationen in Graphdatenbanken erfolgen durch Graphtransformationen oder durch Ändern der Attribut-Wert-Paare.

Abbildung 2: Beispiel-Graphdatenbank zeigt ein beispielhaftes Schema einer Graphdatenbank. Im Beispiel werden Benutzerkonten, Webseiten, sowie die Beziehungen untereinander dargestellt. Es gibt die zwei Knotentypen USER und WEBPAGE sowie die drei Kantentypen FOLLOWS, VISITED und CREATED_BY. Der Knotentyp USER besitzt die Attribute userName, firstName und lastName. Der Kantentyp VISITED enthält das Attribut date. [vgl. Michael Kaufmann, Andreas Meier, Seiten: 267-269]

Graphdatenbanken kommen vor allem dann zum Einsatz, wenn Daten über ein Netzwerk verteilt vorliegen. In diesen Fällen liegt der Fokus nicht auf einem einzelnen Datensatz, sondern vielmehr auf der Verknüpfung der Datensätze untereinander. Graphdatenbanken werden daher oft im Bereich der sozialen Medien oder zur Analyse der Verlinkungen von Webseiten eingesetzt.

Genau wie alle anderen Datenbanken benötigen auch Graphdatenbanken Indizes, um schnelle und gezielte Zugriffe auf einzelne Daten zu ermöglichen. Die Indizierung wird in den meisten Datenbanken über Bäume realisiert, so auch in Graphdatenbanken. In Graphdatenbanken ergibt sich jedoch ein Sonderfall. Da jeder Baum auch als Graph dargestellt werden kann, können Indizes als Graphen direkt in die Graphdatenbank eingebunden werden und stellen somit einen Teil des Gesamtgraphen dar. [vgl. Michael Kaufmann, Andreas Meier, Seiten: 269f]

