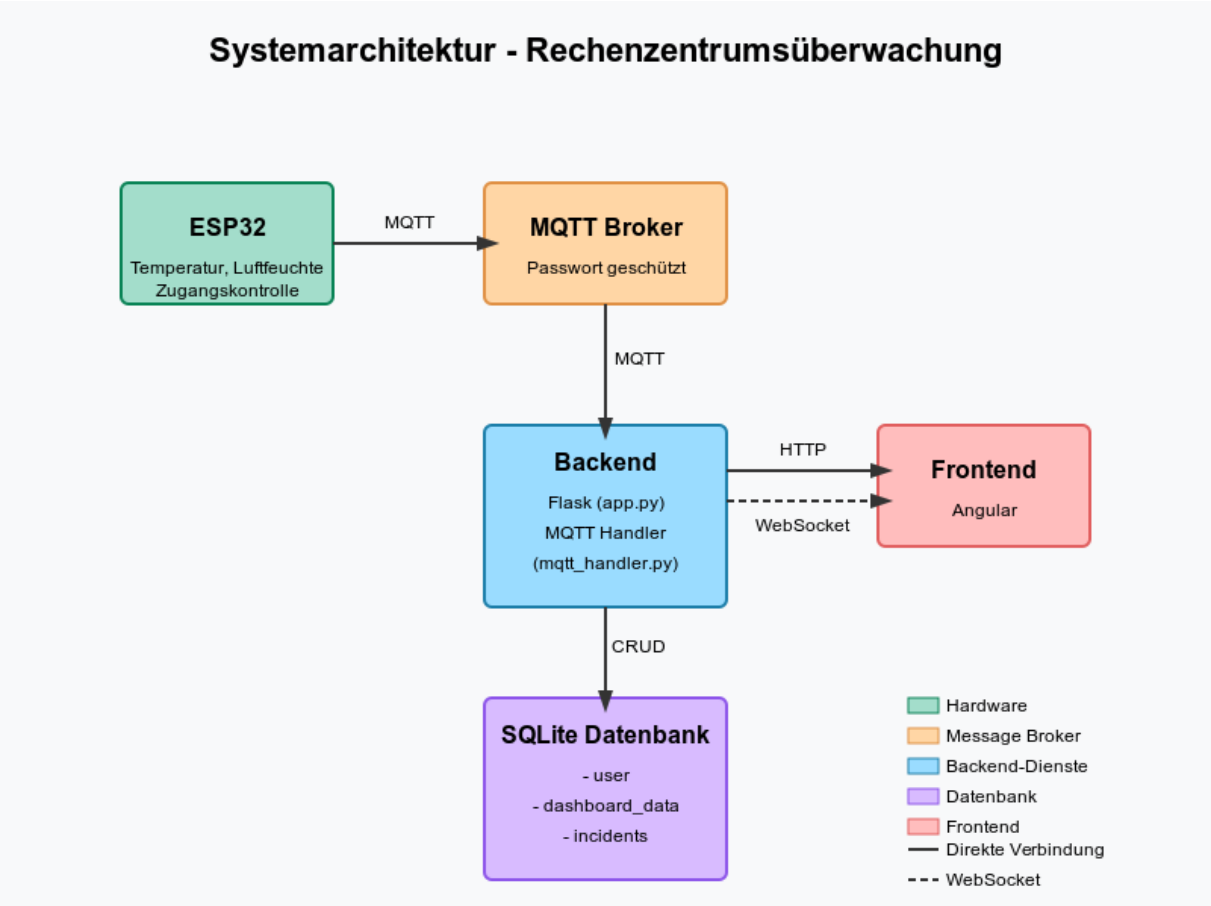
Systemarchitektur

1. Systemübersicht

Das System dient der umfassenden Überwachung eines Rechenzentrums mit integrierter Zugangskontrolle. Es besteht aus vier Hauptkomponenten:

1. **ESP32 Mikrocontroller**: Hardware-Komponente für Sensorik und Zugangskontrolle
2. **MQTT Broker**: Kommunikationsschnittstelle für Echtzeit-Datenaustausch
3. **Backend-Server**: Flask-basierte Anwendung zur Datenverarbeitung und API-Bereitstellung
4. **Frontend-Anwendung**: Angular-basierte Benutzeroberfläche

1.1 Schematischer Aufbau:



2. Komponenten im Detail

2.1 ESP32 Mikrocontroller

- **Funktionen**:

