

Pflichtenheft | Sensor-Projekt

Von: Jörg Broy
Leonard Franke
Martin Braun
Richard Lüdtkke

Fach: Systemintegration
Seminargruppe: CS14
Datum 03.11.2016

Inhaltsverzeichnis

1 Zielbestimmung.....	3
1.1 Musskriterien.....	3
1.2 Kannkriterien.....	3
2 Einsatz.....	4
2.1 Anwendungsbereiche.....	4
2.2 Zielgruppen.....	4
2.3 Betriebsbedingungen.....	4
3 System Voraussetzungen.....	5
3.1 Hardware.....	5
3.1.1 Sensor und Microcontroller.....	5
3.1.3 Server Hardware.....	6
3.1.4 Client Hardware.....	6
3.2 Software.....	6
3.2.1 Server Software.....	6
3.2.2 Client Software.....	6
4 Funktionalität.....	7
4.1 Installation: Organisation der Programmbestandteile in Docker-Images und Containern.....	7
4.2 Einbinden neuer Sensoren.....	7
4.3 Administration im Webfrontend.....	8
4.4 Interkommunikation der Dockercontainer.....	8
4.5 Typische Anwendungsabläufe anhand von UML-Diagrammen.....	10
4.5.1 Aktivitätsdiagramm.....	10
4.5.2 Anwendungsfalldiagramm.....	11
5 Daten.....	12
6 Aufbau des Webfrontends.....	13
6.1 Eigenes Framework.....	13
6.2 Funktionen der Webseite.....	14
6.2.1 Workflow.....	14
6.2.2 Login.....	14
6.2.3 Registrieren.....	15
6.2.4 Loginseite.....	15
6.2.5 Homebildschirm.....	16
6.2.6 Benutzereinstellungen.....	16
6.3 API.....	16
6.4 Design der Webseite.....	17
7 Qualitätsziele.....	18
8 Integration in die Monitoringsoftware "Icinga2".....	19
8.1 Realisierung mit Bash-Script.....	19
8.2 Einfache Installation und Erweiterbarkeit.....	20
8.3 Visualisierung mit Graphite.....	20
9 Quellenverzeichnis.....	21
10 Abbildungsverzeichnis.....	21

1 Zielbestimmung

Die Bereitstellung eines Dockerimages mit einem Webserver + Datenbank, so konfiguriert, dass leicht neue Temperatursensoren in die Weboberfläche integriert werden können und die Daten dazu angezeigt werden.

1.1 Musskriterien

Es soll ein funktionfähiges Skript für den Microcontroller zur Verfügung gestellt werden, sodass dieser Daten vom Temperatursensor erhält und an einen Server schickt. Der Server, der die ermittelten Daten erhält, speichert diese in einer Datenbank. Die Messdaten soll man in der Weboberfläche anschaulich einsehen können.

1.2 Kannkriterien

Für die komplette Infrastruktur soll ein Icinga2- bzw. Nagios-Plugin zur Verfügung gestellt werden, welches die Sensoren respektive die Werte der Datenbank in die entsprechende Monitoringsoftware integriert und so auch für Computerlandschaften interessant macht, die bereits mit Nagiosderivaten arbeiten.

2 Einsatz

Das Einbinden und der Abruf von Sensordaten soll durch eine Weboberfläche möglich sein.

2.1 Anwendungsbereiche

Die Anwendung kann überall eingesetzt werden, wo Strom und ein Wlan-Netz zur Verfügung steht, d.h. es kann für die Messung von Temperaturdaten im Wohnbereich, in Serverräumen und in öffentlichen Gebäuden etc. eingesetzt werden.

2.2 Zielgruppen

Informatiker (und Informatikaffine Nutzer) die den ESP8266 zur Messung und Anzeige von Temperaturdaten nutzen wollen. Die größte Hürde stellt dabei die Programmierung des Microcontrollers dar. Ist der Microcontroller erst einmal korrekt in das System eingebunden, sollen auch Laien dazu in der Lage sein, sich den Temperaturverlauf in einer Weboberfläche anzeigen zu lassen.

