



# 07 – ConnectNow: Real-Time Conferencing

Die Zukunft der Echtzeit-Kommunikation –  
Plattformübergreifend, intuitiv und leistungsstark

# Agenda

## Einleitung

- Techniken (Frontend, backend, Datenbank)
- Zusätzliche Technologien

## Hauptteil

- Frontend (Layout und Struktur)
- SFU (Selective Forwarding Unit)
- SFU realisierung im Projekt

## Schluss

- Zusammenfassung
- zukünftige Features

# Projekt Vorstellung | Informationen

ConnectNow – Real-Time Conferencing

Moderne Alternative

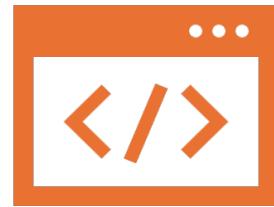
Entwicklung einer  
Web-Applikation für  
Echtzeit-Kommunikation

Plattformen  
übergreifend

Core-Funktionen

- Räume ( Rooms ) erstellen/beitreten
- Echtzeitkommunikation über WebRTC
- Screensharing

# Techniken - Frontend



## Frontend-Technologien

- HTML – Struktur der Website
- CSS – Gestaltung und Design der Benutzeroberfläche
- JavaScript – Interaktive Elemente ( z.B. Buttons , Dynamisches Layout )



## Merkmale des Frontends:

Intuitive Bediengung

Responsive Design (PC oder Smartphone)  
Funktionen ( Räume erstellen , Chat  
nutzen und Video-Feeds anzeigen )

# Technik - backend



## Backend-Technologie

Node.js

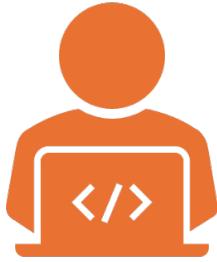


## Aufgaben des Backends

Verwaltung von Benutzer- und  
Raumdaten

Synchronisation von Chat-Nachrichten  
Aufbau und Steuerung der Verbindung  
zwischen Teilnehmern  
Schnittstelle zum Frontend

