meine Weiterbildung

TEKO

www.teko.ch

# Plantwatcher

Pflichtenheft

Linus Pauls
S-TIN-17-Do-z
19.8.2020

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# 1 Einleitung

## 1.1 Sinn und Zweck des Dokumentes

Als Diplomarbeit soll eine Webapplikation erstellt werden, die von Sensoren regelmässig und automatisiert Daten zur Wurzelfeuchtigkeit von Kübelpflanzen erhält und übersichtlich darstellt.

## 1.2 Vision (Inhalt und Ziele)

Auf meiner Terrasse stehen rund 20 teilweise grosse Kübelpflanzen. Da die Terrasse um drei Hausecken verläuft, die Pflanzen an verschiedenen Hausseiten platziert sind und nur in der einen Ecke fliessend Wasser verfügbar ist, lässt sich kein Bewässerungssystem installieren, ohne entweder grössere bauliche Aufwände (Verlegung von Wasserleitungen) oder eine optische Verschandelung durch kreuz und quer verlaufende Schläuche umzusetzen.

Ein weiters Problem ist, dass die Pflanzen teilweise sehr unterschiedliche Bedürfnisse haben, was Wassermenge und Giessfrequenz betrifft. Bisher habe ich während eines Giess-Ganges gleich alle Pflanzen gegossen, unter anderem weil man den Pflanzen nicht unbedingt ansieht, ob sie Wasser benötigen, oder noch genug Feuchtigkeit in den tieferen Erdschichten vorhanden ist. Das hat zur Folge, dass nicht alles so gut gedeiht, wie es könnte.

Ziel ist es, diesen Prozess mittels einer Webapplikation und entsprechender Sensorik zu optimieren. Es soll möglichst nahe an der Wurzel mehrmals täglich die Feuchtigkeit gemessen und mittels Wlan an einen lokalen Server übertragen werden. Dieser stellt dann die Ergebnisse in Verlaufsform dar, woraus sich gut einschätzen lässt, welche Pflanze gerade Wasser benötigt und welche nicht. So soll zielgerichtet und punktuell gegossen werden können. Idealerweise soll das System auch selbständig auf zu starke Trockenheit aufmerksam machen können.

Die Sensorik soll günstiger als gängige, auf dem Markt verfügbare Fertiglösungen, umgesetzt werden und unabhängig vom Stromnetz agieren.

## 1.3 Definitionen und Abkürzungen

ESP32 => Microcontroller von Espressif
API => application programming interface
DOCKER => Containerisierungstechnologie für Anwendungen und Dienste
DEEPSLEEP => Energiesparfunktion des Microcontrollers für extrem niedrigen Stromverbrauch.

## 1.4 Verteiler und Freigaben

| Rolle | Name | E-Mail |
|---|---|---|
| Projektbetreuung | Silvan Wirth | silvan.wirth@fhnw.ch |
| Prüfungsexperte | Pascal Häsler | pascal.haesler@cbre.com |
| Projektleitung / Ausführung | Linus Pauls | linus.pauls@gmail.com |

# 2 Konzept und Rahmenbedingungen

## 2.1 Ziele des Anbieters

Bereitstellen einer intuitiv zu bedienenden Plattform, auf der optisch ansprechend Feuchtigkeitswerte von ausgewählten Kübelpflanzen und deren Verlauf in den letzten 2-3 Tagen dargestellt werden kann.

## 2.2 Ziele und Nutzen des Anwenders

- Einfache, bequeme Benutzung und Übersicht der Applikation.
- Möglichkeit, Feuchtigkeitswerte auch von unterwegs abfragen zu können.

## 2.3 Benutzer / Zielgruppe

Dieses Projekt ist zum Eigennutzen. Grundsätzlich kann aber jeder, der die Möglichkeit hat, die Sensorik für seine Zwecke nachzubauen, diese Nutzung für sich umsetzten. Der Quellcode wird auf Github bereitgestellt.

## 2.4 System-Voraussetzungen

Ein Rechner, der die Rolle des Servers übernehmen kann, mit einem Betriebssystem, für welches Docker verfügbar ist, sowie Materialien für die Sensorik.

