

# Sprint 1



IOT SE-Projekt Wintersemester 2024 Allgemeine Informatik (TIN22)

**Blickbox DHBW Heidenheim**

# Release Notes Sprint 1:

## What's Changed?

- React wurde als Frontend-Framework ausgewählt und aufgesetzt.
- Die React Anwendung wurde Containerisiert (Docker).
- Der Build der Frontend Applikation wurde automatisiert (build.sh)
- Das Dashboard des Frontends wurde geplant (Mockups und Komponentenplanung).
- Das Dashboard des Frontends wurde initialisiert.
- Im Dashboard wurden die ersten Kacheln erstellt (Verbindungsanzeige zu Container und Server, Status des Dockers, Platzhalter).
- Rust wurde als Sprache für das Programm auf dem Raspberry gewählt.
- Erstes Programm, das Sensordaten abfrägt und in File schreibt, wurde initialisiert.
- Firmware zum auslesen von Temperatur und Luftfeuchtigkeitsdaten für den Arduino

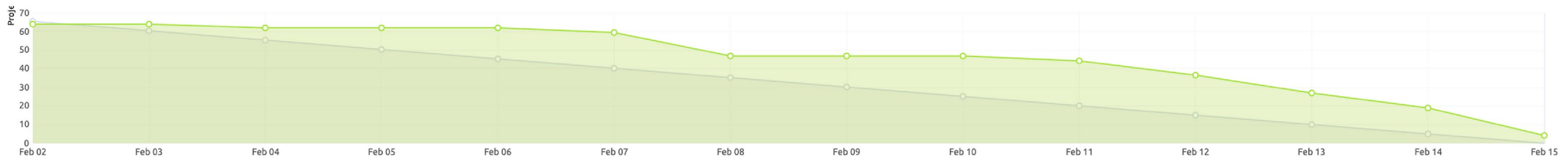
# Release Notes Sprint 1:

## What's Changed?

- Planung welche Software auf dem Server läuft (Schicht 2 C4)
- Docker-Container Konfiguriert
- Erste Routen der API wurden geschrieben
- Influx Datenbank wurde aufgesetzt
- Grafana Server wurde aufgesetzt
- Reverse-Proxy wurde aufgesetzt und konfiguriert
- Swagger Dokumentation wurde geschrieben

# 65 Points

② How this chart works



## Definition of done (DoD)

---

Im Team wurden die folgenden Kriterien definiert, damit eine Aufgabe oder ein Produkt als abgeschlossen gilt:

1. **Code-Review:** Der Code wurde von mindestens einem Teammitglied überprüft.
2. **Unit-Tests:** Es wurden ausreichende Unit-Tests geschrieben, und alle Tests sind erfolgreich durchgelaufen.
3. **Integrationstests:** Der Code wurde erfolgreich in den Hauptentwicklungs Zweig integriert, und alle Integrationstests sind bestanden.
4. **Dokumentation:** Alle relevanten Code-Änderungen wurden in der Projektdokumentation aktualisiert. Ergo sollten spezielle Diagramme erstellt worden sein, müssen diese auch von dem jeweiligen Entwickelnden im Arc42 erläutert und implementiert werden.
5. **Benutzerdokumentation:** Falls erforderlich, wurde die Benutzerdokumentation aktualisiert.
6. **Code-Stil:** Der Code entspricht den vereinbarten Code-Standards und Best Practices.
7. **Performance-Überprüfung:** Die Performance des Codes wurde überprüft und erfüllt die definierten Anforderungen.
8. **Sicherheitsprüfung:** Falls relevant, wurden Sicherheitsprüfungen durchgeführt und alle Sicherheitsanforderungen sind erfüllt.
9. **Akzeptanzkriterien:** Alle in den Akzeptanzkriterien definierten Anforderungen sind erfüllt.
10. **Review mit dem Product Owner:** Der Product Owner hat das Ergebnis überprüft und akzeptiert.

Die DoD wird regelmäßig überprüft und bei Bedarf aktualisiert, um sicherzustellen, dass sie den aktuellen Anforderungen entspricht.

# Definitionen (1/3)

# Git Workflow

---

## 1. Sprint-basierte Entwicklung:

- Für jeden Sprint wird eine eigene Sprint-Branch erstellt.
- Am Ende jedes Sprints wird der Sprint-Branch in den Master-Branch gemerged.

## 2. Story-Banches:

- Jede Story erhält einen eigenen Branch, abgeleitet vom Sprint-Branch.
- Die Benennung erfolgt nach dem Schema: `#id-name`.

## 3. Commits:

- Sprache: Deutsch
- Innerhalb der Story-Banches werden für jeden Task oder mehrere Tasks separate Commits durchgeführt.
- Die Commit-Nachricht enthält die IDs der zugehörigen Tasks und den Namen der Änderung im Format:  
`#id_#id2_#id3-name`.

## 4. Abschluss einer Story:

- Nach Abschluss einer Story wird der entsprechende Branch sowohl lokal als auch remote vom Entwickler gelöscht.

## 5. Merge und Approvals:

- Vor dem Merge einer Story in den Sprint-Branch muss mindestens ein Approver die Änderungen genehmigen.

# Definitionen

(2/3)

## Rollen

---

### Dokumentation

- Release Notes (Aron, Max)
- Diagramme allg. / UML (May)
- arc42 (LaTeX) pflegen (Vivi, Aron)

### PO/Scrum (Max)

- Moderation Freitagsmeeting
- Showcase der Demo
- Backlog pflegen
- Sprinttermine legen

### Techlead Hardware (May, Vivi)

### Techlead Client

- Frontend (Max)
- Backend (Aron)

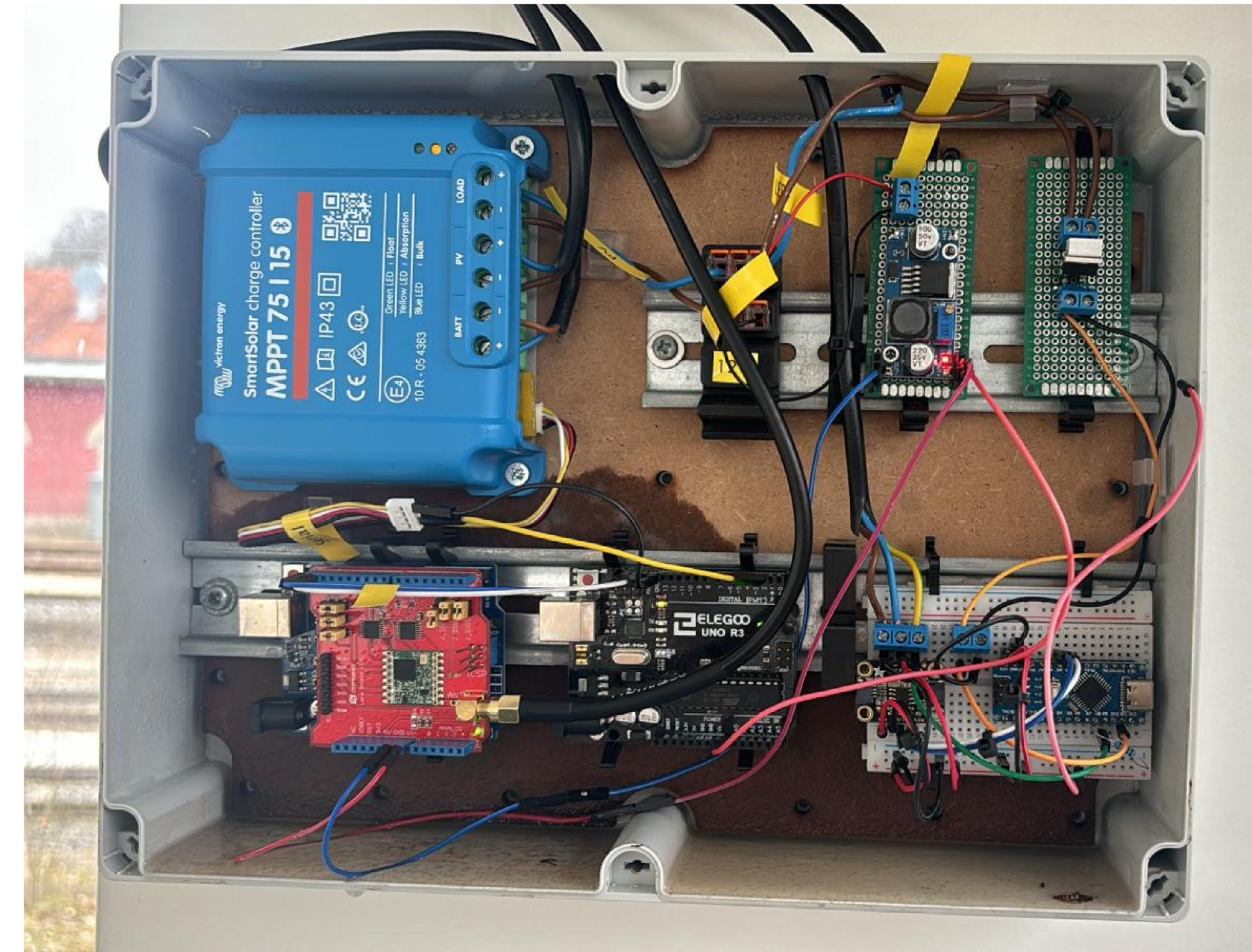
### Techlead Infrastruktur

- Server (Webserver für Client) (Max)
- WLAN
- Sensordaten (May)

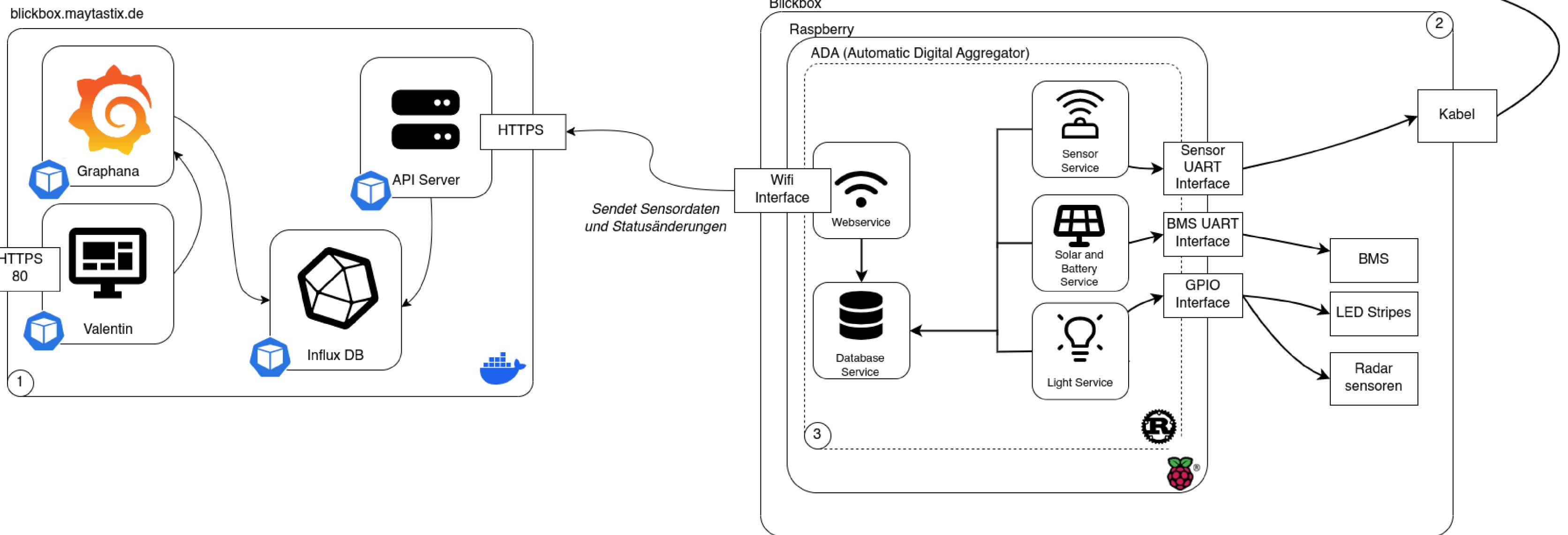
# Definitionen

(3/3)

