

FACHHOCHSCHULE ERFURT

WEB- UND INTERACTION DESIGN

Mindmap

Student:
Bilal ALNAANI

Betreuer:
Prof. Rolf KRUSE

GraphQL Full Stack mit Angular, Nest, Nx und Prisma

Web Engineering und Interaktive Media
Master Angewandte Informatik

19. Februar 2023

„Menschen mit einer neuen Idee gelten solange als Spinner, bis sich die Sache durchgesetzt hat“

Mark Twain

Danksagung

In erster Linie möchte ich mich bei Professor Kruse für die Unterstützung bei der Erfindung der Projektidee, die hilfreichen Anregungen und die freie Wahl der Techniken bedanken.

Inhaltsverzeichnis

Danksagung	ii
1 Begriffsdefinition	1
1.1 Mindmap	1
1.2 Rekrusive Datenabruf	1
1.3 KISS-Prinzip	1
2 Technologien	3
2.1 GraphQL	3
2.1.1 Datentypen	3
2.1.2 schema-first und code-first	6
2.1.3 Apollo server und Apollo client	7
2.1.4 Vergleich zwischen Rest und GraphQL	8
2.2 Typescript	8
2.3 PostgreSQL	8
2.4 Prisma ORM	9
2.5 Docker	9
2.6 NX	9
2.7 Angular	9
2.7.1 Angular material	10
2.7.2 NGX-Graph	10
2.8 Figma	10
3 Konzept	11
3.1 Idee und Motivation	11
3.2 Design	12
3.2.1 Auswahl der Farbe und der Farbtöne	12
4 Umsetzung	13
4.1 Aufbau	13
4.2 Nutzung	13
Wurzelknoten erzeugen	13
Neuen KindKnote erzeugen	14
Neuen KindKnote löschen	14
4.2.1 Systemmerkmale	14
4.3 Features	14
5 Zusammenfassung	16
A Abbildungen	17

Abbildungsverzeichnis

2.1	Listen und Nicht-Null-Typen	4
2.2	Objektyp	4
2.3	Enum Beispiel	5
2.4	Query-Typ Beispiel	5
2.5	Mutation-Typ Beispiel	6
2.6	Subscription-Typ Beispiel	6
2.7	Schema First vs. Code First	7
3.1	Verwendete Farben	12
4.1	Rekursive Datenbankruf	13
A.1	Figma Design	17
A.2	Figma Design	17
A.3	Figma Design	18
A.4	Chatroom aktualisieren	18
A.5	Bestätigungsdialog	19
A.6	Chatroom	19
A.7	Chatroom anlegen	20
A.8	Chatroomliste	20
A.9	Registrieren	21
A.10	Anmelden	21
A.11	Profile	22

Kapitel 1

Begriffsdefinition

1.1 Mindmap

Eine Mindmap ist eine grafische Darstellung von Ideen und Informationen, die in Form eines Diagramms organisiert sind. Es ist ein nützliches Werkzeug zur Visualisierung von Gedanken und Konzepten, die in Beziehung zueinander stehen.

Mindmaps sind in der Regel kreisförmig oder baumartig aufgebaut und bestehen aus einem zentralen Thema oder Hauptgedanken, der mit Zweigen oder Ästen verbunden ist, die weitere Ideen oder Unterkategorien darstellen.

Mindmaps werden häufig verwendet, um komplexe Themen, Konzepte oder Probleme zu organisieren und zu vereinfachen. Sie sind besonders nützlich für die Zusammenfassung von Informationen, die während des Studiums oder bei der Planung eines Projekts gesammelt wurden. Mit einer Mindmap können Sie Ihre Gedanken strukturieren und ein visuelles Bild davon erstellen, wie verschiedene Ideen und Konzepte miteinander verbunden sind.

1.2 Rekrusive Datenabruf

Ein rekursiver Datenabruf in einer Datenbank bedeutet, dass eine Abfrage so strukturiert ist, dass sie sich selbst wiederholt, um Daten aus verschiedenen Ebenen oder Hierarchien zu sammeln.

In SQL kann ein rekursiver Datenabruf mit der WITH RECURSIVE-Klausel ausgeführt werden. Diese Klausel ermöglicht es, eine Abfrage zu schreiben, die sich selbst aufruft, um weitere Datensätze abzurufen.

1.3 KISS-Prinzip

ist ein Design-Prinzip welches in den USA in den 60er Jahren entstand. KISS ist ein Akronym für "keep it simple, stupid". Das Prinzip fordert dazu auf die einfachste Lösung zu einem Problem zu finden. Dabei kann das Prinzip der Einfachheit sowohl das Mittel als auch das Ziel sein. Es gibt verschiedene Bereiche die das KISS-Prinzip einsetzen z. B. das Webdesign, das Produktdesign und die Softwareentwicklung.

Das KISS-Prinzip hat mehrere Vorteile, unter anderem:

- Usability: Ein einfaches und leicht verständliches Produkt hat einen höheren Nutzwert für den User
- Verständlichkeit: Projekte/Produkte werden überschaubarer, wartungsfreundlicher und somit einfacher zu verstehen

- Flexibilität: ein einfaches Produkt kann flexibler geändert werden

Kapitel 2

Technologien

In diesem Kapitel werden zentrale Begriffe erklärt, aktuelle Technologien zum Thema vorgestellt und wichtige Zusammenhänge dargestellt, um die Folgekapitel besser verstehen zu können.

2.1 GraphQL

GraphQL ist eine Open-Source-Abfragesprache und Laufzeitumgebung für APIs, die 2012 von Facebook entwickelt wurde. Im Wesentlichen ermöglicht es Entwicklern, benutzerdefinierte Abfragen an APIs zu erstellen, um nur die benötigten Daten abzurufen und die Struktur der zurückgegebenen Daten zu definieren. Es ist eine Alternative zu REST (Representational State Transfer) als Architekturstil für Web-Services und wird von vielen Unternehmen, darunter auch großen Playern wie Github, Netflix und Shopify, eingesetzt.

Die GraphQL-Syntax besteht aus Typen, Feldern und Argumenten. Ein Typ definiert ein Objekt, das von der API zurückgegeben wird, und seine Felder definieren, welche Daten in diesem Objekt enthalten sind. Ein Argument wird verwendet, um eine Abfrage zu verfeinern, indem es beispielsweise spezifische Filterbedingungen definiert.

Außerdem müssen passend zu jedem Feld Funktionen implementiert werden, die sich um das Bereitstellen der spezifischen Daten kümmern. Diese Funktionen werden auch als **Resolver** bezeichnet, um die angeforderten Daten aus verschiedenen Datenquellen zu sammeln und zu verarbeiten.

Resolver sind in der Regel Teil des Servercodes und werden in der Programmiersprache des Servers geschrieben. In einem Resolver kann man eine Datenbankabfrage, eine API-Anfrage oder jede andere Art von Datenabruf logisch implementieren.

Resolver können auch andere Resolver aufrufen, um komplexe Abhängigkeiten zwischen Datenfeldern zu behandeln. Resolver in GraphQL sind auch verantwortlich für die Validierung von Abfrageparametern und die Verarbeitung von Fehlern. Ein Resolver kann einen Fehler zurückgeben, wenn die Daten nicht verfügbar sind oder wenn ein anderer Fehler aufgetreten ist, was der Client-Anwendung ermöglicht, Fehlermeldungen anzuzeigen oder auf andere Weise darauf zu reagieren.