## 2.5 Abgrenzung

Die Umsetzung bezieht sich aus in der Einleitung genannten Gründen nur um die Messung, es wird keine automatisierte Bewässerung umgesetzt. Die Zeitspanne für die Realisierung ist zu kurz, um das fertige Design und die optimalen Bauteile der Sensorik festlegen zu können. Unter anderem auch, weil die Lieferzeiten für einige Bauteile sich in die Länge ziehen können. Es werden lediglich für reale Bedienungen nutzbare Prototypen erstellt, die den vollen Funktionsumfang bieten. Welche Mess-Sensoren oder Bauteile sich langfristig als optimal für grössere Bestellungen erweisen werden, müssen längere Tests, die über den Zeitraum dieser Arbeit hinaus gehen, zeigen.

## 2.6 Ressourcen

Die Logik für die Sensorik wird mit ESP32 Microcontrollern umgesetzt. Ausserdem stehen verschiedene elektronische Bauteile zur Energieversorgung und Messung der Feuchtigkeitswerte zu Verfügung. Für das Hosting der Applikation ist ein Linux-Server verfügbar. Weiter kann auf die Messenger-App «Telegram» und deren Dokumentation zwecks Erstellung eines Nachrichten-Bots zurückgegriffen werden. Für die Serverapplikation soll Node.js eingesetzt werden. Dies ist eine bewährte, moderne und sehr schnelle Plattform für eine Anwendung wie diese. Als Datenbank wird Mongo-DB verwendet. Das «Document-Modell» eignet sich aufgrund seiner hohen Flexibilität bestens, um die Messdatensätze als Einheit abzulegen. Bei Bedarf kann - einfacher als bei SQL-Datenbanken - nachträglich die Struktur der Datensätze angepasst werden. Durch die Funktionsweise von Mongo-DB sind auf Jahre keine Probleme oder Geschwindigkeitseinbussen zu erwarten.

### 2.7 Übersicht der Meilensteine

| | |
|---|---|
| **Vorbereitungsphase** | |
| Abgabe Pflichtenheft | 21.08.2020 |
| Beschaffung der Materialien für erste Tests | 21.08.2020 |
| Erstellung eines Development Prototypen für die Messung | 23.08.2020 |
| **Implementierung und Test** | |
| Erstellung und Verknüpfung der Dockerfiles für Node-Server und MongoDB | 23.08.2020 |
| Aufbau und Auswertung der Testmetrik für die verschiedenen Sensorik-Optionen | 24.08.2020 |
| Entwickeln der Serverapplikation inkl. API | 10.09.2020 |
| Entwicklung Frontend | 14.09.2020 |
| Fertigstellen der Software für ESP32 | 14.09.2020 |
| Testing und Bugfixing (Software) | 20.03.2020 |
| **Einführung** | |
| Sensorik-Prototyp Varianten für Langzeittests | 23.09.2020 |
| Abgabe Diplomarbeit | 24.09.2020 |
| **Nacharbeiten** | |
| Beginn Langzeittests für „produktives" Design der Sensorik-Elemente | Anschl. |

# 3 Funktionale Anforderungen

### 3.1 Use Case Übersicht

- (P1) Datenübermittlung der Sensorik an das Backend
- (P2) Aufruf der Dashboard Seite mit den Messwerten
- (P3) Verändern der Einstellungen für die Dashboard-Seite
- (P4) Abfrage der Daten von Extern über Trello-Bot
- (P5) Aktive Benachrichtigung des Users bei kritischen Messwerten durch Trello-Bot