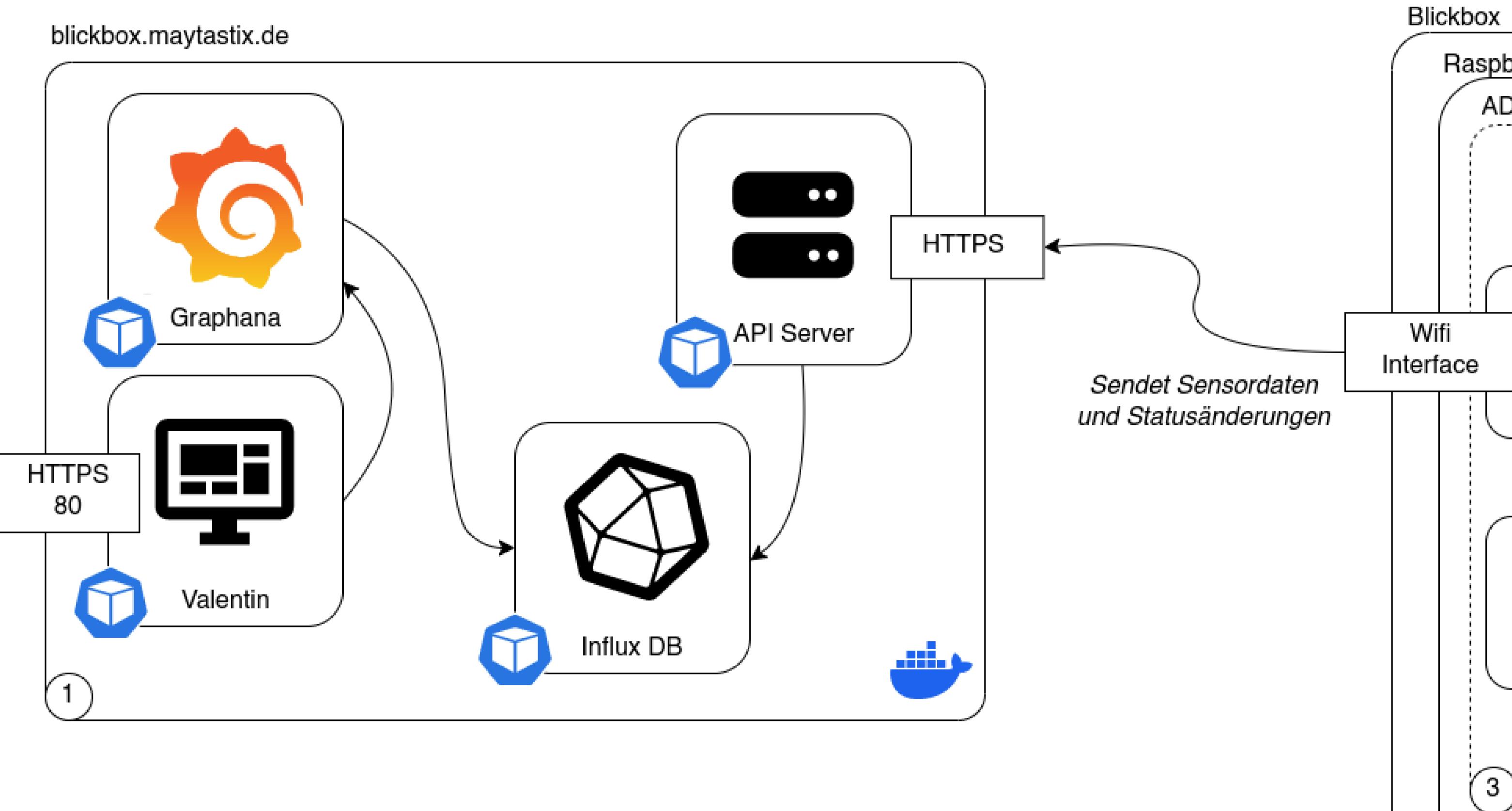
# Ist-Analyse



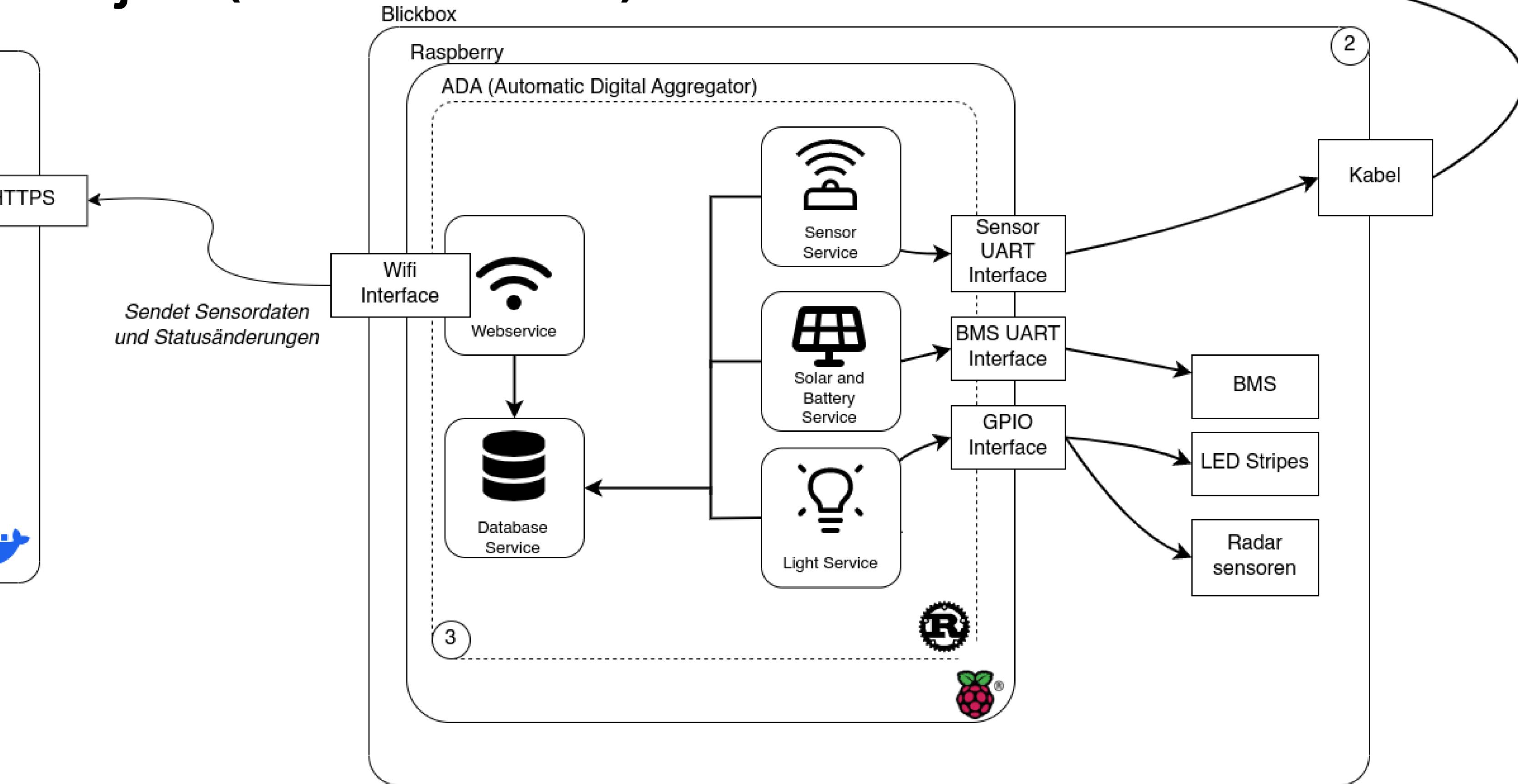
# Übersicht über das Projekt



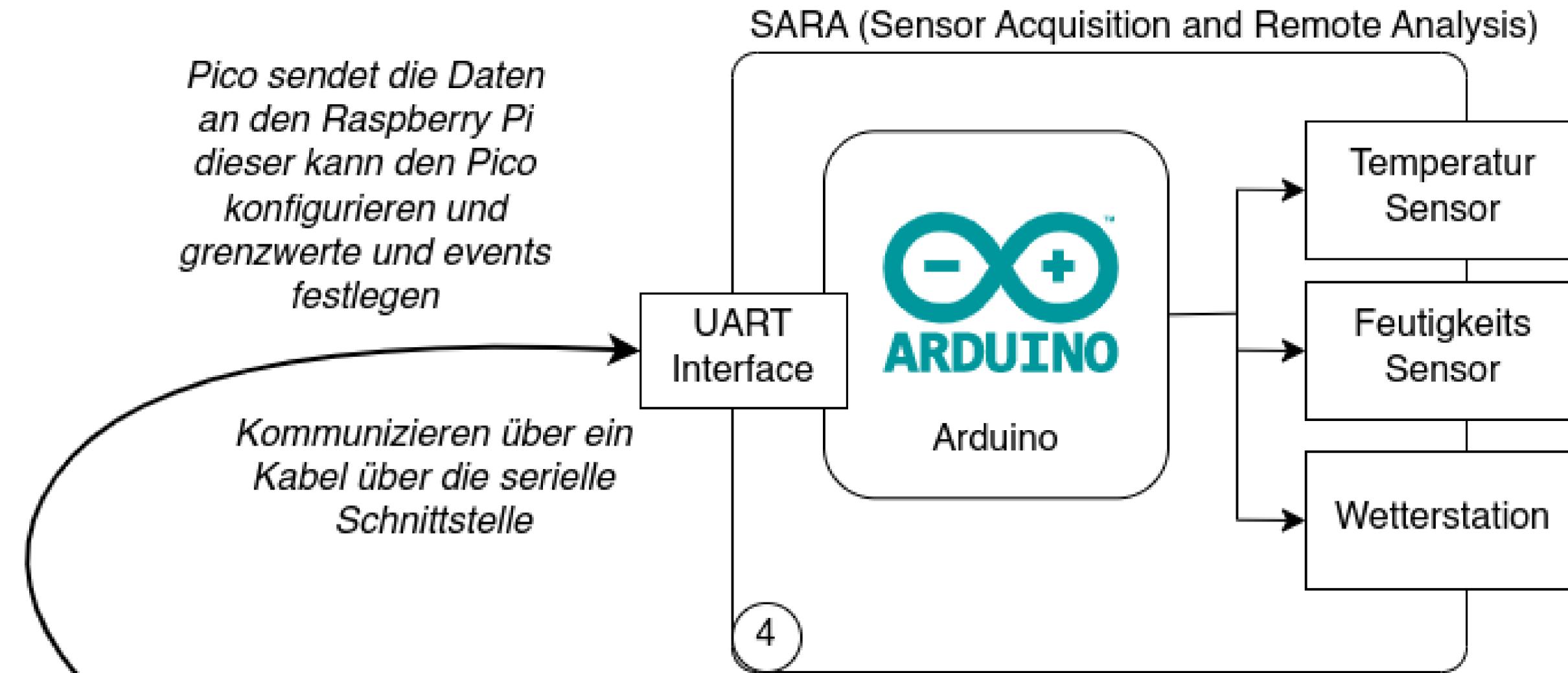
# Übersicht über das Projekt (Detail Cloud)

