
Projektdokumentation

Leistungsbeurteilung LB 1

Modul	431
Eingereicht von	Gruppe A
Projektthema	Temperatursensoren an der gibb IET
Eingereicht bei	Lehrperson
Datum	29. August 2023

Dokumentinformation

Projektleiter	Levyn Schneider	
Projektmitglied	Meer	David
Projektmitglied	Schär	Josia
Diktator	Navagan	Navaajanan

Änderungsverzeichnis:

Datum	Version	Bemerkung / Änderung	Autor
29.08.2023	1	Dokumentation erstellt	Urs Dummermuth
29.08.2023	1.1	Variantenunterscheid, Rollen	Navaajanan Navagan
29.08.2023	1.2	Projektauftrag eingetragen	Levyn Schneider
29.08.2023	1.3	Soll/Ist-Analyse	Josia Schär
29.08.2023	1.4	Projektstrukturplan	David Meer
29.08.2023	1.5	Lösungsvariante eingefügt	Levyn Schneider
05.09.2023	1.6	Ziele	Navaajanan Navagan
05.09.2023	1.4 .1	Projektstrukturplan	David Meer, Levyn Schneider
05.09.2023	1.7	Zeitplan	Josia Schär
05.09.2023	1.8	Materialliste	David Meer
05.09.2023	1.9	Blackbox	Navaajanan Navagan
05.09.2023	1.1 0	Whitebox	Levyn Schneider
12.09.2023	2.1	Bauplan/Installation	Navaajanan Navagan
12.09.2023	2.2	Whitebox, Blackbox und Komponentendiagramm	Levyn Schneider
19.09.2023	2.3	Domain/Testkonzept	NN
19.09.2023	2.4	Webseite	LS
19.09.2023	2.5	Materiellste	JS
19.09.2023	2.6	Webdesign erstellt	JS/DM
17.10.2023	3.1	Arbeitsjournal/Dokumentation	NN
17.10.2023	3.2	Impressum/Bilder für Dokumentation	DM
17.10.2023	3.3	Zeitplan aktualisiert/ Kahoot frage	JS
17.10.2023	3.4	Testscript/ Sensoren mit Hotspot von NN erfolgreich getestet	LS
24.10.2023	3.5	Arbeitsjournal	NN
24.10.2023	3.6		

24.10.2023	3.7
24.10.2023	3.8

Inhaltsverzeichnis

1	Studie (Projektauftrag).....	16
1.1	Ausgangslage.....	16
1.2	SOLL-/IST-Analyse	16
1.3	Verfeinerte Ziele mit Anforderungen & Rahmenbedingungen.....	17
1.4	Variantenentscheid.....	18
1.5	Projektstrukturplan (allenfalls Printscreen aus Tool).....	18
1.6	Zeitplan.....	19
1.7	Aufwand & Kosten.....	19
2	Zusammenfassung	21
3	Initialisierungsphase	22
3.1	Ziele.....	Fehler! Textmarke nicht definiert.
3.2	Rahmenbedingungen:	22
3.3	Anforderungen an das System:.....	22
3.4	Organisation	22
3.4.1	Rollen.....	22
3.5	Zeitplan.....	23
3.6	Arbeitspakete	23
3.7	Lösungsvarianten	23
4	Konzeptphase.....	24
4.1	Blackbox.....	24
4.2	Whitebox	24
4.2.1	Komponentendiagramm.....	25
4.3	Bauplan / Installation	25
4.4	Materialliste	27
4.5	Testkonzept	27
4.6	Testprotokoll	29
5	Realisierungsphase	29
5.1	Entwicklung / Aufbau der Webseite.....	29
5.2	Administration.....	29
5.3	Benutzerverwaltung.....	29
5.4	Testen	29

6	Einführungsphase.....	31
6.1	Präsentation	31
6.2	Schulung	31
6.3	Abnahme und Einführung.....	31
7	Arbeitsjournal / Pendenzen.....	32
8	Reflexion.....	34
8.1	Projektmitglied 1: Funktion X.....	34
8.2	Projektmitglied 2.....	35
8.3	Projektmitglied 3.....	35
8.4	Projektmitglied 4.....	35
9	Anhang	36
9.1	Abbildungen	36
9.2	Quellenverzeichnis	36

Vorwort

Dokumentation "temperatursensoren-gibb.ch"

Dies ist die Dokumentation der Website "temperatursensoren-gibb.ch", entwickelt von Levyn Schneider mit Hilfe von David Meer.

Website

Die Website ist basierend auf Next.js und React.js Framework. Auf der Hauptseite (Landingpage) werden alle Sensoren, welche aktuell im Einsatz sind aufgelistet. Dies funktioniert mithilfe der Datenbank, welche später erwähnt wird.

```
1 // Get all sensors in descending order from database
2 const { data: rooms } = await supabase
3   .from("rooms")
4   .select()
5   .order("floor", { ascending: false });
6
7 console.log(rooms);
8
9 // Generate dict where the keys are the floors and insert the
10 // corresponding sensors into the floors
11 const roomsByFloor = {};
12 rooms.forEach((room) => {
13   const { floor, ...rest } = room;
14   if (!roomsByFloor[floor]) {
15     roomsByFloor[floor] = [rest];
16   } else {
17     roomsByFloor[floor].push(rest);
18   }
19 });
20
21 console.log(roomsByFloor);
22
23 // The current floors / keys in the dict get stored here in descending order
24 const floorKeys = Object.keys(roomsByFloor).sort((a, b) => b - a);
25
26 // Get the data (temp, humidity) from the sensors
27 // where order timestamp ascending: false means get the latest results
28 // and limit them for the amount of sensors
29 const { data: sensorData, error } = await supabase
30   .from("data")
31   .select()
32   .order("timestamp", { ascending: false })
33   .limit(rooms.length);
34
35 console.log(sensorData);
36
37 // Generate dict where the key is the id_room
38 const roomSensorData = {};
39 sensorData.forEach((item) => {
40   roomSensorData[item.room_id] = item;
41 });
42
43 console.log(roomSensorData);
```

```
1 <div className="mt-8">
2   <h1 className="font-poppins text-3xl font-bold mb-6">Räume</h1>
3   { /* Loop through the floors and generate for each floor a
4     container where the sensors from each floor get rendered in
5
6     | used for organisation and design on the website */ }
7   { floorKeys.map((floor) => (
8     <div
9       key={floor}
10      className="flex flex-row flex-wrap columns-3 gap-8 gap-y-6 mb-6"
11    >
12      { roomsByFloor[floor].map((room) => (
13        // Generate component with the title, temp and humidity
14        // The Key prop is used from React, to differentiate the components and identify them
15        return (
16          <Room
17            key={room.id}
18            title={room.name}
19            temperature={roomSensorData[room.id_room].temperature}
20            humidity={roomSensorData[room.id_room].humidity}
21          />
22        );
23      )}
24    </div>
25  )}
26 </div>
```