2.3 Betriebsbedingungen

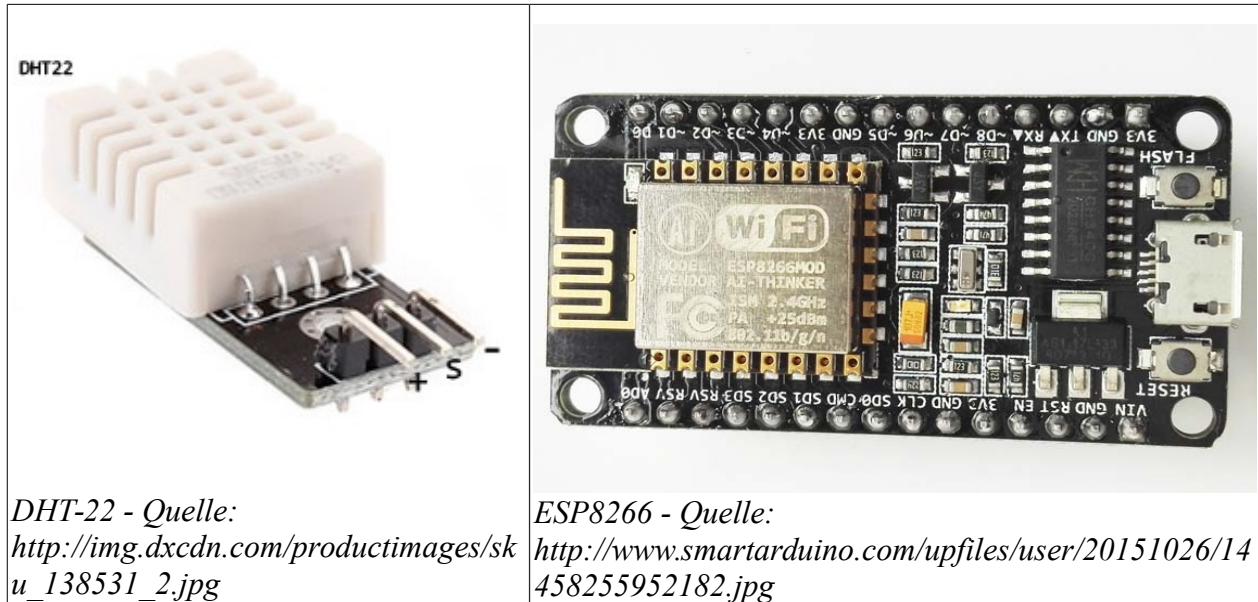
- Betriebsbedingungen: der Einsatz der Technik sollte drinnen geschehen und aus Sicherheitsaspekten lediglich im Intranet betrieben werden
- Betriebszeit: dauerhafter Betrieb 24 Stunden 7 Tage in der Woche
- Beobachtung: unbesichtigter Betrieb

3 System Voraussetzungen

3.1 Hardware

3.1.1 Sensor und Microcontroller

Um die Werte der Temperatur und der Luftfeuchtigkeit zu ermitteln, wird der Sensor DHT22 benutzt. Dieser wird mit dem Microcontroller "ESP8266 NodeMCU" verbunden.



Technische Daten des DHT22:

- Digital Temperatur und Luftfeuchtigkeit ermitteln
- Chip Typ DHT22
- Betriebsspannung: DC 3.3-5.5V
- Luftfeuchtigkeitsmessbereich : 0 bis 100% relative Luftfeuchte
- Feuchtemessgenauigkeit: $\pm 2\%$ RH
- Temperaturbereich: -40 bis +80 C
- Temperaturmessgenauigkeit ± 0.5
- Single-Bus – Digitalsignalausgang, bidirektionale serielle Daten
- zahlreiche Beispielpprogramme für gängige Board im Internet verfügbar
- Maße: 28mm x 12mm x 10mm

Das ESP8266 von dem Hersteller Espressif ist ein programmierbarer WLAN-SoC mit UART- und SPI-Schnittstelle.

Technische Daten des ESP8266:

- 802.11 b/g/n
- Wi-Fi Direct (P2P), soft-AP
- Integrated TCP/IP protocol stack
- Integrated TR switch, balun, LNA, power amplifier and matching network
- Integrated PLLs, regulators, DCXO and power management units
- +19.5dBm output power in 802.11b mode
- Power down leakage current of <10uA
- Integrated low power 32-bit CPU could be used as application processor
- SDIO 1.1/2.0, SPI, UART
- STBC, 1×1 MIMO, 2×1 MIMO
- A-MPDU & A-MSDU aggregation & 0.4ms guard interval
- Wake up and transmit packets in < 2ms
- Standby power consumption of < 1.0mW (DTIM3)
- VCC: 3,3V (Achtung: Eingänge sind NICHT 5V TOLERANT!)

3.1.3 Server Hardware

- Prozessor Architektur x86
- beachte Anforderungen für Betriebssysteme

3.1.4 Client Hardware

- Betriebssystem: Windows, Mac OS X, Linux 32bit/64bit

3.2 Software

3.2.1 Server Software

Betriebssystem:

- CentOS 7.1/7.2 & RHEL 7.0/7.1/7.2 (YUM-based systems)
- Ubuntu 14.04 LTS
- SUSE Linux Enterprise 12
- Docker Version 1.12.2

3.2.2 Client Software

- Arduino IDE Version 1.6.12
- Webbrowser neueste stabile Version(Firefox, Chrome,Opera,...)

