Health Checker

Projektdokumentation

Studiengang Systems Engineering (B. Eng.)
Schwerpunkt I.2

Verfasser/innen:

Jan-Niclas Fenger
Matrikelnummer: 2092175
E-Mail: jan-niclas.fenger@tha.de

Leopold Weber
Matrikelnummer: 2155780
E-Mail: leopold.weber@tha.de

Simon Schneider
Matrikelnummer: 2151357
E-Mail: simon.schneider1@tha.de

Janis Preiß
Matrikelnummer: 2150826
E-Mail: janis.preiss@tha.de

Betreuer/in:

Prof. Dr. Volodymyr Brovkov

1. Februar 2025

Inhaltsverzeichnis

1 Zusammenfassung

Jan

2 Einleitung

Jan

2.1 Kontext/Hintergrund

2.2 Projekt Motivation

2.3 Problemstellung

2.4 Projektziele

2.5 Projektumfang

2.6 Übersicht über die Dokumentation

3 Methodik

Simon

4 Software und User Interface

Leopold

4.1 Messsoftware

Die Messsoftware dieses Projekts setzt sich aus mehreren Programmen zusammen. Die genaue Funktionsweise und Interaktionen der einzelnen Programme miteinander werden nun im Folgenden erklärt.

4.2 Datenbanksystem

4.2.1 Einführung

Das Datenbanksystem (DBS) wurde ursprünglich mit SQLite3 implementiert, da es relativ einfach und intuitiv zu handhaben ist benutz haben wir eine SQL Row based Datenbank. Allerdings ergeben sich hierbei Probleme bei mehrfachen Zugriffen auf die Datenbank. Um diese Herausforderungen zu lösen, wurde auf ein Server-Client-Modell mit PostgreSQL umgestellt.

4.2.2 Problemstellung bei SQLite3

SQLite3 basiert auf Binärcode, was zu Problemen führen kann, wenn zwei Anwendungen gleichzeitig auf die gleiche Datenbank zugreifen. In diesem Szenario schreibt das Prozessprogramm (Python) die Messwerte in die Datenbank, während die GUI (Webapplikation) die Daten aus der Datenbank liest. Wenn das Prozessprogramm gerade Daten in die Datenbank schreibt, wird diese blockiert. Dadurch kann nicht gleichzeitig aus der Datenbank gelesen werden.

4.2.3 Lösung: Umstellung auf PostgreSQL

Um die gleichzeitige Nutzung durch mehrere Anwendungen zu ermöglichen, wurde auf PostgreSQL umgestellt. Als Open-Source-Datenbank bietet PostgreSQL folgende Vorteile:

- Multi-User-Unterstützung: Mehrere Clients können gleichzeitig auf den zentralen Server zugreifen.
- Transaktionssicherheit: Durch den Einsatz von Transaktionen werden Schreib- und Lesevorgänge konsistent und sicher abgewickelt.
- Erweiterbarkeit: Zahlreiche Erweiterungen und Anpassungsmöglichkeiten machen PostgreSQL zu einer flexiblen Lösung.
- Netzwerkbasierter Zugriff: Das System kann als zentraler Datenbankserver in unterschiedlichen Umgebungen eingesetzt werden.

Aktuell läuft der PostgreSQL-Server auf einem Raspberry Pi. Eine zukünftige Weiterentwicklung sieht jedoch den Betrieb auf einem zentralen Server vor, wobei einzelne Komponenten wie die grafische Benutzeroberfläche (GUI) ebenfalls ausgelagert werden könnten.

4.2.4 Schematischer Aufbau

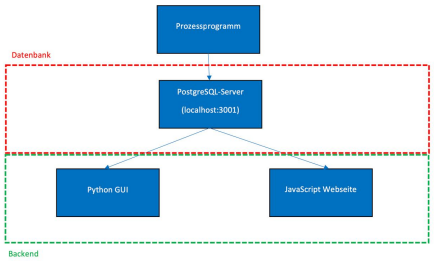

Abbildung 1: Schematischer Aufbau des Datenbanksystems

Bildgröße am Ende anpassen

4.2.5 Zusammenfassung

Durch PostgreSQL wird ein robustes und flexibles Datenbanksystem geschaffen, das die Vorteile beider Technologien vereint. Das Frontend in Python erlaubt eine lokale Steuerung, während die Website als Verwaltungspunkt für einen Admin dient. Insgesamt lässt sich das System als lehr- und praxisnah bewerten.