Auf den Einzelnen Räumen (Sensoren), wird die Temperatur und Feuchtigkeit genauer angezeigt mit zusätzlichen Informationen, wann z.B. die letzte Messung stattgefunden hat.

```
1  const [room, setRoom] = useState([]);
2
3  // calling useEffect to update Sensordata after every refresh
4  useEffect(() => {
5    // Get Current Room
6    getRoom();
7  }, []);
8
9  async function getRoom() {
10   // Get sensor data from database where the column "name" is equal
11   // to the params.room.
12   // params.room is the prop that gets inserted via the link
13   // For example /raum/ie-ie215, where ie-ie215 is the prop/param
14   const { data, error } = await supabase
15     .from("rooms")
16     .select()
17     .eq("name", params.room);
18
19   // check if the room is valid / exists
20   if (data !== []) {
21     setRoom(data);
22   }
23 }
```

```
1  /* Check if room const is not empty / existed in the getRoom() process */
2  {room !== [] ? (
3    <RenderRoom supabase={supabase} room={room} params={params.room} />
4  ) : (
5    <RenderError name={params.room} />
6  )}
```



```

1  async function RenderRoom({ supabase, room, params }) {
2    // Try if room[0] can be accessed
3    // If not means the room in the params is not valid
4    try {
5      console.log(room[0].id_room);
6    } catch (e) {
7      return <RenderError name={params} />;
8    }
9
10   // Select the last entry from the corresponding sensor in the database
11   const { data: sensorData, error } = await supabase
12     .from("data")
13     .select()
14     .eq("room_id", room[0].id_room)
15     .order("timestamp", { ascending: false })
16     .limit(1);
17
18   // Calculate the last measurement of the sensor
19   // Convert the date of the entry
20   const date = new Date(sensorData[0].timestamp);
21   // Get the current date
22   const dateToday = new Date();
23
24   // Calculate the time difference in milliseconds
25   const timeDifference = date - dateToday;
26
27   // Calculate the difference in days
28   // The difference is outputted in minus days
29   const diffInDays = Math.floor(timeDifference / (1000 * 60 * 60 * 24));
30
31   let lastReport = "";
32
33   if (diffInDays == 0) {
34     lastReport = "heute um";
35   } else if (diffInDays == -1) {
36     lastReport = `vor 1 Tag um`;
37   } else {
38     // toString() and slice(1) for removing the minus from the (now) string
39     lastReport = `vor ${diffInDays.toString().slice(1)} Tagen um`;
40   }
41
42   // Convert original output i.E: 2023-10-17T16:30:00+00:00 to 16:30
43   const timestamp = sensorData[0].timestamp
44     .split("T")[1]
45     .split("+")[0]
46     .split(":");
47   const timestampFormatted = timestamp[0] + ":" + timestamp[1];
48
49   return (
50     <
51     <h1 className="font-poppins font-bold text-3xl mt-8">
52       {room[0].name.toUpperCase()}
53     </h1>
54
55     <div className="flex items-center font-nunitosans font-medium text-lg mt-4">
56       <FontAwesomeIcon icon={faTemperatureHalf} className="text-2xl" />
57       <p className="ms-3">
58         Aktuelle Temperatur: {sensorData[0].temperature}&deg;
59       </p>
60     </div>
61
62     <div className="flex items-center font-nunitosans font-medium text-lg mt-2">
63       <FontAwesomeIcon icon={faPercent} className="text-xl" />
64       <p className="ms-3">
65         Aktuelle Feuchtigkeit: {sensorData[0].humidity}%
66       </p>
67     </div>
68
69     <div className="flex items-center font-nunitosans font-medium text-lg mt-2">
70       <FontAwesomeIcon icon={faClock} className="text-xl" />
71       <p className="ms-2">
72         Letzte Messung: {lastReport} {timestampFormatted} Uhr
73       </p>
74     </div>
75   </>
76 );
77 }

```

```

1  export function TemperatureChart({ id, timestamp }) {
2    const chartRef = useRef(null);
3
4    useEffect(() => {
5      async function fetchAndRenderData() {
6        try {
7          // Fetch data from Supabase
8          const { data, error } = await supabase
9            .from("data")
10             .select()
11             .eq("room_id", id)
12             .order("timestamp", { ascending: true });
13
14          if (error) {
15            console.error("Error fetching data from Supabase:", error);
16            return;
17          }
18
19          // Destroy the existing chart if it exists
20          if (chartRef.current) {
21            chartRef.current.dispose();
22          }
23
24          // Create a chart instance
25          const chart = anychart.line();
26
27          const graphData = data.map((row) => [
28            row.timestamp.split("T")[1].split("+")[0].slice(0, 5),
29            row.temperature,
30          ]);
31          console.log(graphData);
32
33          // Create a chart instance
34          const temperatureGraph = chart.line(graphData);
35          temperatureGraph.name("Temperatur");
36
37          // Set chart title
38          chart.title("Temperaturverlauf");
39
40          // Render the chart
41          chart.container("temperature-container");
42          chart.draw();
43
44          // Store the chart instance in the ref
45          chartRef.current = chart;
46        } catch (error) {
47          console.error("Error:", error);
48        }
49      }
50
51      fetchAndRenderData();
52    }, [id]);
53
54    return (
55      <div
56        className="charts mt-6"
57        id="temperature-container"
58        style={{ width: "80%", height: "250px" }}
59      />
60    );
61  }
62

```

```

1 function RenderError({ name }) {
2   // Return this if there was an error returning the room
3   return (
4     <h1 className="font-poppins text-3xl font-bold mt-8">
5       &quot;{name.toUpperCase()}&quot; ist kein gültiger Raum!
6     </h1>
7   );
8 }

```

Datenbank

Als Datenbank wurde wie beschlossen Supabase verwendet. Supabase ist eine JavaScript Datenbank welche einfache Funktionalität in Websites ermöglicht. Die Datenbank Struktur innerhalb der Datenbank ist mit zwei Tabellen aufgebaut: rooms und data.

In der "rooms" Tabelle werden alle Sensoren abgespeichert mit dem Namen des Raums, welcher der Sensor sich befindet und welche Etage. Die Website benutzt wie vielleicht schon bemerkt die einzelnen Namen der Sensoren / Räume, um zu validieren, dass der Link z.B. /raum/ie-ie215 korrekt ist.

