

# Programmierung von Web-Anwendungen

27.06.2025

# Agenda

1. Motivation
2. Projektorganisation
3. Demonstration der Anwendung
4. Technologie-Stack
5. Technische Umsetzung
6. Fazit

# Motivation

# Probleme von Reddit, Twitter, Threads, ...

Motivation



Technische  
Hochschule  
Augsburg



Technische  
Hochschule  
Augsburg

# Idee

## Motivation

- skizze/entwurf/...
- content

# Projektorganisation

# Timeline & Milestones

## Projektorganisation



# Issue Board

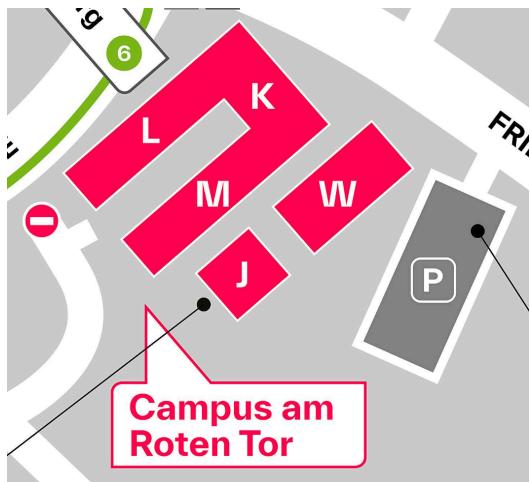
## Projektorganisation

The image shows a digital issue board with four columns: Open, In Progress, Ready for Review, and Closed.

- Open:** Contains two items:
  - Zod für Schema-Validierung integrieren (backend) #10
  - Erweiterungen für den Chat (backend db ui) #15
- In Progress:** Contains three items:
  - Home-Seite integrieren (ui) #4
  - Projektpräsentation (orga) #12
  - DM-Seite integrieren (ui) #5
- Ready for Review:** Contains one item: DM-Seite integrieren (ui) #5.
- Closed:** Contains eight items:
  - Data seeding (backend db) #14
  - Websockets für Chat (backend) #13
  - Datenbank integrieren (db) #2
  - Datenbankschema überlegen und anlegen (db) #1
  - Konzept für State Management im Frontend überlegen (ui) #8
  - Client-side Routing implementieren (ui)

# Kommunikation im Team

Projektorganisation



# API-Dokumentation

## Projektorganisation

 Swagger  
Supported by SMARTBEAR

### API Documentation 1.0.0 OAS 3.0

This is the API documentation for webprog-ss25

**Auth** Authentication related endpoints ^

**POST** /api/auth/register Register a new user. This endpoint requires clients to include credentials (cookies). ▾

**POST** /api/auth/login Login a user. This endpoint requires clients to include credentials (cookies). ▾

**POST** /api/auth/logout Logout a user. This endpoint requires clients to include credentials (cookies). ▾

**Chat** Chat related endpoints ^

**GET** /api/protected/chat/{chatId}/messages Get message history for a chat. This endpoint requires clients to include credentials (cookies). ▾

**GET** /api/protected/chat/chats List user's chats. ▾

**POST** /api/protected/chat/chats Start a new chat. ▾

# Dokumentation

## Projektorganisation

“Good code is its own best documentation. As you’re about to add a comment, ask yourself, ‘How can I improve the code so that this comment isn’t needed?’ Improve the code and then document it to make it even clearer.”

— Steve McConnell, *Code Complete: A Practical Handbook of Software Construction*