# Technik - Datenbank



**Firestore ( NoSQL,Firebase)**



**Arten der gespeicherten  
Daten**

Benutzerinformationen

Chat-Nachrichten

Evtl. Raum und Verbindungsdaten

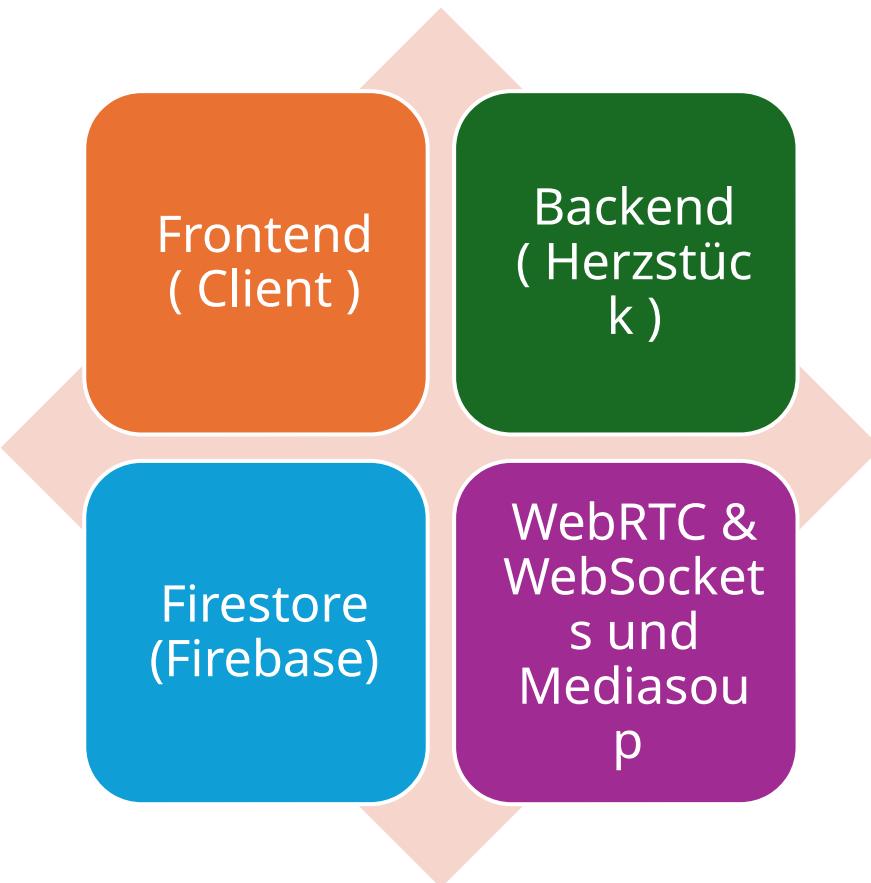
# ZUSÄTZLICHE TECHNOLOGIEN

WebRTC :  
Echtzeitübertragung von Audio und Video

WebSockets: Für den Chat und die Synchronisation von Daten

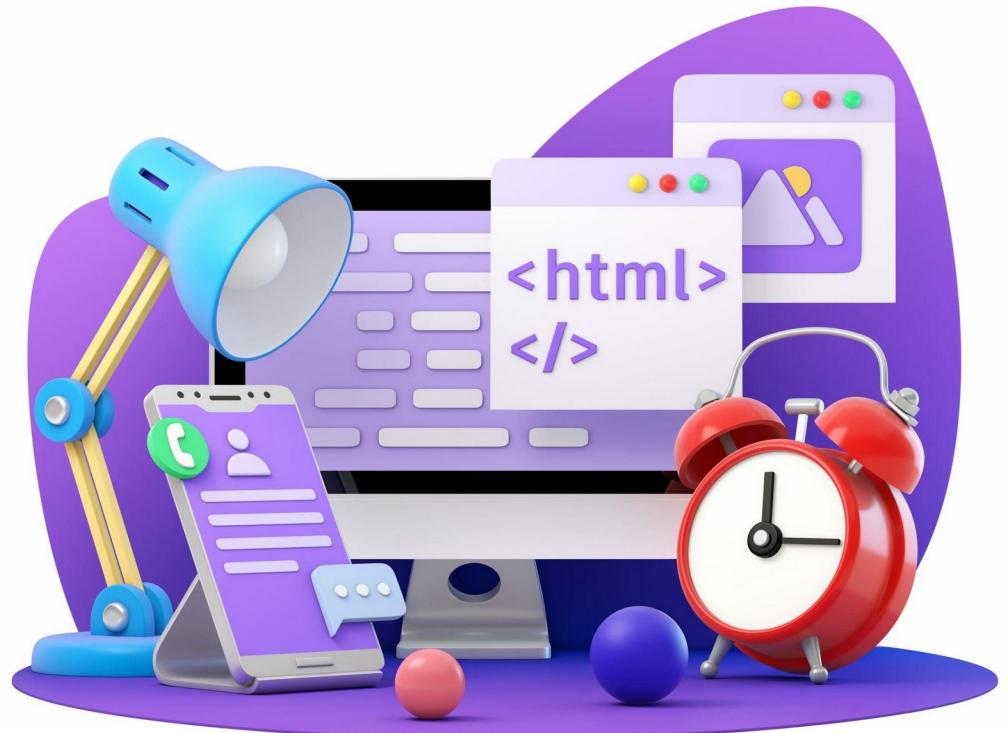
Zusammenarbeit dieser zwei Komponenten

# Zusammenspiel der Technologien



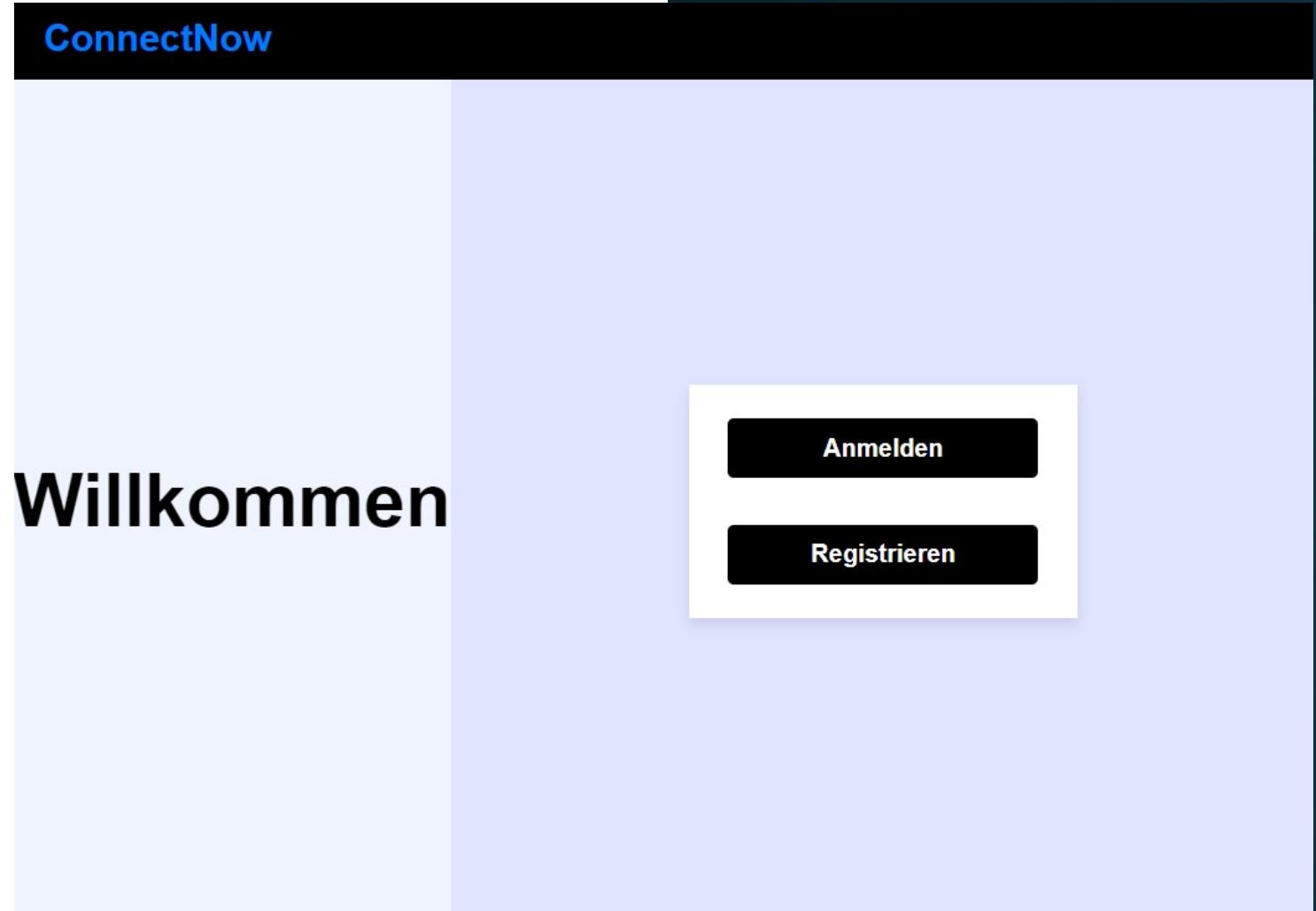
# Frontend

- Vorstellung des User Interfaces
- Codebeispiele



# Einstiegsseite

- Zwei Buttons
  - Anmelden
  - Registrieren



# Registrieren

- Eingabefelder für:
  - Benutzername
  - E-Mail
  - Passwort
- Sendet Eingaben an den Server
  - Server erstellt neuen Nutzer in die Datenbank

ConnectNow

## Registrieren

The registration form consists of four main sections: 'Benutzername' (Username) with a placeholder 'Benutzername eintippen...', 'Email' with a placeholder 'E-Mail', 'Passwort' (Password) with a placeholder 'Passwort', and a large black 'Registrieren' (Register) button at the bottom.

Benutzername  
Benutzername eintippen...