<input type="checkbox"/>	id_r... <small>id...</small>	created_at <small>date</small>	name <small>text</small>	floor <small>int8</small>
<input type="checkbox"/>	5	2023-09-19	ie-ie215	3
<input type="checkbox"/>	6	2023-09-19	ie-ieu07	0

In der "data" Tabelle werden alle Messungen der einzelnen Sensoren gespeichert. Die Sensoren messen jede Halbestunde. Da es aktuell zwei Sensoren gibt und die Messungen 7 tagelang bleiben, gibt es gut mal bis zu 672 einzelne Messungen in der Tabelle.

Die Tabelle ist aufgebaut mit folgenden Attributen: timestamp, room_id, temperature und humidity. Timestamp wird genutzt, um zu speichern, wann die Messung stattgefunden hat. room_id wird als Fremdschlüssel verwendet um zum Sensor / Raum zu verlinken. Dann temperature und humidity sind die eigentlichen Messdaten des Sensors.

<input type="checkbox"/>	id_... <small>id...</small>	timestamp <small>timestamp tz</small>	roo... <small>id...</small>	temperature <small>float4</small>	humidity <small>int2</small>
<input type="checkbox"/>	376	2023-10-16 14:00:00+00	5 →	30	38
<input type="checkbox"/>	377	2023-10-16 14:00:00+00	6 →	20.2	64
<input type="checkbox"/>	378	2023-10-16 14:30:00+00	5 →	30.7	37
<input type="checkbox"/>	379	2023-10-16 14:30:00+00	6 →	20.1	64
<input type="checkbox"/>	381	2023-10-16 15:00:00+00	6 →	20.1	64
<input type="checkbox"/>	380	2023-10-16 15:00:00+00	5 →	29.8	36
<input type="checkbox"/>	382	2023-10-16 15:30:00+00	5 →	28.4	36
<input type="checkbox"/>	383	2023-10-16 15:30:00+00	6 →	20	64

Raspberry Pi

Das Raspberry Pi hat die Funktion den Sensoren jede Halbestunde eine Anfrage zu senden für die Temperatur und Luftfeuchtigkeit. Das Raspberry Pi ist auf einem einfachen Linux Distro konfiguriert mit einem Python File welche ununterbrochen läuft. Um dem RPI, Selbstständigkeit zu vergeben, wird das Python File sofort nach Start ausgeführt, somit muss man das File nicht selbst starten.

Das Python File wurde speziell so entwickelt, dass es jede Halbestunde in der "echten Zeit" ausführt und nicht, ab dann wann der RPI gestartet wurde, resp. das File. Das Python File reguliert auch das Löschen der Daten nach 7 Tagen in der Datenbank.

```

1 def run():
2     print(f"-----{datetime.now()}-----")
3     writeToLog(f"-----{datetime.now()}-----")
4
5     # Create tuyaaapi session with credentials and connect to it
6     openapi = TuyaaOpenAPI(ENDPOINT, ACCESS_ID, ACCESS_KEY)
7     openapi.connect()
8
9     def getDeviceData(device):
10         # Request status from tuyaaapi with the device id
11         data = openapi.get(f"/v1.0/iot-03/devices/{DEVICES[device]}/status")
12
13         # Convert to json
14         parsed_data = json.loads(json.dumps(data))
15         print(parsed_data)
16
17         va_temperature = None
18         va_humidity = None
19
20         # Loop through the result to get the temperature and humidity
21         for item in parsed_data['result']:
22             if item['code'] == 'va_temperature':
23                 va_temperature = item['value']
24             elif item['code'] == 'va_humidity':
25                 va_humidity = item['value']
26
27         # Convert temperature to a float
28         va_temperature = va_temperature / 10.0
29
30         insertData(device, va_temperature, va_humidity)

```

```


1 def insertData(device, temperature, humidity):
2     # Get the current local time
3     current_time = datetime.now()
4
5     # Extract the minutes component
6     minutes = current_time.minute
7
8     # Calculate the rounded minutes value
9     rounded_minutes = (minutes + 15) // 30 * 30
10
11    # Handle cases where rounded_minutes is 60 or higher
12    if rounded_minutes >= 60:
13        current_time += timedelta(hours=1)
14        current_time = current_time.replace(minute=0)
15    else:
16        current_time = current_time.replace(minute=rounded_minutes, second=0, microsecond=0)
17
18    # Insert the data into the supabase table
19    # room_id is the device key in the dict to be defined as foreign key
20    data = supabase.table("data").insert({"timestamp": str(current_time), "room_id": device, "temperature": temperature, "humidity": humidity}).execute()
21    print(data)
22    writeToLog(data)
23
24    stats(temperature, humidity)
25
26    # Loop through each device in the dict
27    for device in DEVICES.keys():
28        getDeviceData(device)

```

```

1 def delete():
2     writeToLogDel(f"-----{datetime.now()}-----")
3     # Calculate time defined days ago
4     timePast = datetime.now() - timedelta(days=DAYS)
5     formattedTime = timePast.strftime("%Y-%m-%dT%H:%M:%S+00:00")
6
7     # Get all data which is older than the defined days
8     response = supabase.table('data').select("*").lte('timestamp', formattedTime).execute()
9
10    dataList = []
11
12    # Append each item to the list above
13    for item in response.data:
14        dataList.append(item)
15
16    # Loop through each element in the list
17    for elem in dataList:
18        # Delete each element from the database table
19        data, count = supabase.table('data').delete().eq('id_data', elem["id_data"]).execute()
20        print(data)
21        writeToLogDel(data)

```



```
1 def waitUntilInterval():
2     while True:
3         # Get the current time in local time
4         current_time = time.localtime()
5
6         # Calculate the seconds remaining until the next interval
7         seconds_until_next_interval = MINUTES * 60 - \
8             (current_time.tm_min % MINUTES) * 60 - current_time.tm_sec
9         print(seconds_until_next_interval)
10        # Wait for the calculated time
11        time.sleep(seconds_until_next_interval)
12
13        # Now you are at the next interval, so you can run your function
14        run()
15        delete()
16
17
18 waitUntilInterval()
```

Abkürzungsverzeichnis

RPI	Raspberry PI
GPL	GNU General Public License
TS	Temperatursensoren

1 Studie (Projektauftrag)

1.1 Ausgangslage

Wir haben bereits zwei Temperatursensoren für einen Prototypen bestellt, welche zwischen 27. August und 4. September ankommen. Ebenfalls haben wir ein gewissen Grundwissen in Skripten, Netzwerktechniken und Webentwicklung. Im Internet haben wir gewisse Inspirationen gefunden, wie wir das Projekt umsetzen könnten.

