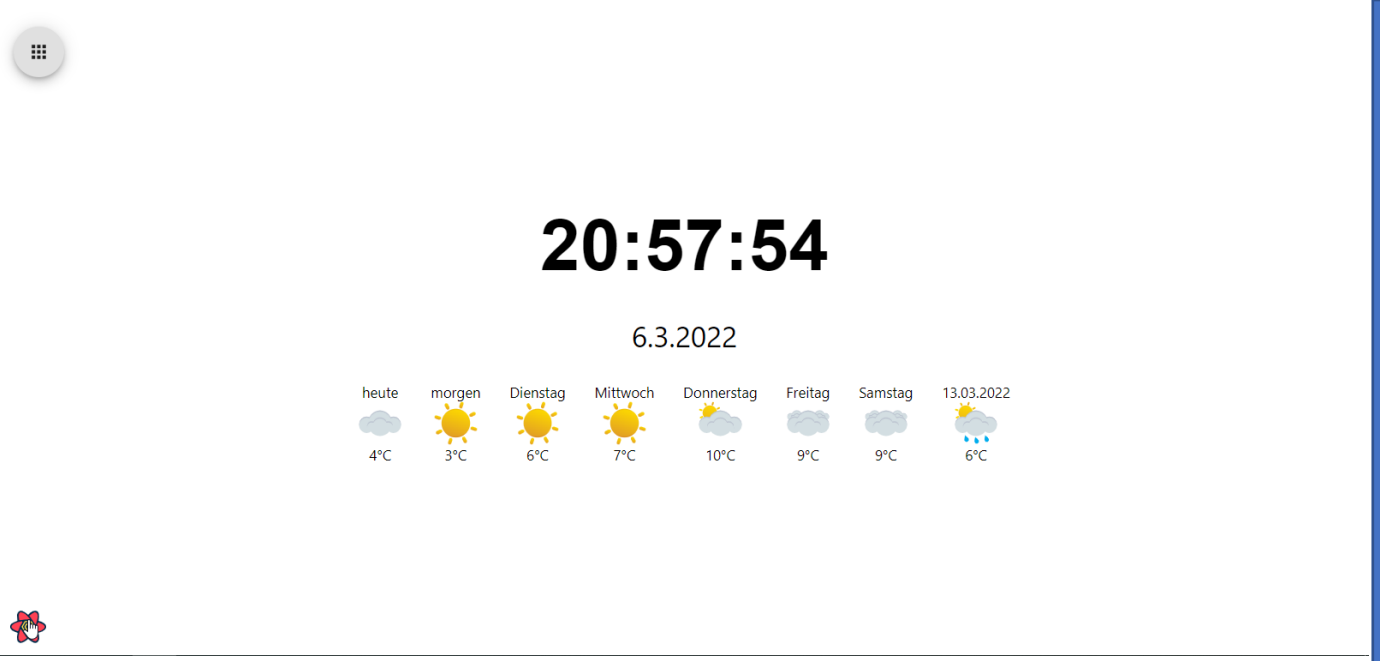