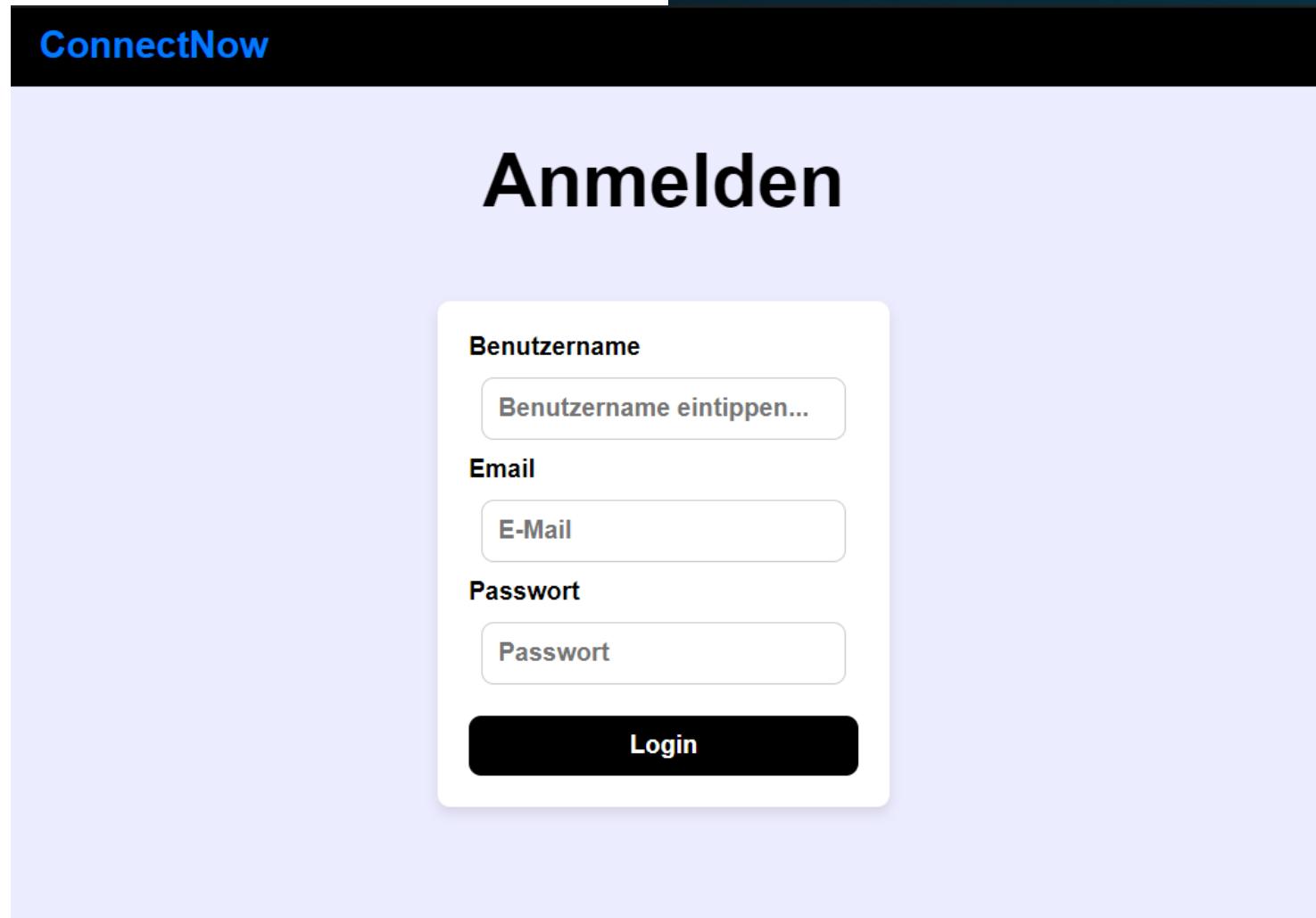
Email  
E-Mail

Passwort  
Passwort

Registrieren

# Anmelden

- Eingabefelder für:
  - Benutzername
  - E-Mail
  - Passwort
- Button sendet Eingaben an den Server
  - Server sendet Nachricht zurück:
    - Fehler oder Erfolg

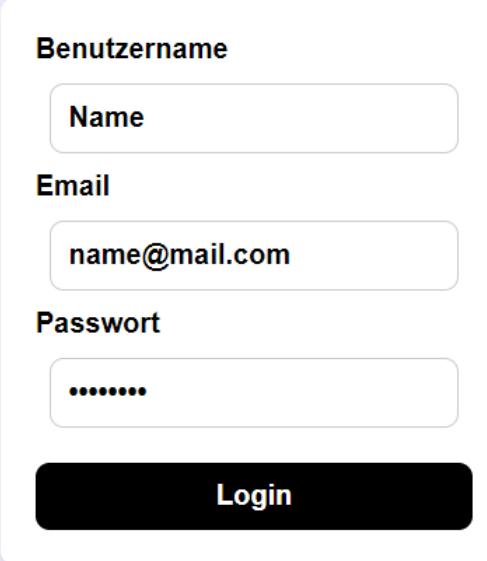


# Anmelden

- Eingabefelder für:
  - Benutzername
  - E-Mail
  - Passwort
- Button sendet Eingaben an den Server
  - Server sendet Nachricht zurück:
    - Fehler oder Erfolg

ConnectNow

## Anmelden



The image shows a login form interface. At the top, the word "Anmelden" is displayed in large, bold black font. Below it is a white rectangular input card with rounded corners. The card contains three text input fields: "Benutzername" with placeholder "Name", "Email" with placeholder "name@mail.com", and "Passwort" with placeholder ".....". At the bottom of the card is a large, dark blue "Login" button.

Benutzername

Name

Email

name@mail.com

Passwort

.....