2.1.1 Datentypen

Es gibt verschiedene **Datentypen**, die die Felder darstellen können. Die am häufigsten verwendeten Datentypen sind:

- **Skalare Datentypen** sind der einfachste Typ und enthält einen Wert von einem bestimmten Typen, welcher immer mit angegeben werden muss.
 - String
 - Int
 - Float
 - Boolean
 - ID: Ein eindeutiger Bezeichner (oft als String dargestellt)
- **Listen und Nicht-Null-Typen:** Neben Skalaren können auch komplexe Datentypen definiert werden.
 - Listen sind eine Sammlung von Werten desselben Datentyps. Sie werden in GraphQL mit eckigen Klammern [] dargestellt. z.B [String]
 - Nicht-Null-Typen: Ein Nicht-Null-Typ stellt einen Wert dar, der immer vorhanden sein muss und nicht null sein darf. Sie werden in GraphQL mit einem Ausrufezeichen ! dargestellt. z.B String!.

```
type Mindmap {  
  id: String!  
  title: String!  
  parent_id: String  
  children: [Mindmap!]!  
  chatroom_id: String  
}
```

ABBILDUNG 2.1: Listen und Nicht-Null-Typen

- **Objekttypen** stellen komplexe Datentypen dar, die aus einer Sammlung von Feldern bestehen. Sie werden verwendet, um Beziehungen zwischen verschiedenen Entitäten zu definieren

```
type User {  
  id: String!  
  firstname: String  
  lastname: String  
  email: String!  
  password: String!  
  deleted: Boolean!  
  role: UserRole!  
  chatroom: Chatroom  
}
```

ABBILDUNG 2.2: Objekttyp

- **Enumerationstypen** stellen eine feste Liste von Werten dar. Sie können verwendet werden, um die zulässigen Werte für ein Feld einzuschränken.

Ein weiterer Vorteil von GraphQL ist die Möglichkeit, die Struktur der Daten zu definieren, die von Usern abgerufen werden, anstatt sich auf die Struktur der vom

```
enum UserRole {  
    USER  
    ADMIN  
}
```

ABBILDUNG 2.3: Enum Beispiel

Server zurückgegebenen Daten zu verlassen. Dies ermöglicht eine bessere Kontrolle darüber, welche Daten abgerufen werden, und minimiert die Wahrscheinlichkeit von Overfetching oder Underfetching.

Außerdem ermöglicht es, die Beziehungen zwischen Daten in einer API zu modellieren und Abfragen zu verschachteln, um Daten von mehreren Objekten abzurufen. Dies bedeutet, dass komplexe Abfragen erstellt werden können, ohne dass mehrere Anfragen an die API gesendet werden müssen.

Ein weiteres wichtiges Konzept ist das Schema, das als Vertrag zwischen der API und dem Client dient. Es definiert die Typen, die in der API verfügbar sind, und wie sie miteinander verknüpft sind. Anhand des Schemas können die Nutzer klar erkennen, welche Daten in der API verfügbar sind und wie sie darauf zugreifen können.

Im Gegensatz zu REST-basierten APIs, bei denen Daten in vordefinierten Endpunkten abgerufen werden, ermöglicht GraphQL, Abfragen an eine einzige Endpunkt-URL zu senden, die dann die angeforderten Daten zurückgibt. Dies macht GraphQL besonders nützlich für mobile Anwendungen und Single-Page-Anwendungen, bei denen es wichtig ist, die Datenmenge, die über das Netzwerk gesendet wird, zu minimieren.

Insgesamt bietet GraphQL eine leistungsfähige Möglichkeit, maßgeschneiderte Abfragen an APIs zu erstellen und eine bessere Kontrolle über die zurückgegebenen Daten zu haben.

Eine GraphQL-Abfrage beginnt immer mit einem Operationstypen, der entweder eine Abfrage (siehe [Abbildung 2.4](#)), eine Mutation oder eine Abonnement (subscription) sein kann. Die **Abfrage** selbst definiert dann die erforderlichen Felder und Argumente, sie entspricht dem Verb get in Rest-API.

```
query getRoom($id: String!){  
  chatroom(id:$id){  
    id,  
    name,  
    type,  
    users{  
      id,  
      firstname,  
      lastname,  
      email,  
    }  
  }  
}
```

ABBILDUNG 2.4: Query-Typ Beispiel

In diesem Beispiel wird eine Abfrage an den Server gesendet, um die Daten eines bestimmten Raums abzurufen, einschließlich Name, ID, Typ und alle Benutzer mit

der ID, Vorname, Nachname und E-Mail. Die Antwort des Servers enthält genau die angeforderten Daten in der angeforderten Struktur.

Eine **Mutation** ist eine Operation, mit der Daten auf dem Server verändert, gelöscht oder erstellt werden können. Im Gegensatz zur Abfrage (Query), die nur lesend ist, können Mutationen schreibend sein und Änderungen an Daten auf dem Server vornehmen (siehe Abbildung 2.5). Wie auch bei einer Query muss eine Mutation immer einen Namen und mindestens ein Feld haben. Eine einfache Mutation könnte wie folgt aussehen:

```
mutation createMap($id: String!){
  createMindmap(
    createMindmapInput: {
      title: "new child",
      parent_id: $id
    }
  ){
    title,
    parent_id
  }
}
```

ABBILDUNG 2.5: Mutation-Typ Beispiel

In diesem Beispiel wird eine Mutation mit dem Namen “createMindmap” ausgeführt, die eine neue Mindmap-Karte mit einem Titel erstellt. Die Server-Antwort enthält die Parent-ID des neuen Beitrags sowie dessen Titel.

Subscription ist eine Operation, die es Clients ermöglicht, in Echtzeit auf Änderungen auf dem Server zu reagieren. Eine einfache Subscription könnte wie folgt aussehen:

```
Subscription {
  newMessage(roomId: "123"){
    content,
    from,
    roomId
  }
}
```

ABBILDUNG 2.6: Subscription-Typ Beispiel

In diesem Beispiel sind die Nachrichten auf einen bestimmten Raum mit der ID “123” abonniert. Wenn eine neue Nachricht an diesen Raum gesendet wird, sendet der Server eine Benachrichtigung an den Client und gibt die Raum-ID und den Inhalt der neuen Nachricht sowie die Id der Absender zurück.

2.1.2 schema-first und code-first

In der Entwicklung von GraphQL-basierten APIs gibt es zwei Ansätze, um das Schema der API zu definieren: Schema-First und “Code-First”.

Schema-First: Bei diesem Ansatz wird das Schema der GraphQL-API zuerst entworfen und in einer SDL-Datei (Schema Definition Language) festgelegt.

In dieser SDL-Datei werden die Datenobjekte, Abfragetypen, Mutationen und Typdefinitionen für die API definiert. Dann wird auf Basis dieses Schemas der gesamte Code für den Server generiert, der die API implementiert.

Dies kann entweder manuell oder automatisch mit Hilfe von Tools wie Apollo CLI oder Prisma erfolgen. Der Schema-First-Ansatz erleichtert die Erstellung und Wartung der API, da das Schema die Grundlage für die gesamte API bildet und somit leichter verständlich und verwaltbar ist.

Code-First: Beim Code-First-Ansatz wird der Server-Code zuerst geschrieben, um die Datenquellen und die Geschäftslogik der API zu implementieren. Dann wird das Schema der API aus dem Code generiert.