Die Idee entstand aus einem Chat-GPT Vorschlag für eine IOT-Wetterstation. Der Grundbaustein für diese Idee entstand aus einem Witz, da die Gibb für ihre warme Zimmertemperaturen bekannt ist.

1.2 SOLL-/IST-Analyse

› Material / Ressourcen

Ist: Wir haben 2 Temperaturmessgeräte bestellt. Raspberry Pi von Levyn Schneider, denn wir für das Projekt benutzen können.

Soll: *Unser Temperatur Messgeräte müssen noch geliefert werden. Levyn muss sein Raspberry Pi von zu Hause mitbringen.*

› Skills

Ist: Wir haben einige Kenntnisse über das Programmieren von Webseiten, Kenntnisse über verschiedene Designs und wir wissen, wie man die Temperatursensoren mit dem Netzwerk verbindet.

Soll: *Wir müssen das Knowhow über das Programmieren und Design noch verbessern.*

› Interessen

Ist: Wir möchten uns im Webseiten programmieren verbessern. Wir möchten den andern Lernenden helfen sich auf der Raumtemperatur vorzubereiten. Wir möchten unser Teamwork Fähigkeit verbessern.

Soll: Wir müssen auf YT oder anderen Informationsquellen, Infos zum Programmieren holen. Wir müssen als Gruppe zusammenarbeiten und nicht als vier Individuen in einer Gruppe.

› Bestehende Elemente aus dem Umfeld

Ist: im Moment ist es heiss in der IET-Abteilung und weshalb wir uns auf die Raumtemperatur vorbereiten wollen.

Soll: Wir möchten anderen Lernenden helfen sich auf die ausgeprägten Temperaturen vorzubereiten.

1.3 Ziele mit Anforderungen & Rahmenbedingungen

Ziel 1:

Die Lernenden und die Lehrer von der Gibb IET sollen von unserer Webseite herauslesen können, was die aktuellen Raumtemperaturen sind. Diese werden durch Implementierung eines zuverlässigen Temperaturüberwachungssystem auf die Webseite dargestellt.

Anforderung zum Ziel 1:

Dass die Lernenden von der Website profitieren können, indem sie sich auf den Raum vorbereiten können (passende Kleidung, evtl. Ventilator, genügend Flüssigkeit).

Ziel 2:

Eine Webseite, welche aktuelle Temperaturen und Luftfeuchtigkeit von den Räumen anzeigt.

Anforderung zum Ziel 2:

Website, um die aktuellen Temperaturen und Luftfeuchtigkeit von den Räumen and der Gibb-IET anzuzeigen.

Ziel 3:

Die Daten auf der Webseite protokollieren, um im späteren Zeitpunkt anzuzeigen, wie die Temperatur verläuft, um Massnahmen für die kommenden Jahren zu unternehmen. Die Daten werden grafisch aufgezeigt.

Anforderung zum Ziel 3:

Protokollieren der Daten, um anzuzeigen wie die Temperatur verläuft, um Massnahmen für die kommenden Jahren zu unternehmen.

1.4 Variantenentscheid

Variante 1: Wir kaufen Temperaturmessgeräte ein und bauen diese in allen IET-Zimmern ein. Danach erstellen wir einen Teamskanal. In diesem Teamskanal sind die Temperaturen der einzelnen Zimmern ersichtlich. Die Lehrer und die Lernenden können im Teamskanal die einzelnen Zimmern auswählen und sehen dort die Temperaturen für das ausgewählte Zimmer.

Vorteile:

- Wir benötigen weniger Arbeitszeit
- Der gesamte Auftrag ist weniger aufwendig

Nachteile:

- Für die User ist die Verwendung von Webseiten eventuell einfacher.

Variante 2: Wir kaufen Temperaturmessgeräte ein und bauen diese in allen IET-Zimmern ein. Des Weiteren erstellen wir eine eigene Webseite, wo wir auch die Temperaturen von den vergangenen Tagen lesen können. Dies soll, denn Lehrpersonen wie auch den Studierenden aufzeigen, dass es immer wärmer oder kälter wird.

Vorteile:

- Wir lernen somit eine Webseite Programmieren.

Nachteile:

- Wir lernen nicht, wie wir ein Temperatur Messgerät bauen können.

Variante 3: Wir arbeiten mit eigenen Temperaturmessgeräte, dazu benötigen wir ein Microcontroller (Raspberry Pi), Breadbord, ein Jumper-Kabel, Display und natürlich eine Stromversorgungsquelle. Diese Daten werden wir via Webseite, allen zur Verfügung stellen.

Vorteile:

- Wir arbeiten hier mit mehr Hardware.

Nachteile:

- Die Teile für die Temperaturmessgeräte kosten mehr Geld als eine fertige zu kaufen.
- Wir brauchen viel Zeit, um die Temperaturmessgeräte herzustellen, weshalb wir nicht genügend Zeit haben, um die Webseite selber zu programmieren.