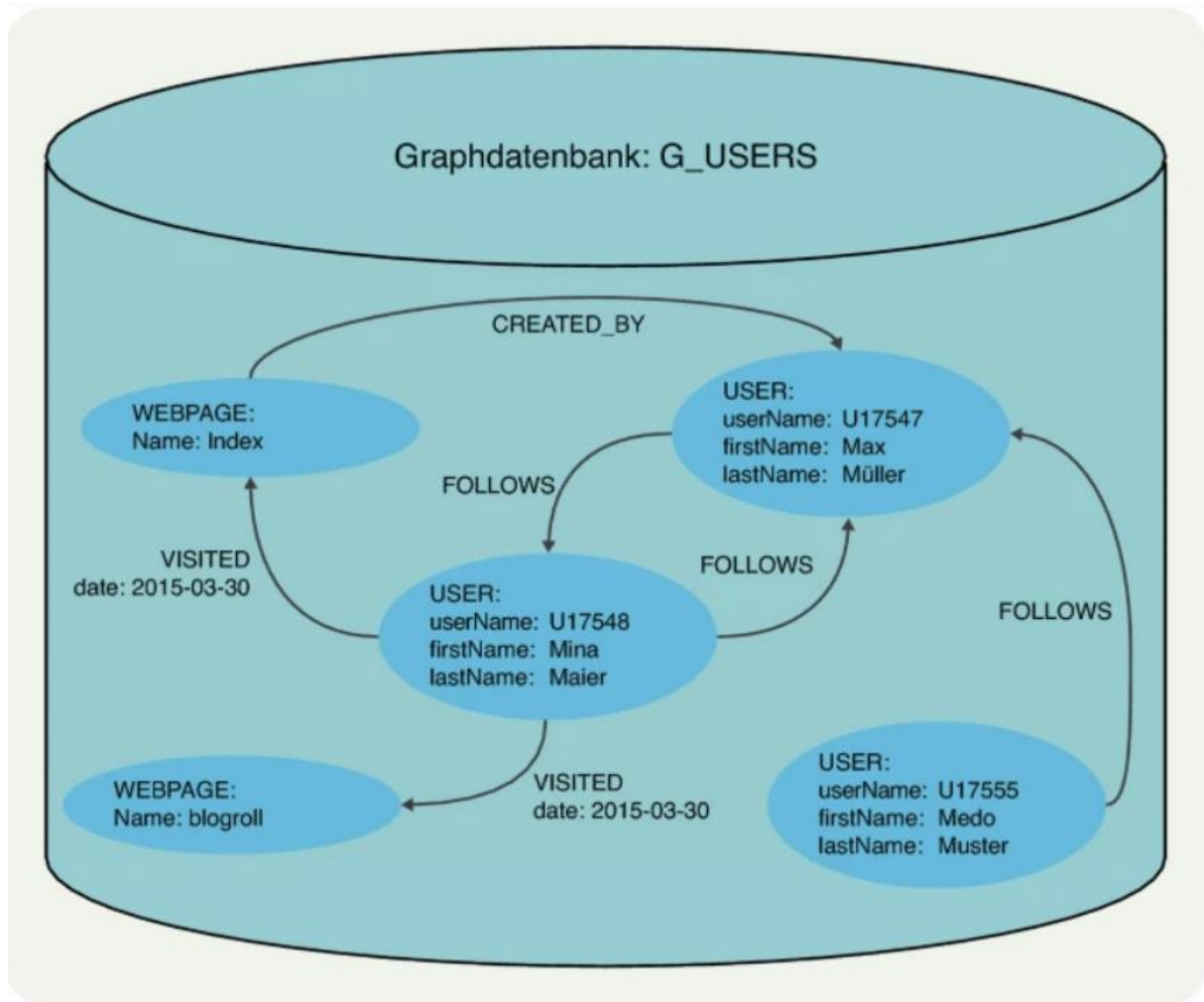


Abbildung 2: Beispiel-Graphdatenbank²

Wie bereits oben erwähnt ist **Gun.js** eine Java-Script Bibliothek für eine dezentrale Graphdatenbank, welche durch den Entwickler Mark Nadal entwickelt wurde. Anders als bei zentralen Datenbanken, welche die Daten nur an einem zentralen Ort speichern, werden die Daten bei Gun.js in einem Netzwerk auf viele verschiedene Peers oder User aufgeteilt. Dabei werden die notwendigen Daten lokal in den Browsern der einzelnen Peers gespeichert. Jeder Peer stellt einen Knoten der Graphdatenbank dar. Die Kommunikation sowie der Datenaustausch zwischen den einzelnen Peers erfolgt mit Hilfe von „Web Real-Time Communication“ (WebRTC). WebRTC ermöglicht eine Echtzeitkommunikation über Rechner-Rechner-

² [Michael Kaufmann, Andreas Meier, Seite: 269]

Verbindungen. Die Technologien und Protokolle beruhen auf einem offenen Standard und sind in allen gängigen Browsern verfügbar. Peers in Gun.js sind meist die einzelnen Rechner der Endbenutzer. Diese haben jedoch den Nachteil, dass in der Regel nur 5-10MB Speicherkapazität im Browser zur Verfügung stehen. Außerdem kann nicht garantiert werden, dass die Endbenutzer und damit auch die notwendigen Daten dauerhaft im Netzwerk verfügbar sind. Um diese Probleme zu beheben, gibt es in Gun.js jedoch die Möglichkeit zusätzlich zu den Endbenutzern einige node.js-Relay-Server in das Netzwerk einzubinden. Diese Relay-Server können weit mehr Daten speichern und es kann einfacher sichergestellt werden, dass die Daten dauerhaft verfügbar sind. Dadurch werden die Stabilität und Zuverlässigkeit des Netzwerks erhöht. [vgl. GUN-ECO]

Jeder Endbenutzer benötigt für die Verwendung der Webseite oder App Zugriff auf bestimmte andere Knoten im Netzwerk. Somit ergibt sich für jeden Benutzer ein individueller Graph, über den er seine Daten beziehen kann. Alle einzelnen Teilgraphen sind jedoch wieder über gemeinsame Knoten miteinander verbunden und stellen somit die gesamte Datenbank dar. [vgl. GUN-ECO]

2.3 Angular

Angular ist ein bekanntes, clientseitiges JavaScript-Web-Framework, das zur Erstellung von Single-Page-Webanwendungen genutzt wird. Angeführt von Google wurde das Framework aus der Vorgängerversion AngularJS weiterentwickelt. Im Zuge dessen wurde unter anderem die auf JavaScript basierende Sprache „TypeScript“ als neue Code-Basis eingeführt. Inzwischen stellt Angular auch viele Entwicklungs-Tools sowie Generatoren bereit und unterstützt ein einfaches Einbinden von Third-Party-Libraries, wodurch das Entwickeln in Angular vereinfacht wird. [vgl. Platzi IT GmbH]

3 Umsetzung

In diesem Kapitel wird die technische Umsetzung der „Peer-To-Peer“-Chat-Anwendung betrachtet.