Dies geschieht oft automatisch mit Hilfe von Tools wie GraphQL Nexus oder TypeGraphQL, die es dem Entwickler ermöglichen, das Schema durch einfaches Schreiben von Code zu definieren.

Der Code-First-Ansatz ist hilfreich, wenn es darum geht, die API-Logik schnell zu implementieren und dabei die Kontrolle über die Details zu behalten. Im Mindmap-Projekt wird der Code First-Ansatz verwendet.

Beide Ansätze haben Vor- und Nachteile und die Wahl zwischen ihnen hängt von den Anforderungen des Projekts ab. Schema-First kann dabei helfen, eine klare Trennung zwischen der API-Spezifikation und der Implementierung zu schaffen, während Code-First eine schnellere Iteration in der API-Entwicklung ermöglicht.

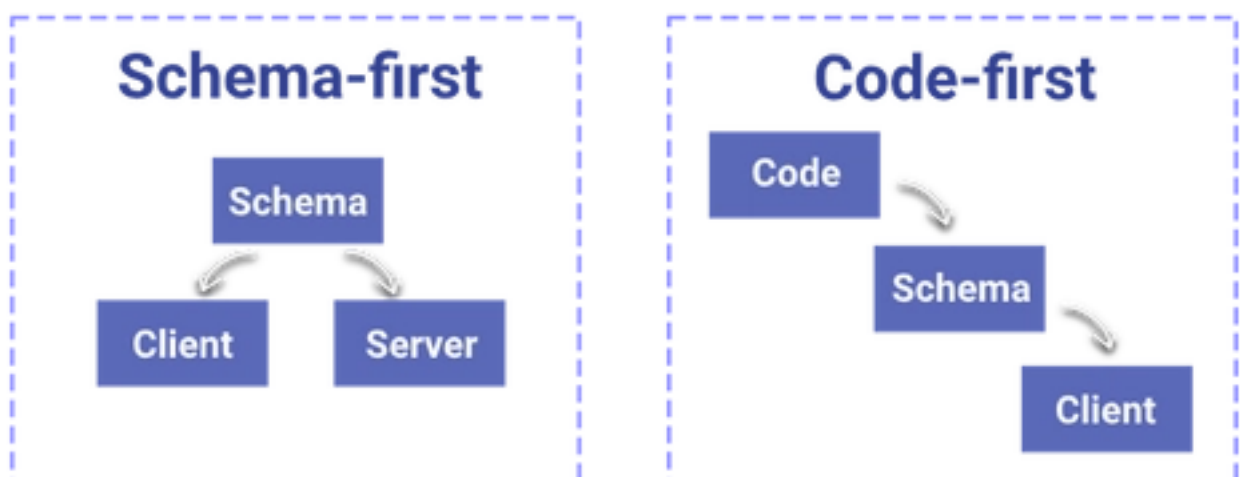


ABBILDUNG 2.7: Schema First vs. Code First

2.1.3 Apollo server und Apollo client

Apollo-Server: Ein Apollo-Server implementiert das GraphQL-API-Schema und bietet Endpunkte an, um Daten abzurufen und zu aktualisieren. Der Server kann Daten aus verschiedenen Quellen wie Datenbanken, externen APIs oder anderen Datenquellen sammeln und sie in einem einzigen GraphQL-Schema zusammenführen.

Apollo-Client: Ein Apollo-Client ist eine Bibliothek, die es ermöglicht, Abfragen und Mutationen an den GraphQL-Server zu senden und die Daten in der Client-Anwendung zu verarbeiten. Der Client nutzt das Schema der API, um die benötigten Daten zu definieren und zu abonnieren.

2.1.4 Vergleich zwischen Rest und GraphQL

Rest (Representational State Transfer)

- Größe und Form des Ergebnisses wird vom Server vorgegeben.
- Multiple Endpoints call
- Datenabruf durch CRUD-Operationen an verschiedenen Endpoints
- Ein Routing-Handler pro Aufruf

GraphQL

- Größe und Form des Ergebnisses wird vom Benutzer vorgegeben.
- Single-Request-Prinzip
- Eine URL (Datenabrufe nach Schlüsselwörtern)
- Mehrere Resolver pro Aufruf

2.2 Typescript

TypeScript ist eine typisierte Programmiersprache, die auf JavaScript aufbaut. Sie wurde von Microsoft entwickelt und ist eine Open-Source-Sprache. TypeScript bietet zusätzliche Funktionen und Verbesserungen gegenüber JavaScript, darunter Typisierung, erweiterte Klassen und Schnittstellen, Dekoratoren, Optionale Parameter und mehr.

Die Typisierung ist eine der wichtigsten Funktionen von TypeScript. Sie ermöglicht es, Variablen, Funktionen und Objekten Typen zuzuordnen, um sicherzustellen, dass der Code zur Kompilierzeit funktioniert. Durch die Typisierung können potenzielle Fehler im Code erkannt werden, bevor sie zur Laufzeit auftreten. Dies verbessert die Wartbarkeit und Qualität des Codes.

2.3 PostgreSQL

PostgreSQL ist ein leistungsfähiges relationales Datenbanksystem, das Open-Source und kostenlos verfügbar ist. Es wurde erstmals 1986 als PostgreSQL-Projekt an der University of California in Berkeley gestartet und hat sich seitdem zu einer der beliebtesten und fortschrittlichsten Open-Source-Datenbanken entwickelt.

PostgreSQL bietet eine umfassende Palette an Funktionen und Möglichkeiten für Datenbankentwickler und -administratoren. Es unterstützt alle wichtigen SQL-Funktionen, einschließlich Joins, Unterabfragen und Transaktionen, sowie eine Vielzahl von Erweiterungen und Add-Ons für erweiterte Funktionen.

Es bietet eine hervorragende Skalierbarkeit und unterstützt parallele Abfragen für eine bessere Leistung und Geschwindigkeit.

PostgreSQL ist plattformunabhängig und läuft auf den meisten Betriebssystemen, einschließlich Windows, Linux, macOS und Unix. Es wird von einer großen und aktiven Community unterstützt, die regelmäßig neue Funktionen und Verbesserungen bereitstellt.

2.4 Prisma ORM

Prisma ist ein Open-Source-ORM (Object-Relational Mapping)-Tool, das eine einfachere Datenbankabstraktion und -verwaltung in modernen Webanwendungen ermöglicht. Es ist in TypeScript geschrieben und bietet eine benutzerfreundliche API zum Arbeiten mit Datenbanken.

Prisma bietet eine datenbankunabhängige Schnittstelle, die es Entwicklern ermöglicht, Datenbankmodelle in einer DSL (Domain-Specific Language) zu definieren und dann Datenbankabfragen in nativem Code zu schreiben, wobei die Abstraktion der zugrunde liegenden Datenbanktechnologie durch Prisma übernommen wird. Prisma unterstützt eine Vielzahl von Datenbanken, darunter PostgreSQL, MySQL und SQLite.

Prisma ist besonders nützlich für Webentwickler, die moderne Backend-Technologien wie GraphQL verwenden, da es nahtlos in den GraphQL-Resolver-Code integriert werden kann und so eine einfache Verbindung zwischen Datenbanken und GraphQL-Schemas ermöglicht.