4 Funktionalität

4.1 Installation: Organisation der Programmbestandteile in Docker-Images und Containern

Die Bestandteile des Programmes befinden sich in Docker-Images, die auf dem eingesetzten Server zu Docker-Containern ausgerollt werden. Der Nutzer benötigt dafür lediglich Kenntnisse über das Instanzieren von Docker-Containern. Diese Informationen können über das Internet leicht beschafft werden¹ und sind überdies im Github-Repository dieses Projektes erklärt. Alles, was der Anwender an Dateien braucht, ist eine "Docker-Compose"-Datei, die er vom Github-Repository herunterladen und dann nach der Installation der Programme "Docker" und "Docker-Compose" einfach per Kommandozeile ausführen kann. Das Weitere geschieht automatisch: Die Docker-Images werden heruntergeladen, in die dockerüblichen Verzeichnisse kopiert und orchestriert gestartet (Sinn und Zweck von Docker-Compose), so, dass die Anwendung ohne weiteres Zutun und in Gestalt des Webfrontends unter einem festgelegten Socket (Default ist localhost:80) bereit steht.

4.2 Einbinden neuer Sensoren

Um neue Sensoren in die Infrastruktur zu integrieren, sind einige Vorbereitungen nötig. Es wird benötigt:

- Sensor DHT22
- Wlan-Chip esp8266 Development-Board
- Software Arduino

Als erstes wird ein neuer Sensor in der Weboberfläche erstellt. Dazu logt man sich in der Weboberfläche ein und navigiert zur Übersicht aller Sensoren. Dort erstellt man einen neuen Sensor. Nachdem man einen Namen und einen Ort vergeben hat, sieht man in der Tabelle einen neuen Eintrag mit dem erstellten Sensor. In der ersten Spalte steht die ihm zugewiesene Id. Diese merkt man sich.

Der Wlan-chip wird über die Software Arduino programmiert. In dem Verzeichnis "esp8266" findet man den benötigten Code. Am Anfang müssen

```
const char* ssid = "*****";  
const char* password = "*****";
```

konfiguriert sein, damit sich der Wlan-Chip mit dem vorhandenen W-Lan-Netzwerk verbinden kann.

¹ <https://docs.docker.com/engine/getstarted/>

In der Zeile

```
client.println("GET /info.php?chipid=ID&temperature=" + tstring +  
"&humidity=" + hstring);
```

muss noch "ID" durch die gemerkte ID des Sensors ersetzt werden. Nachdem der Sensor richtig mit dem Wlan-Chip verkabelt wurde, kann er über ein Micro-USB-Kabel mit Strom versorgt werden.

Verkabelung Sensor-Wlan-Chip:

- Sensor(-) an Wlan-Chip(GND)
- Sensor(out) an Wlan-Chip(D4)
- Sensor(+) an Wlan-Chip(3V3)

Alsdann ist der Chip in das System integriert und funkt seine Daten an die MariaDB-Datenbank, aus der wiederum das Webfrontend seine Daten bezieht.

4.3 Administration im Webfrontend

Die Webseite bietet eine Nutzerverwaltung (Nutzer hinzufügen, entfernen, verändern und mit Passwörtern versehen) und die Möglichkeit, Sensoren hinzuzufügen, zu entfernen und deren Metadaten zu verändern. Durch die Nutzerverwaltung soll die Sichtbarkeit der verschiedenen Sensoren auf bestimmte Personen begrenzt sein.

4.4 Interkommunikation der Dockercontainer

Bei all diesen Tätigkeiten kommuniziert die Webapp mit einer MariaDB im Hintergrund, die sich in einem separaten Dockercontainer befindet. Konkret geschieht dies über Sockets. Die Dockercontainer kennen gegenseitig ihre IPs bzw. Domainnamen und können per Kombination IP(bzw. Domainname):Port interagieren, da sie per default im selben Subnetz verortet sind. Die DNS-Einträge sind als Links in der "docker-compose.yml" hinterlegt und erlauben eine unabhängige Erreichbarkeit der Container, falls sich die IP-Adressen einmal ändern sollten.