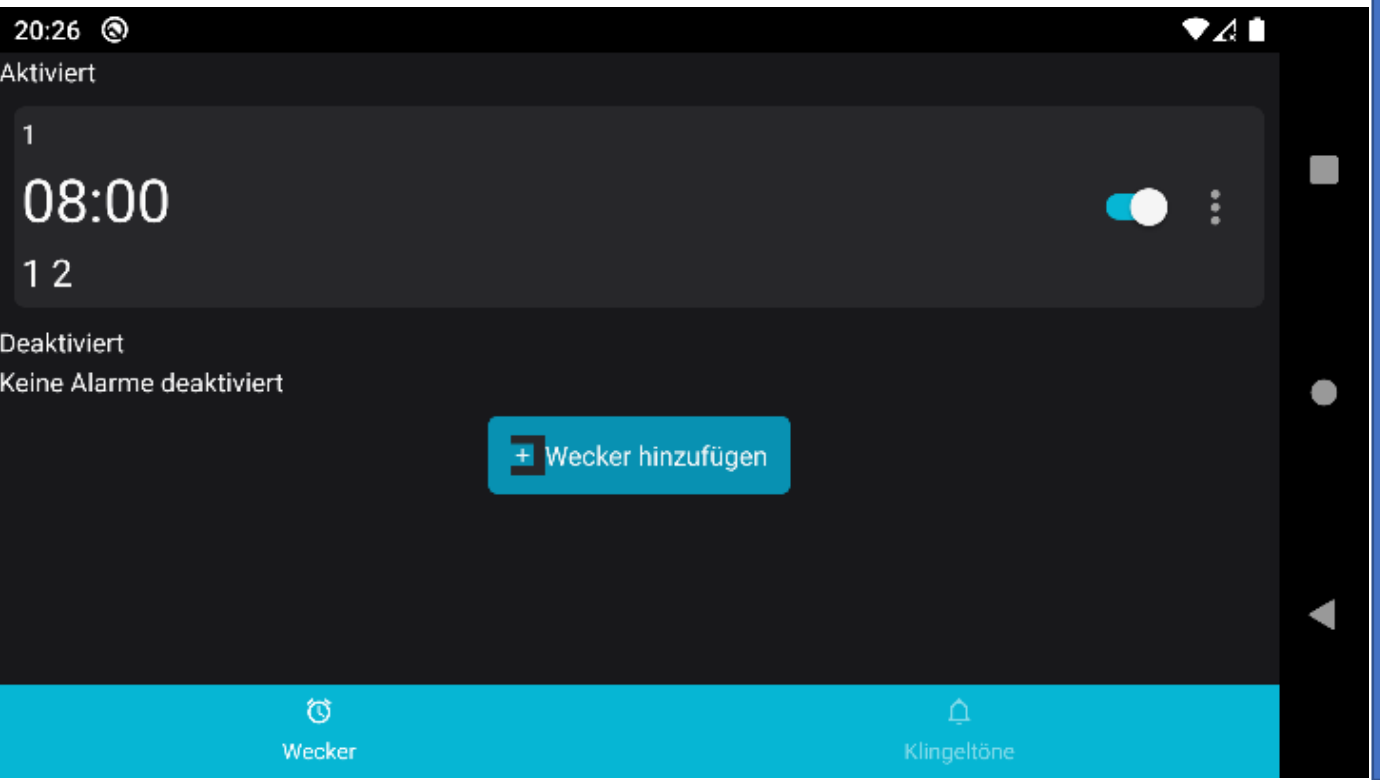