4.3 Technologien und Architektur

Für die Umsetzung der Webanwendung haben wir uns für einen aktuellen Technologie-Stack entschieden, der uns viel Flexibilität und eine klare Struktur bietet:

- **HTML5 & CSS3:**
  Die Grundstruktur der Anwendung basiert auf HTML5. Für das Styling nutzen wir CSS3 in Kombination mit Tailwind CSS, was uns ein responsives und ansprechendes Design ermöglicht.

- **JavaScript & Chart.js:**
  Dynamische Inhalte und interaktive Elemente werden mit JavaScript realisiert. Für die Darstellung von Diagrammen, etwa zur Visualisierung von Temperaturverteilungen oder Mitarbeiterrollen, setzen wir auf Chart.js.

- **RESTful API:**
  Um eine reibungslose Kommunikation zwischen Frontend und Backend sicherzustellen, verwenden wir eine RESTful API, die im JSON-Format arbeitet. Über diese Schnittstelle werden Funktionen wie die Benutzerregistrierung, der Login, die Mitarbeiterverwaltung und diverse Datenbankabfragen abgewickelt.

- **Responsive Design:**
  Dank moderner CSS-Frameworks und responsiven Layouts passt sich die Anwendung automatisch an verschiedene Geräte an – egal, ob auf dem Desktop oder auf mobilen Geräten.

4.4 Benutzerinterface und Funktionalitäten

Das Benutzerinterface ist in mehrere Bereiche unterteilt, um die Bedienung möglichst einfach zu gestalten.

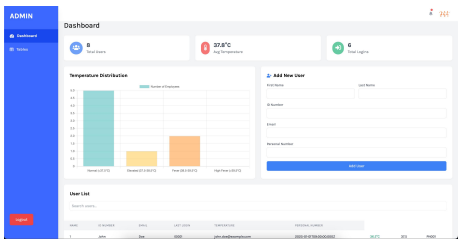

Abbildung 2: Ausschnitt aus dem Graphical User Interface (GUI)

Bildgröße am Ende anpassen

Über eine ständig sichtbare Sidebar mit passenden Icons gelangt man schnell zu wichtigen Bereichen wie dem Dashboard oder den Tabellenansichten. Im Dashboard

bekommt man einen Überblick über wichtige Kennzahlen – etwa die Anzahl der Nutzer, die Durchschnittstemperatur und die Login-Aktivitäten – die in Echtzeit aktualisiert werden.

Neue Benutzer oder Mitarbeiter können über das Login hinzugefügt werden, wobei Eingaben sofort validiert werden und Fehler direkt angezeigt werden.

Die Messdaten werden übersichtlich in Diagrammen dargestellt, zum Beispiel zeigt ein Balkendiagramm die Temperaturverteilung und ein anderes Diagramm die Verteilung der Mitarbeiterrollen.

Zudem sorgen Such- und Filterfunktionen in den Tabellen dafür, dass man die Daten schnell durchsuchen und analysieren kann. Die Anmeldung erfolgt über ein Login-System, das auf JSON Web Tokens (JWT) basiert, sodass nur berechtigte Nutzer Zugriff auf die Anwendung haben.

Fazit:
Durch die Kombination dieser Technologien entstand eine benutzerfreundliche Webanwendung, die alle wichtigen Daten zentral anzeigt und verwaltet. Die modulare Architektur erlaubt zudem eine einfache Erweiterung, falls zukünftig weitere Anpassungen notwendig werden falls unser Projekt jemals in Produktion gehen würde.