# Demonstration der Anwendung

# Demonstration der Anwendung

The screenshot shows a web application interface with a dark theme. On the left, there is a sidebar with navigation links: 'Explore' (with a magnifying glass icon), 'Share' (with a plus sign icon), 'Chat' (with a speech bubble icon), and 'Profile' (with a person icon). The main area displays a feed of four posts, each enclosed in a light blue rounded rectangle. Each post contains a timestamp at the bottom. The first post reads: "Interesting thoughts on thoughts about recent presentations and seminars! I have a slightly different take — The workshop on public speaking was interactive and helpful." (08.06.2025, 17:26 Uhr). The second post reads: "Interesting thoughts on thoughts about recent presentations and seminars! I have a slightly different take — I liked how the keynote speaker emphasized emotional intelligence." (08.06.2025, 17:26 Uhr). The third post reads: "Interesting thoughts on thoughts about recent presentations and seminars! I have a slightly different take — The leadership workshop gave me practical tools I could use immediately." (08.06.2025, 17:26 Uhr). The fourth post, which is highlighted with a teal border, reads: "Interesting thoughts on thoughts about recent presentations and seminars! I have a slightly different take — I liked how the keynote speaker emphasized emotional intelligence." (08.06.2025, 17:26 Uhr). This highlighted post has a teal header bar with white text that says "Thoughts about recent presentations and seminars". Below the header, there is a teal button labeled "Join Discussion". Above the header, there are two circular icons: one with a speaker symbol and another with a comment bubble symbol.

The screenshot shows a mobile application interface with a dark theme. At the top, there is a header bar with various icons: battery level (23:46), signal strength, and battery percentage (22%). The main area displays a feed of four posts, each enclosed in a light blue rounded rectangle. Each post contains a timestamp at the bottom. The first post reads: "Interesting thoughts on science and history trivia! I have a slightly different take — Bananas are berries, but strawberries aren't." (08.06.2025, 17:26 Uhr). The second post reads: "Interesting thoughts on science and history trivia! I have a slightly different take — The Great Fire of London in 1666 destroyed much..." (08.06.2025, 17:26 Uhr). The third post reads: "Interesting thoughts on science and history trivia! I have a slightly different take — Bananas are berries, but strawberries aren't." (08.06.2025, 17:26 Uhr). The fourth post, which is highlighted with a teal header bar with white text that says "Science and history trivia", has a teal button labeled "Join Discussion". Above the header, there are two circular icons: one with a speaker symbol and another with a comment bubble symbol. The bottom of the screen features a navigation bar with icons for 'Explore' (magnifying glass), 'Share' (plus sign), 'Chat' (speech bubble), and 'Profile' (person).

# Demonstration der Anwendung

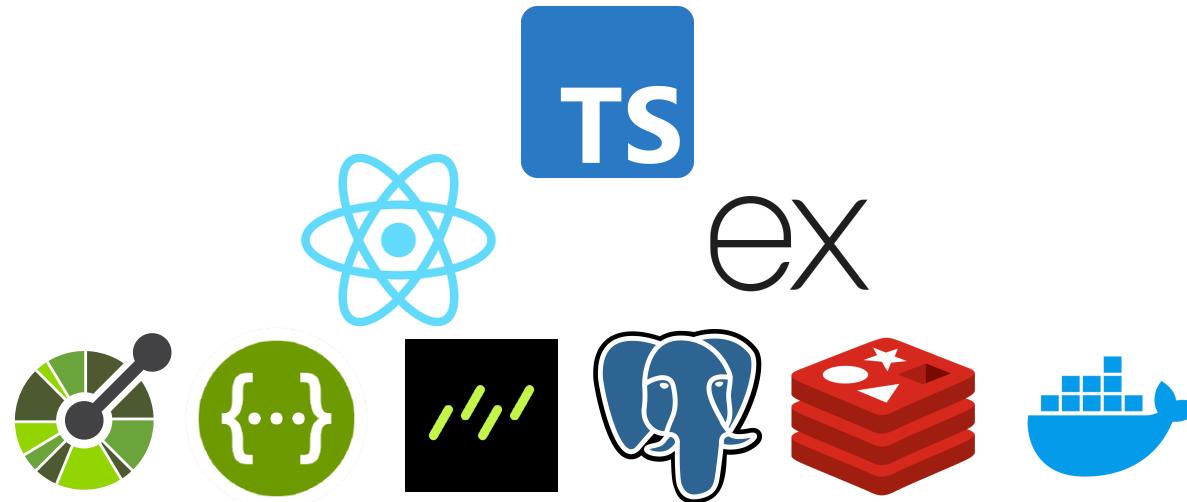
Jetzt selber ausprobieren!