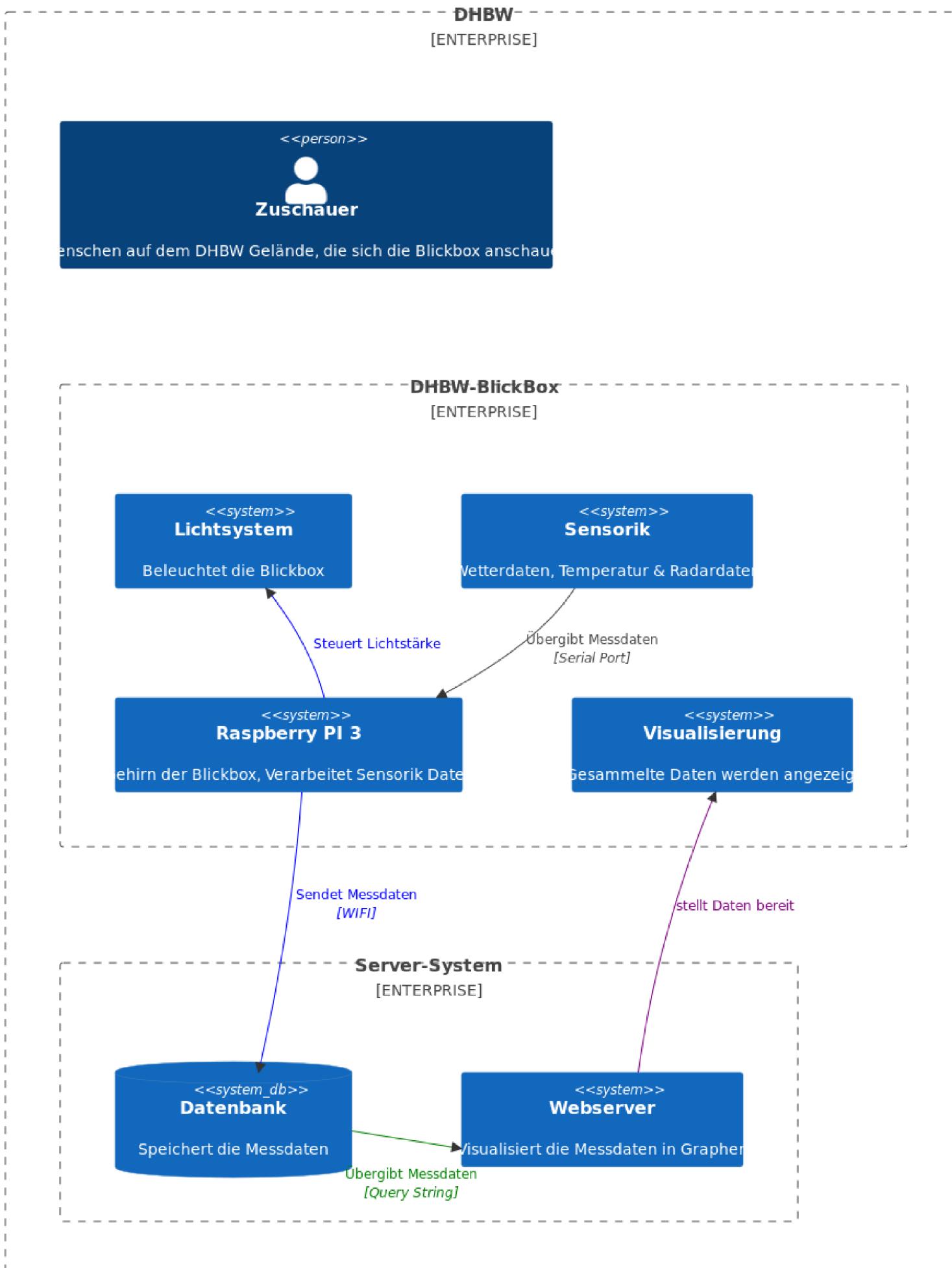


# Übersicht über das Projekt (Detail Blickbox)



# Übersicht über das Projekt (Detail Blickbox)





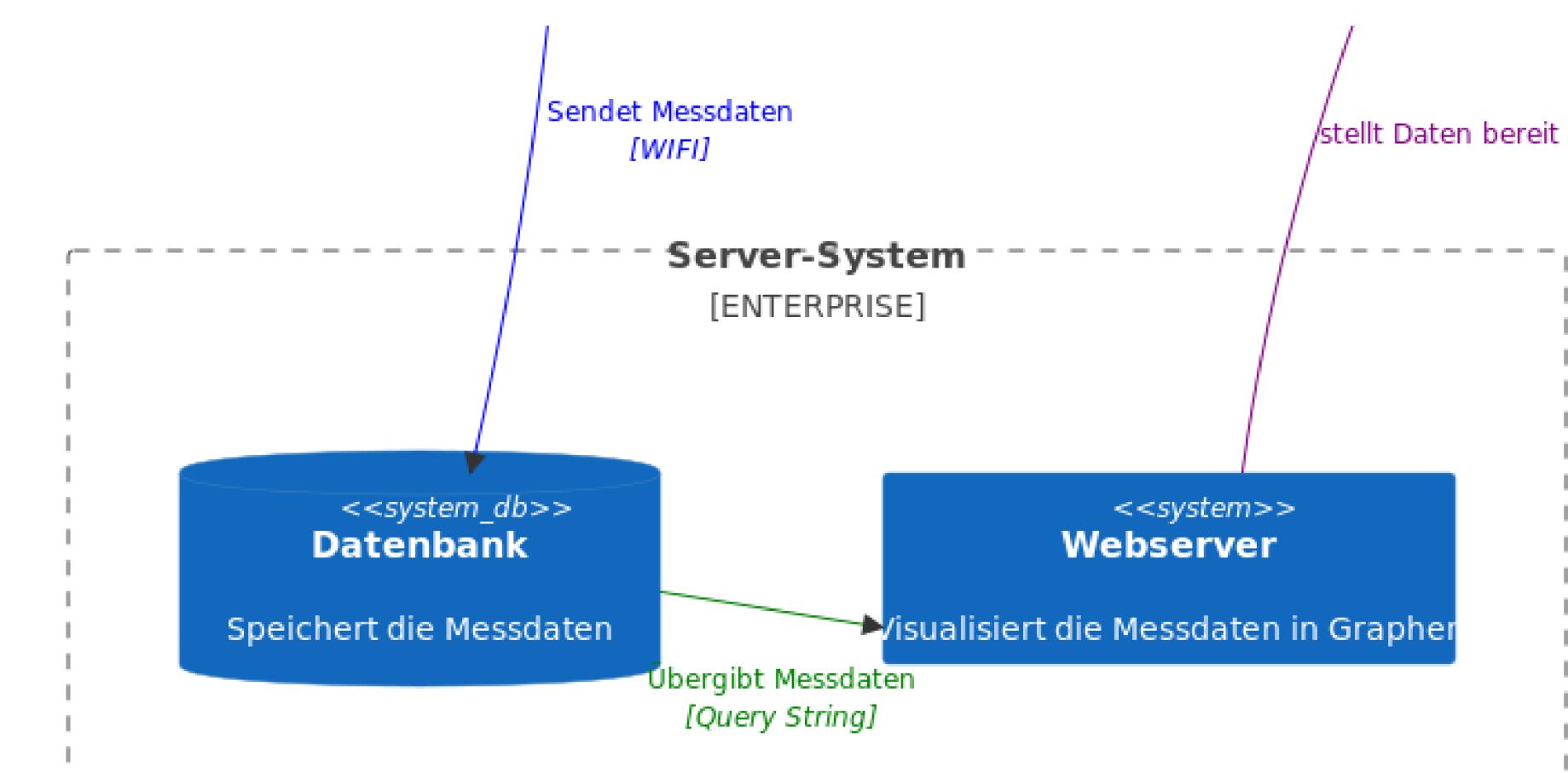
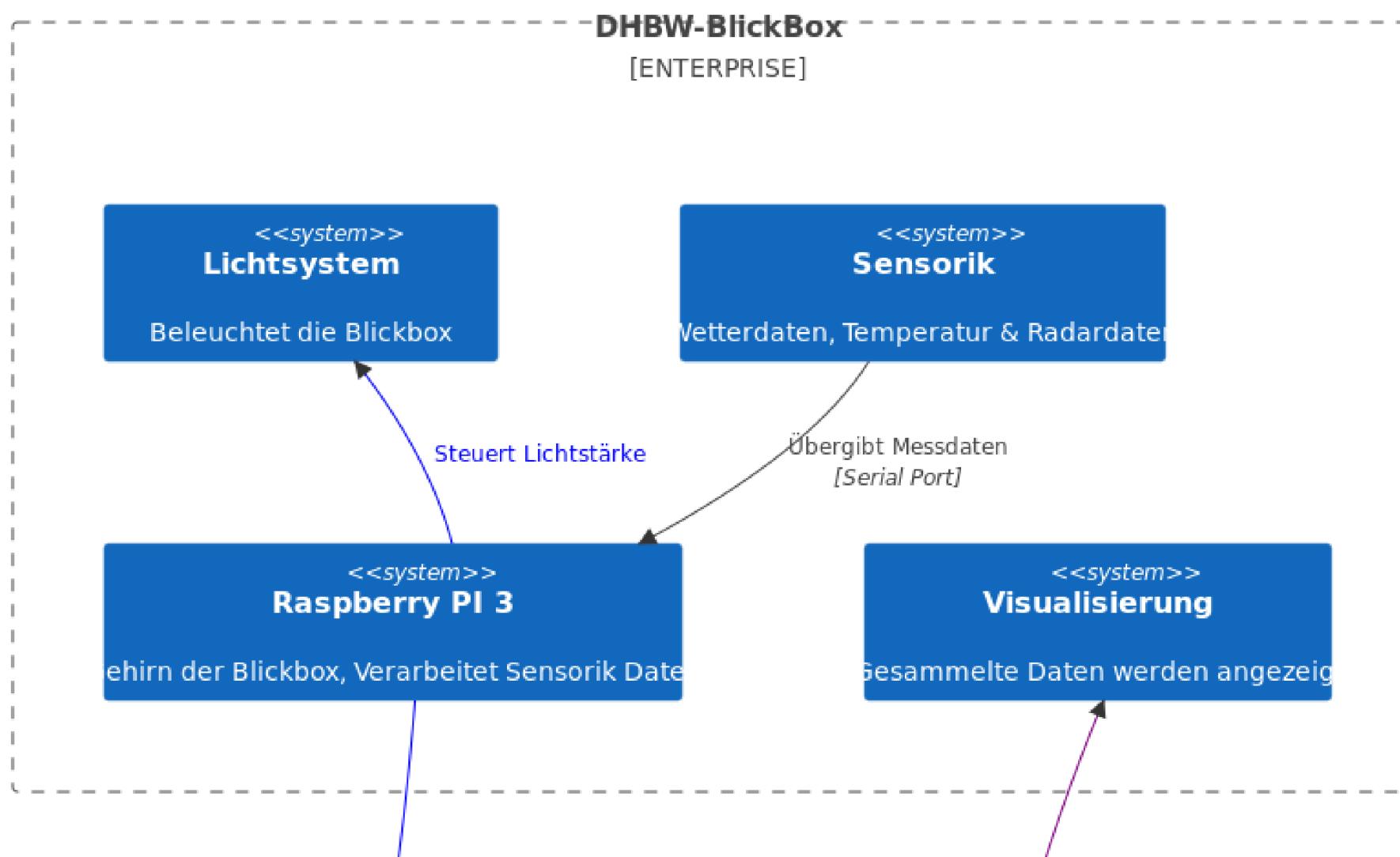
# System Context Diagram

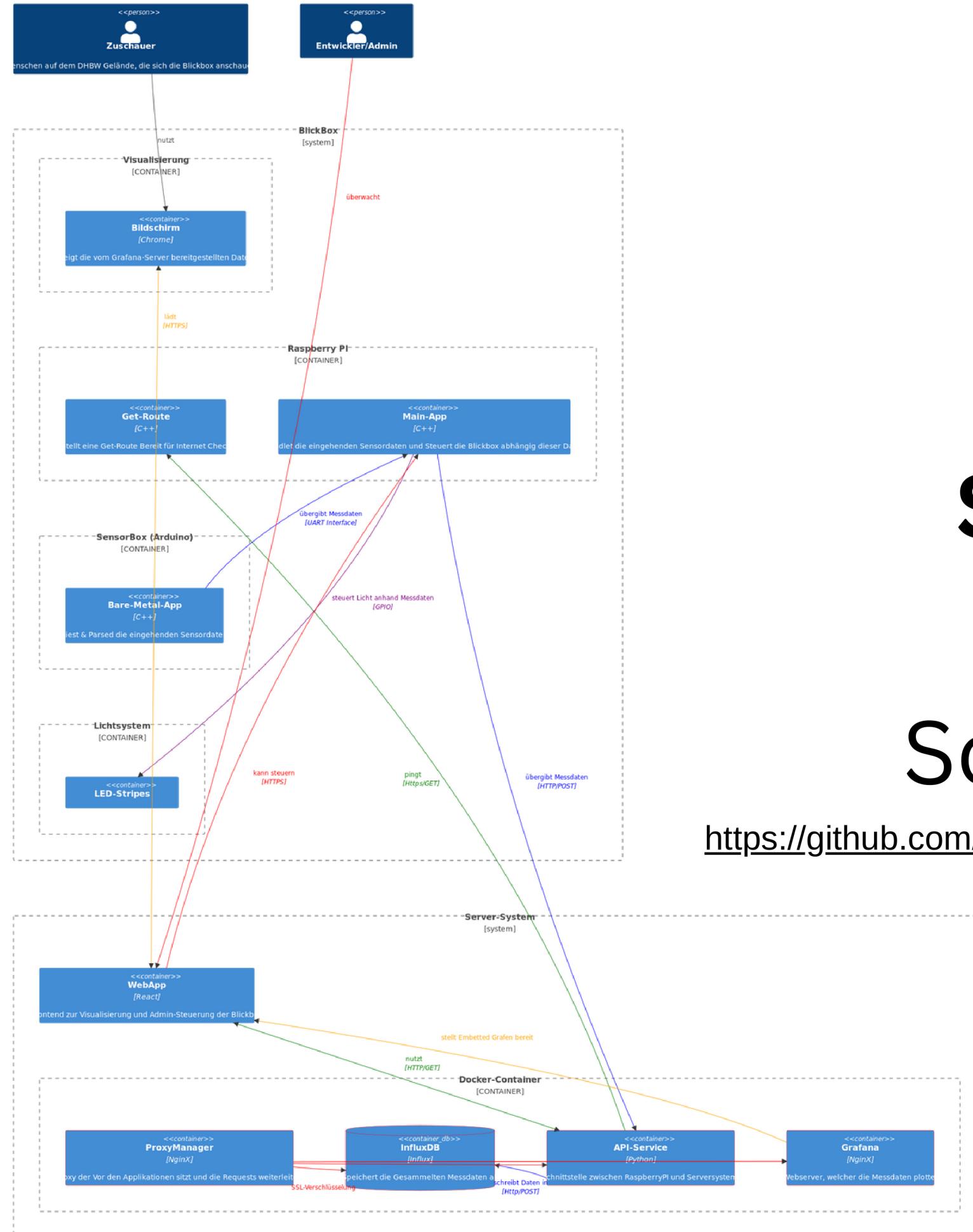
## Schicht 1 - C4 Modell

<https://github.com/mxmueller/DHBW-Blickbox/blob/sp1/docs/c4/1.%20C4%20Context.png>