## 3.2 Use Case Diagramm

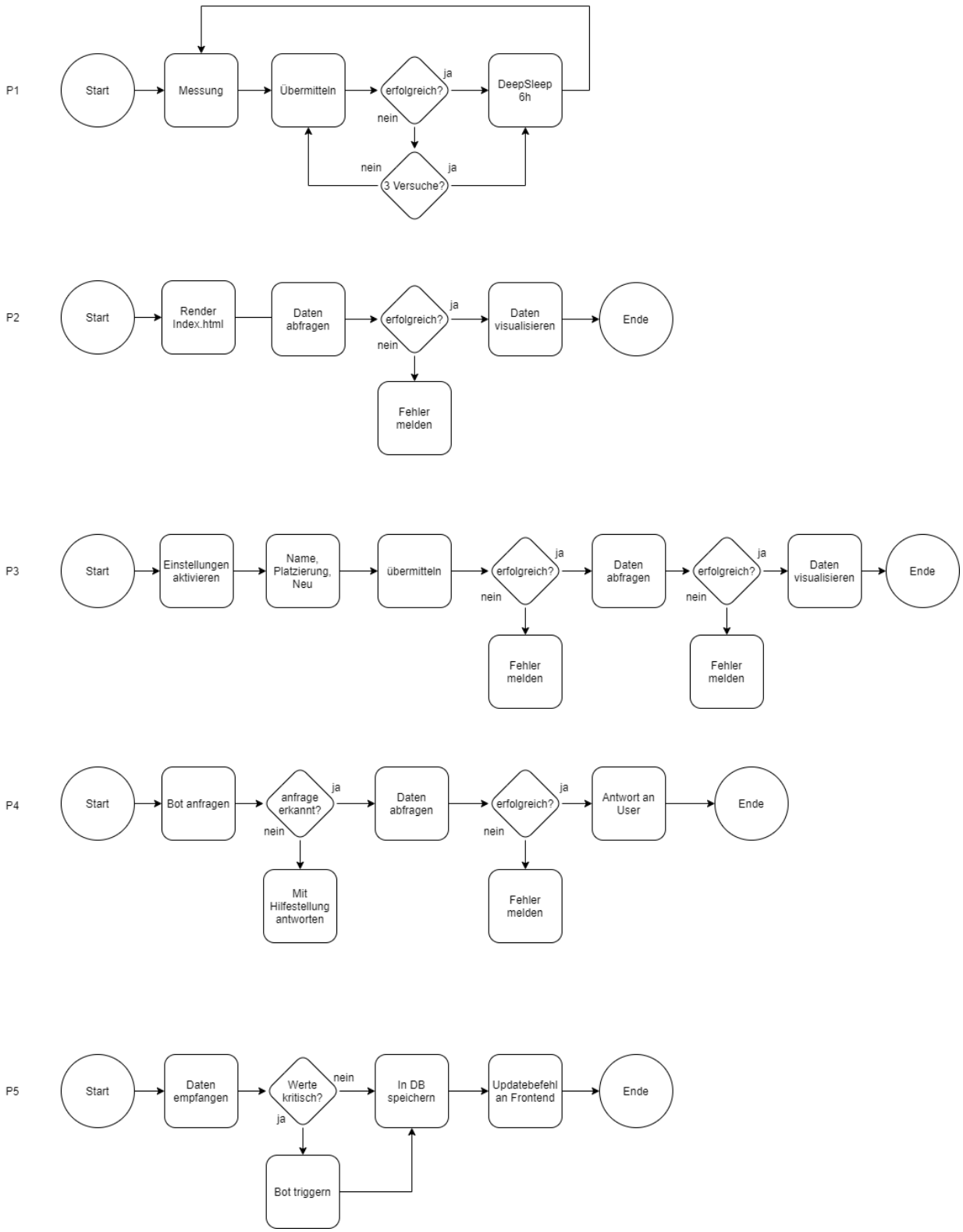


**Abbildung 1: Use Case Diagramm**

## 3.3 Use-Case Details

Bei Prozess 1 erfasst die Sensorik die Messdaten, nachdem sie aus dem Deepsleep aufwacht und sendet sie danach per «POST-Request» an die API des Backend's. Die Verbindung erfolgt über Wlan. Danach wird wieder der Deepsleep aktiviert bis zum nächsten Messzeitpunkt. Sollte die Übermittlung scheitern, wird der Vorgang noch drei Mal versucht. Klappt die Übermittlung weiterhin nicht, wird ohne Datenversand in den Deepsleep gewechselt, um nicht zu viel Energie zu verbrauchen, und beim nächsten Zeitfenster wird ein neuer Versuch gestartet.

Prozess 2 behandelt den Dashboard-Aufruf des Nutzers. Nachdem das Backend die Roh-Seite gerendert hat, werden die Daten über Ajax vom Frontend abgefragt und die Karten mit den grafischen Elementen für die Darstellung der Messwerte erzeugt.

Prozess 3 behandelt die Änderung von Einstellungen des Nutzers. Pflanzen sollen bearbeitet und neu Erstellt werden können, ohne manuell in die Datenbank eingreifen zu müssen. Dafür kann die Einstellungsfunktion aktiviert werden, worauf hin der Nutzer Änderungen vornehmen kann. Die Daten werden dann wieder per AJAX an die API des Backend übermittelt, in der Datenbank gespeichert und nach Verlassen des Settings-Modus erneut abgefragt und dargestellt.

Prozess 4 zeigt den Ablauf bei Anfrage des Nutzer über den Trello-Bot. Erkennt der Bot die Anfrage, werden die aktuellen Messdaten aus der Datenbank abgefragt und in zusammengefasster Form an den Nutzer gemeldet. Über diesen Prozess kann auch ausserhalb des Netzwerks auf die Daten zugegriffen werden.