# Technologie-Stack

# Übersicht

Technologie-Stack



# React

Technologie-Stack

# React: JSX

Was ist JSX?

# React: JSX

Vorteile von JSX

# React: Hooks

Was sind Hooks?

# React: Hooks

useState → lokaler State in Komponenten

```
import { useState } from "react";

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <button onClick={() => setCount(count + 1)}>
      Count: {count}
    </button>
  );
}
```

# React: Hooks

useEffect → Side-Effects (z.B. API-Aufrufe, DOM-Manipulation)

```
import { useState, useEffect } from "react";

function DataLoader() {
  const [data, setData] = useState<string>("");
  const [searchString, setSearchString] = useState<string>("");

  useEffect(() => {
    fetch("/api/data/?query=${searchString}")
      .then((res) => res.text())
      .then((text) => setData(text));
  }, [searchString]);

  return <div>{data}</div>;
}
```

# React: Hooks

useContext → Zugriff auf Kontext (z.B. globale Zustände)

```
import { createContext, useContext, useState, ReactNode } from "react";

const AuthContext = createContext(undefined);

export const AuthProvider = ({ children }) => {
  const [userId, setUserId] = useState(null);

  const login = (userId) => setUserId(userId);
  const logout = () => setUserId(null);

  return (
    <AuthContext.Provider value={{ userId, loggedIn: Boolean(userId), login, logout }}>
      {children}
    </AuthContext.Provider>
  );
};

// Verwendung in Komponenten:
const { userId, login, logout } = useContext(AuthContext);
```

# Browser DOM

This is shown on the left

```
<html>
  <head>
    <title>This is the title</title>
  </head>
  <body>
    <h1>This is a header</h1>
    <p> This is a paragraph</p>
  </body>
</html>
```

- Repräsentiert die Struktur der Webseite als Baum von Knoten (*Nodes*)  

- DOM-Operationen sind teuer (*Performance*)

# Virtual DOM

Von React generierter virtueller DOM-Baum

```
const domTree = {
  tagName: 'html',
  attributes: {},
  children: [
    {
      tagName: 'head',
      attributes: {},
      children: [
        {
          tagName: 'title',
          attributes: {},
          children: ['Href Attribute Example'],
        },
      ],
    },
    {
      tagName: 'body',
      attributes: {},
      children: [
        {
          tagName: 'h1',
```

# React Internals: Virtual DOM

Wie React Änderungen effizient rendernt

# Express.js

Technologie-Stack

# Technische Umsetzung

# Frontend

Technische Umsetzung

BrowserRouter

Auth-Provider | Chat-Provider | Opinions-Provider

AppShell

Navigation

Sidebar | Bottombar

Page

ExplorePage | FeedPage | ChatPage | ProfilePage

# Tailwind

## Technische Umsetzung

- wenn wir noch Zeit brauchen, dann ausformulieren

# Animationen

Warum Motion?

- Einfache deklarative Implementierung von Animationen
- `motion`-Komponente können wie normale HTML-Elemente verwendet werden
- Hohe Performance durch optimierte Rendering-Pipelines

# Animationen

Verwendung in unserer Anwendung

- `motion.div`:

```
<motion.div
  drag="x"
  animate={animationController}
  whileDrag={{ scale: 1.02 }}
  onDragEnd={({_, dragInfo}) => handleSwipe(dragInfo)}
>
  { /* ...DiscussionCards... */ }
</motion.div>
```

- `useAnimation`:

```
const animationController = useAnimation();
const handleDismissed = async () => {
  await animationController.start({ x: -500, opacity: 0, transition: { duration: 0.2 } });
  // fetch new data
  animationController.set({ x: 50, opacity: 0 });
  await animationController.start({ x: 0, opacity: 1, transition: { duration: 0.4 } });
};
```