2.5 Docker

Docker ist eine Open-Source-Plattform, die es ermöglicht, Anwendungen in isolierten Containern auszuführen. Docker-Container sind einfach zu erstellen, zu verteilen und zu verwalten. Sie können in jeder beliebigen Umgebung ausgeführt werden, die Docker unterstützt. Docker ist plattformübergreifend und kann auf Windows, macOS und Linux ausgeführt werden.

2.6 NX

NX ist ein Open-Source-Tool für die Verwaltung von monolithischen und modular aufgebauten Anwendungen in einer einzigen Codebasis. Es ist ein sogenanntes "Dev-Tool", das speziell für die Verwaltung großer Codebasen entwickelt wurde, die aus mehreren Anwendungen oder Modulen bestehen.

Das NX-Tool bietet eine Reihe von Funktionen, die die Arbeit mit großen Codebasen erleichtern sollen. Zum Beispiel bietet es eine schnelle und effektive Möglichkeit, Codeänderungen in verschiedenen Anwendungen oder Modulen vorzunehmen, indem es Abhängigkeiten automatisch aktualisiert und Tests durchführt, um sicherzustellen, dass die Änderungen keine unerwünschten Nebenwirkungen haben. NX unterstützt auch verschiedene Sprachen und Frameworks, darunter Angular, React, Node.js, NestJS, Express, und mehr.

2.7 Angular

Angular ist ein Open-Source-Framework zur Entwicklung von Webanwendungen. Es wurde von Google entwickelt und ist in der Programmiersprache TypeScript geschrieben.

Angular basiert auf dem Konzept der Komponenten, wobei jede Komponente eine bestimmte Funktion in der Anwendung erfüllt. Diese Komponenten können miteinander interagieren, um eine dynamische und reaktionsfähige Benutzeroberfläche zu erstellen.

Das Framework bietet eine umfangreiche Funktionalität für die Entwicklung von Single-Page-Anwendungen, einschließlich Datenbindung, Routing, Formularverarbeitung und serverseitiger Kommunikation. Es ist auch einfach erweiterbar durch das Hinzufügen von Drittanbieter-Modulen und Plug-ins.

Angular hat sich in den letzten Jahren zu einem der beliebtesten Frameworks für die Entwicklung von Webanwendungen entwickelt und wird von vielen großen Unternehmen und Entwicklern weltweit genutzt.

2.7.1 Angular material

Angular Material ist eine von Google entwickelte Designbibliothek für Angular-Anwendungen. Es bietet vorgefertigte UI-Komponenten, die auf den von Google empfohlenen Material Design-Spezifikationen basieren, einschließlich Schaltflächen, Formularelementen, Dialogfeldern, Tabellen, Karten und vielem mehr.

Angular Material ermöglicht es Entwicklern, ansprechende und konsistente Benutzeroberflächen für ihre Anwendungen zu erstellen, indem sie eine einheitliche Designsprache und einheitliche Interaktionen für die Benutzer bieten. Die Komponenten sind responsiv und können an verschiedene Bildschirmgrößen und Geräte angepasst werden. Darüber hinaus bietet Angular Material verschiedene Themen und Farbpaletten, mit denen Entwickler das Aussehen ihrer Anwendung anpassen können.

2.7.2 NGX-Graph

NGX-Graph ist eine Open-Source-Bibliothek für Angular, die es Entwicklern ermöglicht, benutzerdefinierte Diagramme und Grafiken in Angular-Anwendungen zu erstellen.

Die Bibliothek basiert auf D3.js, einer mächtigen JavaScript-Bibliothek für die Visualisierung von Daten. Es bietet eine Vielzahl von Diagrammtypen, darunter Balkendiagramme, Liniendiagramme, Mindmaps und mehr. Die Bibliothek verfügt über eine Vielzahl von Anpassungsmöglichkeiten, mit denen das Aussehen und Verhalten der Diagramme an die spezifischen Anforderungen angepasst werden können. Es ist auch möglich, benutzerdefinierte Diagrammstile zu erstellen und diese mit der Bibliothek zu integrieren.

2.8 Figma

Figma ist ein browserbasiertes Design- und Prototyping-Tool, das von einem Unternehmen mit demselben Namen entwickelt wurde. Es ermöglicht es Designern, benutzerfreundliche Benutzeroberflächen, Grafiken und Vektor-Illustrationen zu erstellen und zu bearbeiten.

Es bietet eine Vielzahl von Funktionen, die die Zusammenarbeit erleichtern, einschließlich Echtzeit-Zusammenarbeit und Kommentierung. Dies ermöglicht es Designern und anderen Teammitgliedern, in Echtzeit an einem Design oder Prototypen zu arbeiten und Feedback zu geben.

Figma ist bekannt für seine leistungsstarke Funktion zum Erstellen von Prototypen. Designer können interaktive Prototypen erstellen, um Benutzerverhalten zu simulieren und zu testen, bevor das Design in eine voll funktionsfähige Web- oder Mobilanwendung umgesetzt wird.

Kapitel 3

Konzept

3.1 Idee und Motivation

Die Hauptidee für das Projekt ist es , eine Anwendung zu entwickeln, die es ermöglicht, im Live-Chat durch die Eingabe von Schlüsselwörtern eine Meindmap (Siehe [1.1](#))) zu erstellen.

Die Hauptmotivation für mich aus technischer Sicht war es, GraphQL mit ihren Ansätzen zu erkunden und mit dem Rest zu vergleichen, insbesondere die Verwendung von GraphQL mit NestJS, Angular und einigen anderen.

3.2 Design

Bei der Gestaltung der Mindmap-Anwendung habe ich mich für das KISS-Prinzip entschieden (Siehe [1.3](#)), da der Schwerpunkt der Anwendung auf der Funktionalität und nicht auf dem Design liegen sollte (Siehe [A.1](#)).

3.2.1 Auswahl der Farbe und der Farbtöne

Bei der Farbgestaltung habe ich großen Wert darauf gelegt, dass nicht nur die Farbkombination harmonisch sein sollte, sondern dass die verschiedenen Objekte durch die Akzentfarbe und den Farbkontrast voneinander hervorgehoben werden können.

Ich habe Blau als Akzentfarbe gewählt, weil Blau eine sehr ruhige, angenehme Farbe ist und sie durch die anderen Komplementärfarben gut hervorgehoben werden kann.

Um einen guten Farbkontrast zu erzielen, habe ich Weiß und Schwarz als Komplementärfarben eingesetzt.

Um ein gutes Nutzererlebnis zu gewährleisten, wurden die Farben Rot und Grün für die Benutzerhinweise, die Bestätigung und die Warnung verwendet. dies hilft, den Nutzer auf mögliche Risiken aufmerksam zu machen oder Warnungen zu übermitteln.



ABBILDUNG 3.1: Verwendete Farben

Der Designentwurf wurde mithilfe des Prototyping-Tools Figma (Siehe [2.8](#)) erstellt

Kapitel 4

Umsetzung

4.1 Aufbau

Die Hauptfunktion der Anwendung besteht darin, Knoten in Echtzeit zu erstellen und zu speichern. Da eine Mind Map eine Baumstruktur hat, d.h. jeder Knoten kann ein Kind- und ein Elternknoten sein, muss diese Struktur bei der Speicherung und beim Abruf der Daten berücksichtigt werden, damit die MindMap korrekt angezeigt werden kann.