Prozess 5 ist ein Ablauf, der ohne Useraktion startet. Wenn Messdaten von der Sensorik beim Backend eintreffen, wird vor dem Abspeicherns geprüft, ob die Feuchtigkeitswerte unter eine festgelegte Toleranz fallen. Sollte dies der Fall sein, wird der Nutzer über den Trello-Bot proaktiv informiert.

## 3.4 Risiken

Die grössten Risiken werden bei der Sensorik erwartet. Die Feuchtigkeitssensoren müssen aussagekräftige und realistische Werte liefern. Es gibt hier verschiedene Ansätze. Die Zeit könnte knapp werden, falls sich alle Sensormodelle, für die ich mich zu Beginn entscheide, als unzureichend erweisen.
Ein weiteres Problem könnte die Energieversorgung der Microcontroller werden. Ich bin kein Elektroniker und es ist möglich, dass ich hier aufgrund fehlender Erfahrung zu viel Zeit aufwenden muss. Kernthema ist eigentlich die Software, aber es könnte sein, dass sich die Hardware als grössere Knacknuss erweist.

## 3.5 Testhinweise

Es wird eine Art Labor eingerichtet, wo in mehreren Gefässen die gleiche Erde mit verschiedenen Feuchtigkeitsstufen bereit stehen. Hier können verschiedene Sensortypen über einen längeren Zeitraum messen und die Daten untereinander verglichen werden. So kann ermittelt werden, wie sensibel die Modelle sind, wo die Unterschiede liegen und was sinnvolle Toleranzgrenzen sind.

### 3.6 Grobschätzung des Aufwands

| | |
|---|---|
| **8h** | Pflichtenheft erstellen |
| **25h** | Recherche über ganzen Projektverlauf |
| **5h** | Grobskizzierung der Funktionen |
| **10h** | Aufbau Testumgebung Messwerte |
| **25h** | Einarbeiten in MongoDB und Mongoose |
| **20h** | Materialbeschaffung, Aufbau und Programmierung Mess-Prototyp |
| **30h** | Erstellen der Dockerumgebung für Development und Production |
| **40h** | Programmierung des Servers und der Backendfunktionalität |
| **10h** | Programmierung Trello-Bot |
| **30h** | Programmierung Frontend |
| **25h** | Testing und Bugfixing |
| **40h** | Materialbeschaffung, Aufbau, Programmierung definitiver Prototyp |
| **30h** | Dokumentation und Präsentation der Arbeit |

# 4 Nicht-funktionale Anforderungen

### 4.1 Benutzbarkeit

Die Applikation soll intuitiv zu bedienen sein, in Haptik und Optik soll sie gegen andere, kommerzielle Produkte nicht abfallen. Beim Nutzer soll den Eindruck einer soliden Software entstehen. Sämtliche Funktionen des Dashboards sollen auch über das Smartphone oder Tablet nutzbar sein. Datenabfragen sollen für den Nutzer möglichst unbemerkt im Hintergrund stattfinden.

### 4.2 Sicherheit

Die Applikation ist direkt nur vom internen Netzwerk erreichbar. Darum kann auch zwecks Benutzerfreundlichkeit auf Nutzer-Authentifizierung verzichtet werden. Es wird davon ausgegangen, dass jede Person im Haushalt mit Zugang zum Wlan die Applikation bedienen darf. Trotzdem soll der Server grundlegende Sicherheitsaspekte wie die Validierung der übermittelten Daten bieten; Dies aber primär, um falsche Bedienung oder Fehler in der Übermittlung abzufangen.

### 4.3 Implementierung, Prozess

Für jede neue Pflanze muss ein Sensormodul gebaut und programmiert werden. Bei der ersten Installation der Serverapplikation muss zusätzlich ein Nutzer für die Datenbank erstellt werden. Alles weitere soll durch Docker oder direkt in der Applikation erfolgen können.

### 4.4 Kosten

Für die Entwicklungsphase werden für Materialien und Werkzeug 150CHF angesetzt.
Der finale Prototyp soll unter 25CHF realisiert werden können.

# 5 Schnittstellen

## 5.1 Hardwareschnittstellen

Die Feuchtigkeits-Sensoren werden über die GPIO Schnittstelle des Microcontrollers ausgelesen. Die Energiezufuhr erfolgt über Photovoltaik oder Batterien.

## 5.2 Softwareschnittstellen

Das Backend stellt eine API für die Übermittlung und Abfrage der Daten zu Verfügung.