1.5 Projektstrukturplan (allenfalls Printscreen aus Tool)

https://miro.com/app/board/uXjVMoLtpAk=?share_link_id=285771068323

1.6 Zeitplan

Zeitplan	Zeit in Stunden	1		2		3		4		5		6		7		8	
		K0/02		K0/04		K0/05		K0/06		K0/07		K0/08		K0/09		K0/10	
		DL	Z	DL	Z	DL	Z	DL	Z	DL	Z	DL	Z	DL	Z	DL	Z
Initialisierung	Soll-Total: 39.5 Ist-Total: 39.5																
1. Dokumentation	Soll: 0 Ist: 0																
1.1 Ausgangslage	Soll: 5 Ist: 5			0.5		0.5											
1.2 Soll/Ist Analyse	Soll: 5 Ist: 5			0.5		0.5											
1.3 Verfeinerte Ziele	Soll: 7.5 Ist: 7.5			0.5		0.5		1.5		0.5							
1.4 Variantenunterschied	Soll: 0.5 Ist: 0.5			0.5		0.5		1.5									
1.5 Aufwand und Kosten	Soll: 5 Ist: 5			0.5		0.5		0		0.5							
2. Projektstrukturplan	Soll: 5 Ist: 5.25					2		0		0.25							
3. Zeitplan	Soll: 0 Ist: 0							0		0.25							
4. Arbeitskette	Soll: 0.5 Ist: 0.5							0.5									
Konzept	Soll-Total: 12 Ist-Total: 10.8																
1. Systemarchitektur	Soll: 5 Ist: 5							5									
2. Blockbau	Soll: 1.5 Ist: 1.5							1		0.5							
3. Whitbox	Soll: 2 Ist: 1.25							1		0.25							
4. Bauplan	Soll: 2 Ist: 2							2									
5. Materialliste	Soll: 1.5 Ist: 1.5							1		0.5							
6. Taskkonzept	Soll: 2 Ist: 1							1									
7.1 Fontauswählen	Soll: 0.5 Ist: 0.25							0.5		0.25							
7.2 Farbauswählen	Soll: 1 Ist: 0.25							1		0.25							
7.3 Designerauswählen	Soll: 1 Ist: 0.25							1		0.25							
Realisierung	Soll-Total: 50.3 Ist-Total: 32.3																
1. Entwicklung	Soll: 0 Ist: 0																
1.1 Webseite entwickeln	Soll: 10 Ist: 1.7									10							
1.2 Webdesign realisieren	Soll: 1.5 Ist: 7									1.5							
2. Verbindung RPI mit Sensoren	Soll: 5 Ist: 5									5							
3. API-Entwickeln für Webseite	Soll: 10 Ist: 5									10							
4. Temperatursensoren bestellen	Soll: 0.25 Ist: 0.25									0.25							
5. Dokumentieren/Arbeitsjournal	Soll: 2 Ist: 4													2			
Einführung	Soll-Total: 10 Ist-Total: 0																
1. Vorbereitung Präsentation	Soll: 10 Ist: 5													10			
1.1 Präsentation	Soll: 0 Ist: 0															0.5	
Total	Soll-Total: 112 Ist-Total: 82.8																

1.7 Aufwand & Kosten

An diesem Projekt arbeiten 4 Personen. Der Aufwand schätzen wir auf 35 Arbeitsstunden insgesamt für ungefähr alle 4 Personen. Der Stundenansatz berechnen wir auf CHF 120 pro Person.

CHF 36.90 für zwei Temperaturmessgeräte als Prototyp + 35h pro Person CHF 120.
Total: CHF 16'836.90

- externe kosten:

Produktkosten: 150 Fr

Transportkosten: 0 Fr.

Lizenz- und Zertifizierungskosten: aktuell 0 Fr.

Marketingausgaben: 0 Fr.

Versicherungskosten:

Steuern und Abgaben:

- interne kosten:

Personalkosten: An diesem Projekt arbeiten 4 Personen. Der Aufwand schätzen wir auf 35 Arbeitsstunden insgesamt für ungefähr alle 4 Personen. Der Stundenansatz berechnen wir auf CHF 120 pro Person. Das entspricht 16'800 Fr

Materialkosten: 36.90 Fr.

Versandkosten: 0 Fr

Verwaltungskosten:

Supportkosten: Der Preis für die IT-Service-Stunde liegt bei 60 Fr pro Stunde

Sonstige Kosten: aktuell keine Vorhanden

2 Zusammenfassung

3 Initialisierungsphase

3.1 Rahmenbedingungen:

- 1) Unsere Temperaturmessgeräte müssen so deponiert werden, dass unbefugte unsere Messgeräte verstellen/fälschen.
- 2) Wir müssen sicherstellen, dass unbefugte keinen Zugriff auf unsere Datenbank oder auf Raspberry Pi haben.
- 3) Wir können die Systemdefekte/Fehler nicht berücksichtigen

3.2 Anforderungen an das System:

- Das System sollte so konfiguriert sein, dass bei Fehlfunktionen oder Sensoren ausfallen eine Benachrichtigung auf der Webseite angezeigt.
- Die Temperaturdaten müssen mit einer Genauigkeit von mindestens $\pm 0,5^\circ$ erfasst und auf der Webseite angezeigt werden.

3.3 Organisation

3.3.1 Rollen

Auftraggeber:

- Simon Witter

Projektleiter:

- Levyn Schneider

Projektmitglieder:

- David Meer, Josia Schär

Dokumentierer:

- Navaajanan Navagan

3.4 Zeitplan

Zeitplan	Zeit in Stunden	1		2		3		4		5		6		7		8	
		K0/23		K0/24		K0/25		K0/26		K0/27		K0/28		K0/29		K0/30	
		DL	Z	DL	Z	DL	Z	DL	Z	DL	Z	DL	Z	DL	Z	DL	Z
Initialisierung	Soll-Total: 39,5 Ist-Total: 39,5																
1. Dokumentation	Soll: 0 Ist: 0																
1.1 Ausgangslage	Soll: 5 Ist: 5			2,5		2,5											
1.2 Soll/Ist Analyse	Soll: 5 Ist: 5			2,5		2,5											
1.3 Verfeinerte Ziele	Soll: 7,5 Ist: 7,5			2,5		2,5		2,5		2,5							
1.4 Variantenunterschied	Soll: 6,5 Ist: 7,5			2,5		2,5		1,5		0,5							
1.5 Aufwand und Kosten	Soll: 5 Ist: 7,5			2,5		2,5		2,5		0,5							
2. Projektstrukturplan	Soll: 5 Ist: 5,25					2,5		2,5		0,25							
3. Zeitplan	Soll: 0 Ist: 0							2,5		0,25							
4. Arbeitskette	Soll: 0,5 Ist: 0,5							0,5		0,5							
Konzept	Soll-Total: 12 Ist-Total: 10,8																
1. Systemarchitektur	Soll: 5 Ist: 2							5		2							
2. Blackbox	Soll: 1,5 Ist: 1,5							1,5		1,5							
3. Whitbox	Soll: 2 Ist: 1,25							2		0,25							
4. Bauplan	Soll: 2 Ist: 2							2		2							
5. Materialliste	Soll: 1,5 Ist: 0,5							1,5		0,5							
6. Taskkonzept	Soll: 2 Ist: 1							2		1							
7.1 Punkteauswählen	Soll: 0,5 Ist: 0,25							0,5		0,25							
7.2 Farbeauswählen	Soll: 1 Ist: 0,25							1		0,25							
7.3 Dreiecksauswählen	Soll: 1 Ist: 0,25							1		0,25							
Realisierung	Soll-Total: 50,3 Ist-Total: 32,3																
1. Entwicklung	Soll: 0 Ist: 0																
1.1 Webseite entwickeln	Soll: 20 Ist: 1,7									20							
1.2 Webdesign realisieren	Soll: 1,5 Ist: 7									1,5							
2. Verbindung RPI mit Sensoren	Soll: 5 Ist: 5									5							
3. API entwickeln für Webseite	Soll: 10 Ist: 5									10							
4. Temperatursensoren bestellen	Soll: 0,25 Ist: 0,25									0,25							
5. Dokumentieren/Arbeitsjournal	Soll: 2 Ist: 4													2			
Einführung	Soll-Total: 10 Ist-Total: 0																
1. Vorbereitung Präsentation	Soll: 10 Ist: 5													10			
1.1 Präsentation	Soll: 0 Ist: 0															0,5	
Total	Soll-Total: 112 Ist-Total: 82,8																