Um dies zu bewältigen, musste eine rekursive Datenabfrage (Siehe 1.2) in der Datenbank durchgeführt werden. Diese Abfrage sieht in der Mindmap-Anwendung so aus:

```
WITH RECURSIVE mindmap_tree AS (
  SELECT id, title, parent_id, chatroom_id
  FROM "Mindmap"
  WHERE parent_id = \${parent_id}
  UNION ALL
  SELECT c.id, c.title, c.parent_id, c.chatroom_id
  FROM "Mindmap" c
  INNER JOIN mindmap_tree ct ON ct.id = c.parent_id
)
SELECT DISTINCT *
FROM mindmap_tree;';
```

ABBILDUNG 4.1: Rekursive Datenbankabfrage

Hier wird die Mindmap-Tabelle durchlaufen und nach den Kindern der eingegebenen parent_id durchsucht.

4.2 Nutzung

Die Mindmap-Anwendung bietet die Möglichkeit, während eines Live-Chats Knoten zu erzeugen oder zu löschen. Dazu muss man ein paar Schlüsselwörter kennen.

Wurzelknoten erzeugen

Um den Wurzelknoten zu erzeugen, sollte man einfach ein "#" mit dem Knotennamen eingeben. Dadurch wird automatisch ein neuer Knoten mit dem eingegebenen

Namen erzeugt.

z.B. #root

Neuen KindKnote erzeugen

Um einen Kindknoten zu erzeugen, muss man einfach einen “#” mit dem Namen des Elternknotens und dann einen “#” mit dem Namen des neuen Knotens eingeben.

Wenn man zwischen den beiden Knotennamen einen Text geschrieben hat, wird dieser vom System ignoriert.

z.B. #root child

Neuen KindKnote löschen

Um einen untergeordneten Knoten zu löschen, muss man einfach einen “# delete_” mit dem Namen des übergeordneten Knotens, gefolgt von einem “_” dem Namen des Knotens, eintippen.

Das System zeigt dem Benutzer eine Warnung an, wenn der gewünschte Knoten nicht gefunden wird.

z.B. #delete_root_child

4.2.1 Systemmerkmale

- Das System beachtet die Groß- oder Kleinschreibung nicht
- Der Elternknoten kann nicht gelöscht werden
- Da die Anwendung mit einem Websocket verbunden ist, muss die Seite nicht aktualisiert werden, um ein Ergebnis zu sehen.
- für jeden Chatraum kann nur eine Mindmap erstellt werden.
- Der Name eines Knotens darf keinen Zeilenumbruch, Unterstrich oder Leerzeichen enthalten.

4.3 Features

- Authentifizierung (Anmelden (siehe [A.10](#)), Registrieren (siehe [A.9](#)))
- Profil bearbeiten und Löschen (siehe [A.11](#))
- Lokalisierung (englisch, Deutsch)
- Chatroom anlegen, updaten und löschen (siehe [A.7](#))
- Knoten anlegen und löschen durch tippen
- Validierung der Knoten vor der Erstellung

- Privater, öffentlicher Live chat (siehe [A.6](#))

Kapitel 5

Zusammenfassung

Die Verwendung von GraphQL hat sich als äußerst entwicklerfreundlich erwiesen und GraphQL kann als zukünftige Konstante im Bereich der API-Entwicklung voll und ganz bestätigt werden.

Es mag sein, dass sich ihre Vorteile bei kleinen APIs kaum bemerkbar machen und die Lernkurve und das Umdenken für den Entwickler zu Beginn sehr anspruchsvoll sind, aber sie ist eine leistungsstarke Alternative zu herkömmlichen REST-APIs.