Login

# Codebeispiel

- html:
  - Sorgt für das Layout
- CSS
  - Sorgt für design
- JavaScript
  - Sorgt für Funktion

```
<!-- Registrierungsseite -->
<div class="registerPage">
  <h2>Registrieren</h2>
  <div class="registerFeld">
    <p>Benutzername</p>
    <input id="usernameInputFeldReg" type="text" placeholder="Benutzername eintippen...">
    <p>Email</p>
    <input id="registerMail" type="email" placeholder="E-Mail">
    <p>Passwort</p>
    <input id="registerPass" type="password" placeholder="Passwort">
    <button onclick="registerUser()">Registrieren</button>
  </div>
</div>
```

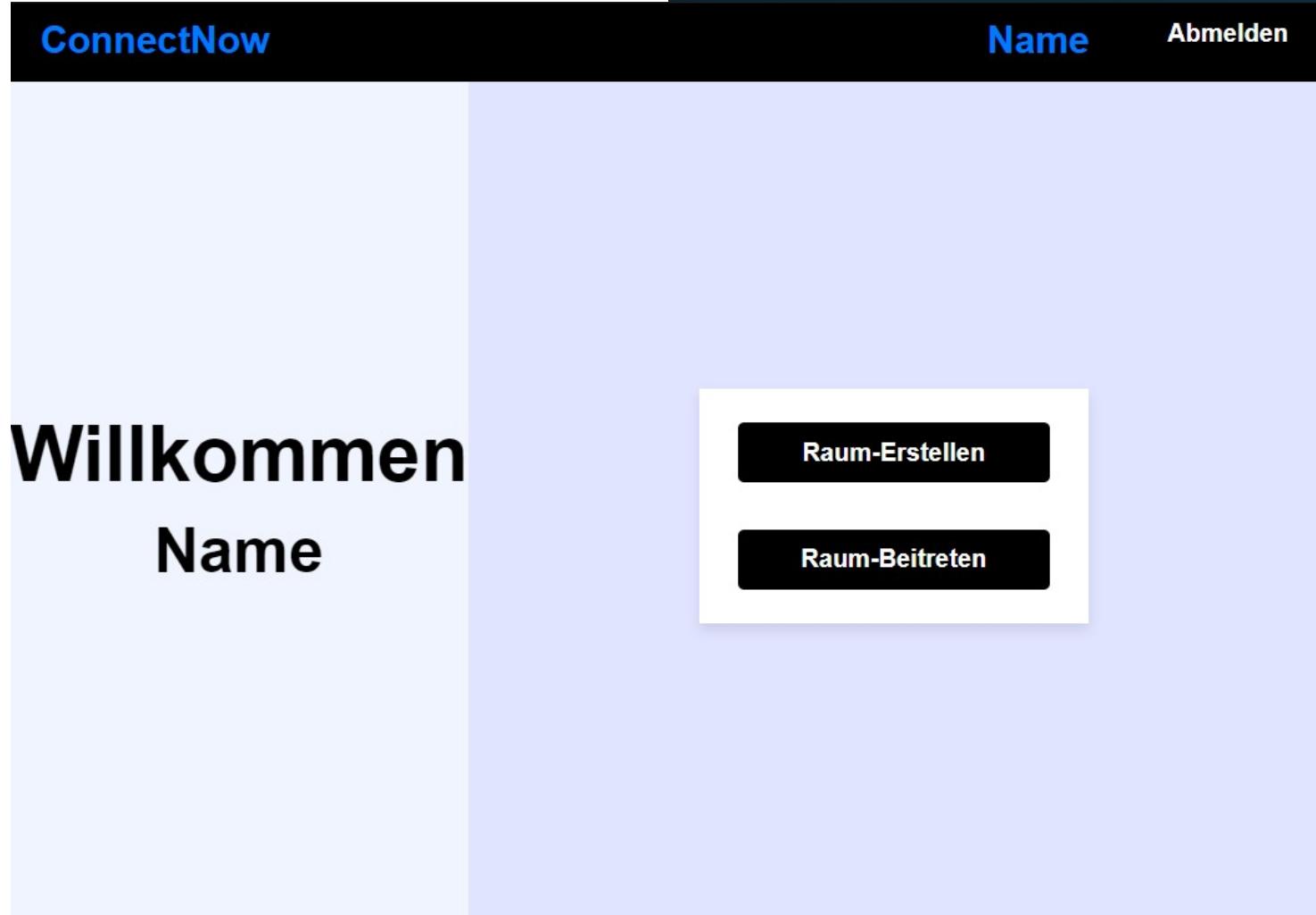
```
.registerPage button{
  background-color: black;
  color: white;
  margin-top: 10px;
  padding: 10px;
  border-radius: 8px;
  font-size: 16px;
  text-align: center;
  width: 250px;
  cursor: pointer;
}
```

```
async function registerUser() {
  console.log("In der RegisterUser methode")
  const loginName = document.getElementById("usernameInputFeldReg").value;
  const passwort = document.getElementById("registerPass").value;

  try {
    const response = await fetch(`${API_URL}/users/register`, {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({ loginName, passwort }),
    });
    const data = await response.json();
    if (response.ok) {
      localStorage.setItem("userID", data.userid);
      localStorage.setItem("username", data.loginName);
      alert("Registrierung erfolgreich!");
      window.location.href = "loginPage.html";
    } else {
      alert(data.error || "Registrierung fehlgeschlagen");
    }
  } catch (error) {
    console.error("Fehler bei der Registrierung:", error);
  }
}
```