3.5 Arbeitspakete

Unter diesem Link findet man unsere Arbeitspakete:

https://miro.com/app/board/uXjVMoLtpAk=?share_link_id=285771068323

3.6 Lösungsvarianten

Die Lösung, die wir gewählt haben, ist, dass wir einsatzbereite Temperaturmessgeräte kaufen und bauen diese in allen IET-Zimmern ein. Des Weiteren erstellen wir eine eigene Webseite, wo wir die aktuellen Temperaturen an der gibb IET sehen aber auch die Temperaturen von den vergangenen Tagen lesen können. Dies soll, denn Lehrpersonen wie auch den Studierenden aufzeigen, wie die aktuelle Temperatur ist oder wie auch dass es immer wärmer oder kälter wird.

4 Konzeptphase

4.1 Blackbox

Unsere Webseite soll einfach zum Bedienen sein. Auf der Webseite sollten die Lernenden die einzelnen Räume auswählen können und dort die aktuelle Raumtemperatur lesen können.

In der Navigationsleiste kann man zu jedem Raum, an der gibb-IET die Temperatur Verläufe ansehen.

Unsere Temperaturen werden wir voraussichtlich auf den Schränken platzieren, damit sie weniger Aufmerksamkeit erregen.

Die Schriftarten der Website sind bestehenden aus: Poppins, für Title Elemente und Nunito Sans für diversere kleinere Texte.

Das Farbdesign der Website sind bestehenden aus:

Primary: #F7FAFF

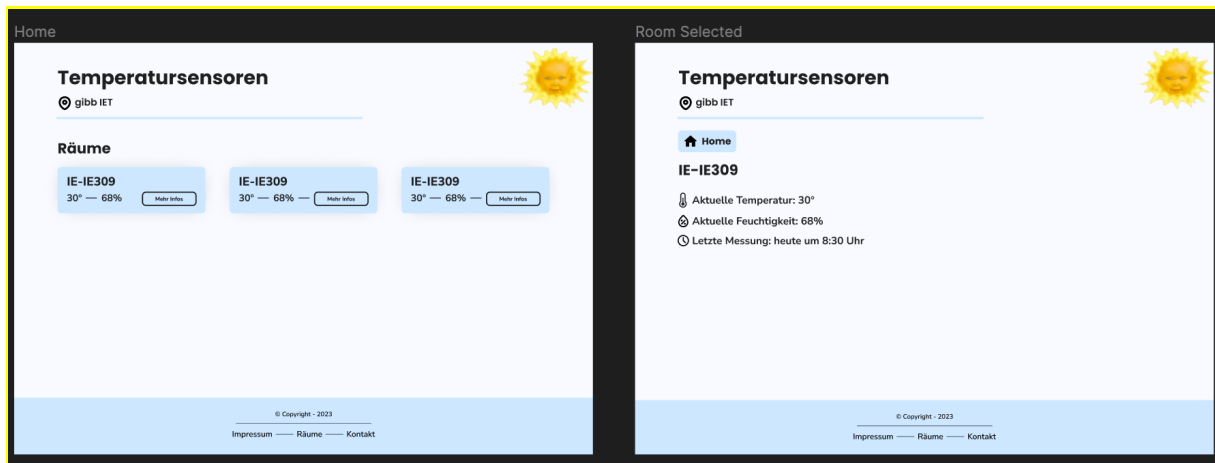
Secondary: #CDE8FF

Accent: #C9FFBE

Text: #202020

Die Designart soll aus einem minimalistischen, räumlichen, aber auch übersichtlichem Design bestehen.

Mockup:



4.2 Whitebox

Das System ist aufgebaut auf mehreren erweiterbaren Temperatursensoren. Die Temperatursensoren werden über die Tuya "SmartLife App" initialisiert und mit dem Netzwerk verbunden.

Die Temperatursensordaten werden über das Raspberry PI mit Hilfe des CLI-Betriebssystems "Home Assistent" über eine API auf die Website übermittelt.

Das Frontend der Website ist auf React basiert und das Backend basierend auf Node.js.

Der Temperaturverlauf wird stündlich auf die "Supabase" Datenbank geschrieben. Die Daten werden nach einer Woche wieder gelöscht, um Überlastung und Speicher zu sparen.

Bei einem Request von einem User ruft die Website die aktuellen Daten von der Datenbank ab.

4.2.1 Komponentendiagramm

Name	Funktion
Navigation	Rendered die Navigation mit den einzelnen Punkten
Footer	Rendered den Footer
Sensor	Eine Komponente, welcher für jeden Sensor die Daten abrufen und eine Seite generiert
Räume	Komponente für die Räume.
Logik	Logik Komponente für generelle Logik

4.3 Bauplan / Installation

Module;

Temperatursensoren: Dieses Modul umfasst die Hardwarekomponenten, welche in den Klassenzimmern installiert werden, um die Raumtemperatur zu erfassen.

Raspberry Pi: Das Raspberry Pi dient als Blackbox, um Daten von den Temperatursensoren zu sammeln und diese in die Datenbank zu übertragen.

Datenbank: Die Datenbank dient damit die Temperaturdaten, welche vom Raspberry Pi gesammelt werden und auf der Webseite anzuzeigen werden.

Webseite: Das Webseitenmodul dient für die Benutzer, um die aktuellen Raumtemperaturen anzuzeigen.

Domain: Dieses Modul umfasst die Bereitstellung einer Webseite. Damit User und Benutzer der Webseite den entsprechenden Zugriff haben.

externe Schnittstellen;

Lehrperson: Der Lehrperson ist eine externe Schnittstelle, weil er das Projekt Genehmigen oder nicht genehmigen kann.

Die IET-Räume (Eventuell Schulleiter): Der Schulleiter muss eine Genehmigung erteilen, um die Temperatursensoren in den Klassenzimmern zu teilen.

Lehrkräfte und Schüler: Lehrpersonen, Schüler, & Hauswärter/innen sind die Endnutzer der Website, welche die Raumtemperaturen anzuzeigen.

Temu: Temu ist Onlineshop, welche uns die Temperatursensoren zuliefert, bei einer Bestellung.