# Detail

## Schicht 1 - C4 Modell





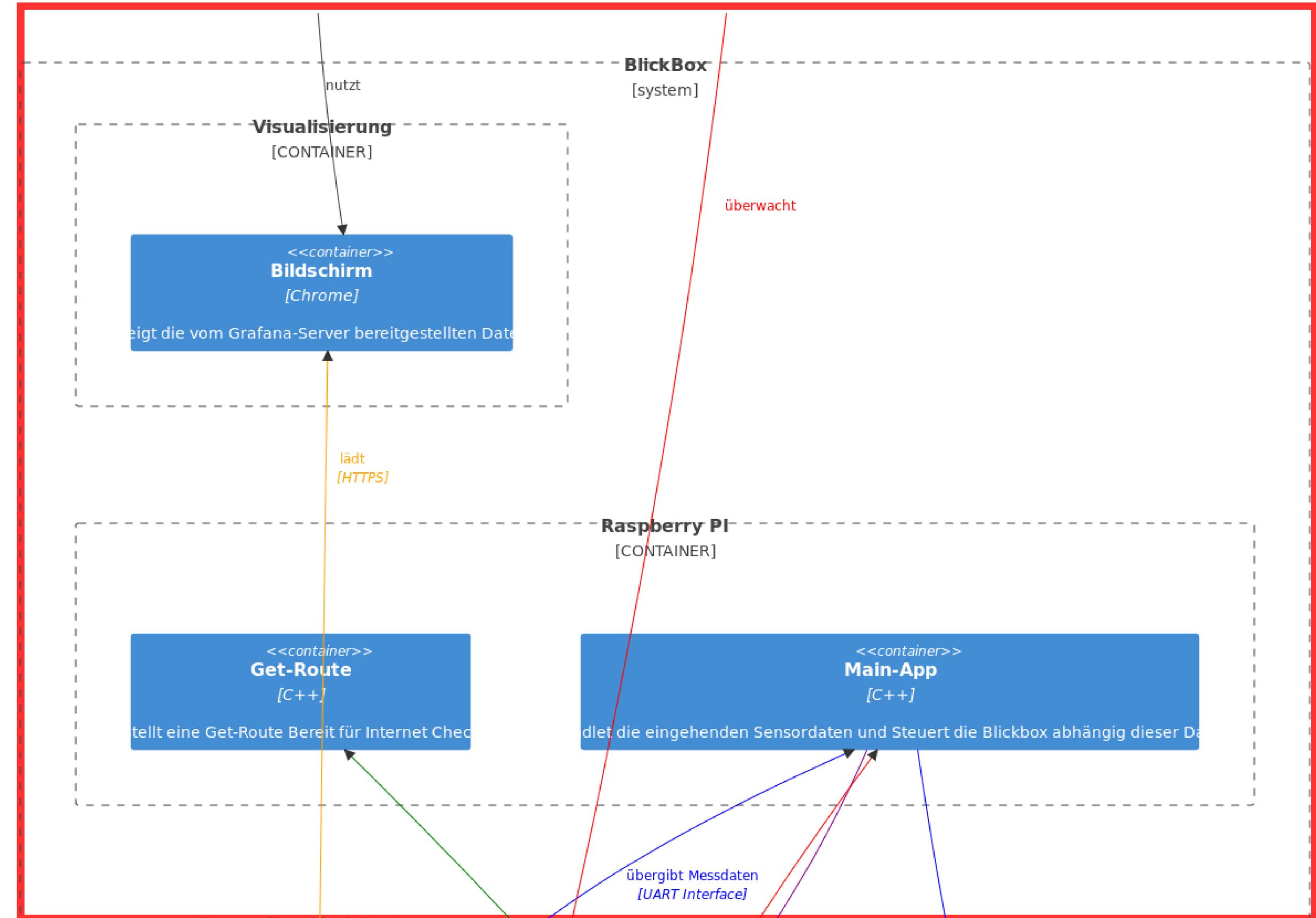
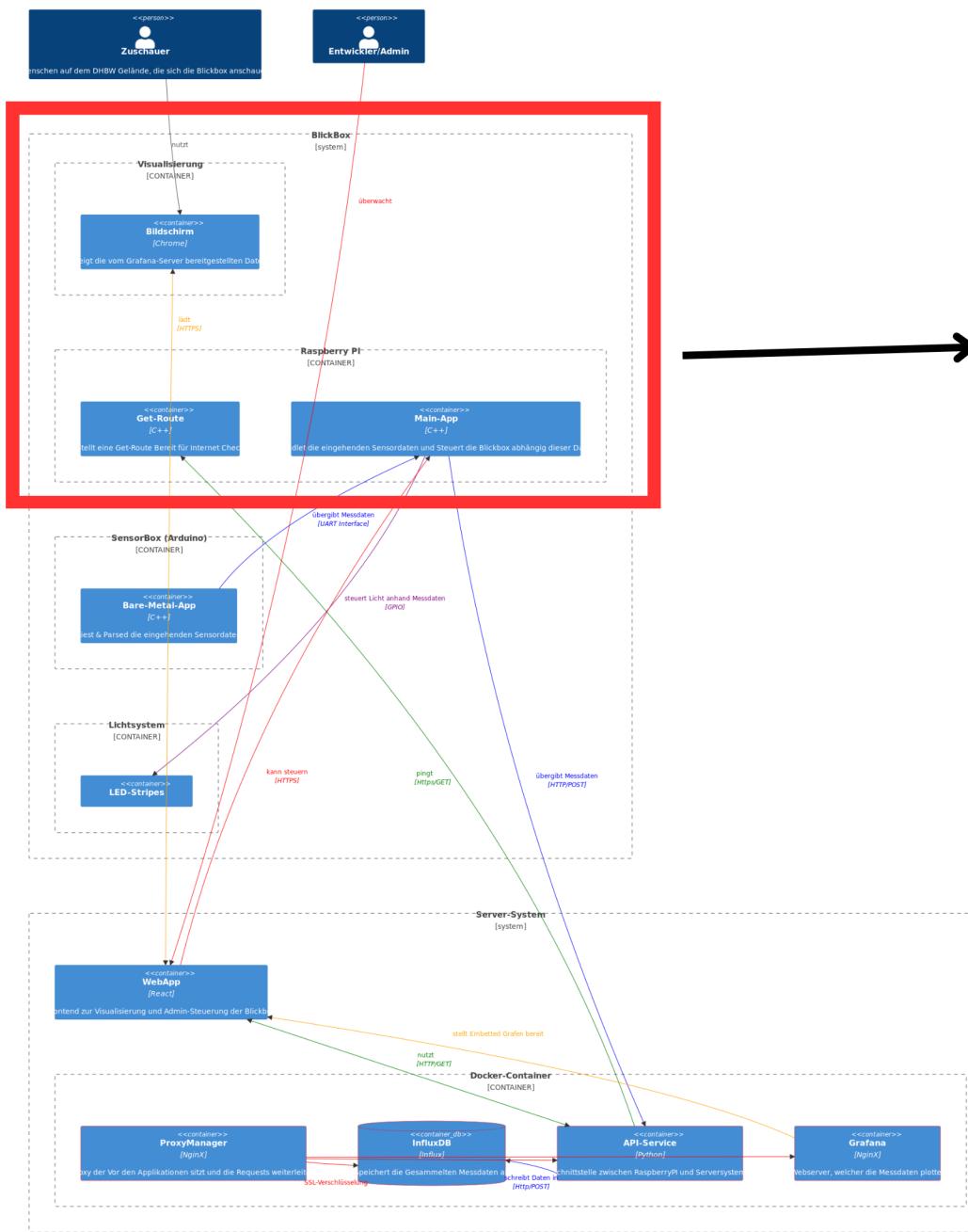
# System Container Diagram

## Schicht 2 - C4 Modell

<https://github.com/mxmueller/DHBW-Blickbox/blob/sp1/docs/c4/2.%20C4Container.png>