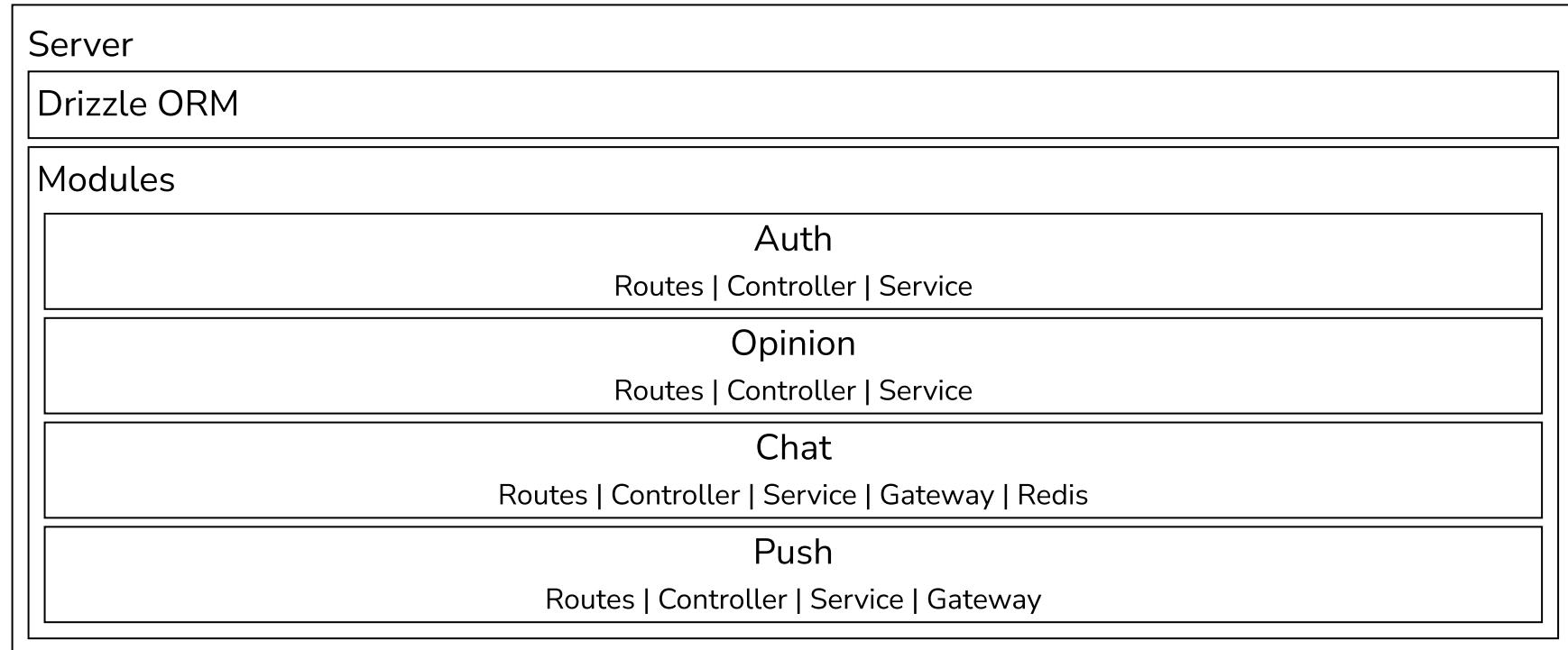
# Animationen

## Gesten und Dragging

```
<motion.div
  drag
  onDragEnd={(e, info) => console.log(info.offset.x)}
>
  Drag Me
</motion.div>
```