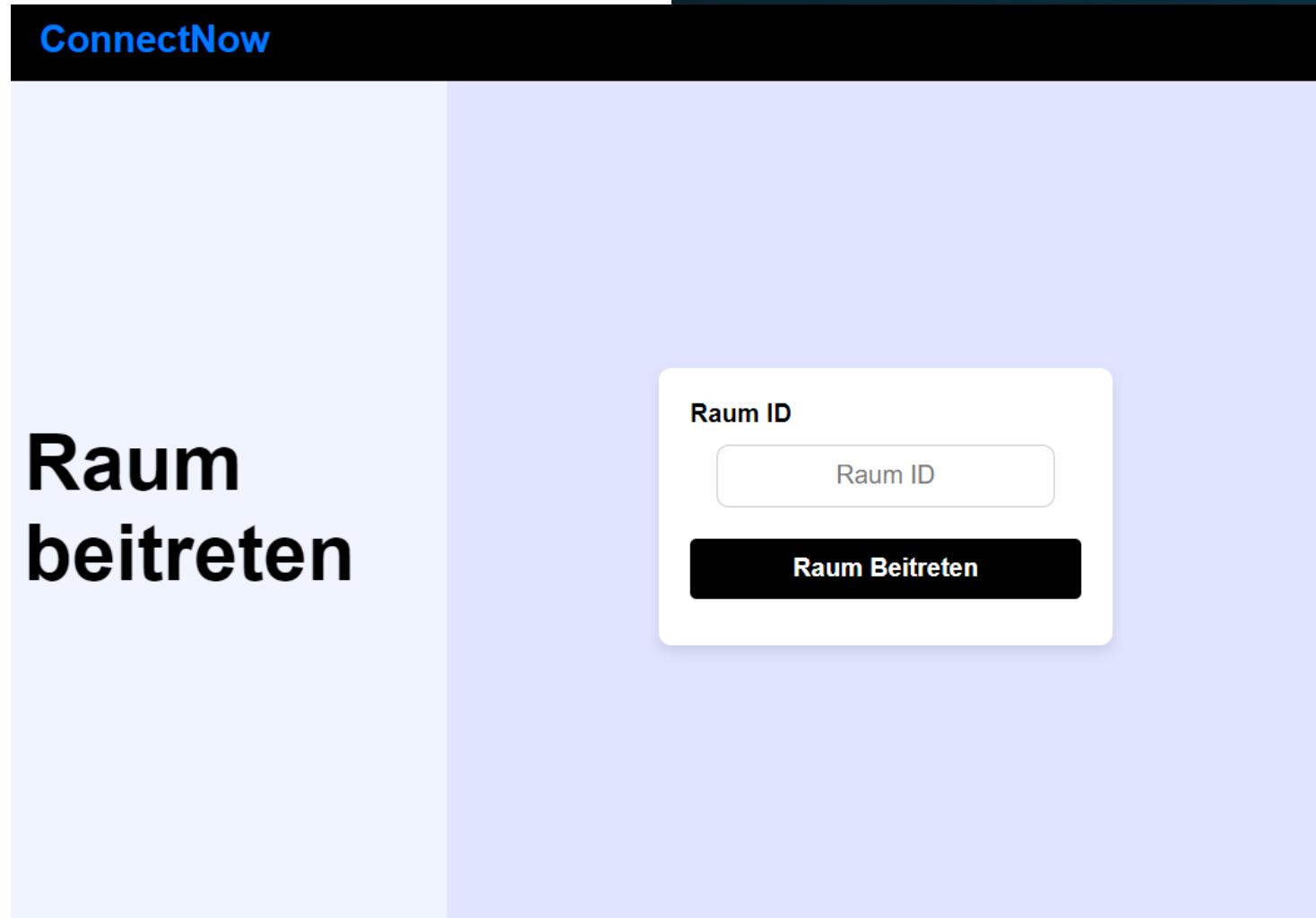
# Angemeldet

- „Abmelde“- Button
- Raum erstellen/beitreten
  - Führen zu weiteren Seiten



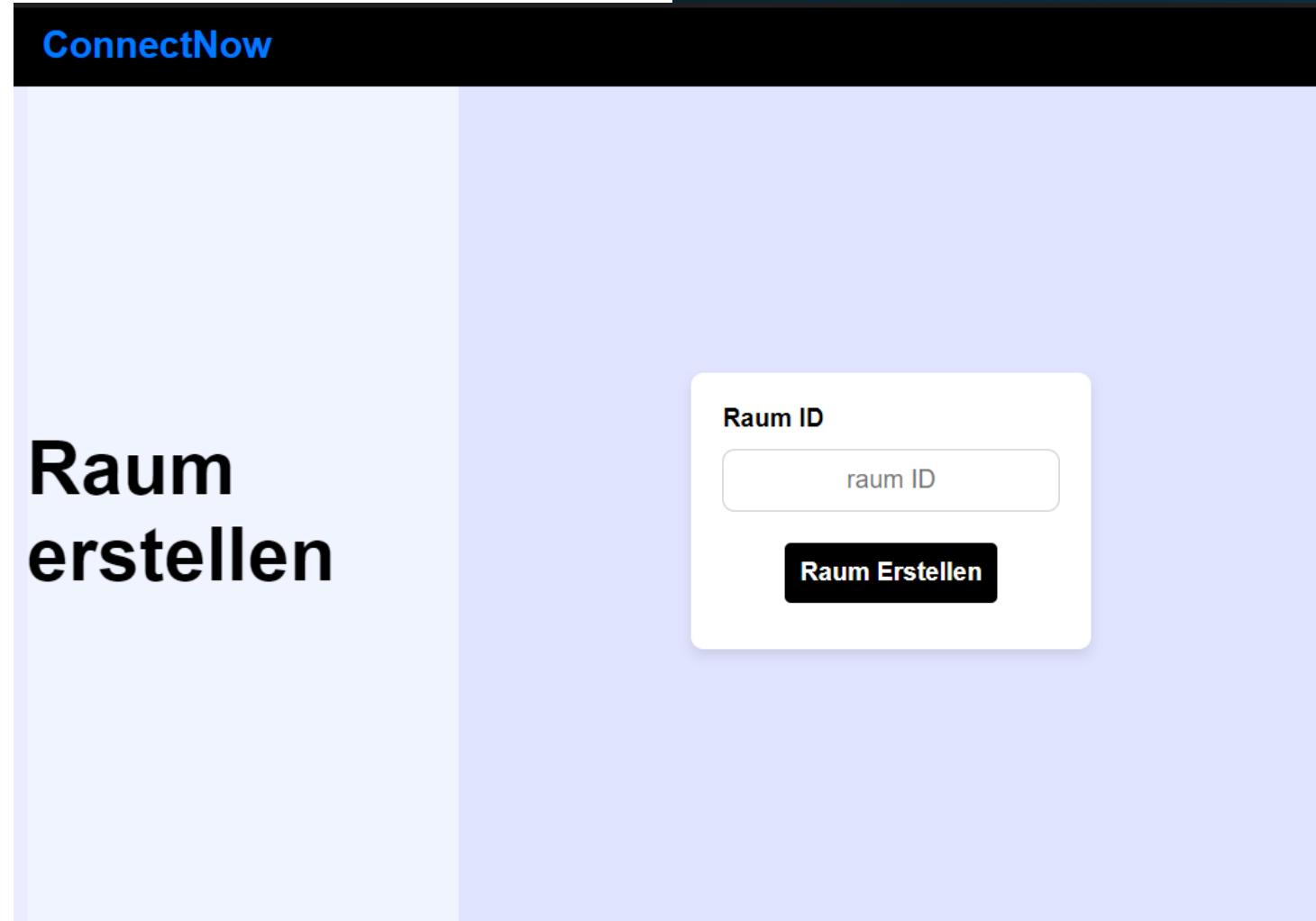
# Raum Beitreten

- Eingabefelder zum Betreten des Raumes
- Button sendet an den Server
  - erstellt einen neuen Raum mit Daten der Eingabe