Auszug aus der "docker-compose.yml":

```
mariadb:
...

phpmyadmin:
...
  ports:
    - 8080:80
  links:
    - mariadb:db

php:
...
  ports:
    - 80:80
  links:
    - mariadb:mysql

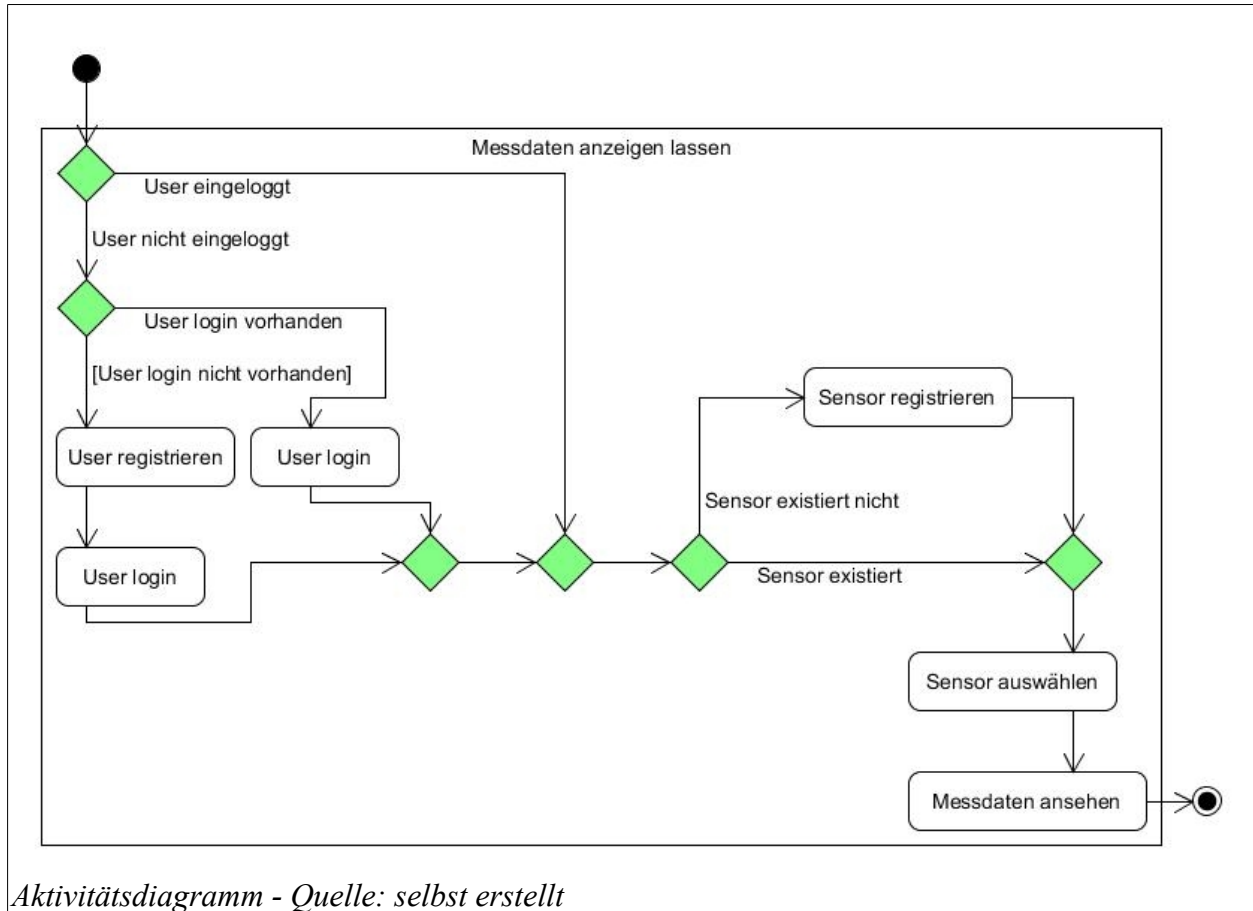
icinga2:
...
  ports:
    - 60000:80
  links:
    - mariadb:mysql
```

"docker-compose.yml" (Auszug) - Quelle: selbst erstellt

Die Container Icinga2 und PHP können damit den Container MariaDB per Domainname erreichen und auf die Datenbank zugreifen.