# Backend

Technische Umsetzung



# Datenbank

## Technische Umsetzung

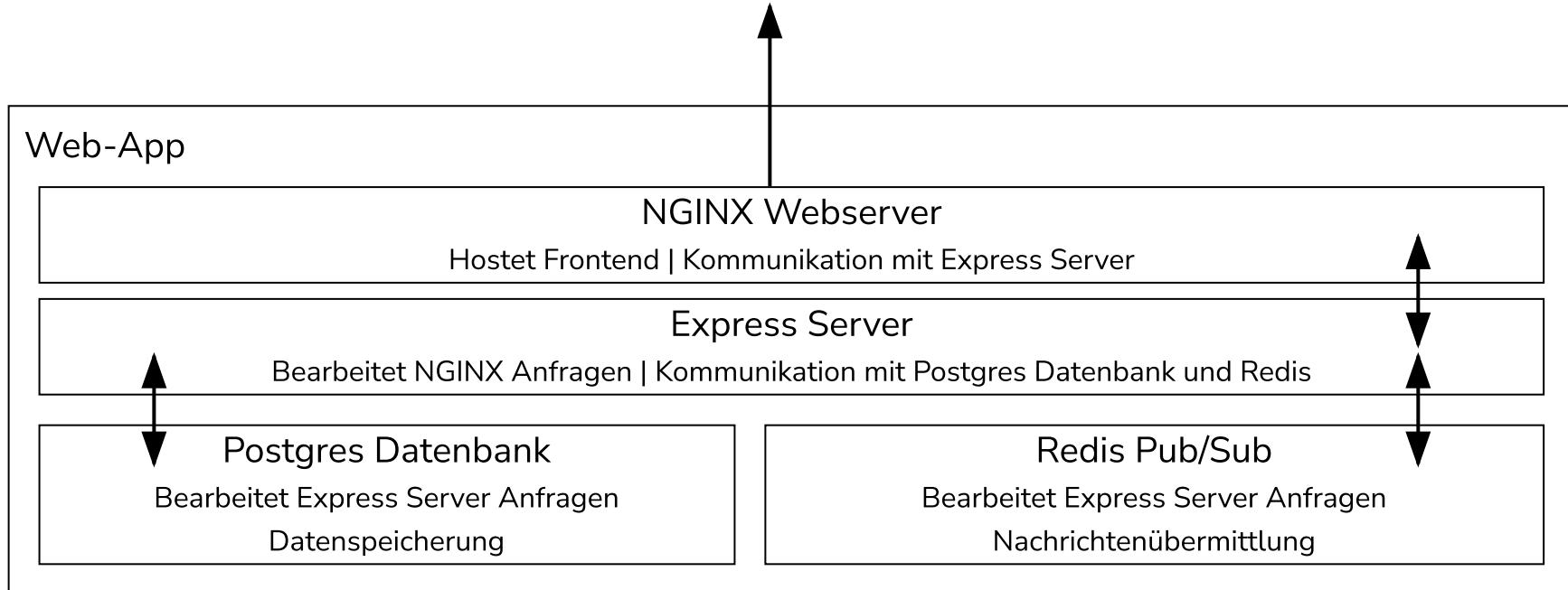
- todo: Drizzle ORM
- Postgres
- Datenbank-Schema etc.

# Authentifizierung

Technische Umsetzung

# DevOps – Überblick

Technische Umsetzung



# DevOps – Frontend

## Technische Umsetzung

```
# BUILD APP
FROM node:lts-alpine as builder

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY ..

RUN npm run build

# SERVE APP
FROM nginx:stable-alpine

WORKDIR /usr/share/nginx/html

RUN rm -rf ./*

COPY --from=builder /app/build /usr/share/nginx/html
```

# DevOps – Backend

## Technische Umsetzung

```
ARG NODE_VERSION=22.15.0
FROM node:${NODE_VERSION}-alpine

WORKDIR /app

COPY package*.json .

RUN npm install

COPY ..

RUN npm run build

EXPOSE 80

CMD ["npm", "run", "start"]
```

# DevOps – Docker Compose

## Technische Umsetzung

```
services:
  client:
    build: ./client
    restart: always
    container_name: webprog-ss25-fe
    volumes:
      - ./client/nginx/client.conf:/etc/nginx/conf.d/client.conf
    ports:
      - "8084:80"
    env_file:
      - ./client/.env.production
    depends_on:
      - server

  server:
    build: ./server
    restart: always
    container_name: webprog-ss25-be
    ports:
      - "8085:80"
    environment:
```