W-Seminar Robotik

Smart Alarm

Bernd Storath

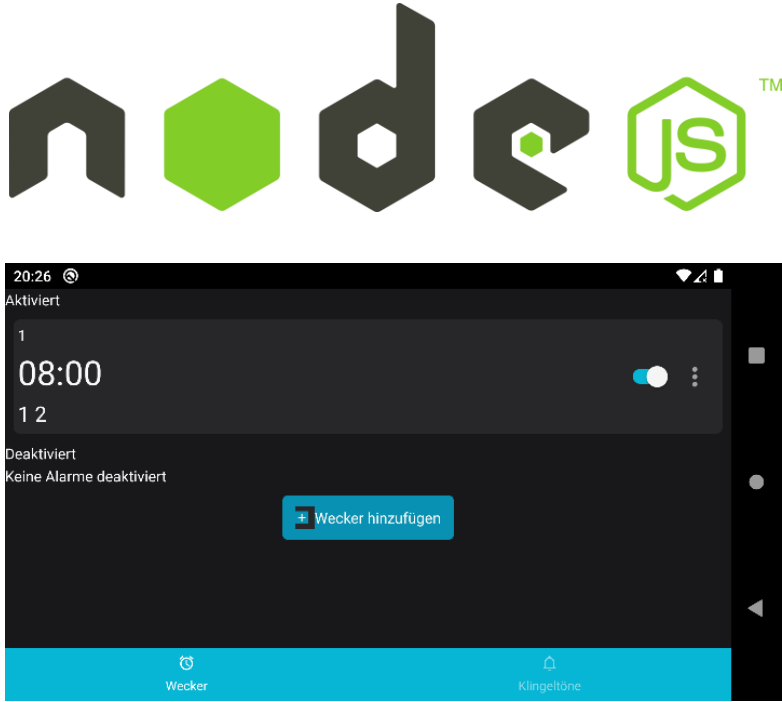
Aufbau

Frontend

Backend

Auf dem Wecker

App



Frontend

Electron + React

Zeigt:
- Uhrzeit
- Wetter
- Einstellungen
- Aus-Schalter

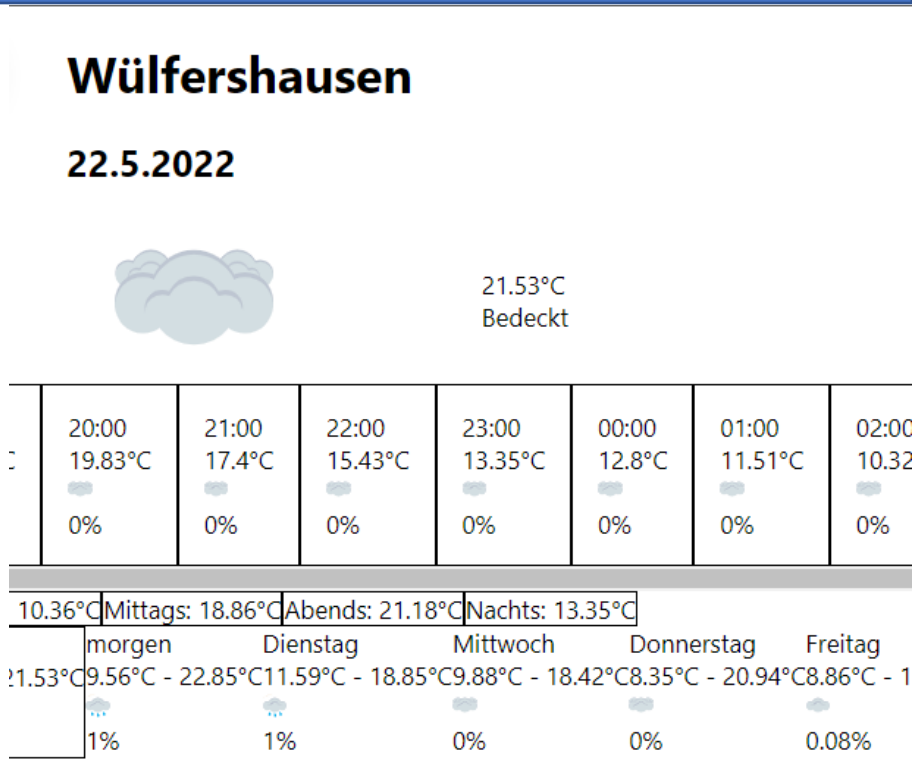
An

Noch nicht fertig:
- Wecker
- Musik

Frontend

Wetter

OpenWeatherMap
- Kostenlos
- 60 Anfragen /Minute
- 1 Mio. Anfragen /Monat



Backend

Node.JS Server

Steuert:

- Kommunikation zwischen App und Frontend
- Datenbank
- Alarm Dienst
- Speicher für Musik und Klingentöne

```
import { Server } from 'socket.io';

import Alarm from './Alarm';
import Routing from './routes';
import SocketIO from './socket';

config();

const zodConfig = createConfig({
  cors: true,
  logger: {
    level: 'debug',
    color: true
  },
  server: {
    upload: true,
    listen: process.env.PORT || 8080
  },
  startupLogo: false
});

const { httpServer } = createServer(zodConfig, Routing);

export const socketIO = new SocketIO(new Server(httpServer, { cors: { origin: '*' } }));

const alarm = new Alarm();

export const stopServer = () => {
  alarm.stop();
  return new Promise<true>((res, rej) => {
    httpServer.close((err) => {
      if (err) {
        rej(err);
      }
      res(true);
    });
  });
```

Backend

API

REST:

- POST /api/alarms
  - ➢ Erstellt neuen Alarm
- DELETE /api/alarms
  - ➢ Löscht einen Alarm

Etc.