Webhosting-Anbieter: Die Website wird auf einem Webhosting-Server gehostet, der eine externe Schnittstelle darstellt. Voraussichtlich benutzen wir Netlify als Webhoster.

Domain: DNS-Server sind externe Schnittstellen, weil sie die Zuordnung von Domainnamen zu IP-Adressen bereitstellen. Diese sind für den Zugriff der Domain im Internet entscheidend.

Interne Schnittstellen;

Kommunikationsschnittstelle zwischen Temperatursensoren, Raspberry Pi: Die erfassten Daten von den Sensoren werden durch Raspberry P auf die Webseite übertragen.

Supabase: Die erfassten Temperaturdaten werden in der Supabase gespeichert.

Webanwendung: Die Schnittstelle zwischen dem Überwachungssystem und der Website ermöglicht die Übertragung und Anzeige der Temperaturdaten auf der Website.

Systemgrenze;

Wir können die allgemeine Stromversorgung und die Netzwerkinfrastrukturen der Schule nicht beeinflussen.

Wir können nicht sicherstellen das unsere Temperaturgeräte manipuliert werden.

Das Webhosting über Netlify, bietet eine Uptime von 99.98%. Allerdings gibt es keine Garantie beim Starter oder Pro Plan.

Unsere Datenbank, Supabase, hat ein Uptime von 99.9%.

Domain: Da wir eine Datenbank (Supabase) verwenden, besteht eine Schnittstelle zwischen dem Webserver und dem Datenbankserver.

Software;

Informationen finden Sie hier:

~ 4.2.1 Komponentendiagramm

Hardware;

Informationen finden Sie hier:

~ 4.1 Blackbox

~ 4.2 Whitebox

4.4 Materialliste

Anzahl	Name	Link	Lizenz	Preis	Bestellt/Geliefert
2	Tuya WiFi Temperatur Feuchtigkeitsensor	https://bit.ly/temperaturatsensor_gibb	-	18.99 CHF pro Stück	Geliefert
1	Raspberry PI 2	https://bit.ly/rasberry2-gibb	-	40 CHF	Schon vorhanden
-	SmartLife	bit.ly/smartlifegibb	GNU General Public License	-	
-	Home Assistant	https://www.home-assistant.io/	Apache License (free and open source)	-	
-	LocalTuya	https://github.com/rospogrigio/localtuya	GNU General Public License v.3.0	-	
-	VSCode	https://code.visualstudio.com/	Standard MIT license	-	
-	Node.js	https://nodejs.org/de	Permissive MIT license	-	
-	React JS	https://react.dev/	MIT License	-	

4.5 Testkonzept

Testfall Nummer	Name	Vorgehen	Erwartetes Resultat
1	Aufruf	Der Benutzer ruft die Website auf.	Die Website öffnet sich.

2	Raum Sub Page	Der Benutzer klickt auf einen Raum.	Der Raum öffnet sich auf der Website und die Temperatur wird angezeigt.
3	Sensorverlauf Analytik	Der Benutzer benötigt, die Temperatur-daten der vergangenen Wochen und öffnet die Webseite.	Es werden ihm/ihr die aktuellen Daten angezeigt und oben in der Mitte kann man mit einem Klick auf den Pfeil die letzten Daten einsehen.
4	Clouddaten	Wir öffnen die Tuya Webseite.	Wir können die Sensoren- Daten von der Cloud ansehen
5	Sensor hinzufügen	Sensor in der App "Smartlife" hinzufügen. Verbindung hinzufügen in der Tuya IOT Cloud.	In der API-Rückgabe sollte der neue Sensor inkl. die Alten vorhanden sein.
6	Sensor löschen	Sensor von der App "Smartlife" entfernen. Sensor von der Tuya IOT Cloud löschen.	In der API-Rückgabe sollte der Sensor nicht mehr vorhanden sein, und gelöscht sein in der Datenbank.

4.6 Testprotokoll

Testfall Nummer	Name	Fazit
1	Aufruf	Dies hat erfolgreich funktioniert. Die Testperson kam wie erwartet auf unsere Webseite.
2	Raum Sub Page	Der Raum öffnete sich und man konnte die Temperatur lesen.
3	Sensorverlauf Analytik	Die Testperson konnte die Daten von der letzten Woche nachsehen.
4	Clouddaten	Ein Interner Testperson konnte bestätigen, die Sensoren-Daten von der Cloud ansehen
5	Sensor hinzufügen	Dieser Test war erfolgreich
6	Sensor löschen	Sensor löschen war erfolgreich

5 Realisierungsphase

5.1 Entwicklung / Aufbau der Webseite

5.2 Administration

5.3 Benutzerverwaltung

5.4 Testen

Checkliste

--	--	--

6 Einführungsphase

6.1 Präsentation

6.2 Schulung

6.3 Abnahme und Einführung

7 Arbeitsjournal / Pendenzen

KW	Zeit	Kürzel	Zielsetzung / Tätigkeit	Erfüllungsgrad Positives / Negatives	Reflexion/Fazit
38	12:15	LS	Das Ziel ist es, ein Webdesign und einer Webseite zu erstellen. Mein Tagesziel ist es eine Webseite mit next und react zu kreieren.	Ich habe mein Ziel erreicht. Dabei hatte ich Probleme bei der Internetverbindung mit den TS. Dies habe ich anschliessend mit dem Hotspot-Verbindung gelöst. Was mir auch Schwierigkeiten verursacht hatte, war die Datenbankverbindung zwischen TS und der Webseite. Das Problem dabei war, dass ich kein Recht zum Zugriff auf die Datenbank hatte, dann habe ich mir die Richtlinien verlieht und konnte dann weiterarbeiten.	Ich habe viel Zeit damit verbracht auf die Datenbankverbindung zuzugreifen, ohne mir den Zugriff zu gewähren. Das nächste Mal werde ich das schon von Anfang an machen. Ich war auch froh, meiner Gruppe über den aktuellen Stand zu erklären.
38	12:00	DM	Das Ziel ist es, ein Webdesign und eine Webseite zu erstellen. Mein Ziel für heute ist es mit JS eine simples und schönes Webdesign zu erstellen.	Das Webdesign haben wir zuerst auf wix.com versucht zu erstellen und waren damit unzufrieden, wegen der eingeschränkten Designfreiheit und wir haben gelesen, dass bei Wix.com erstellte Webseiten, Werbung von Wix.com erhalten, was unprofessionell ist. Wir haben dann für figma.com entschieden und dies war einfach zu bedienen und wir konnten damit unseren Webdesign erstellen.	Das nächste Mal werden wir mehr Zeit in die Vorarbeit investieren. Da wir die Webseite erstellt haben, war es für uns klar, wo wir die TS-Daten sieht. Wir mussten aber die Webseite Benutzer freundlich und einfach gestalten. Im Großen und Ganzen bin ich froh, dass das Webdesign mir meinen Teamkameraden gefallen hat.
38	12:15	JS	Das Ziel ist es, ein Webdesign und eine Webseite zu erstellen.	Wie oben bei DM schon erwähnt, haben wir bei figma.com weiterarbeiten können. DS und ich, wir beide waren uns schnell einig, dass wir 2 Grafen: 1 für Temperatur und 1 für die Luftfeuchtigkeit erstellen. Die Farben haben wir auch neutral gestalten. Problem dabei hatten	Ich bin froh, dass unser Webdesign einfach und freundlich darstellen konnten. Für weitere Webseiten werde ich auf figma.com zurückgreifen.