# DevOps – Nginx Proxy Manager

## Technische Umsetzung

The screenshot shows the Nginx Proxy Manager web interface. A modal window titled "Edit Proxy Host" is open in the center. The "Domain Names" field contains "voic.rajsingh.it". The "Scheme" field is set to "http", "Forward Hostname / IP" is "pi", and "Forward Port" is "8086". There are three active toggle switches: "Cache Assets", "Block Common Exploits", and "Websockets Support". The "Access List" field contains "Publicly Accessible". At the bottom of the modal are "Cancel" and "Save" buttons. Below the modal, a row of proxy host entries is visible: "http://pi:8084", "Let's Encrypt", "PUBLIC", and "http://pi:8086", "Let's Encrypt", "Public". In the background, a list of proxy hosts is shown with columns for "NAME", "STATUS", and "Actions". All hosts are marked as "Online". The top right corner shows a user profile for "Raj Administrator".

# OpenAPI – Konfiguration

## Technische Umsetzung

```
import swaggerJSDoc from "swagger-jsdoc";  
  
const swaggerDefinition = {  
    openapi: "3.0.0",  
    info: {  
        title: "API Documentation",  
        version: "1.0.0",  
        description: "This is the API documentation for webprog-ss25",  
    },  
    components: {  
        schemas: {  
            Message: {  
                type: "object",  
                required: ["message"],  
                properties: {  
                    message: {  
                        type: "string",  
                        example: "Request failed/successful.",  
                    },  
                },  
            },  
        },  
    },  
};
```

# OpenAPI – Initialisierung

## Technische Umsetzung

```
import swaggerUi from "swagger-ui-express";
import { openapiSpec } from "./openapi";

const app = express();

app.use("/api-docs", swaggerUi.serve, swaggerUi.setup(openapiSpec));
app.get("/openapi.json", (req, res) => {
    res.setHeader("Content-Type", "application/json");
    res.send(openapiSpec);
});
```

# OpenAPI – Dokumentation

## Technische Umsetzung

```
import { Router } from "express";
import { testController } from "./test.controller";

/**
 * @swagger
 * tags:
 *   name: Protected
 *   description: Protected endpoints.
 */
const testRouter = Router();

/**
 * @swagger
 * /api/protected/test:
 *   post:
 *     summary: Test if user is logged in.
 *     tags: [Protected]
 *     operationId: testProtected
 *     responses:
 *       200:
```

# OpenAPI – Client-Generierung

## Technische Umsetzung

```
$ npm run openapi-ts
```

```
$ tree client/src/api/
client/src/api/
├── auth.ts
├── client.gen.ts
├── index.ts
├── schemas.gen.ts
├── sdk.gen.ts
├── setup.ts
└── types.gen.ts
```

# WebSockets & Redis Pub/Sub – Initialisierung 1

## Technische Umsetzung

```
import http from "http";
import express from "express";
import WebSocket from "ws";
import { initRedis } from "./modules/chat/chat.redis";
import {
    upgradeWebSocket,
    handleWebSocketConnection,
} from "./modules/chat/chat.gateway";

const app = express();
const server = http.createServer(app);
const wss = new WebSocket.Server({ noServer: true });

server.on("upgrade", (req, socket, head) => {
    upgradeWebSocket(wss, req, socket, head);
});

wss.on("connection", handleWebSocketConnection);

initRedis().then(() => {
    const PORT = process.env.PORT || 3001;
    app.listen(PORT, () => {
        console.log(`Listening on port ${PORT}`);
    });
});
```

# WebSockets + Redis Pub/Sub – Initialisierung 2

## Technische Umsetzung

```
import { createClient } from "redis";

export const redisConfig = {
  url: process.env.REDIS_URL,
};

export const redisPub = createClient(redisConfig);

export async function initRedis() {
  await redisPub.connect();
}
```

# WebSockets + Redis Pub/Sub – Initialisierung 3

## Technische Umsetzung

```
export const upgradeWebSocket = (
  server: WebSocketServer,
  request: IncomingMessage,
  socket: any,
  head: any,
) => {
  // Überprüfung ob Cookie vorhanden und gültig
};

export const handleWebSocketConnection = async (
  ws: WebSocket,
  request: IncomingMessage,
) => {
  // get chats

  const subClient = createClient(redisConfig);
  await subClient.connect();

  await Promise.all(
    chats.map((chat) =>
```