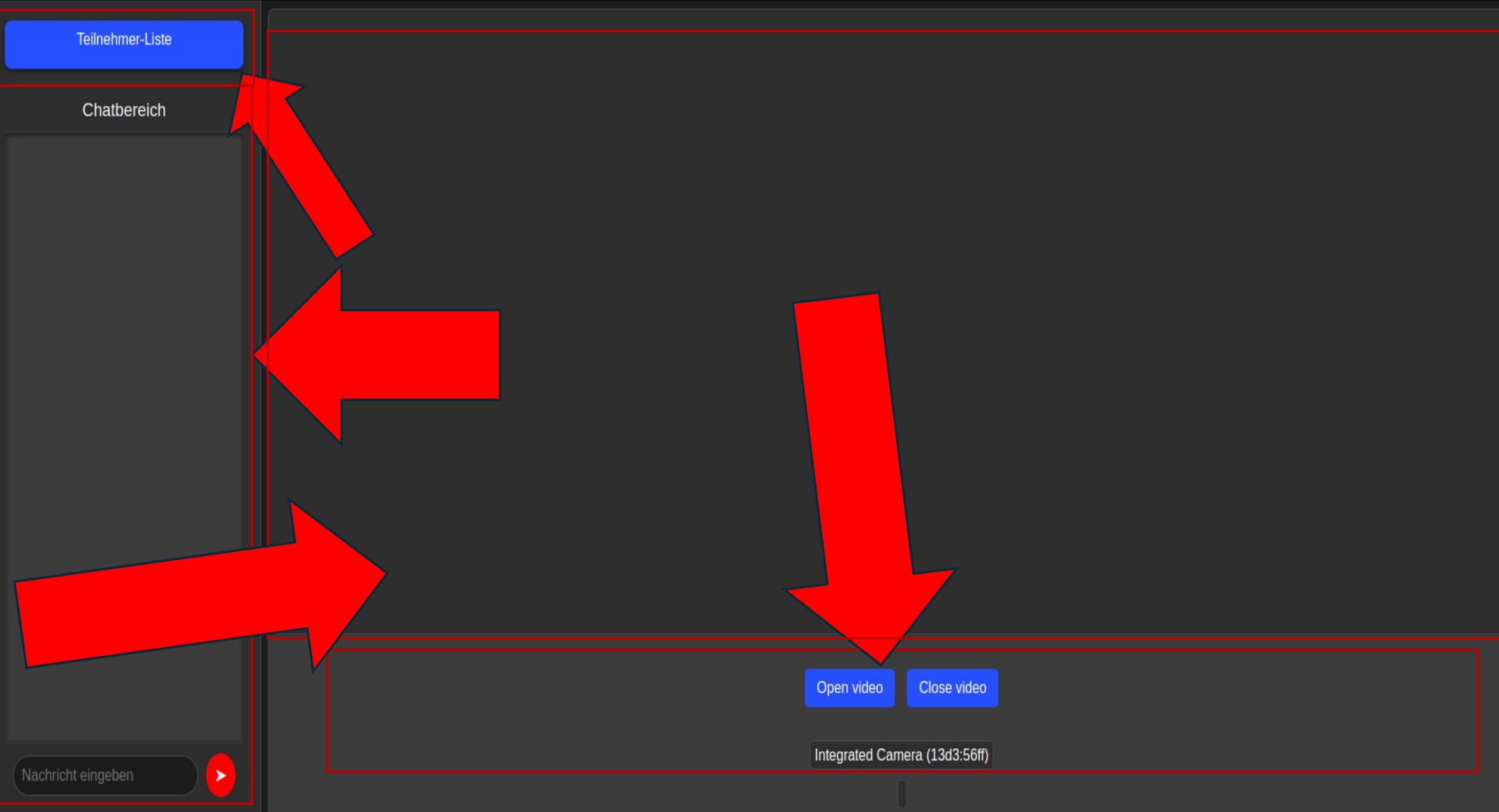
# Raum Beitreten

- Eingabefelder zum erstellen des Raum
- Button sendet an den Server
  - erstellt einen neuen Raum mit Daten der Eingabe



# Layout und Struktur der Room Page

Erklärung des Designs und der Elemente der Room Page



Die Kopfzeile von der Seite

Logo **CONNECTNOW**

Navigationsleiste für den aktuellen Namen der Seite

Der eigentliche Raum

Die komplette linke Seite

Teilnehmerliste

Beispielnamen für die Simulation

Chatbereich

Chatbox, indem die Nachrichten abgelegt werden

Nachrichten eingabeleiste

Eingabe auch mit Enter-Taste realisierbar machen

Die komplette rechte Seite

Video Container

Kontrolle der Medien

```
1 <html>
18   <body>
19     <!-- Header -->
20     <header class="header-main">
21       <div class="name">
22         <a href="createPage.html">ConnectNow</a>
23       </div>
24       <nav class="nav">
25         <span class="username">SpongeBob</span>
26       </nav>
27     </header>
28   </body>
29 </html>
```