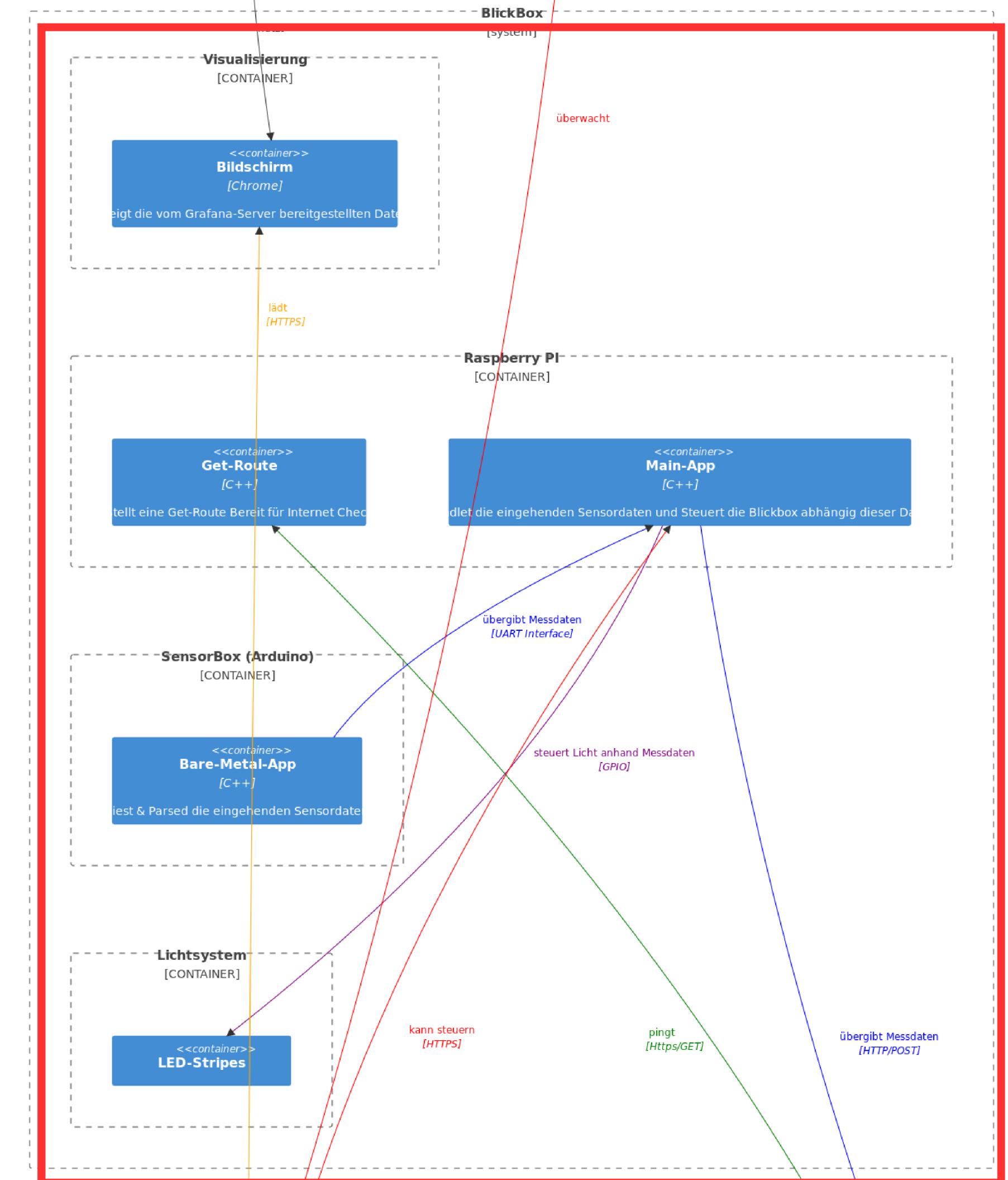
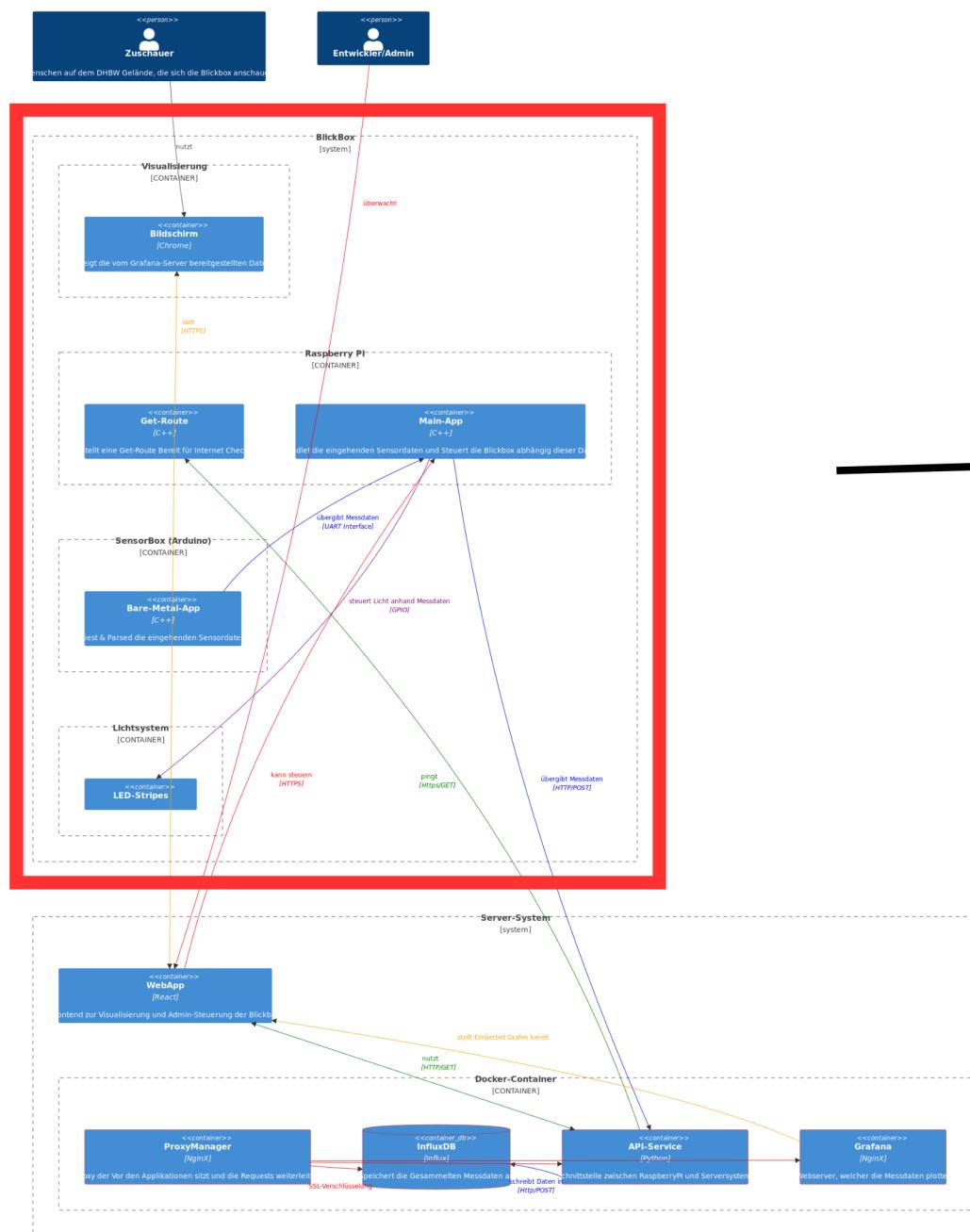
# Detail (1/3)