3.1 Development- und Deploymentworkflow

Die folgende Abbildung 3: Workflow stellt den Development- und Deploymentworkflow dar.

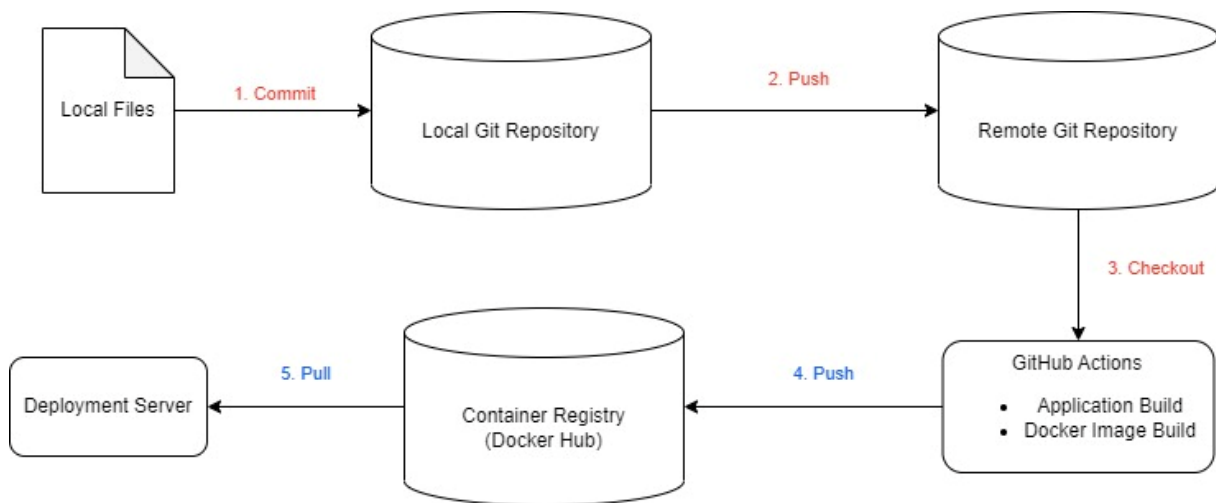


Abbildung 3: Workflow

Nach erfolgreicher lokaler Entwicklung werden die Änderungen im Git-Repository committet und in das Remote-Git-Repository übertragen.

Die Automatisierung von Continuous Integration (CI) und Continuous Deployment (CD) erfolgt durch GitHub Actions. Bei jedem Commit des Main-Banches im Remote-Git-Repository wird automatisch das Dockerfile gebaut und das entstandene Image in die Container Registry „Docker Hub“ gepusht.

Abschließend wird das aktualisierte Image von der Container Registry auf den Deployment-Server heruntergeladen, gestartet und die Webanwendung bereitgestellt.

Durch diesen durchdachten Workflow wird nicht nur die Qualität der Anwendung sichergestellt, sondern auch eine effiziente und konsistente Bereitstellung in einer verteilten Umgebung gewährleistet. Die Automatisierung durch GitHub Actions in Verbindung mit Docker ermöglicht einen reibungslosen Entwicklungs- und Bereitstellungsprozess von der lokalen Entwicklung bis zum Deployment auf dem finalen Server.

Abbildung 4: Architektur zeigt die Architektur der Anwendung.