# WebSockets + Redis Pub/Sub – Client

## Technische Umsetzung

```
// ...

export const ChatProvider = ({ children }: { children: ReactNode }) => {
  const wsRef = useRef<WebSocket | null>(null);

  useEffect(() => {
    const loadChats = async () => {
      // Use REST API for initial load
    };

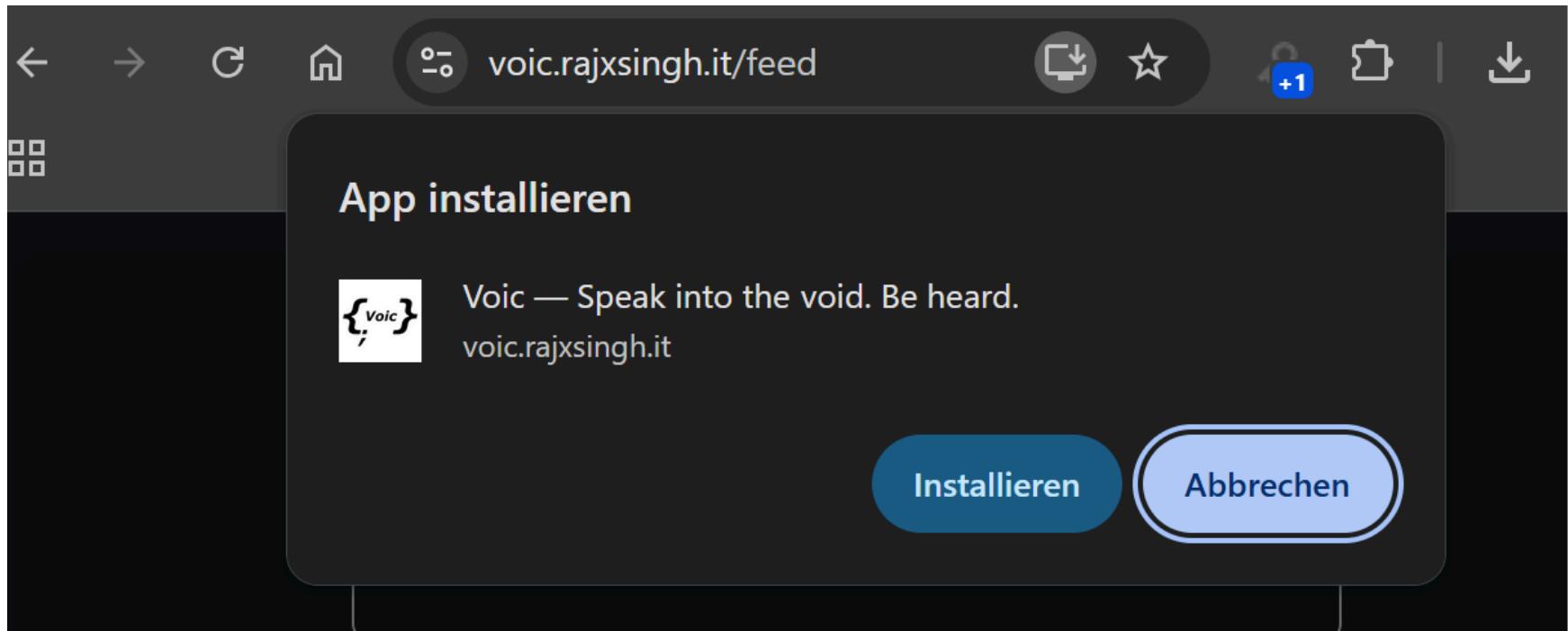
    void loadChats();
  }, [userId]);

  useEffect(() => {
    const ws = new WebSocket(import.meta.env.VITE_WEBSOCKET);
    wsRef.current = ws;

    const handleMessage = (event: MessageEvent) => {
      try {
        const data = JSON.parse(event.data);
      }
    };
  });
}
```