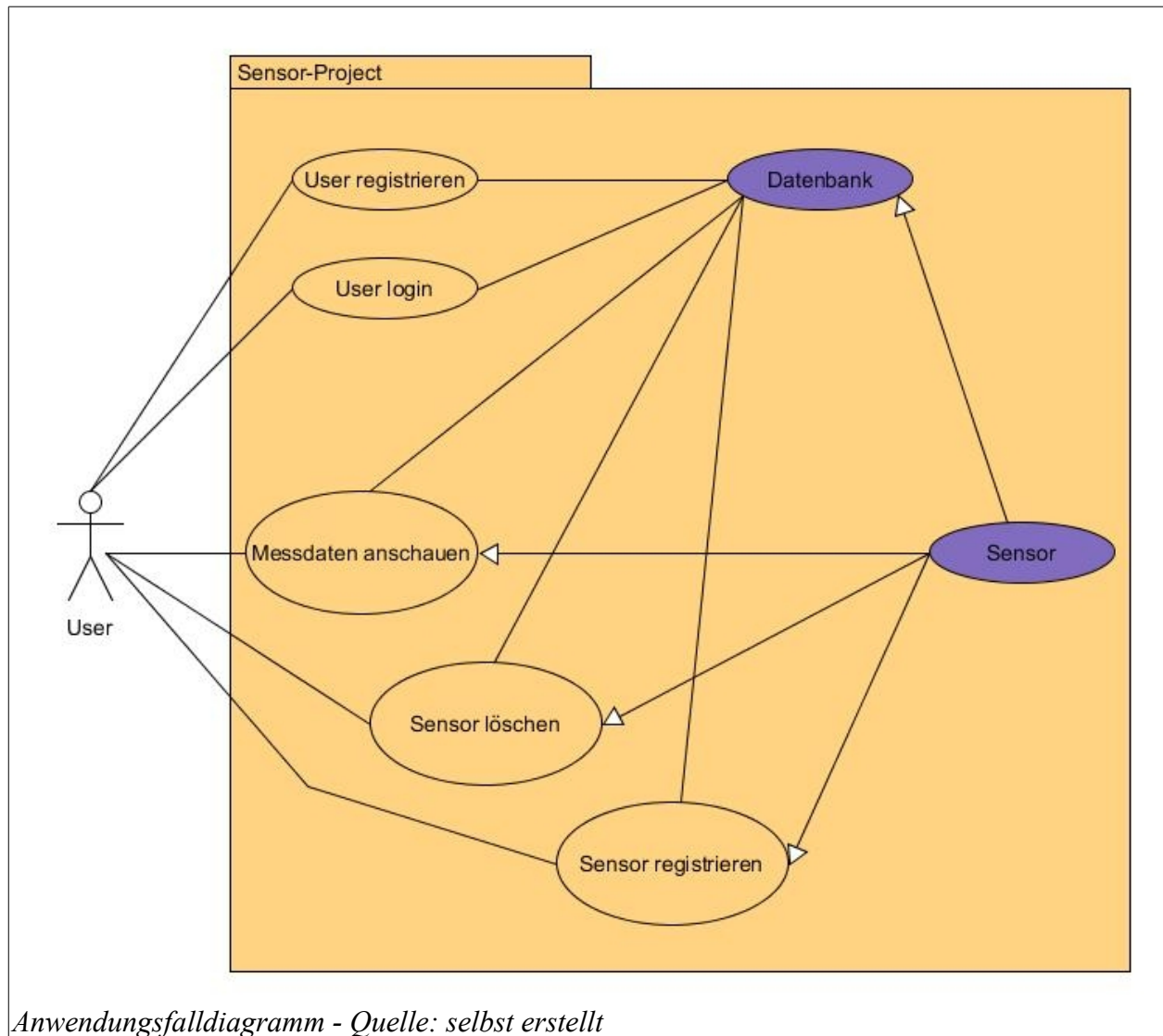
4.5 Typische Anwendungsabläufe anhand von UML-Diagrammen

4.5.1 Aktivitätsdiagramm



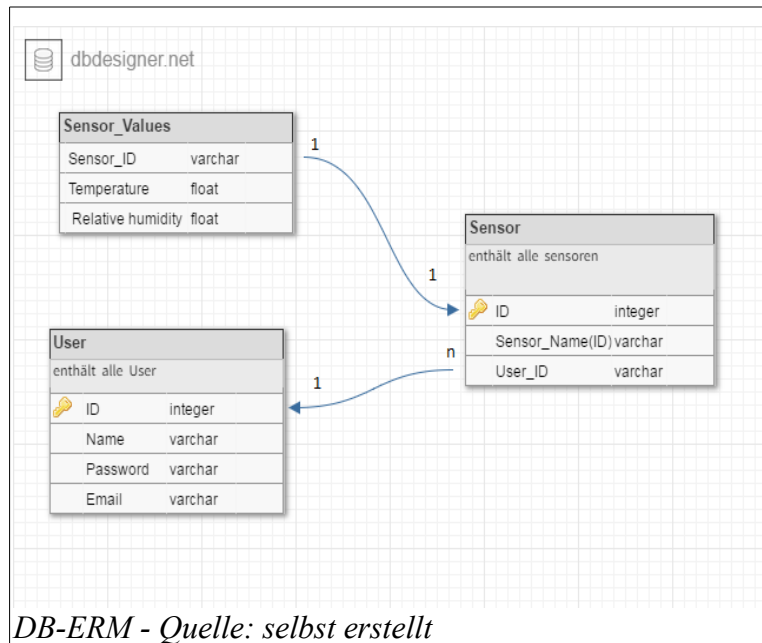
Aktivitätsdiagramm - Quelle: selbst erstellt

4.5.2 Anwendungsfalldiagramm



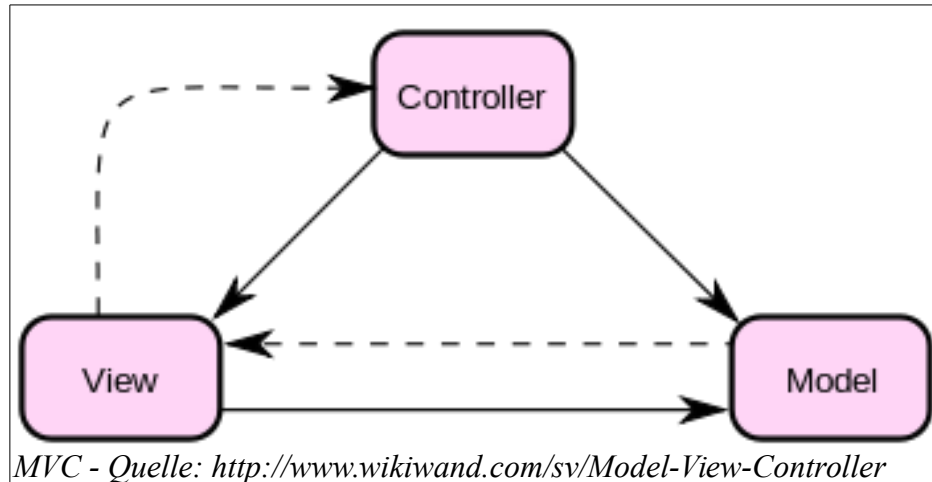
5 Daten

Nutzerdaten, Daten, die zum Sensor geschickt werden sowie die gesammelten Sensordaten werden in einer Datenbank gespeichert.