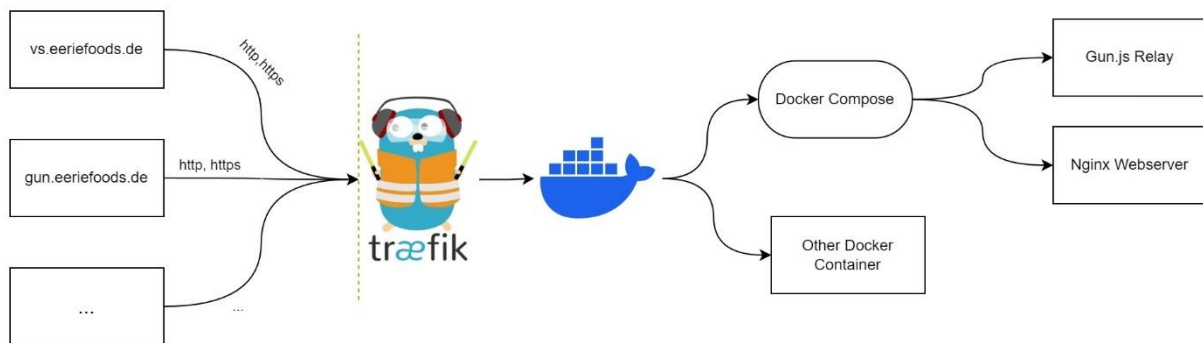


Abbildung 4: Architektur

Zur geräteunabhängigen Entwicklung und für die finale Bereitstellung von sowohl der Webanwendung als auch dem Relay-Server wird Docker Compose verwendet.

Das Docker-Compose-File orchestriert zwei Dienste: "gun" (der Relay-Server) und "chat" (die Webanwendung). Der "gun"-Dienst basiert auf dem gundb/gun-Image und wird über den Traefik-Proxy mit einem öffentlichen Host (gun.eeriefoods.de) sowie TLS-Unterstützung bereitgestellt. Ebenfalls über Traefik erfolgt die Bereitstellung des "chat"-Dienstes, der das Image der Webanwendung nutzt und einen öffentlichen Host (vs.eeriefoods.de) mit TLS-Unterstützung aufweist. Beide Dienste sind in einem dedizierten Netzwerk namens 'traefik_proxy' organisiert und verweisen extern auf den Traefik-Proxy.

Das Bauen der Webanwendung erfolgt durch das Dockerfile, welches einen zweistufigen Build-Prozess initiiert. Zunächst wird eine Node.js-21-Alpine-Umgebung erstellt, in der das Arbeitsverzeichnis auf /app festgelegt wird. Anschließend werden sämtliche Dateien in dieses Verzeichnis kopiert, die npm-Abhängigkeiten installiert und das Angular.js-Frontend gebaut. Im zweiten Abschnitt des Dockerfiles wird ein Nginx-Alpine-Image verwendet, um das zuvor erstellte Frontend in den Webserver zu integrieren. Dabei wird der Port 80 für externe Zugriffe freigegeben.

Traefik ermöglicht die nahtlose Integration von HTTPS durch automatische Zertifikatsgenerierung und -erneuerung, was die Sicherheit der Anwendungen gewährleistet. Durch die Konfiguration von Traefik wird sichergestellt, dass sämtlicher Datenverkehr zwischen Benutzern und Diensten verschlüsselt ist, wodurch die Vertraulichkeit und Integrität der übermittelten Informationen gewährleistet wird. [vgl. Traefik Maintainer Team]

Traefik leitet den eingehenden HTTPS-Verkehr dann an die entsprechenden Dienste weiter, basierend auf den definierten Regeln und Konfigurationen. Dadurch wird die Skalierbarkeit von Anwendungen verbessert, da neue Container oder Dienste dynamisch hinzugefügt werden können, ohne dass aufwändige manuelle Konfigurationen erforderlich sind.

3.2 Webanwendung

Durch das verwendete Angular-Framework teilt sich das Projekt in verschiedene Angular-Komponenten auf:

Components

Components beinhalten sowohl HTML-als auch TypeScript-Code für das Aussehen und die Funktionalität der Website.

Services

Services beinhalten Component-übergreifende Funktionalitäten wie Authentifizierung, Datenbankzugriffe oder ähnliches.

Models

Models beinhalten die Datenmodelle, welche sowohl von Services als auch Components zur Verarbeitung gebraucht werden.

Die Speicherung in der Graphdatenbank erfolgt gemäß einem definierten Schema, das in diesem Fall die Struktur eines Nachrichtenobjekts widerspiegelt:

```
{
  "TIME1":
    {
      "type": "text",
      "text": "My Message",
      "sender": "User Alias",
      "timestamp": 1603090200
    },
  ...
}
```

Hier wird jede Nachricht unter einem eindeutigen Zeitstempel abgelegt. Das Nachrichtenobjekt enthält Attribute wie den "type", der konstant auf "text" gesetzt ist, den eigentlichen Text der Nachricht ("text"), den Absender der Nachricht ("sender") und den Zeitstempel der Nachricht ("timestamp").