# Zu viel Code?

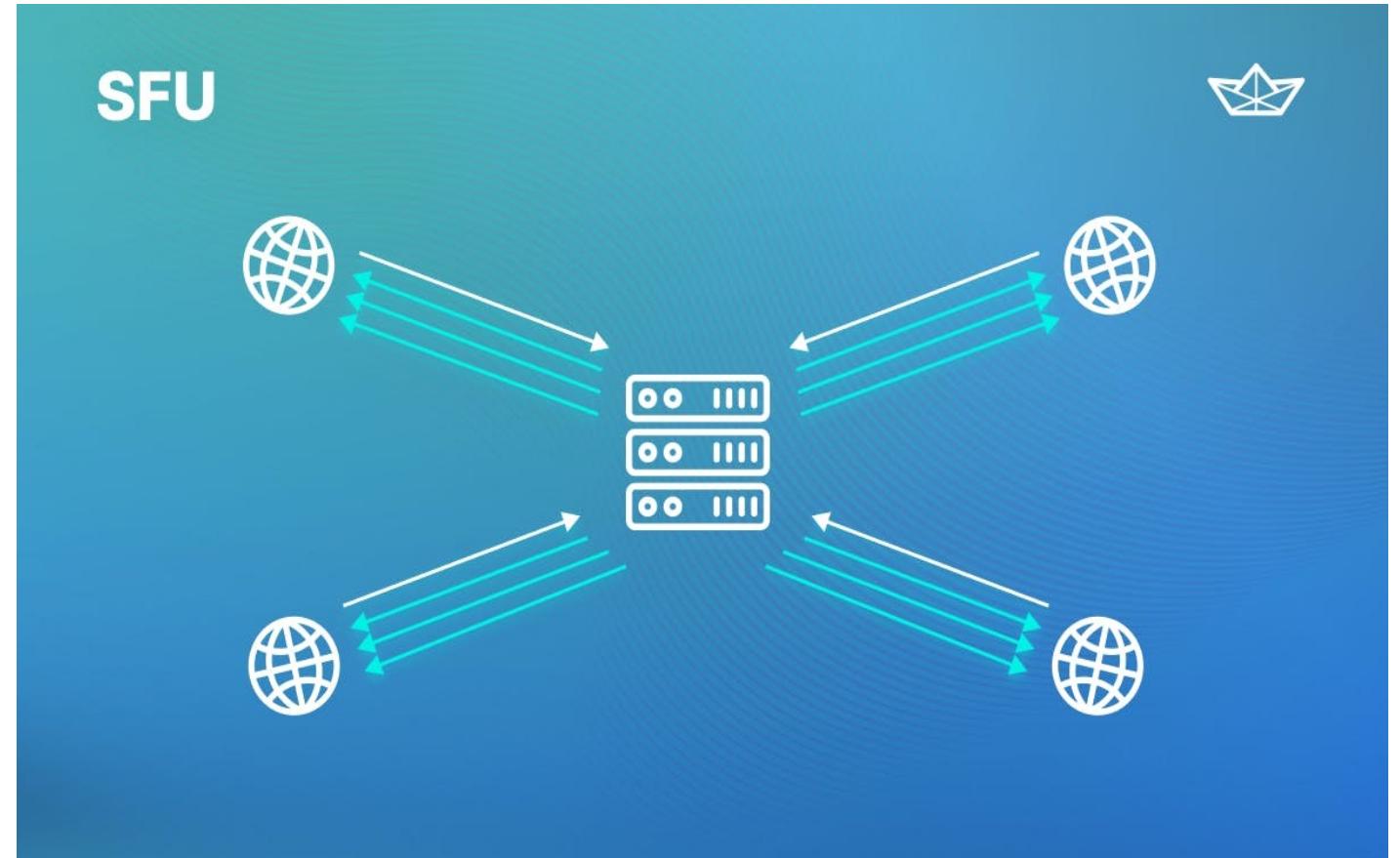
```
57   <!-- Chatbereich -->
58   <div id="chatBox" class="chat-box">
59     <!-- Chat-Inhalte -->
60   </div>
61   <div class="chat-input">
62     <input type="text" id="chatMessage" placeholder="Nachricht eingeben" onkeydown="if (event.key === 'Enter') sendMessage()">
63     <button onclick="sendMessage()"></button>
64   </div>
65 </div>
66 </div>
67 <!-- Die rechte seite-->
68 <div class="right">
69   <!-- Alle Videos und der Videocontainer-->
70   <div id="videoMedia" class="hidden containers">...
71 </div>
72 <!-- Kontrolfunktionens elemente (Kamera, Audio, Screensharing)-->
73 <div id="control" class="hidden">...
74 </div>
75 </div>
76 </div>
77 </div>
78 </div>
79 </div>
80 </div>
81 </div>
82 </div>
83 </div>
84 </div>
85 </div>
86 </div>
87 </div>
88 </div>
89 </div>
90 </div>
91 </div>
92 </div>
93 </div>
94 </div>
95 </div>
96 </div>
97 </div>
98 </div>
99 </div>
100 </div>
101 </html>
```

# SFU (Selective Forwarding Unit) und Mediasoup-Bibliothek

Erklärung der Funktionalität von SFU und Mediasoup

# Was ist SFU

- Ein Server, der Audio- und Videostreams selektiv weiterleitet
- Teil von WebRTC-Technologien für Echtzeitkommunikation (z. B. Videokonferenzen)
- Leitet Streams weiter, ohne sie zu dekodieren
- Wir bei Zoom, Microsoft Teams, Google Meets verwendet



# Funktionsweise der SFU (Selective Forwarding Unit)

## Senden von Streams

-Teilnehmer senden ihre Audio-/Videostreams an die SFU.

## Selektive Weiterleitung

-Die SFU entscheidet an wenn weiter geleitet wird.

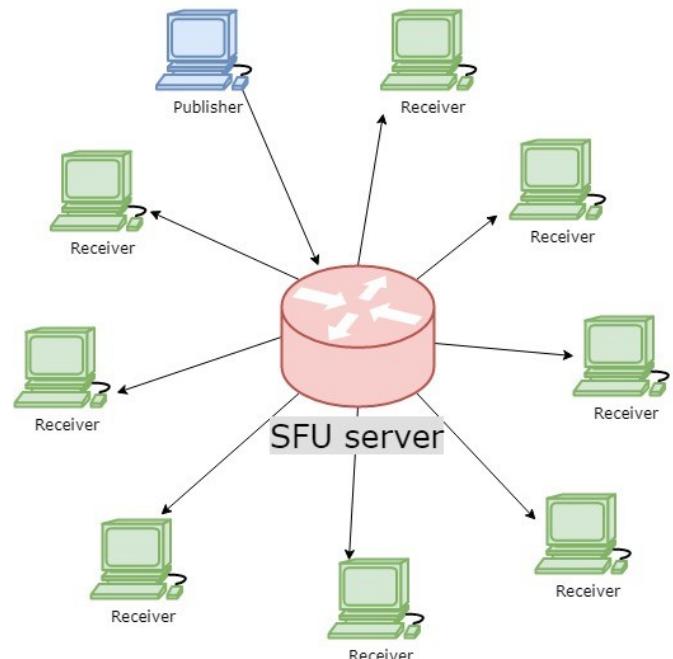
## Keine Transkodierung

-Die Streams werden unverändert weitergeleitet, was die Latenz reduziert und Rechenleistung spart.