Nutzt express-zod-api

```
import { DependsOnMethod, Routing, ServeStatic } from 'express-zod-api';
import { join } from 'path';

import deleteAlarmsController from './Controllers/deleteAlarms';
import deleteRingtonesController from './Controllers/deleteRingtones';
import getAlarmsController from './Controllers/getAlarms';
import getRingtonesController from './Controllers/getRingtones';
import getSettingsController from './Controllers/getSettings';
import patchAlarmsController from './Controllers/patchAlarms';
import postAlarmsController from './Controllers/postAlarms';
import postRingtonesController from './Controllers/postRingtones';
import postSettingsController from './Controllers/postSettings';

const routing: Routing = {
  api: {
    alarms: new DependsOnMethod({
      get: getAlarmsController,
      delete: deleteAlarmsController,
      patch: patchAlarmsController,
      post: postAlarmsController
    }),
    ringtones: new DependsOnMethod({
      get: getRingtonesController,
      delete: deleteRingtonesController,
      post: postRingtonesController
    }),
    settings: new DependsOnMethod({
      get: getSettingsController,
      post: postSettingsController
    })
  },
  ringtones: new ServeStatic(join(__dirname, '../Ringtones'), {
    index: false,
    redirect: false,
    setHeaders: (res) => {
      res.setHeader('Access-Control-Allow-Origin', '*');
    }
  })
};
```

Backend

Kommunikation

Verbindung von Backend zu Frontend und App mit Socket.IO

Ereignisse:
„databaseChange“: Frontend und App aktualisieren ihre Daten
„alarm“: Frontend zeigt Wecker an

```
import { Server, Socket } from 'socket.io';
import { ExtendedError } from 'socket.io/dist/namespace';
import { DefaultEventsMap } from 'socket.io/dist/typed-events';

class SocketIO {
  server: Server;
  clients: Socket[] = [];
  private _frontend: Socket[] = [];
  constructor(server: Server) {
    this.server = server;
    this.server.on('connection', (socket: Socket) => {
      if (socket.handshake.query.type === 'frontend') {
        this._frontend.push(socket);
      } else if (socket.handshake.query.type === 'client') {
        this.clients.push(socket);
      }
      console.log(
        `CONNECT ${socket.id} ${socket.handshake.address} ${socket.handshake.query.type}`
      );
      socket.on('disconnect', () => {
        if (socket.handshake.query.type === 'frontend') {
          this._frontend = [];
        } else if (socket.handshake.query.type === 'client') {
          this.clients = this.clients.filter((s) => s.id !== socket.id);
        }

        console.log(
          `DISCONNECT ${socket.id} ${socket.handshake.address} ${socket.handshake.query.type}`
        );
      });
    });
  }
```

Backend

Datenbank

Speichert

- Einstellungen
  - Position
  - Zeitzone
- Alarme
  - Name
  - Klingelton
  - Tage
  - Zeit
  - Aktiviert
- Klingeltöne
  - Name
  - URL


```
{
    "settings": {
        "timezone": "Europe/Berlin",
        "location": {
            "city": "Wülfershausen",
            "countryCode": "DE",
            "lat": 50.0540866,
            "lon": 9.9995626
        }
    },
    "ringtones": [{
            "name": "Alarm",
            "location": "/ringtones/Alarm.mp3"
        }
    ],
    "alarms": [{
            "ringtone": "Alarm",
            "time": "08:00",
            "days": [1, 2, 6],
            "enabled": true,
            "name": "1"
        }
    ],
    "initialized": true
}
```

Backend

Alarm Dienst

Überprüft jede Minute ob es einen aktivierten Wecker für den jeweiligen Tag gibt:
→ Ja: sendet Ereignis an Frontend