# PWA

Technische Umsetzung



# PWA – manifest.json

## Technische Umsetzung

```
{  
  "name": "Voic – Speak into the void. Be heard.",  
  "short_name": "Voic",  
  "description": "Voic is an anonymous space for honest thoughts and open conversation. No profiles, no likes, no fol",  
  "start_url": ".",  
  "display": "standalone",  
  "background_color": "#00d3bb",  
  "theme_color": "#00d3bb",  
  "icons": [  
    {  
      "src": "icon-192x192.png",  
      "sizes": "192x192",  
      "type": "image/png"  
    },  
    {  
      "src": "icon-512x512.png",  
      "sizes": "512x512",  
      "type": "image/png"  
    }  
  ]}
```

# PWA – service-worker 1

## Technische Umsetzung

```
self.addEventListener("push", function (event) {
  const data = event.data ? event.data.json() : {};
  const title = data.opinionTitle;
  const options = {
    body: data.body,
    icon: "/icon-192x192.png",
    data: data,
  };
  event.waitUntil(self.registration.showNotification(title, options));
});

self.addEventListener("notificationclick", function (event) {
  event.notification.close();
  event.waitUntil(
    clients.matchAll({ type: "window" }).then(function (clientList) {
      for (const client of clientList) {
        if (client.url && "focus" in client) {
          return client.focus();
        }
      }
    })
  );
});
```

# PWA – service-worker 2

## Technische Umsetzung

```
if ("serviceWorker" in navigator) {  
  window.addEventListener("load", () => {  
    navigator.serviceWorker.register("/sw.js").catch(console.error);  
  });  
}
```

# PWA – WebPush API 1

## Technische Umsetzung

```
useEffect(() => {
  const vapidPublicKey = import.meta.env.VITE_VAPID_PUBLIC_KEY;

  async function subscribeUserToPush(publicKey: string) {
    if ("serviceWorker" in navigator && "PushManager" in window) {
      const reg = await navigator.serviceWorker.ready;
      const sub = await reg.pushManager.subscribe({
        userVisibleOnly: true,
        applicationServerKey: urlBase64ToInt8Array(publicKey),
      });
      return sub;
    }
  }

  if (userId && vapidPublicKey) {
    subscribeUserToPush(vapidPublicKey).then(async (sub) => {
      // save subscription to db using REST API
    });
  }
}, [userId]);
```

# PWA – WebPush API 2

## Technische Umsetzung

```
import webpush from "web-push";
import { getUserSubscriptions, removeSubscription } from "./push.service";

// Generate VAPID keys only once and store them securely. For demo, you can generate and log them here:
// const vapidKeys = webpush.generateVAPIDKeys();
// console.log('VAPID Public Key:', vapidKeys.publicKey);
// console.log('VAPID Private Key:', vapidKeys.privateKey);

webpush.setVapidDetails(
  process.env.VAPID_CONTACT_EMAIL as string,
  process.env.VAPID_PUBLIC_KEY as string,
  process.env.VAPID_PRIVATE_KEY as string
);

export async function sendPushNotification(userId: string, payload: any) {
  const userSubs = await getUserSubscriptions(userId);
  for (const sub of userSubs) {
    const pushSubscription = {
      endpoint: sub.endpoint,
      keys: {
        p256dh: sub.p256dh,
        auth: sub.auth
      }
    };
    webpush.sendNotification(pushSubscription, payload);
  }
}
```

# Fazit

# Fazit

## Fazit

- Neue Technologien erfolgreich eingesetzt
- Alle haben Neues gelernt und ausprobiert
- Hohe Motivation durch kreative Freiheit
- Viel Spaß durch praxisnahes Arbeiten

# Ausblick

## Fazit

- Anonyme Plattform als bewusstes Experiment
- Raum für ehrliche Diskussion – aber Missbrauch möglich
- Bedarf an Mechanismen zur Erkennung von problematischem Content
- Ziel: Balance zwischen Freiheit und Verantwortung
- Viel Potenzial für Weiterentwicklung

Fragen?

Meinungen?