## Bandbreitenanpassung

-Streams werden an die Netzwerkqualität der Teilnehmer angepasst (z.B. HD oder niedrigere Auflösung).

## One to Many



# Wichtige Komponenten fürs SFU im Projekt

## Worker

- Prozess, der Router verwaltetet

## Router

- Kernkomponente der SFU
- verbindet Producer mit Consumer
- Entscheidet welche Streams weitergeleitet werden

## Transport

- Verbindungskanal
- SendTransport
- ReceiveTransport

## Pruducer

- Streams die vom Teilnehmer an den Router gesendet werden

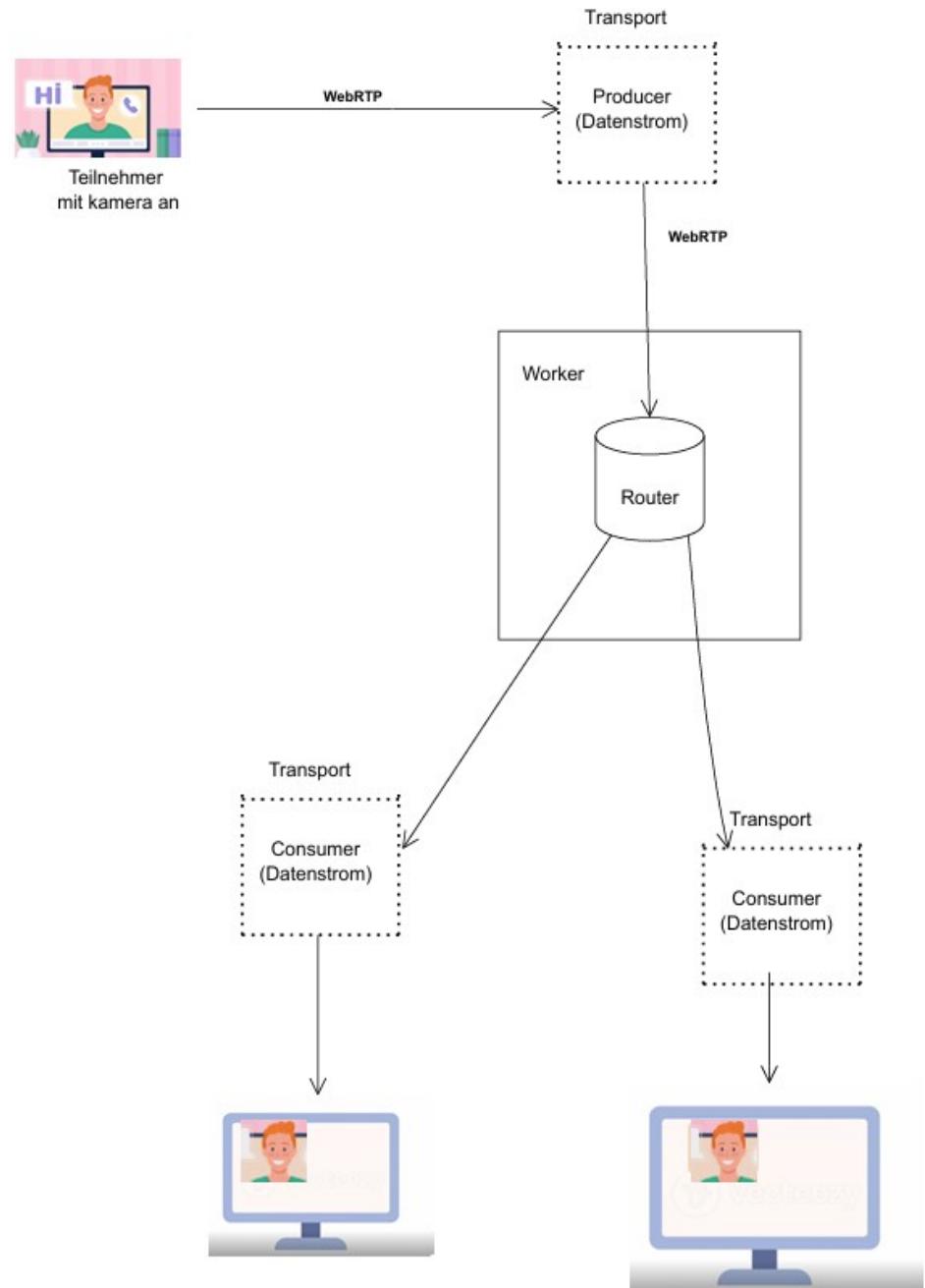
## Consumer

- Streams vom Router die an Teinehmer gesendet werden



# Wie funktioniert SFU im Projekt

1. Ein Teilnehmer sendet seinen Videostream über WebRTC-Send-Transport an den Router (z. B. Kamerastream)
2. Der Router registriert den Stream als Producer.
3. Router entscheidet, welche Teilnehmer den Stream empfangen sollen.
4. Erstellt für jeden Teilnehmer einen Consumer.
5. Teilnehmer, die verbunden sind, empfangen den Consumer über WebRTC-Receive-Transport.
6. Der Stream wird im Browser angezeigt.



Fazit



# Zusammenfassung: Was wurde schon geschafft?

- Account Verwaltung mit Datenbank
- Videochat mit SFU
- Ansprechendes Frontend



# Was werden wir im fertigen Produkt noch hinzufügen?



## Im Videochatroom

Chat Funktion  
Teilnehmerliste  
Screensharing



## Datenbank

Nutzerauthentifizierung



## Anpassungen im Frontend und Backend

# Danke für Ihre Aufmerksamkeit



SIND NOCH FRAGEN