6 Aufbau des Webfrontends

Für den grundlegenden Aufbau des Webfrontends wurde sich am MVC-Muster (Model View Controller) orientiert.



Hierbei kann man auch sagen, dass es bereits viele vorgefertigte Lösungen gibt, welche man einfach verwenden kann, jedoch sind diese meistens zu komplex bzw. schwierig zu verstehen und zu erweitern. Desweiteren ist man mit diesen Lösungen zu eingeschränkt und kann nicht genau nach seinen eigenen Vorstellungen arbeiten. Aus diesen Gründen wurde ein eigenes „Framework“ erstellt, welches sich am MVC orientiert.

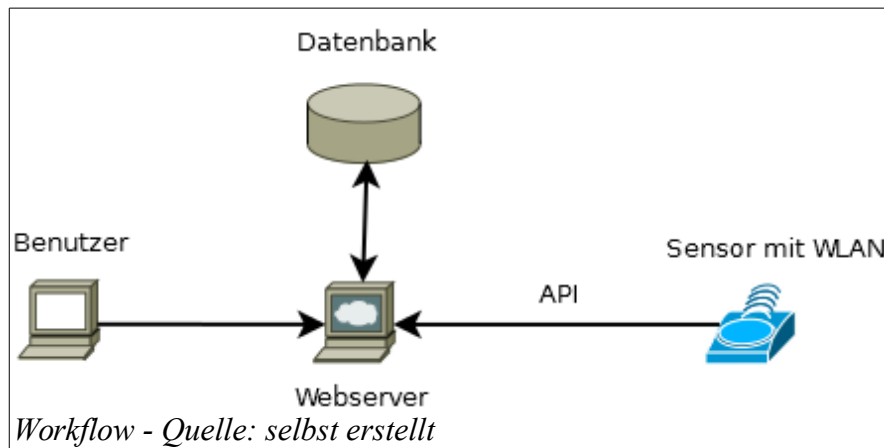
6.1 Eigenes Framework

In der Praxis bedeutet dies, dass das Arbeiten vereinfacht wird. Durch eine Grundstruktur, deren Aufwand der Erstellung hoch ist, lässt sich die Webseite schnell erweitern. Alle Anfragen, die an den Webserver kommen, werden auf die `index.php` umgeleitet, welche die Anfrage auswertet. Über das Format `localhost/Controller/Action` werden so die einzelnen Funktionen aufgerufen. Um auf die Startseite zu gelangen, ist dies zum Beispiel `localhost/index/index`. (Aufruf bedeutet `IndexController` und `indexAction`). Model beinhaltet:

- Datenbankverbindungen und Funktionen (PDO zur Realisierung von OOP). Um dann die Daten darstellen zu können, muss sich hinter dem Controller im Ordner „views“ eine gleichnamige `.phtml`-Datei befinden, die dem Namen der Action entspricht, also im Falle der `indexAction` die `/views/index.phtml`



6.2 Funktionen der Webseite


6.2.1 Workflow




6.2.2 Login

- Login-Funktionalität mit Datenbanküberprüfung der eingegeben Daten
- Auswahl der Sprache (Mehrsprachigkeit)
- Link zum Registrieren

Einloggen  

 leonardfranke



Einloggen

Ich habe noch keinen Account! [Hier registrieren](#)

Login-Screen - Quelle: selbst erstellt

6.2.3 Registrieren

- Neuanlegen eines Benutzeraccounts mit verschiedenen Feldern
- RegEx Überprüfung hinter jedem Feld mit Erläuterung im Infofeld jeder Zeile (Frontend und Backend)
- Überprüfung, dass Benutzername nur einmalig ist (Frontend und Backend)

Neuen Account erstellen

Benutzername ⓘ Ihr Benutzername

Email ⓘ Ihre Email

Passwort ⓘ Ihr Passwort

Passwort bestätigen ⓘ Ihr Passwort bestätigen

Abbruch Registrieren

Neuer Account - Quelle: selbst erstellt

6.2.4 Loginseite

- geschützt durch Login, ohne dieses nicht erreichbar
- Menü, um verschiedene Funktionalitäten aufzurufen