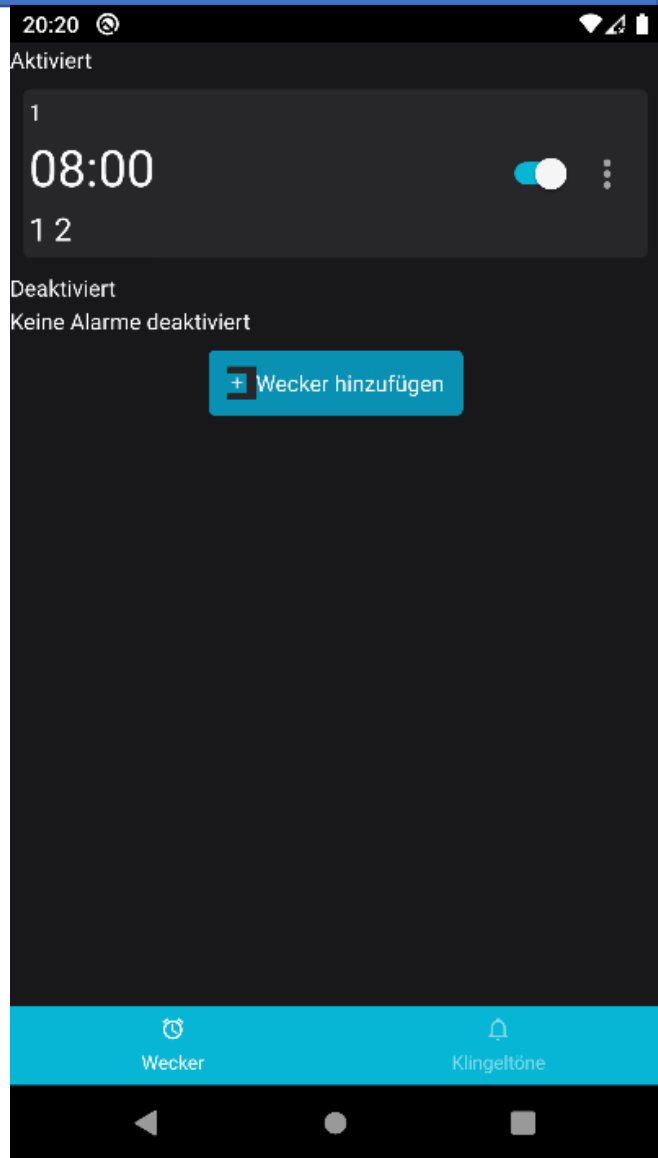
```
export default class Alarm {
  days = ['Montag', 'Dienstag', 'Mittwoch', 'Donnerstag', 'Freitag', 'S
  interval?: NodeJS.Timeout;
  constructor() {
    this.initializeInterval();
  }
  /** Start interval at beginning of minute */
  initializeInterval() {
    if ([0, 1, 2, 3].includes(new Date().getSeconds())) {
      console.log('Starting Alarm Service');
      this.check();
      this.interval = setInterval(() => this.check(), 1 * 60 * 1000);
    } else {
      setTimeout(() => this.initializeInterval(), 3000);
    }
  }
  check() {
    const alarms = db.getAlarms();
    const date = new Date();
    /** `Day, HH:MM:SS` */
    const currentDate = date.toLocaleTimeString('de-DE', {
      timeZone: db.getSettings().timezone,
      weekday: 'long'
    });
    /** In `HH:MM` format */
    const currentTime = currentDate.slice(0, -3).slice(-5);
    const currentDay = currentDate.split(',')[0];
    const currentDayIndex = this.days.indexOf(currentDay) + 1;
    for (const alarm of alarms) {
      if (alarm.enabled && alarm.time === currentTime && alarm.days.inc
        this.ring(alarm);
      }
    }
  }
  ring(alarm: database['alarms'][0]) {
    console.log('Ringing', alarm);
    socketIO.emitFrontend('alarm', alarm);
  }
  stop() {
    this.interval && clearInterval(this.interval);
```

App

React Native

Steuert das Frontend

- Wecker
  - Aktivieren/deaktivieren
  - Bearbeiten
  - Löschen
- Klingeltöne, Musik, Einstellungen
  - ➢ Noch nicht implementiert