4 Ergebnis

Das Ergebnis dieser Projektarbeit ist die Peer-to-Peer Chatanwendung „Spaghetti-Taco“. Hierbei handelt es sich um eine Webanwendung, welche mithilfe von GUN.js und Angular erstellt wurde. Wie in Kapitel 2.2 GUN.js und Graphdatenbanken beschrieben, ist GUN.js eine dezentrale Graphdatenbank. Die Daten werden als verschiedene Knoten der Graphdatenbank auf allen Teilnehmern (Peers) im Netzwerk gespeichert. Daher kann die Chatanwendung als Peer-to-Peer System bezeichnet werden. Genauer gesagt handelt sich um ein hybrides Peer-to-Peer System (siehe Kapitel 2.1 „Peer-to-Peer“-Systeme), da neben den einzelnen Anwender-Peers noch einige Relay-Server im Netzwerk vorhanden sind. Diese Relay-Server fungieren als Super-Peers, um größere Datenmengen zu speichern und die Performance der Anwendung zu erhöhen.

Die Peer-to-Peer Chatanwendung kann über diesen Link aufgerufen und genutzt werden: <https://vs.eeriefoods.de/>

Im Folgenden werden die wesentlichen Funktionen der Chatanwendung kurz vorgestellt.

Beim Aufruf der Chatanwendung gelangen alle Benutzer zuerst auf die Login Seite, welche in Abbildung 5: Login-Fenster dargestellt ist. Dort kann der Benutzer sich mithilfe seines Benutzernamens und seines Passwortes über den Button „LOG IN“ einloggen. Neue Benutzer können sich nach Eingabe eines noch nicht vergebenen Benutzernamens über den Button „SIGN IN“ registrieren.

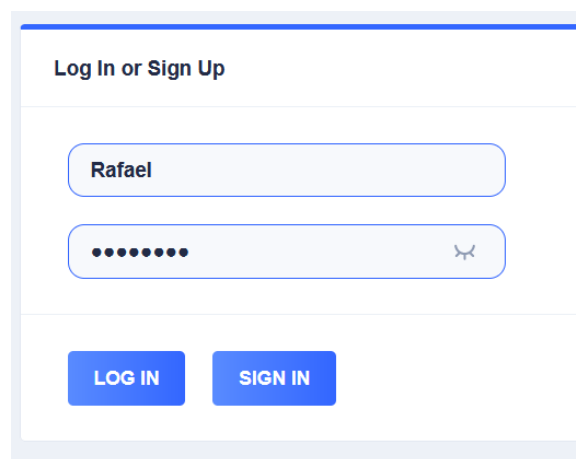


Abbildung 5: Login-Fenster

Nach erfolgreicher Anmeldung bzw. Registrierung wird der Benutzer auf die Hauptseite weitergeleitet, welche in Abbildung 6: Chatanwendung dargestellt ist.