4.5 API-Dokumentation

Die API unterstützt die Benutzer- und Mitarbeiterverwaltung sowie die Kommunikation mit der Datenbank. Im Folgenden sind die wichtigsten Endpunkte und deren Funktion kurz zusammengefasst.

4.5.1 Allgemeines

- **Datenformat:** Alle Anfragen und Antworten erfolgen in JSON.
- **Fehler:** Bei ungültigen Anfragen wird eine entsprechende Fehlermeldung mit HTTP-Statuscode zurückgegeben.
- **Port:** Standardmäßig läuft der Server auf Port 3001 (konfigurierbar über .env).

4.5.2 Benutzerverwaltung

Registrierung

- **Methode:** POST
- **Endpoint:** /api/register
- **Body:**

```
{
  "username": "testuser",
  "password": "securepassword"
}
```

- **Antworten:**
  - *201 Created:* Benutzer erfolgreich registriert
  - *400 Bad Request:* Benutzername existiert bereits oder erforderliche Felder fehlen
  - *500 Internal Server Error:* Fehler beim Speichern in der Datenbank

Login

- **Methode:** POST
- **Endpoint:** /api/login
- **Body:**

```
{
  "username": "testuser",
  "password": "securepassword"
}
```

- **Antworten:**
  - *200 OK:* Login erfolgreich, liefert ein JWT-Token (gültig für eine Stunde)
  - *401 Unauthorized:* Falscher Benutzername oder Passwort
  - *500 Internal Server Error:* Fehler beim Abrufen des Benutzers

4.5.3 Datenbankverwaltung

Datenbankverbindung testen

- **Methode:** GET
- **Endpoint:** /test-db
- **Antworten:**
  - *200 OK:* Verbindung erfolgreich
  - *500 Internal Server Error:* Fehlerhafte Verbindung

4.5.4 Mitarbeiterverwaltung

Mitarbeiter abrufen

- **Methode:** GET
- **Endpoint:** /api/employees
- **Antworten:**
  - *200 OK:* Array mit allen Mitarbeitern

- *500 Internal Server Error:* Fehler beim Abrufen der Daten

Neuen Mitarbeiter hinzufügen

- **Methode:** POST
- **Endpoint:** /api/employess
- **Body:**

```
{
    "first_name": "Max",
    "last_name": "Mustermann",
    "id_number": "12345",
    "email": "max@example.com",
    "personal_number": "67890"
}
```

- **Antworten:**
  - *201 Created:* Mitarbeiter erfolgreich gespeichert
  - *400 Bad Request:* Fehlende oder ungültige Felder
  - *500 Internal Server Error:* Fehler beim Speichern des Mitarbeiters

5 Ergebnisse und Diskussion

Janis

6 Fazit und Ausblick

6.1 Zusammenfassung der Arbeit

Janis

6.2 Diskussion der Implikationen

Leopold

Unsere Untersuchung hat wesentlich dazu beigetragen, grundlegende Zusammenhänge in der Informatik und Elektrotechnik besser zu verstehen. Besonders hervorzuheben ist dabei die Anwendung eines webbasierten Systems welches in Zusammenhang mit einem lokalen Prototyp zusammenspielt. Die PostgreSQL Datenbank könnte durch Zukunftssicherheit auch direkt in Produktion deployt werden. Dasselbe gilt (mit Ausnahme der Authentizität) auch für die Webanwendung.

Besonders positiv ist, dass die benutzerfreundliche GUI, die auf mobilen Geräten nutzbar ist, zeigt, wie technische Messungen auch für Laien verständlich aufbereitet werden können. Dies fördert nicht nur den Alltagseinsatz neuer Technologien, sondern steigert auch das Bewusstsein für Energieeffizienz und Nachhaltigkeit.

Insgesamt liefert das Projekt eine fundierte Basis für weiterführende Studien und Anwendungen, die dazu beitragen können, die die Gesundheit und Nachhaltigkeit von Mitarbeitern und Applikationen zu verbessern.

6.3 Bereiche für zukünftige Arbeiten

Janis

Liste der noch zu erledigenden Punkte