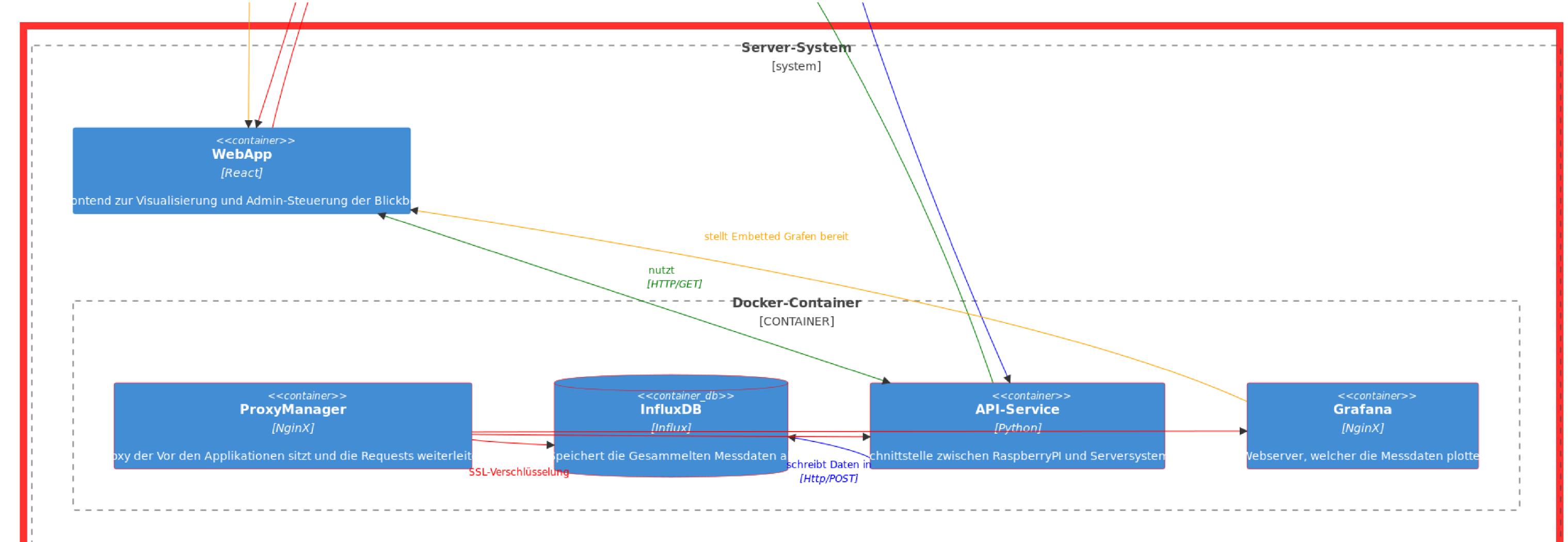
## Schicht 2 - C4 Modell



# Detail (2/3)

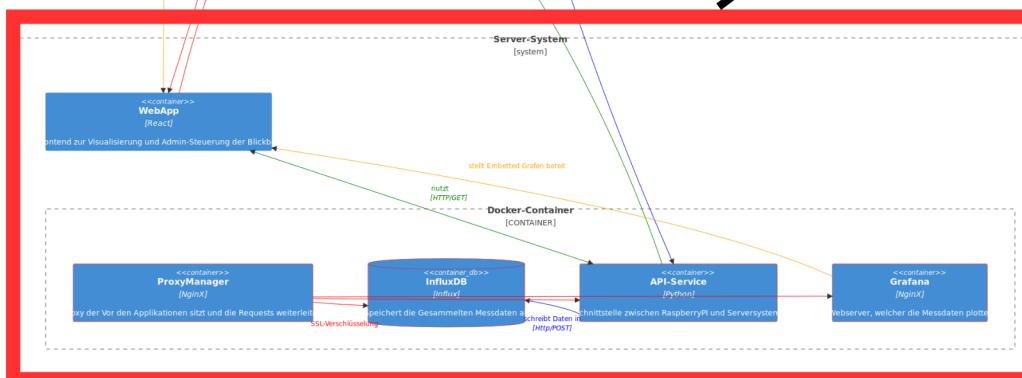
## Schicht 2 - C4 Modell





## Detail (3/3)

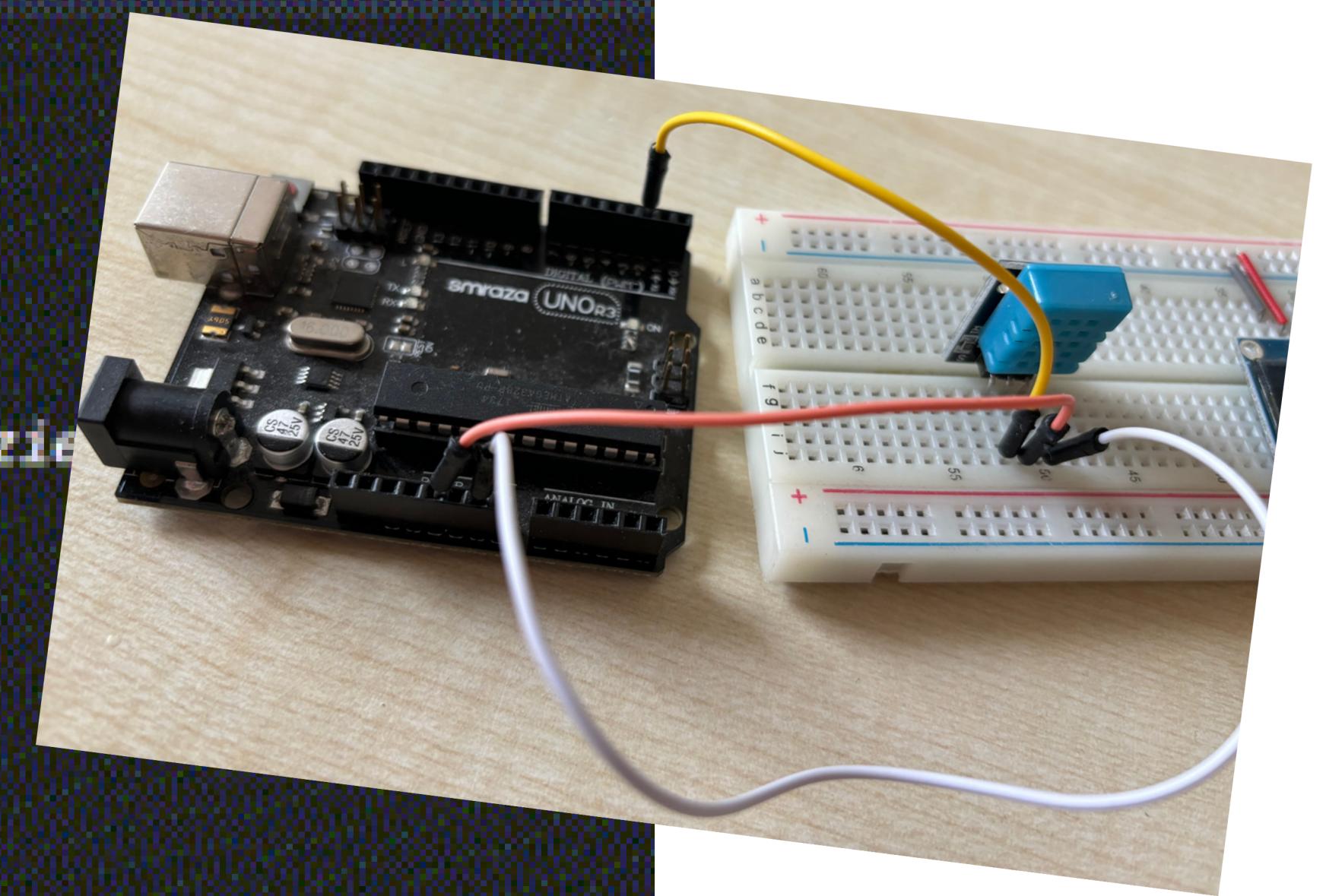
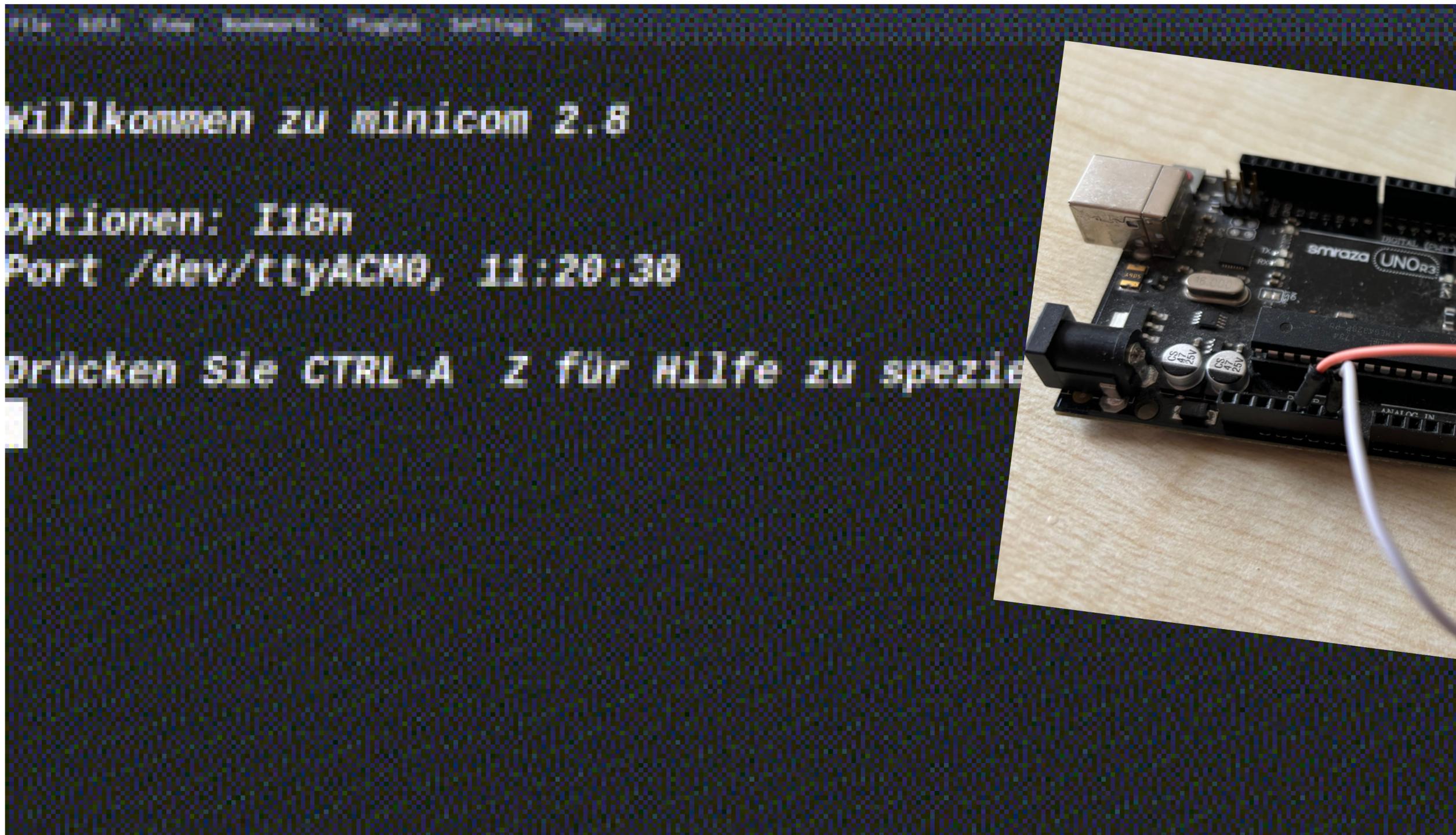
### Schicht 2 - C4 Modell



# **Disclaimer**

Innerhalb der Hardware bzw. Sensorik musste aufgrund der Gegebenheiten zur **Evaluation** in **Sprint 1** zunächst ein **Prototyping Approach** gewählt werden. Die genaue **Planung** und **Modellierung** kann aufgrund der gewonnenen Erkenntnisse in **Sprint 2** erfolgen.

# Sara und serielle Kommunikation



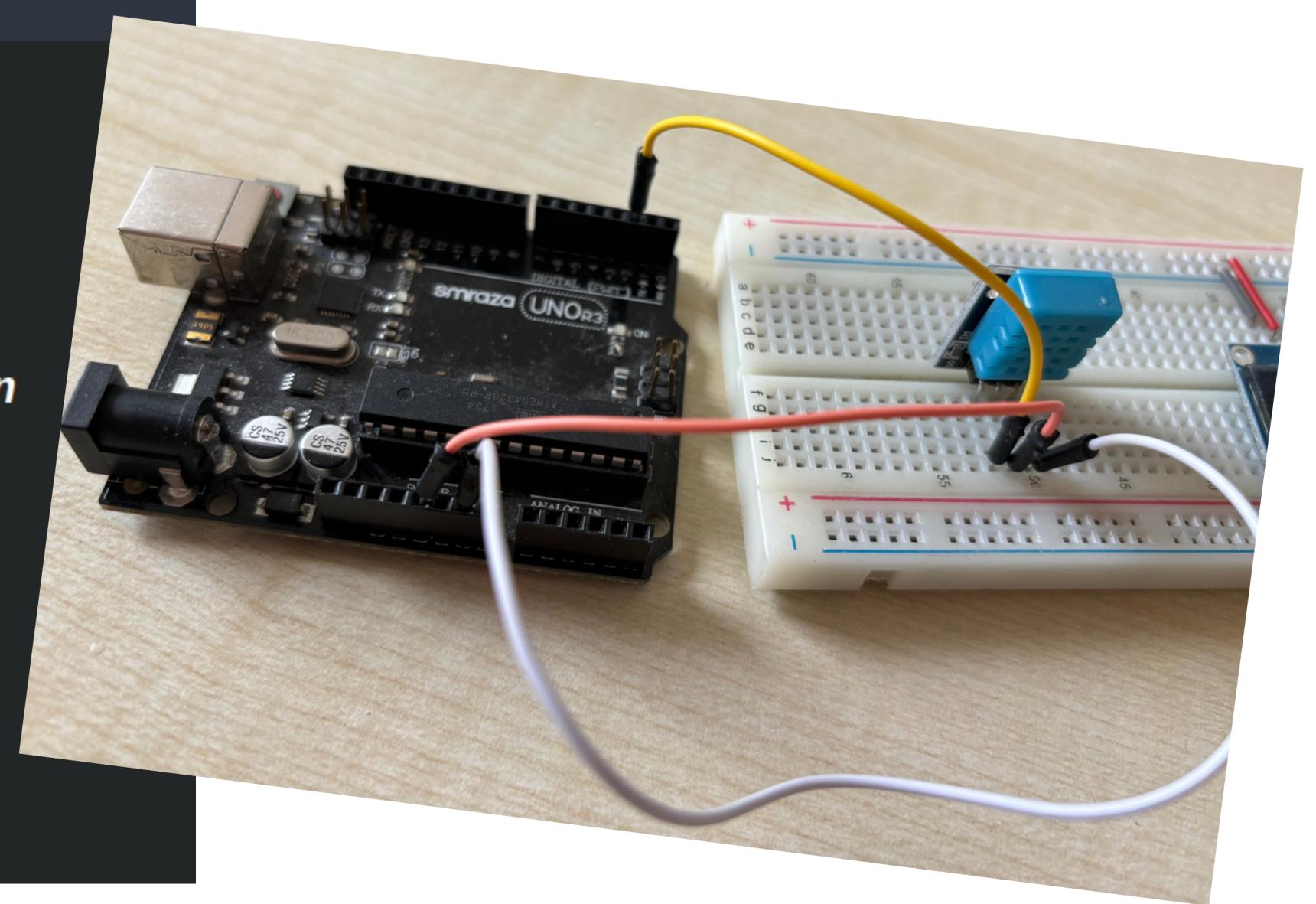
# Sara und serielle Kommunikation

```
File Edit View Bookmarks Plugins Settings Help

Willkommen zu minicom 2.8

Optionen: I18n
Port /dev/ttyACM0, 11:20:30

Drücken Sie CTRL-A Z für Hilfe zu speziellen Tasten
Ello Sara here :D
I am initialized!
Commands: t - temp, h - humidity
t
t:537.60
h
h:1100.80
[]
```



# Eventgesteuert



```
1  /**
2  * Sobald Daten über in den Buffer eintreffen wird ein Serial Event ausgelöst
3  * welches die Daten verarbeitet
4  */
5 void serialEvent() {
6     on_serial_message_recieve ();
7 }
```

# Eventgesteuert

```
1 void on_serial_message_recieve (){
2     while (Serial.available()) {
3         char inChar = (char)Serial.read();
4
5         if (inChar == '\n' || inChar == '\r') {
6
7             stringCompleter = true;
8             currentPatter = COMMAND;
9             return;
10        }
11
12        inputString += inChar;
13
14        switch (currentPatter)
15        {
16            case COMMAND:
17                command += inChar;
18                if(inChar == ' '){
19                    currentPatter = ARGUMENT;
20                }
21                break;
22        }
23    }
24 }
```

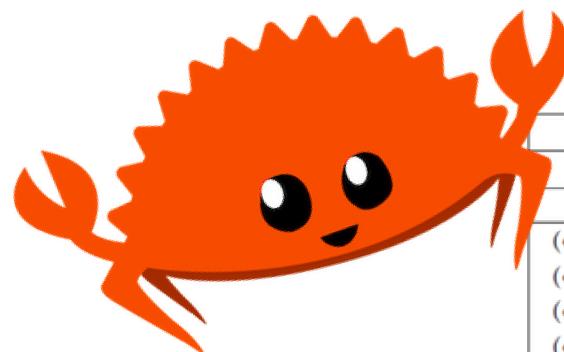
# Eventgesteuert



```
1  /**
2   * Verarbeitet die Kommandos, die über das
3   * Serielle Protokoll übertragen wurden
4   */
5  void handle_serial_reques (){
6      if (!stringComplet ) {
7          if (command == F("t")){
8              print_temperatur ();
9          }else if (command == F("h")){
10             print_humidit ();
11         }
12         stringComplet  = false;
13         command = "";
14         inputString = "";
15     }
16 }
```

# ADA - Automatic Digital Aggregator

Über Programm, das auf Raspi laufen soll:  
Daten anfragen und in File speichern



Total			
	Energy	Time	Mb
(c) C	1.00	(c) C	1.00
(c) Rust	1.03	(c) Rust	1.04
(c) C++	1.34	(c) C++	1.56
(c) Ada	1.70	(c) Ada	1.85
(v) Java	1.98	(v) Java	1.89
(c) Pascal	2.14	(c) Chapel	2.14
(c) Chapel	2.18	(c) Go	2.83
(v) Lisp	2.27	(c) Pascal	3.02
(c) Ocaml	2.40	(c) Ocaml	3.09
(c) Fortran	2.52	(v) C#	3.14
(c) Swift	2.79	(v) Lisp	3.40
(c) Haskell	3.10	(c) Haskell	3.55
(v) C#	3.14	(c) Swift	4.20
(c) Go	3.23	(c) Fortran	4.20
(i) Dart	3.83	(v) F#	6.30
(v) F#	4.13	(i) JavaScript	6.52
(i) JavaScript	4.45	(i) Dart	6.67
(v) Racket	7.91	(v) Racket	11.27
(i) TypeScript	21.50	(i) Hack	26.99
(i) Hack	24.02	(i) PHP	27.64
(i) PHP	29.30	(v) Erlang	36.71
(v) Erlang	42.23	(i) Jruby	43.44
(i) Lua	45.98	(i) TypeScript	46.20
(i) Jruby	46.54	(i) Ruby	59.34
(i) Ruby	69.91	(i) Perl	65.79
(i) Python	75.88	(i) Python	71.90
(i) Perl	79.58	(i) Lua	82.91

```
[package]
name = "ada"
version = "0.1.0"
edition = "2021"

# See more keys and their definitions at https://

[dependencies]
serialport = "4.3.0" 4.3.0
clap = "4.5.0" 4.5.0
chrono = "0.4.34" 0.4.34
```

```

1 use std::io;
2 use std::fs::File;
3
4 use std::io::{Read, Write};
5 use std::time::{Duration, SystemTime, UNIX_EPOCH};
6 use chrono::{DateTime, Utc};
7
8 fn main() {
9     // Opens file in append mode (and creating it if it doesn't exist)
10    let mut file :File = OpenOptions::new()
11        .write(true) :&mut OpenOptions
12        .append(true) :&mut OpenOptions
13        .create(true) :&mut OpenOptions
14        .open(path: "command_history.txt").unwrap();
15    let ports :Vec<SerialPortInfo> = serialport::available_ports().expect(msg: "No ports found!");
16    for p:SerialPortInfo in ports {
17        println!("{}: {}", p.port_name);
18    }
19
20    let mut port :Box<dyn SerialPort> = serialport::new(path: "/dev/ttyACM0", baud_rate: 115200)
21        .timeout(Duration::from_millis(millis: 1000)) :SerialPortBuilder
22        .open().expect(msg: "Failed to open port");
23
24    thread::sleep(Duration::from_millis(millis: 5000));
25
26    let mut serial_buf: Vec<u8> = vec![0; 100];
27