Start

Sensoren

Benutzer Einstellungen

Ausloggen

Detail - Quelle: selbst erstellt

6.2.5 Homebildschirm

- kleine Begrüßung, Übersicht
- Funktionalität bzw. Bild fehlt noch Tabelle der Sensoren
- Übersicht über alle Sensoren mit zugehörigem Benutzer und zusätzlichen Infos
- Möglichkeit einen neuen Sensor zu erstellen
- Aktualisierung, Suche, Sortierung, Exportieren der Daten
- Kontextmenü über Rechtsklick auf Datensatz: Informationen: Diagramm mit eingetragenen Daten und die dazu gehörige API Bearbeiten: Bearbeiten des Sensors z.B. Ort ändern Löschen: Sensor und dazu gehörige Daten löschen

[+ Neuen Sensor anlegen](#)

SensorID ↕	Sensor Name ▼	Standort ↕	Benutzername ↕	Erstellt am ↕
1	leos_test	Badezimmer	leonardfranke	2016-10-21 12:26:39

Zeige 1 bis 1 von 1 Zeile

Hauptbildschirm - Quelle: selbst erstellt

6.2.6 Benutzereinstellungen

- Funktionalitäten bezüglich des Benutzers z.B. Passwortänderung
- Ausloggen mit Redirect auf Startseite

6.3 API

- API (Programmierschnittstelle) zum Speichern der Sensordaten in der Datenbank. Aufruf der entsprechenden ControllerAction mit den entsprechenden Parametern (ID des Sensors, Temperatur und Luftfeuchtigkeit)
- speichert die Daten in der sensor_datas (beinhaltet Sensordaten aller Sensoren) mit Zeitstempel

6.4 Design der Webseite

Für die Gestaltung der Webseite wurde das öffentlich zugängliche Bootstrap verwendet, welches von Twitter entwickelt wurde. Es ist vor allem darauf konzipiert, Webseiten einfach zu gestalten und ebenfalls optimiert für die Anwendung auf mobilen Endgeräten wie Tablets oder Smartphones. Im Allgemeinen wird es als CSS-Framework beschrieben und bietet Gestaltungsvorlagen für verschiedenste Elemente einer Webseite und lässt diese durch wenige Handgriffe bzw. Anpassungen ansehnlich aussehen. Es gibt für Bootstrap mehrere Erweiterungen, so genannte Plugins, die JavaScript beinhalten und speziellere Funktionen bieten, wie die Erstellung von Tabellen, welche dann einige Grundfunktionen bieten, wie interaktive Aktualisierung, Sortierung bzw. Suche, als auch den Export von Daten. Solche Plugin wurden ebenfalls für dieses Projekt verwendet.

7 Qualitätsziele

- Zuverlässigkeit: das System soll in der Lage sein Temperaturen innerhalb des Toleranzbereiches des eingesetzten Sensors (DHT22) von $\pm 0.5\text{ }^{\circ}\text{C}$ anzuzeigen und zu speichern
- Benutzerfreundlichkeit:
- Modularität: modularer Aufbau durch den Einsatz Docker, so dass die MariaDB Datenbank durch eine MySQL Datenbank mit geringfügigen Änderungen ausgetauscht werden kann.
- Preis/Leistung: Einsatz von Standardkomponenten mit geringen Energieverbrauch und geringen Kosten
- Portierbarkeit: durch den Einsatz von Standardbibliotheken soll Anwendung portierbar gehalten werden
- Robustheit: das System soll innerhalb von Gebäuden ohne Störung konstant funktionieren

8 Integration in die Monitoringsoftware "Icinga2"

Ein mögliches Anwendungsgebiet des Temperatursensors ist die Überwachung bspw. der Kühlung von Serverräumen. Dafür ist es sinnvoll, ihn in bestehende Monitoringsoftwarelösungen zu integrieren. Zu diesem Zweck wird ein Plugin für Icinga2, einer Weiterentwicklung von Nagios, entwickelt. Mithilfe des Addons Graphite werden die Daten innerhalb von Icinga2 visualisiert. Die Daten werden der bestehenden Datenbank entnommen.

8.1 Realisierung mit Bash-Script

Das Icinga2-Plugin wird mithilfe eines Bashscriptes realisiert, das vom Icinga2-Server in wohldefinierten Intervallen aufgerufen wird und mit einem von folgenden Exitcodes schließt:

Exit Code	Status
0	OK
1	WARNING
2	CRITICAL
3	UNKNOWN

Exit-Codes(1) - Quelle: selbst erstellt

Neben dem Exit-Code wird der Temperaturwert an Icinga2 übermittelt. Wenn die Temperatur über einen bestimmten Schwellwert steigt, wird ein „Warning“ bzw. ein „Critical“ ausgegeben. Dabei können die einzelnen Werte diskutiert und angepasst werden.