## 5.3 Kommunikationsschnittstellen

Die Sensorik nutzt zur Kommunikation ausschliesslich das lokale Wlan. Das Backend interagiert mit dem Trello-Bot über Internet. Der Bot ist über die Messenger-App «Trello» nutzbar.

# 6 Hardware

## 6.1 Microcontroller

Als Microcontroller wird der ESP32 verwendet. Er bietet onboard Wifi Funktionalität und kann gegenüber dem Vorgänger mehrere Stunden im Deepsleep verbringen, bei sehr niedrigem Energieaufwand (0.01mA, laut Herstellerangabe)

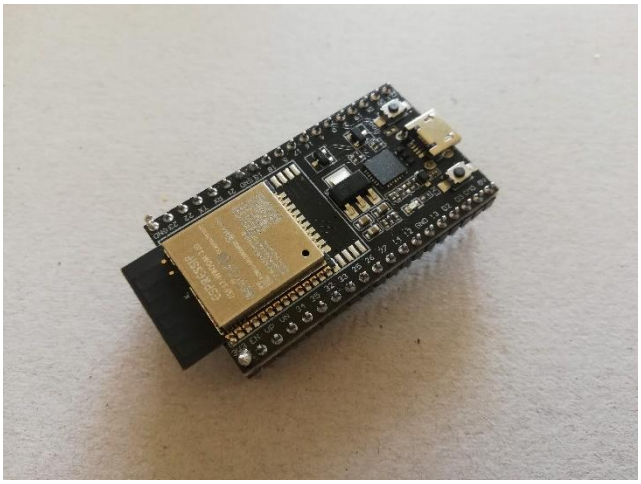

**Abbildung 2: ESP32 von Espressif**

## 6.2 Sensoren

Zur Messung der Feuchtigkeit werden verschieden Modelle, kapazitiv und resistiv getestet.

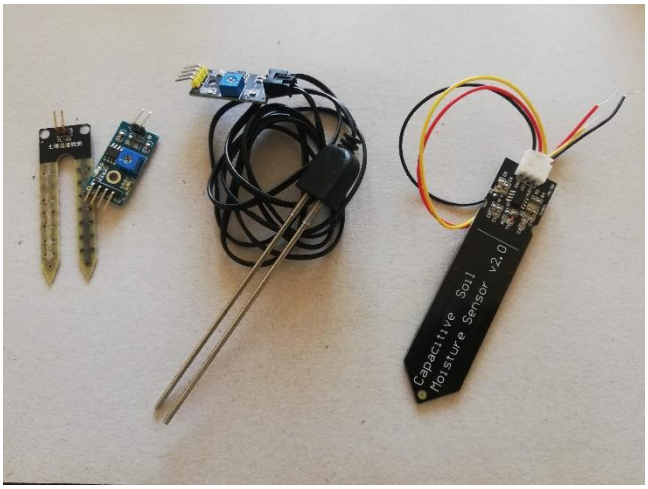


**Abbildung 3: Sensoren**

## 6.3 Energieversorgung

Es sollen verschiedene Formen der Energieversorgung geprüft werden. Eine Möglichkeit sind Lithium Batterien, sie können kurzfristig genügend Strom liefern, um den ca. 120-180mA-Bedarf beim Senden über Wlan zu decken. Eine andere Variante sind Photovoltaik-Zellen mit Superkondensator als Energiespeicher. Der Vorteil wäre hier klar, dass keine Batterien ausgetauscht werden müssen. Andererseits muss genügend Lichteinstrahlung vorhanden sein, um den Kondensator zu laden, und die Schaltung wird aufwendiger.

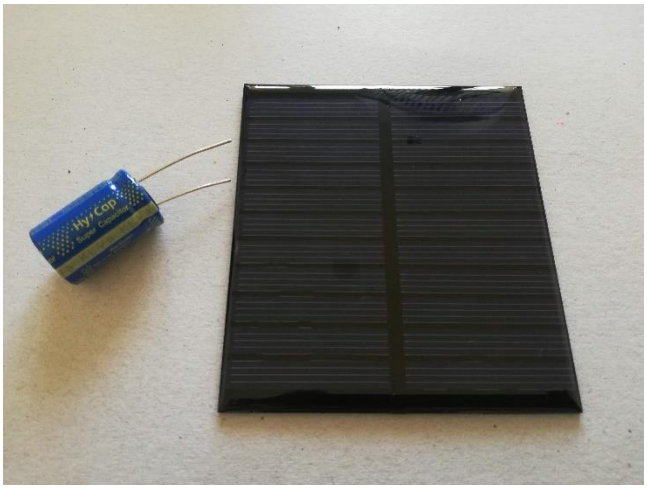


**Abbildung 5: Solarzelle und Kondensator**


**Abbildung 4: Lithium Batterien**

# 7 Interface

## 7.1 Main/Dashboard

Dies sind Mokups zur UI-Funktionalität. Ziel ist es, lediglich die Funktionsweise zu skizzieren. Genaue Umsetzung und Inhalt können variieren.