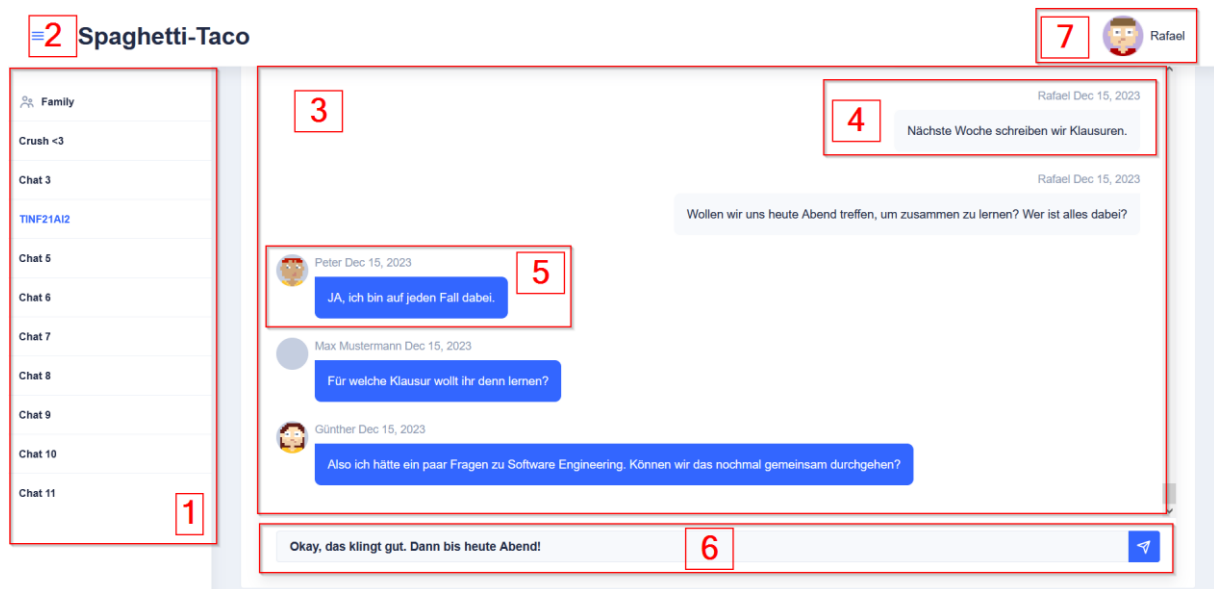


Abbildung 6: Chatanwendung

Zur Erläuterung der Funktionen:

- **1** Am linken Bildschirmrand können aus einer Liste verschiedene Chatverläufe ausgewählt werden.
- **2** Mit einem Klick auf das Hamburger-Menü links oben kann die Liste mit den Chatverläufen aus- bzw. eingeblendet werden.
- **3** In der Mitte des Fensters ist der Chatverlauf des ausgewählten Chats abgebildet.
- **4** Eigene Nachrichten werden am rechten Rand des Chatfensters dargestellt.
- **5** Nachrichten von anderen Chatteilnehmern werden am linken Rand des Chatfensters dargestellt. Sie enthalten den Namen des Absenders sowie das Datum, an dem die Nachricht geschrieben wurde.
- **6** In der Eingabeleiste am unteren Bildschirmrand, können neue Nachrichten verfasst werden. Durch einen Klick auf den blauen Button in der Eingabeleiste werden die Nachrichten versendet.
- **7** Über das Profil oben rechts kann zwischen Light- und Dark Mode unterschieden werden. Des Weiteren gibt es einen „Logout“-Button, um sich von der Seite abzumelden.

Autorenverzeichnis

Kapitel <u>1 Einleitung</u>	Rafael Simon
Kapitel <u>2 Grundlagen</u>	Rafael Simon
Kapitel <u>3 Umsetzung</u>	Jan-Luca Wolf
Kapitel <u>4 Ergebnis</u>	Rafael Simon
Quellcode	Jan-Luca Wolf

Quellenverzeichnis

1. **Peter Mahlmann, Christian Schindelhauer.** Peer-to-Peer-Netzwerke. URL: <https://link.springer.com/book/10.1007/978-3-540-33992-2>, Zugriff: 22.11.2023 : Springer, 2007.
2. **R.Clement, D.Schreiber, P.Bossauer, S.Pakusch.** Internet-Ökonomie. URL: <https://link.springer.com/book/10.1007/978-3-662-59829-0> Zugriff: 22.11.2023 : Springer Gabler, 2019.
3. **Konrad Miller.** Karlsruhe Institute of Technology. *Strukturierte Peer-to-Peer*. [Online] https://os.itec.kit.edu/downloads/Structured_P2P.pdf. Zugriff: 22.11.2023.
4. **IONOS SE.** Digital Guide. *Was ist Peer-to-Peer?* [Online] <https://www.ionos.de/digitalguide/server/knowhow/was-ist-peer-to-peer/>. Zugriff: 22.11.2023.
5. **Michael Kaufmann, Andreas Meier.** SQL- & NoSQL-Datenbanken. <https://link.springer.com/book/10.1007/978-3-662-67092-7>, Zugriff: 25.11.2023 : Springer Vieweg, 2023.
6. **GUN-ECO.** GUN Docs. *Introduction*. [Online] <https://gun.eco/docs/Introduction>. Zugriff: 25.11.2023.
7. **Platri IT GmbH.** Platri IT. *Was ist Angular?* [Online] <https://platri.de/was-ist-angular/>. Zugriff: 23.11.2023.
8. **Traefik Maintainer Team.** Traefiklabs. *Welcome*. [Online] <https://doc.traefik.io/traefik/>. Zugriff: 20.12.2023.