Temperatur	Status
< 29° C	OK
29-35° C	WARNING
>35° C	CRITICAL
kaputt	UNKNOWN

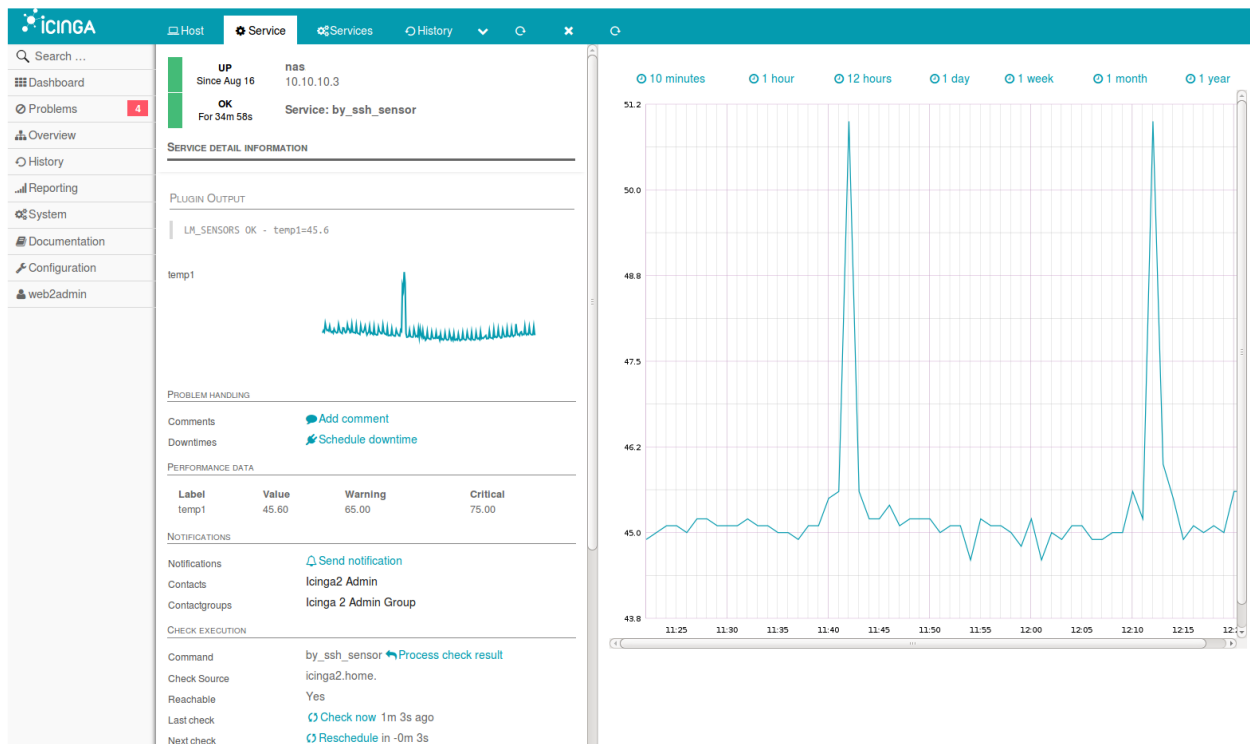
Exit-Codes(2) - Quelle: selbst erstellt

8.2 Einfache Installation und Erweiterbarkeit

Die Integration eines neuen Temperatursensors in Icinga2 soll so einfach wie möglich sein. Pro Sensor soll es nur ein einzelnes Script geben. D.h. wird ein weiterer Raum und/oder Sensor in die Überwachung aufgenommen, muss ein weitgehend geklontes Script in Icinga2 integriert werden. Dazu werden die Scripts der Übersichtlichkeit halber in ein separates Verzeichnis kopiert. Es wird vermieden, lange Scripts vorhalten zu müssen, deren Konfiguration und Erweiterung unübersichtlich ist. Es gilt der Grundsatz „One Service – One File“.

8.3 Visualisierung mit Graphite

Für die graphische Aufbereitung wird die Software „Graphite“ eingesetzt. Sie speichert die ermittelten Daten in einer weiteren Datenbank und gewährt so Einblicke in den zeitlichen Verlauf der Temperaturen in den Serverräumen. Graphite ist eine eigenständige Software, die dank vordefinierter Schnittstellen als Plugin in Icinga2 integriert werden kann. Zu beachten ist, dass die Maschine, auf der der Icinga2-Server mit Graphite läuft, mindestens 1 GB RAM vorweist. Graphite benötigt viel Hauptspeicher um seine Grafiken anzuzeigen.



Icinga2 mit Graphite - Quelle: <https://www.linuxfrickeln.de/visualisierung-von-performancedaten-in-der-icingaweb2-gui/>

9 Quellenverzeichnis

Bootstrap	http://v4-alpha.getbootstrap.com/getting-started/browsers-devices/
Dht22	http://www.mikrocontroller-elektronik.de/dht22-am2302-luftfeuchte-und-temperatursensor/
Docker	https://docs.docker.com/cs-engine/install/
Docker-Tutorial	https://docs.docker.com/engine/getstarted/
Esp8266	http://www.mikrocontroller.net/articles/ESP8266

10 Abbildungsverzeichnis

siehe Bildunterschriften