```

- File handling
- Port zu Sara-Programm
- Buffer für Daten

## Loop:

- angekommen Daten zu checken
- in File schreiben

```

27
28    // here data is requested
29    let temperature_command :&str = "t\n";
30    port.write_all(temperature_command.as_ref()).expect(msg: "Failed to send command");
31
32    loop {
33        match port.read(serial_buf.as_mut_slice()) {
34            Ok(t:usize) => {
35                let received_data :&[u8] = &serial_buf[.. < t];
36                let interpreted_data :String = interpret_data(received_data);
37
38                // writing the received data to file system
39                // data to be written should be a struct sooner or later... :/
40                if interpreted_data != "" {
41                    let date_time_format: DateTime<Utc> = SystemTime::now().into();
42                    let time :String = date_time_format.format(fmt: "%Y-%m-%d %H:%M:%S").to_string();
43                    let data :String = format!("{}: {}", time, interpreted_data);
44                    file.write_all(data.as_bytes()).unwrap();
45
46                }
47
48                thread::sleep(Duration::from_millis(millis: 1000));
49                continue;
50            }
51            Err(ref e:&Error) if e.kind() == io::ErrorKind::TimedOut => {
52                println!("Timeout occurred");
53                continue;
54            }
55            Err(e:Error) => {
56                eprintln!("Failed to read from port: {}", e);
57                break;
58            }
59        }
60    }

```

```

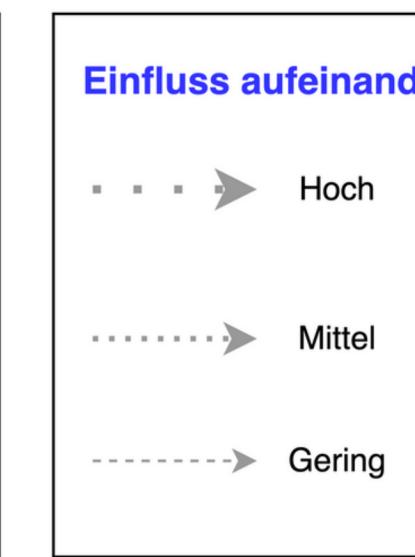
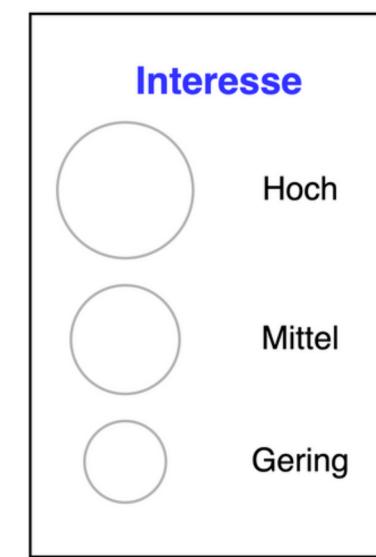
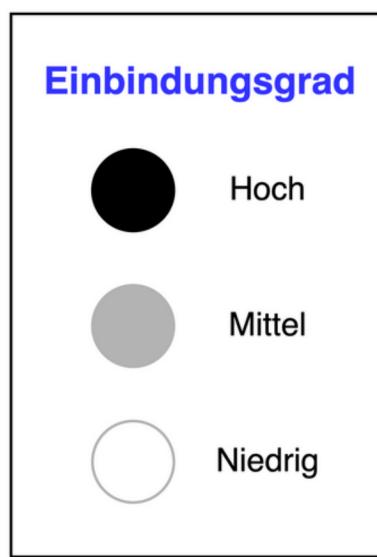
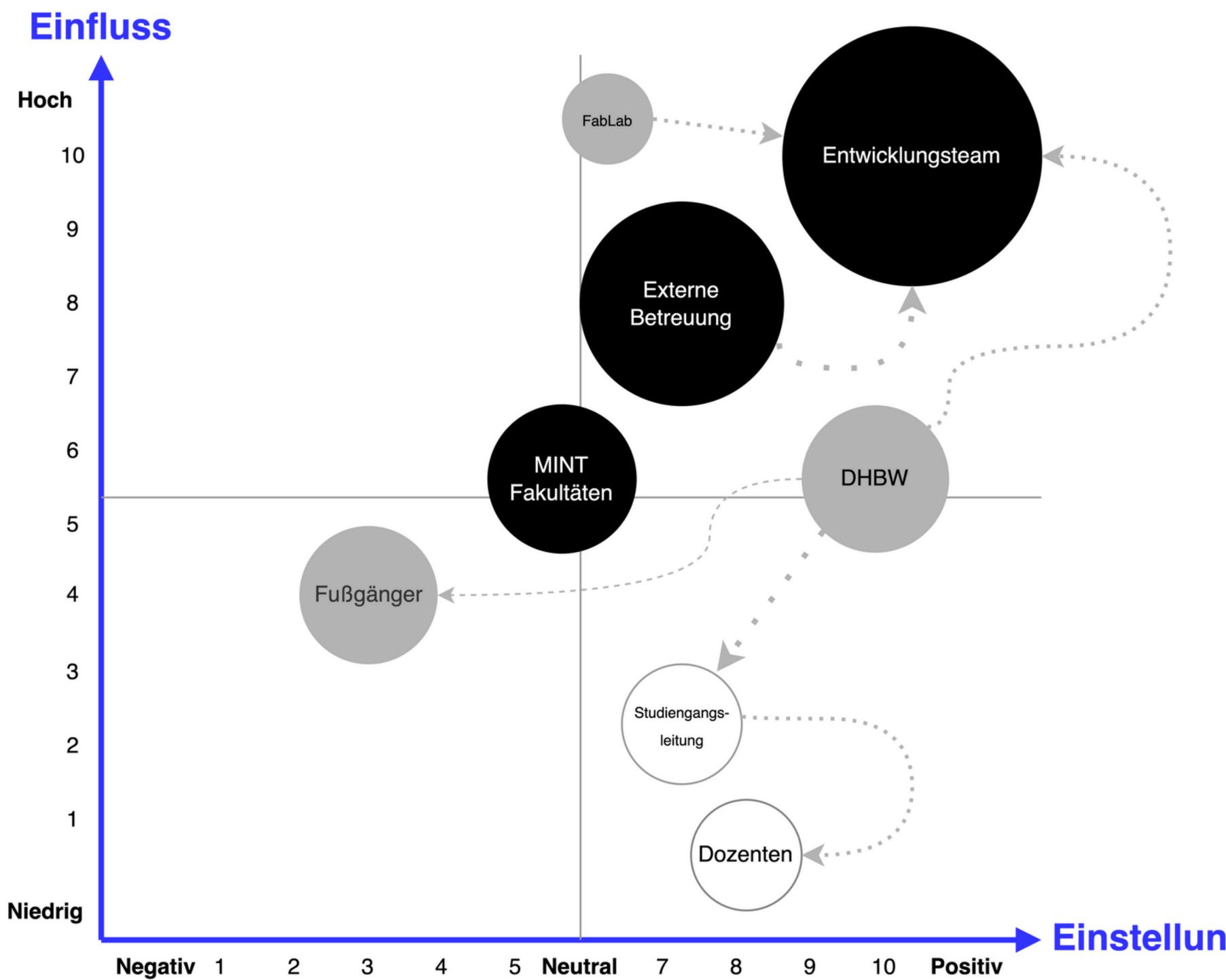
62 pub fn interpret_data(data: &[u8]) -> String {
63     /// no error handling
64     // here i am interpreting the received string before storing it into the buffer :)
65     // : als Separator zwischen Key und Value der Werte und String wird mit \n beendet
66     // println!(" data: {:?}", data);
67     let mut string: String = String::from_utf8_lossy(data).to_string();
68
69     println!("received data to be interpreted: {:?}", string);
70
71     return match string.as_str() {
72         "Hello Sara here :D\r\nI am initialized!\r\nCommands: t - temp, h - humidity\r\n" => {
73             String::new()
74         }
75         _ => {
76             // Remove trailing \r\n or \n characters
77             if string.ends_with(pat: "\n") {
78                 string.pop();
79             }
80             if string.ends_with(pat: "\r") {
81                 string.pop();
82             }
83
84             let separated_string_by_colon: Vec<&str> = string.split(pat: ':').collect::<Vec<&str>>();
85
86             // if there would be more commands in the received_data
87             // ...I'm not sure it would catch the correct data as interpreted_data everytime
88             if separated_string_by_colon.len() >= 2 {
89                 println!("Interpreted data: {:?}", separated_string_by_colon[1].clone().to_string());
90                 separated_string_by_colon[1].clone().to_string()
91             } else {
92                 println!("damn, didnt work");
93                 String::new()
94             }
95         }
96     }