				wir nicht. Wir haben unser Design an LS und NN gezeigt und sie waren damit auch zufrieden.	
38	12:15	NN	<p>Angeschaut wer wann Ferien hat, und besprochen wie man erreichbar ist, wenn es brennt.</p> <p>Mein Tagesziel für heute ist es, den Arbeitsjournal fertig zu schreiben. Und den anderen so gut wie möglich zu helfen.</p>	<p>Wir haben alle in verschiedenen KW-Ferien/ wir sind nicht alle gezeitigt erreichbar.</p> <p>LS habe ich geholfen seine Probleme zu analysieren wie z.B. Datenbankverbindung und habe mit ihm nach möglichen Lösungen gesucht. LS und Ich, wir waren auch sehr zufrieden mit dem simplen Design von DM und JS.</p>	<p>Via WhatsApp sind wir am besten erreichbar.</p> <p>Ich hatte die Möglichkeit in beide arbeiten reinzuschauen und mir Notizen zu machen für das Dokumentieren. Das nächste Mal beginne ich früher mit dem Arbeitsjournal damit ich am Ende nur noch die jeweilige Reflexion schreiben kann.</p>
42	12:00	LS	Mein Ziel für heute ist es, die Webseite fertig zu coden.	Ich habe die Webseite mit JS fertig gecodet und habe mit DM das Impressum fertig geschrieben. Ich bin sehr überrascht von meinen Ergebnissen, weil sie sehr positiv ausgefallen sind.	Ich bin zufrieden mit meiner Arbeit heute. Ich konnte heute durchgehend problemlos fertig coden.
42	12:00	DM	Mein Tagesziel für heute ist es, das Impressum fertig zu schreiben.	Ich habe mich informiert, wie man ein Impressum schreibt und das habe ich auch geschrieben.	Ich bin auch sehr zufrieden, da mein Impressum mir gelungen ist und alle vom Team einverstanden sind.
42	12:00	JS	Ich möchte heute die Webseite fertig gecodet haben mit LS.	Ich habe mit LS die Webseite fertig gecodet und den Zeitplan aktualisiert.	Ich bin auch zufrieden. Ich habe gemerkt, dass ich noch einiges zum Lernen habe, was das coden angeht. In der Zukunft werde ich mehr daran arbeiten, um im

					Team mehr helfen zu können.
4 2	12:00	NN	Ich muss heute die Dokumentation fertig schreiben.	Ich habe mich informiert, wer was macht und damit die Dokumentation fertig geschrieben, da ich der Dokumentierer.	Ich bin froh, dass ich das Team nicht viel unterbrechen musste, um nachzufragen an was sie arbeiten und wie es aktuell läuft. Ich konnte mir die Infos gut merken und das hier notieren.

8 Reflexion

«Das Argumentieren besteht aus neun Teilen: Sie werden für einen bestimmten Zweck argumentieren (1), Sie werden versuchen, ein Problem zu lösen (2), zumindest eines, Sie werden Informationen verwenden, die Sie von irgendwoher bekommen haben (3), Sie werden diese Informationen mit Konzepten und Ideen interpretieren (4). Sie werden zu einigen Schlussfolgerungen kommen (5). Diese Schlussfolgerungen werden Auswirkungen haben (6). Die Schlussfolgerungen werden auf Annahmen beruhen (7), die Sie mit Ihrem Standpunkt begründen werden (8). Unabhängig davon, ob Sie es erkennen oder nicht, haben Sie in Ihrem Kopf eine Rechtfertigung (9) für Ihre Schlussfolgerungen. Warum die Schlussfolgerungen für Sie richtig erscheinen.» (Paul, 2022)

8.1 Projektmitglied 1: Funktion X

Welche Arbeiten habe ich für die Projektgruppe übernommen und realisiert?

Was ist mir gut gelungen?

Wenn es Schwierig wurde, wie habe ich die Probleme gelöst?

Wie habe ich mich beholfen? Welche Hilfestellung habe von wem erhalten?

Reflexion, Fazit, Massnahme

8.2 Projektmitglied 2

8.3 Projektmitglied 3

8.4 Projektmitglied 4

9 Anhang

9.1 Abbildungen

Abb. 1 Beschriftung

9.2 Quellenverzeichnis

Cohnen, T. (13. 03 2013). Placemat (Platzdeckchen-Methode). Rheinland Pfalz.

Diepenhorst, H. (11. 01 2020). *Teamentwicklung Lab*. Von <https://teamentwicklung-lab.de/tuckman-phasenmodell> abgerufen

Hübscher, H., Petersen, H.-J., Rathgeber, C., Richter, K., & Scharf, D. D. (2015). *IT-Handbuch* (9. Aufl.). Braunschweig: Westermann.

Oefner, M. (2013). *In 20 Schritten zum Redeprofi* (1. Aufl.). Zürich: Verlag SKV AG.

Paul, D. R. (04. 08 2022). Write: How to Teach Students to Write Well. Von https://youtu.be/YDlrN3DfZ_M abgerufen

Universität Leipzig Schreibportal - Zitationsregeln. (03. 08 2020). Von <https://home.uni-leipzig.de/schreibportal/zitationsregeln/> abgerufen

Wikipedia - Ablauforganisation. (05. 08 2020). Von <https://de.wikipedia.org/wiki/Ablauforganisation> abgerufen

Wikipedia - Projektmanagement. (04. 08 2020). Von Projektmanagement: <https://de.wikipedia.org/wiki/Projektmanagement> abgerufen

Wikipedia - Projektorganisation. (04. 08 2020). Von <https://de.wikipedia.org/wiki/Projektorganisation> abgerufen