Anhang A

Abbildungen

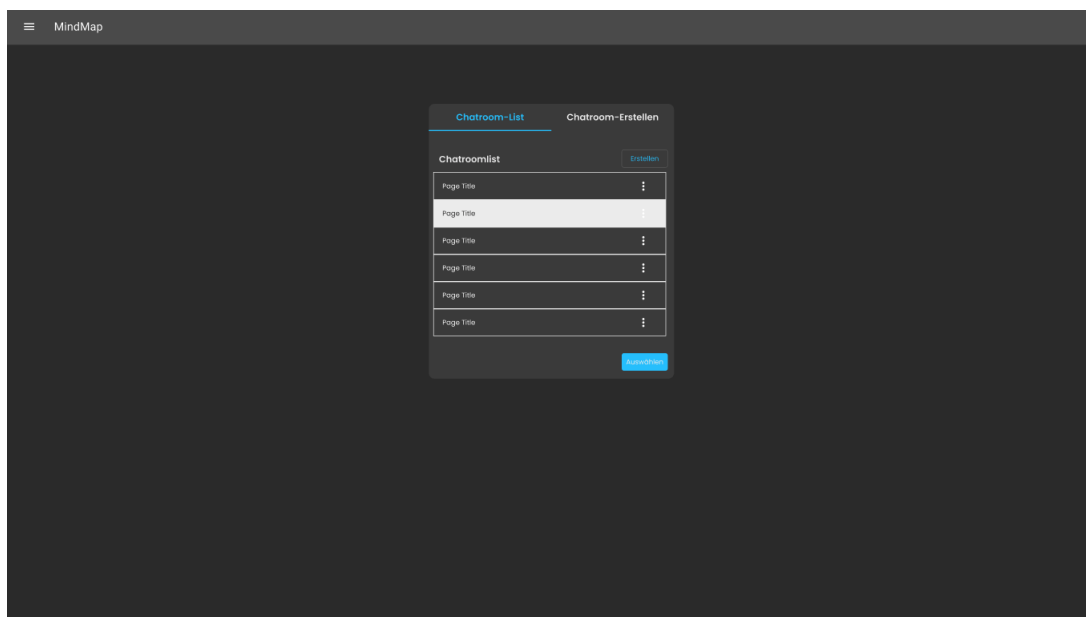


ABBILDUNG A.1: Figma Design

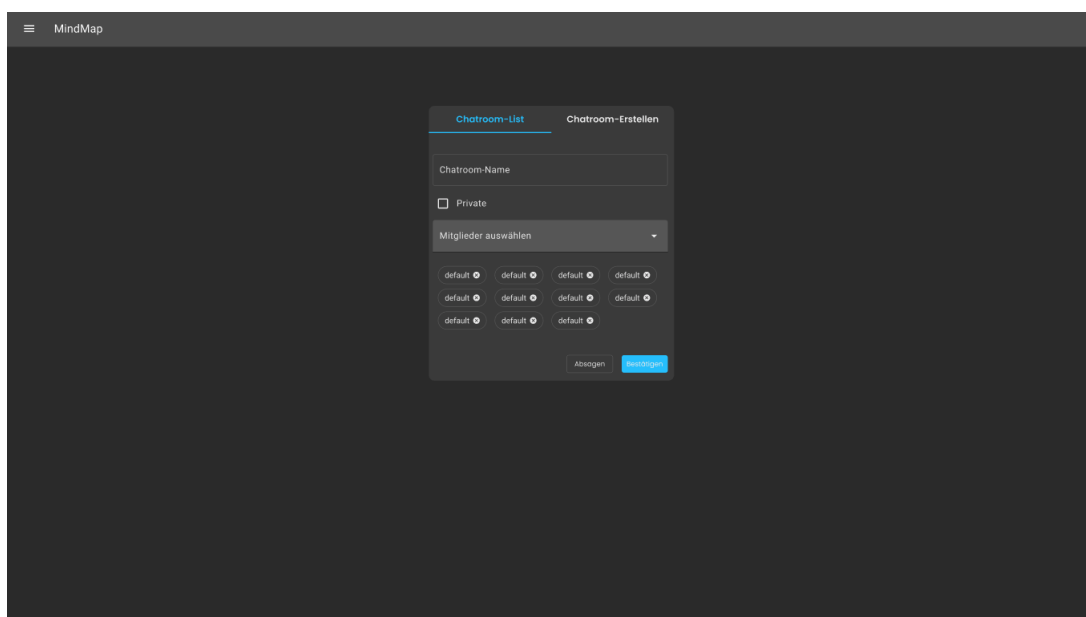


ABBILDUNG A.2: Figma Design

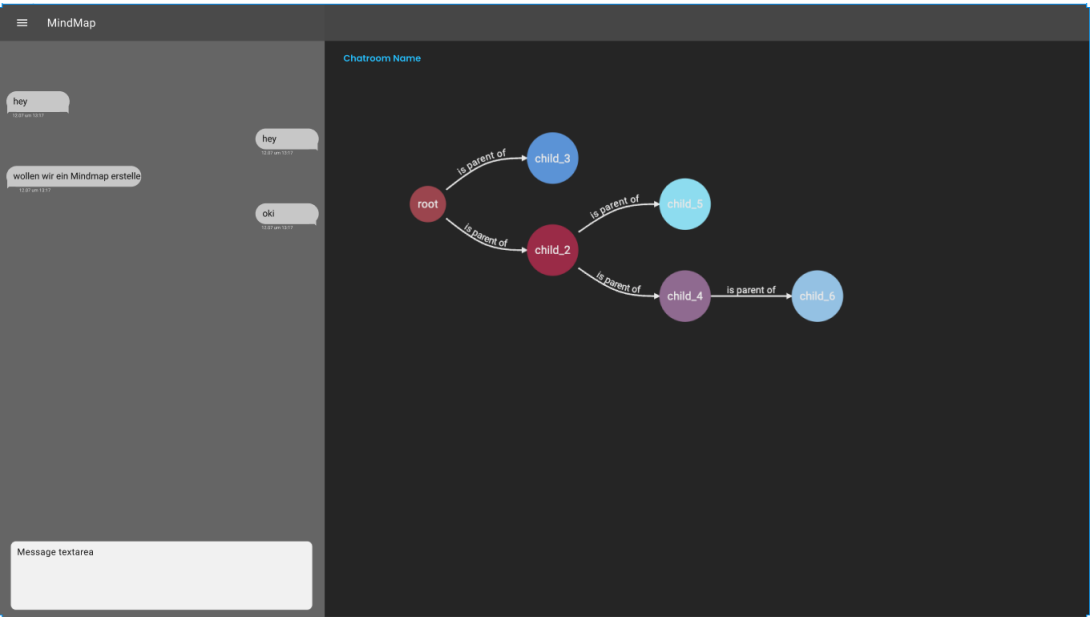


ABBILDUNG A.3: Figma Design

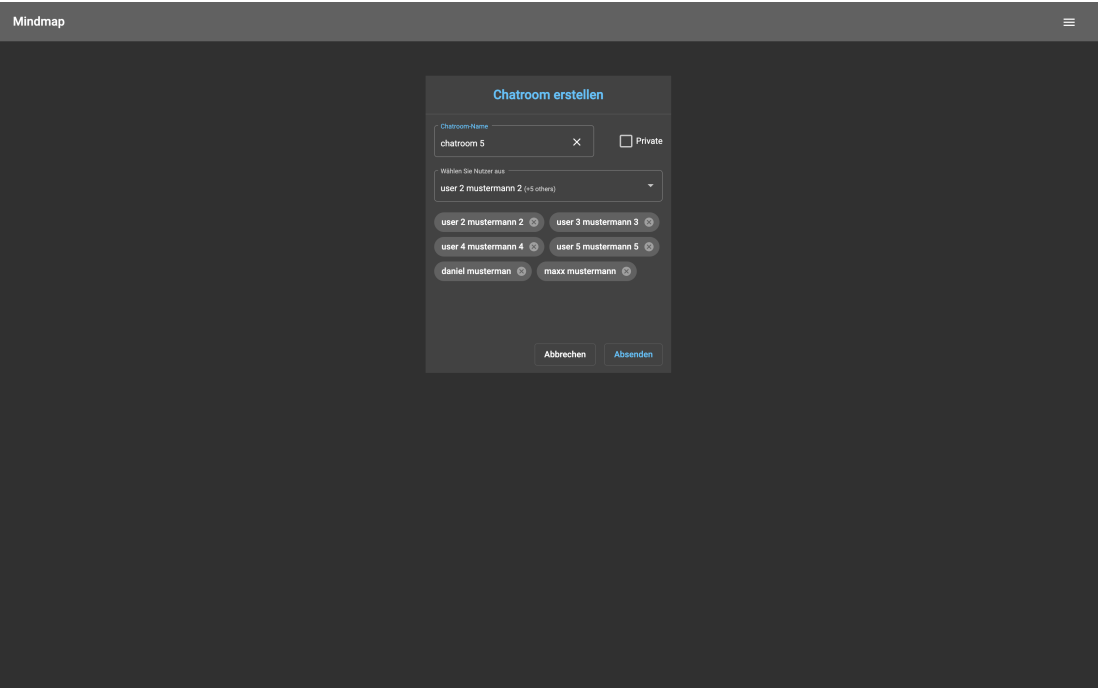


ABBILDUNG A.4: Chatroom aktualisieren

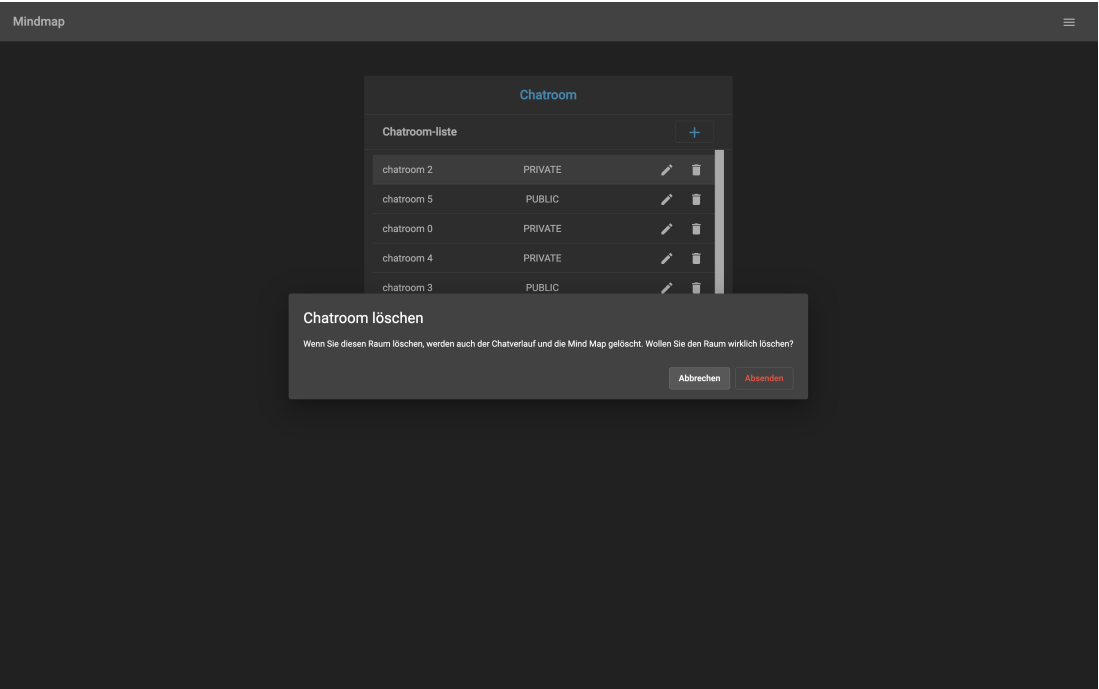


ABBILDUNG A.5: Bestätigungsdialog

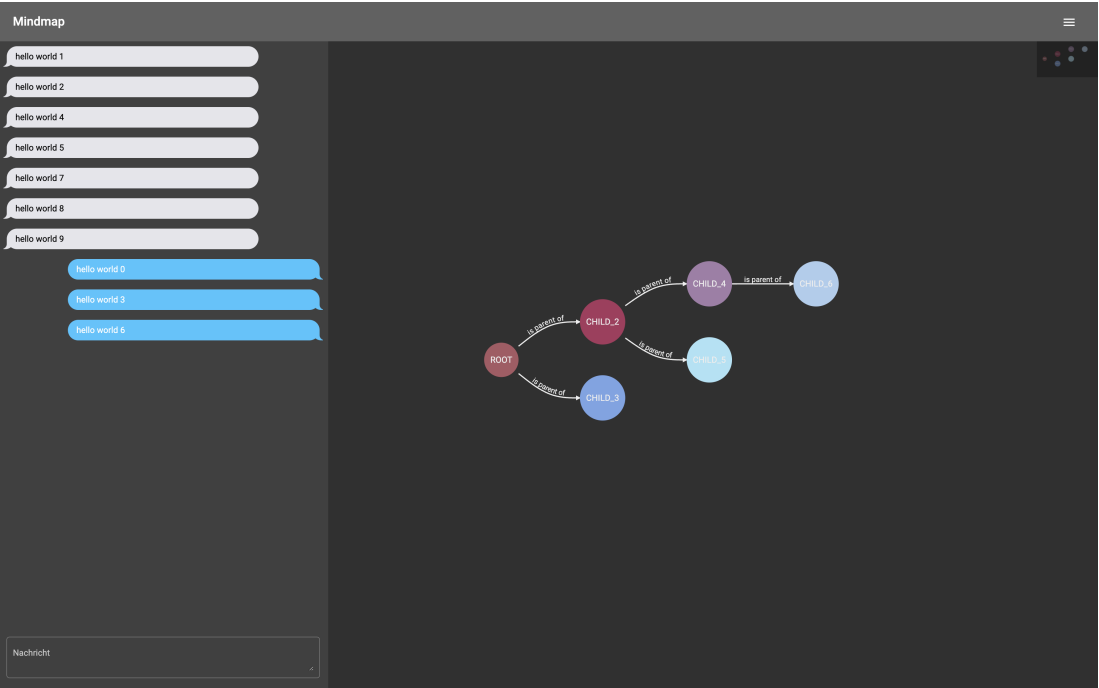


ABBILDUNG A.6: Chatroom

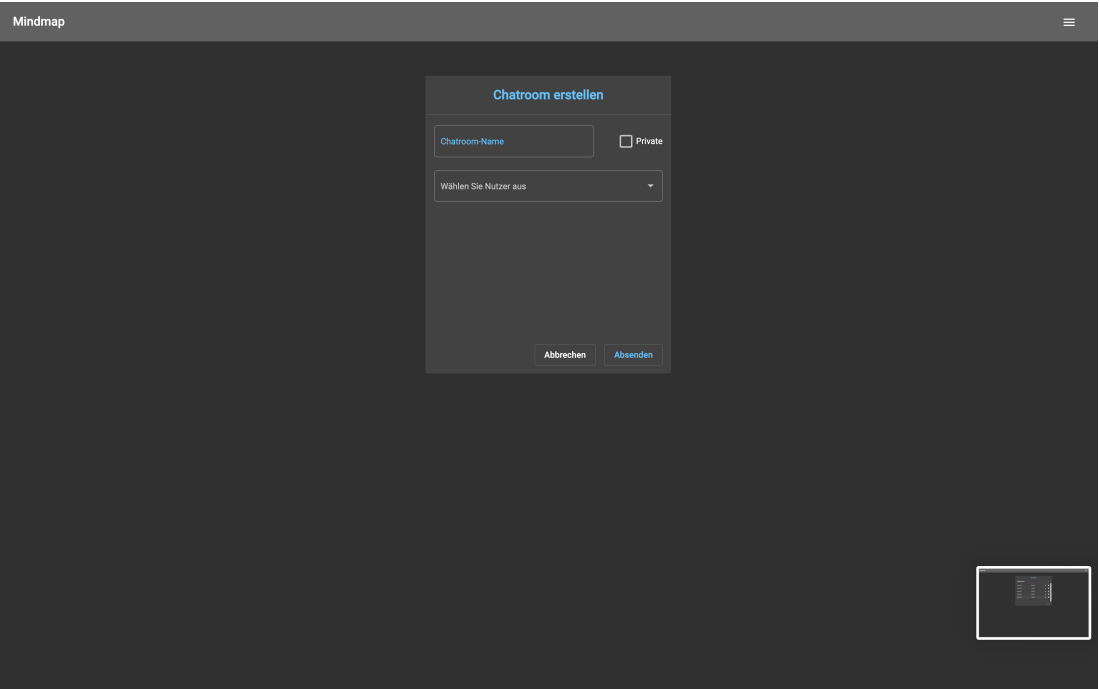


ABBILDUNG A.7: Chatroom anlegen

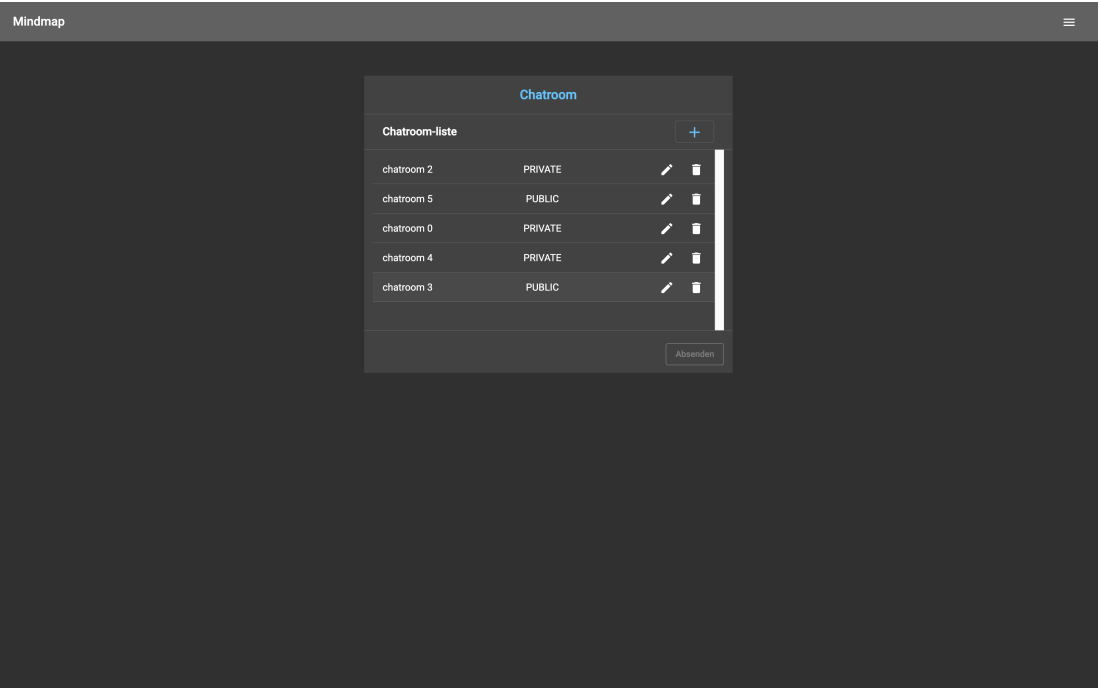
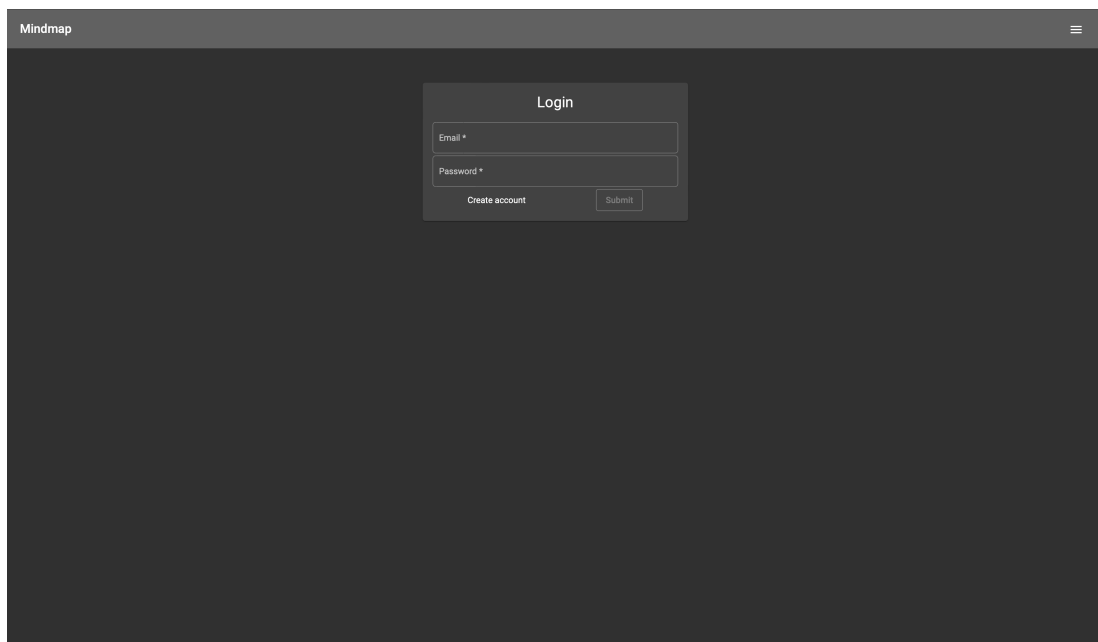
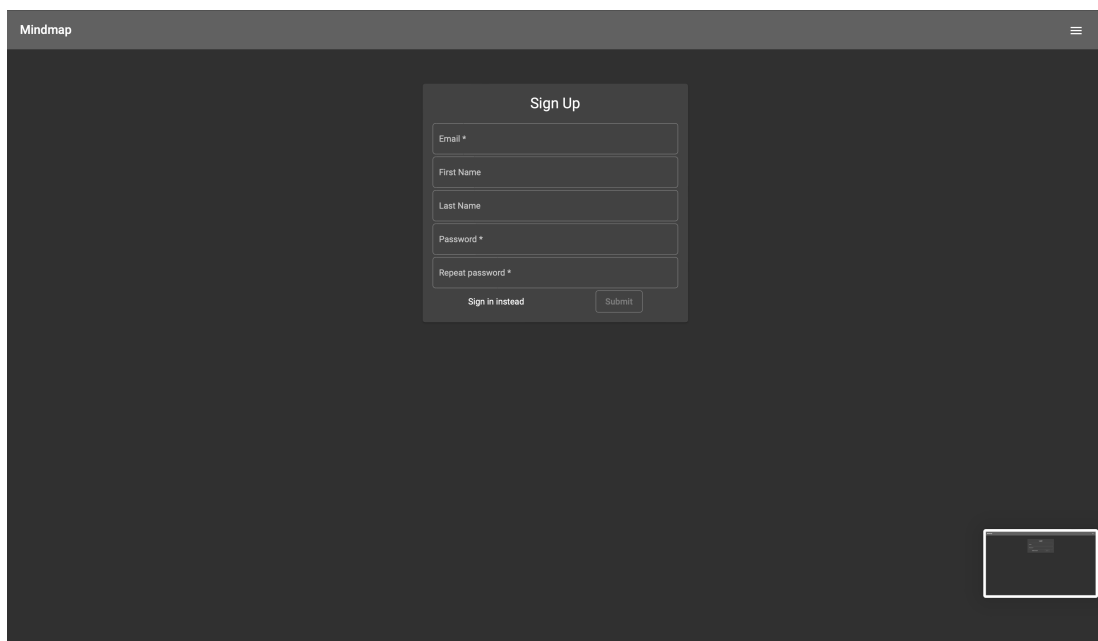


ABBILDUNG A.8: Chatroomliste



The screenshot shows a dark-themed web interface for a 'Mindmap' application. At the top left, the word 'Mindmap' is displayed in a small, light font. At the top right, there is a hamburger menu icon. The main content