```

- **Generelle Logik zur Interpretation der angekommenen Daten**



# Stakeholder Analyse



# Frontend technische Analyse

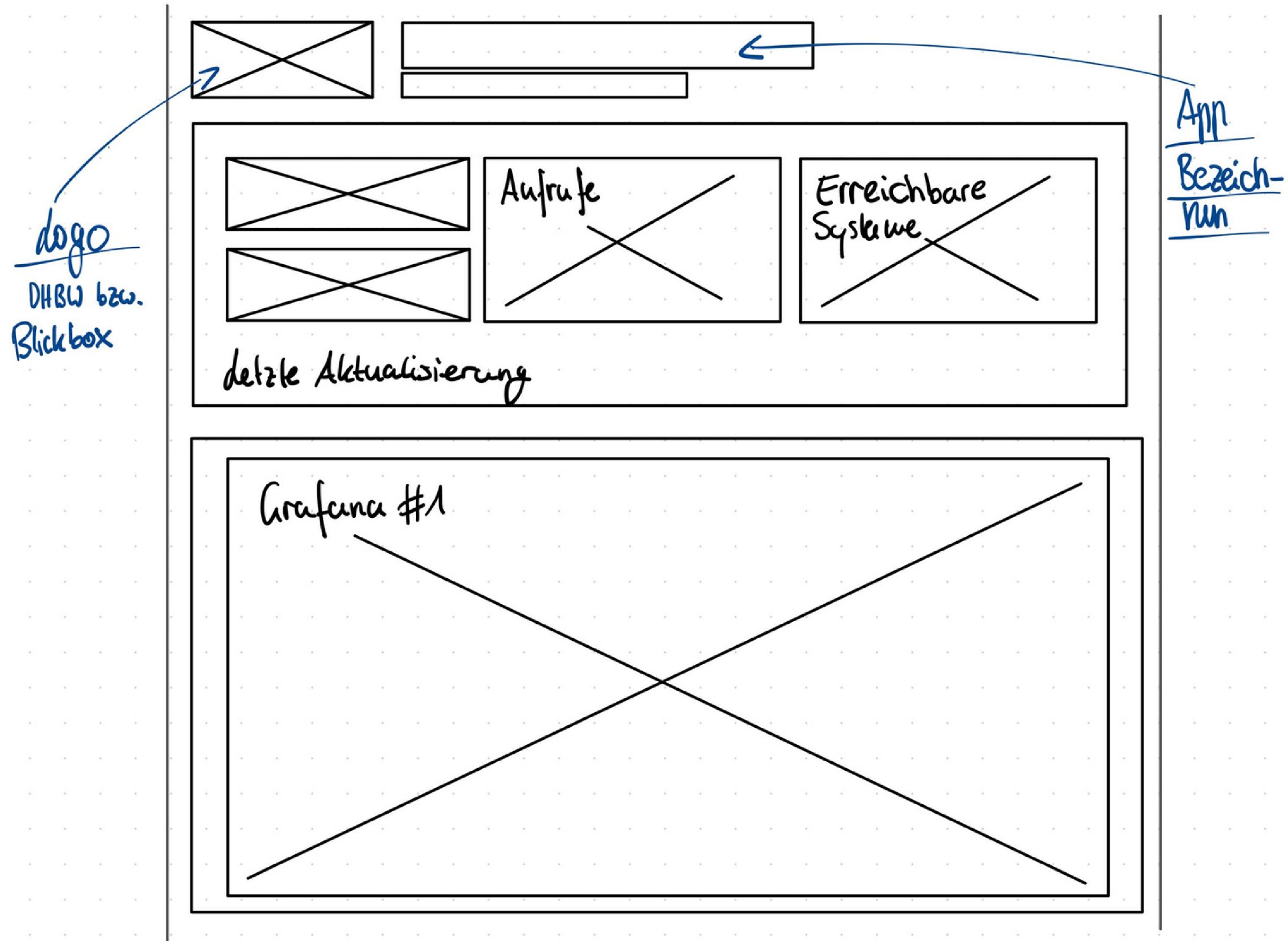
## Wahl des Frontend Frameworks

	Gewichtung	React	Angular	Vue	
Popularität (1)	2	80	<b>160</b>	56	<b>112</b>
Awareness (1)	3	100	<b>300</b>	100	<b>300</b>
Vorkenntnisse	5	70	<b>350</b>	5	<b>25</b>
Client Rendering Geschwindigkeit (2)	5	67	<b>335</b>	68	<b>340</b>
<b>Summe</b>	15	317	<b>1145</b>	229	<b>777</b>
				240	<b>855</b>
					<b>3563</b>

(1: <https://www.sitepoint.com/most-popular-frontend-frameworks-compared/>)

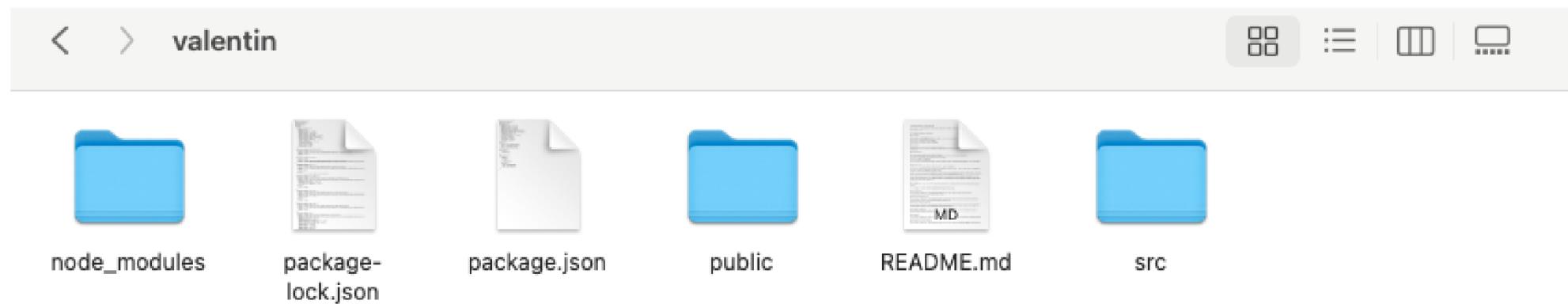
(2: <https://blog.smile.eu/en/digital/what-frontend-framework-fastest>)

# Frontend Dashboard Mockup



# Frontend Initialisierung

Valentin = Value and Notification



```
FROM node:17-alpine as builder
WORKDIR /app
COPY ./src/package*.json .
COPY ./src/yarn*.lock .
RUN npm install
COPY ./src ./
RUN npm run build

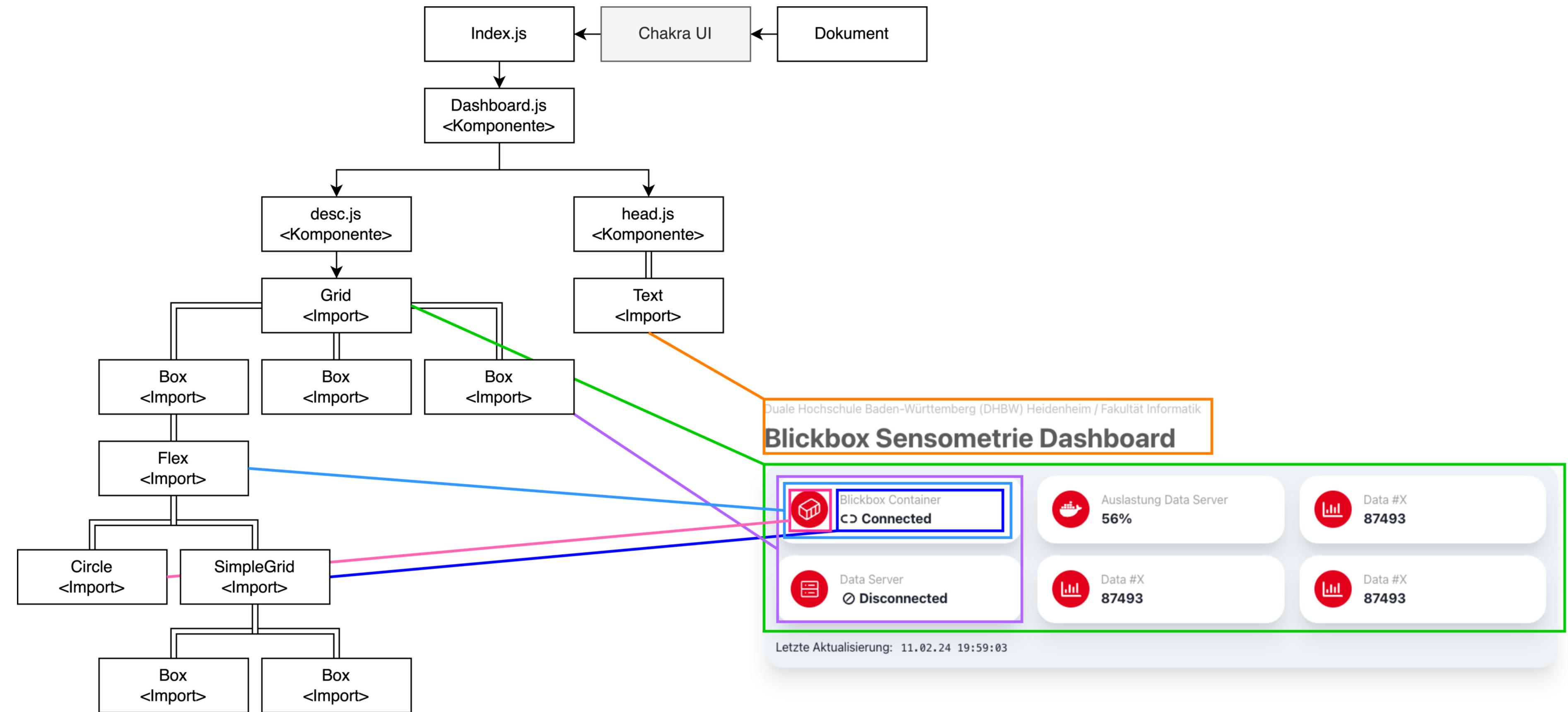
FROM nginx:latest
WORKDIR /usr/share/nginx/html
RUN rm -rf /*
COPY --from=builder /app/build .
ENTRYPOINT ["nginx", "-g", "daemon off;"]
```

```
#!/bin/bash

sudo chown -R $(whoami) ~/.docker

docker build -t valentin .
docker run -d -p 4000:80 --name valentin valentin
```

# Frontend Komponenten Planung



# Frontend Implementierung Basis Dashboard

Duale Hochschule Baden-Württemberg (DHBW) Heidenheim / Fakultät Informatik

## Blickbox Sensometrie Dashboard



Blickbox Container  
 Connected



Auslastung Data Server  
56%



Data #X  
87493



Data Server  
 Disconnected



Data #X  
87493



Data #X  
87493

Letzte Aktualisierung: 11.02.24 19:59:03

# Warum Python?

- Einfachheit und Lesbarkeit  
-> Einfache Syntax
- Große Auswahl an Frameworks
- Skalierbarkeit
- Große Community

# Flask vs. Django

	 Flask	 django	
Größe & Komplexität	x		Von der Größe und Komplexität ist Flask kleiner
Kostenlos	x	x	Beide Frameworks sind kostenlos
Benutzerfreundlich	x		Flask ist leichter zu lernen und somit für Anfänger besser
Geschwindigkeit	x		Für Messdaten sind schnelle Frameworks geeignet
Community		x	Django hat eine sehr aktive und große Community, die eine Fülle von Ressourcen, Bibliotheken und Erweiterungen zur Verfügung stellt.
Summe:	4	2	

```
● ● ●  
@app.route('/iot/api/insert/temperature', methods=['POST'])  
def insert_temperature():  
    try:  
        data = request.json  
        if 'temperature' not in data:  
            return jsonify({"message": "Falscher Input!"}), 400  
        temperature = float(data['temperature'])  
        if(temperature < -60.0 or temperature > 100):  
            return jsonify({"message": "Falscher Input!"}), 400  
  
        json_body = [  
            {  
                "measurement": "temperature",  
                "fields": {  
                    "value": temperature  
                }  
            }  
        ]  
        influx_client.write_points(json_body)  
        return jsonify({"message": "Daten erfolgreich eingefügt"}), 200  
    except Exception as e:  
        return jsonify({"error": str(e)}), 500
```

```
● ● ●  
@app.route('/iot/api/pingGF', methods=['GET'])  
def pingGrafana():  
    url = "http://grafana-server:3000/api/health"  
    try:  
        response = requests.get(url)  
        if response.status_code == 200:  
            return jsonify({"message": "Verbindung zu Grafana steht"}), 200  
        else:  
            return jsonify({"message": "Grafana Server hat geantwortet mit Statuscode  
{response.status_code}"}), response.status_code  
    except requests.ConnectionError:  
        return jsonify({"Fehler": "Keine Verbindung zum Grafana Server"}), 503
```



**POST** /insert/temperature Ein neuen Temperatur Wert hinzufügen

Füg einen neuen Temperatur Wert hinzu.

**Parameters**

No parameters

**Request body required**

application/json

Erstelle einen neuen Temperatur Wert

Example Value | Schema

```
{  
  "temperature": 27.8  
}
```

**Responses**

Code	Description	Links
200	Wert erfolgreich eingefügt.	No links
	Media type	
	application/json	
	Controls accept header.	
	Example Value   Schema	
	{ "message": "Daten erfolgreich eingefügt" }	
400	Falscher Input	No links
	Media type	
	application/json	
	Example Value   Schema	
	{ "message": "Falscher Input" }	
500	Interner Fehler	No links



**GET** /pingBB Checkt ob die Blickbox mit dem Internet verbunden ist.



Parameters

[Try it out](#)

No parameters

Responses

Code Description

Links

200 Verbindung steht

[No links](#)

Media type

application/json



Controls Accept header.

[Example Value](#) | [Schema](#)

```
{  
  "message": "BlickBox ist erreichbar"  
}
```

503 Fehler bei der Verbindung

[No links](#)

Media type

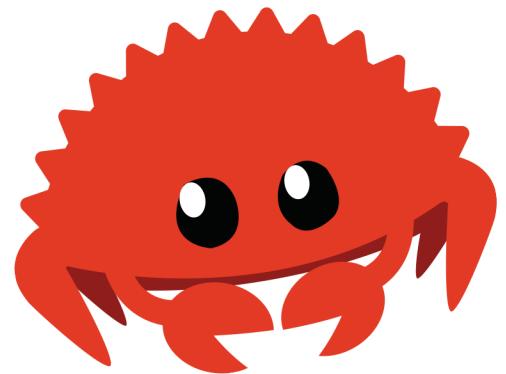
application/json



[Example Value](#) | [Schema](#)

```
{  
  "Fehler": "Keine Verbindung zur Blickbox"  
}
```

# Demo



grafana



api A black Python logo icon, which is a stylized lowercase letter 'p' enclosed in a circle.

Frontend  
Ada und Sara