App

Design

Figma

Komponenten

Raspberry Pi

Display

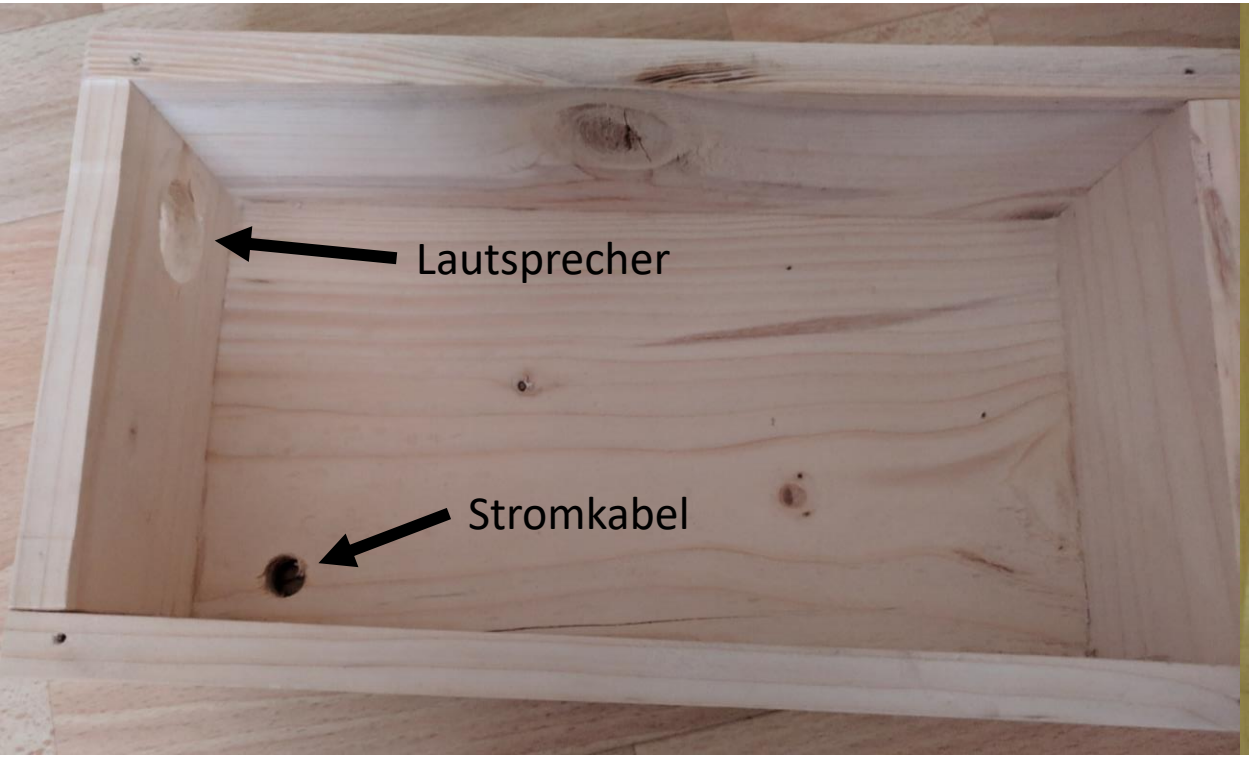
Lautsprecher

Verstärker

Gehäuse

→ Preis: ≈105€

Gehäuse



Aus Holz

Maße (HxLxT): 16cm x 29cm x 10cm

Vergleich

Nest Hub

Preis: 99,99€

Vorteile:
- Billiger (-5€)
- Bessere Lautsprecher
- Mehr Funktionen

Nachteile:
- Keine Anpassungsmöglichkeiten
- Google Account erforderlich
- Nur Plastik

Right to Repair

Reparaturmöglichkeit

- Logisch getrennte Komponenten
  → Kaputt -> Ersetzen

- Nicht verklebt, sondern verschraubt

Zukunft

Ziele

- Sprachbefehle
- Smart Home
- Integrationen
  - Nachrichten
  - Stau