area is a dark gray rectangle. In the center of this rectangle is a white 'Login' form. The form has a title 'Login' at the top. Below the title are two input fields: 'Email *' and 'Password *'. At the bottom of the form are two buttons: 'Create account' on the left and 'Submit' on the right.

ABBILDUNG A.9: Registrieren



The screenshot shows a dark-themed web interface for a 'Mindmap' application, similar to the previous one. At the top left, the word 'Mindmap' is displayed. At the top right, there is a hamburger menu icon. The main content area is a dark gray rectangle. In the center of this rectangle is a white 'Sign Up' form. The form has a title 'Sign Up' at the top. Below the title are five input fields: 'Email *', 'First Name', 'Last Name', 'Password *', and 'Repeat password *'. At the bottom of the form are two buttons: 'Sign in instead' on the left and 'Submit' on the right. In the bottom right corner of the dark gray area, there is a small, light gray rectangular box containing a small, illegible icon or logo.

ABBILDUNG A.10: Anmelden

The screenshot shows a web application interface with a dark theme. At the top, a header bar contains the text 'Mindmap' on the left and a hamburger menu icon on the right. The main content area is a dark gray rectangle. Centered within this area is a white form titled 'Profil Einstellungen'. The form contains several input fields: 'E-Mail *' with the value 'max@gmail.com', 'Vorname *' with the value 'max', 'Nachname *' with the value 'mustermann', 'Altes password *' with a masked value '*****', 'Neues password *', and 'Neues Passwort wiederholen *'. Below these fields is a button labeled 'Änderungen speichern'. At the bottom of the form, there is a language selector showing 'Sprache DE' and a red button labeled 'Konto löschen'.

ABBILDUNG A.11: Profile