    - Erfassung von Umgebungsdaten (Temperatur, Luftfeuchtigkeit)
    - Steuerung der Zugangskontrolle
    - Überwachung und Protokollierung von Zutrittsversuchen
    - Alarmfunktion
- **Kommunikation**:
    - Verbindung als MQTT-Client mit dem zentralen MQTT-Broker
    - Sendet regelmäßig Sensordaten
    - Empfängt Konfigurations- und Steuerungskommandos

2.2 MQTT Broker

- **Eigenschaften**:
    - Passwort-geschützte Instanz
    - Dient als zentraler Kommunikationsknoten für alle Echtzeit-Daten
- **Sicherheit**:
    - Authentifizierung für alle Clients erforderlich
    - Verschlüsselte Kommunikation

2.3 Backend-Server (Flask)

- **Hauptkomponenten**:
    - app.py: Hauptanwendungsdatei mit REST API-Endpunkten
    - mqtt_handler.py: Verarbeitet MQTT-Kommunikation
- **Funktionen**:
    - HTTP-API für Frontend-Anfragen (historische Daten, Konfiguration)
    - WebSocket-Server für Echtzeit-Updates
    - Datenbank-Interaktion (Lesen/Schreiben)
    - Verarbeitung und Weiterleitung von MQTT-Nachrichten
    - Bidirektionale Kommunikation mit ESP32 über MQTT:
        - Empfangen von Sensordaten und Ereignissen
        - Senden von Konfigurationsbefehlen und Steuerungsinformationen

2.4 SQLite Datenbank

- **Tabellen**:
    - user: Benutzerverwaltung (id, username, pass)

    - dashboard_data: Sensormesswerte (id, timestamp, value)
    - incidents: Protokoll von Vorfällen und Zugriffen (id, timestamp, type, value)
- **Datensicherung**:
    - Lokale Speicherung mit regelmäßigem Backup (empfohlen)

2.5 Frontend (Angular)

- **Funktionen**:
    - Benutzeroberfläche für Systemüberwachung
    - Dashboard für Echtzeitdaten und historische Verläufe
    - Konfigurationsschnittstelle für ESP32
    - Anzeige und Filterung von Vorfällen und Zugriffen
- **Kommunikation**:
    - HTTP-Requests für API-Zugriffe und Konfigurationsänderungen
    - WebSocket-Verbindung für Echtzeit-Updates

3. Datenfluss

3.1 Sensordatenerfassung

1. ESP32 erfasst Sensordaten (Temperatur, Luftfeuchtigkeit)
2. Daten werden über MQTT an den Broker gesendet
3. Backend (mqtt_handler.py) empfängt diese Daten
4. Daten werden in die dashboard_data Tabelle geschrieben
5. Gleichzeitig werden Daten über WebSocket an das Frontend gesendet
6. Frontend zeigt aktuelle Werte im Dashboard an

3.2 Zugangskontrolle

1. Zugangsversuch wird am ESP32 registriert
2. ESP32 sendet Ereignis über MQTT
3. Backend verarbeitet das Ereignis
4. Ereignis wird in der incidents Tabelle protokolliert
5. Benachrichtigung wird über WebSocket an das Frontend gesendet
6. Frontend aktualisiert die Anzeige der Vorfälle

3.3 Konfigurationsänderungen

1. Benutzer ändert Parameter im Frontend

2. Frontend sendet HTTP-Request an Backend
3. Backend verarbeitet die Anfrage
4. Bei Bedarf wird eine MQTT-Nachricht an den ESP32 gesendet
5. ESP32 passt seine Konfiguration entsprechend an
6. ESP32 sendet Bestätigung über MQTT zurück
7. Bestätigung wird an das Frontend weitergeleitet

4. Sicherheitsaspekte

4.1 Authentifizierung

- Frontend-Benutzer: Anmeldung über Benutzername/Passwort
- MQTT-Kommunikation: Passwortgeschützt
- API-Zugriffe: Session-basierte Authentifizierung
- JWT Token

4.2 Zugangskontrolle

- Protokollierung aller Zutrittsversuche
- Alarmmeldungen bei unbefugten Zugriffen & Überschreiten der Sensorthresholds
- Historische Nachverfolgbarkeit durch incidents Tabelle

5. Technische Anforderungen

5.1 Hardware

- ESP32 Mikrocontroller mit entsprechenden Sensoren
- Server für Backend und MQTT-Broker

5.2 Software

- Python mit Flask für das Backend
- Angular für das Frontend
- MQTT-Broker (z.B. Mosquitto)
- SQLite als Datenbank