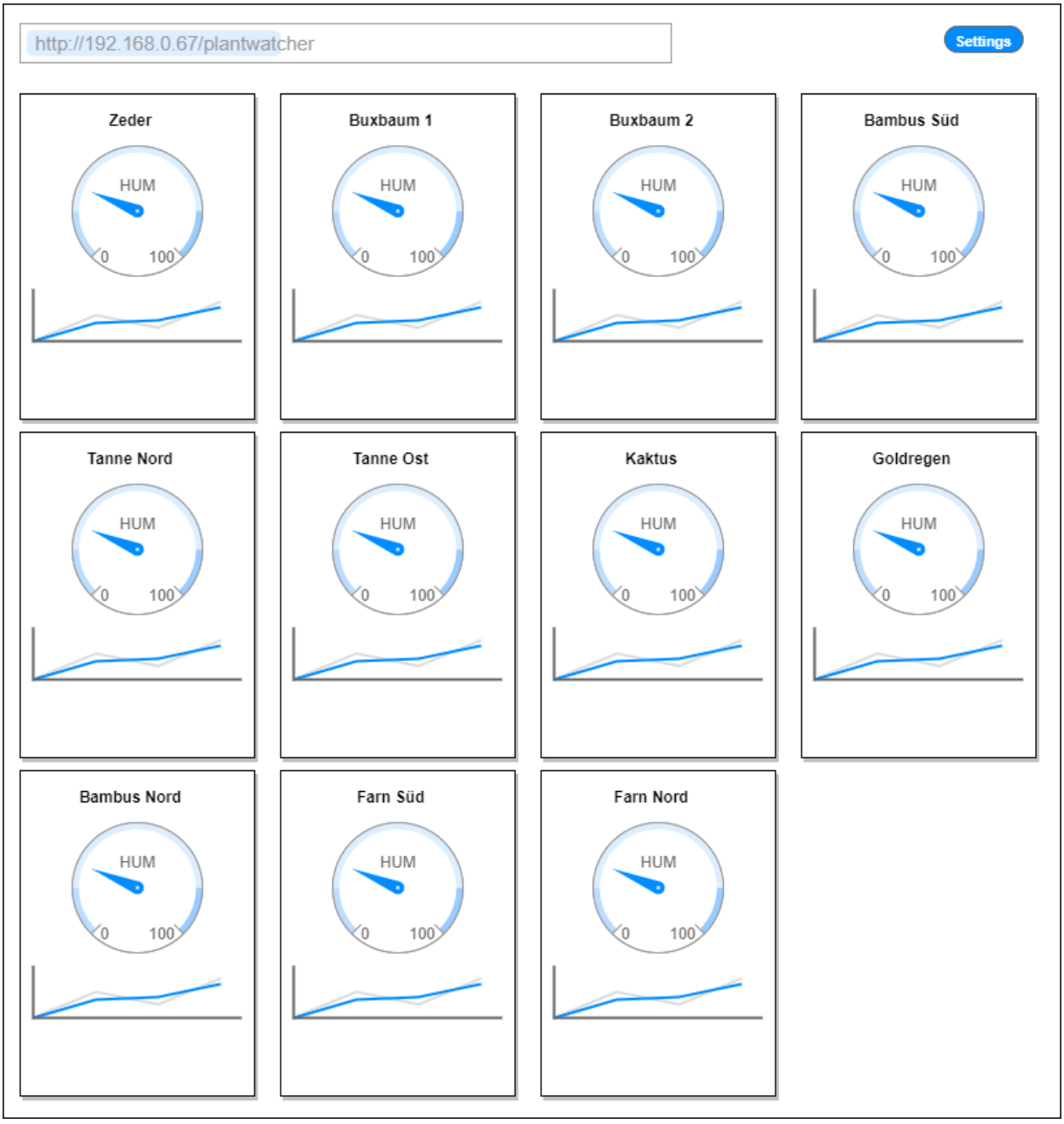


**Abbildung 6: Mokup Dashboard**

## 7.2 Settings

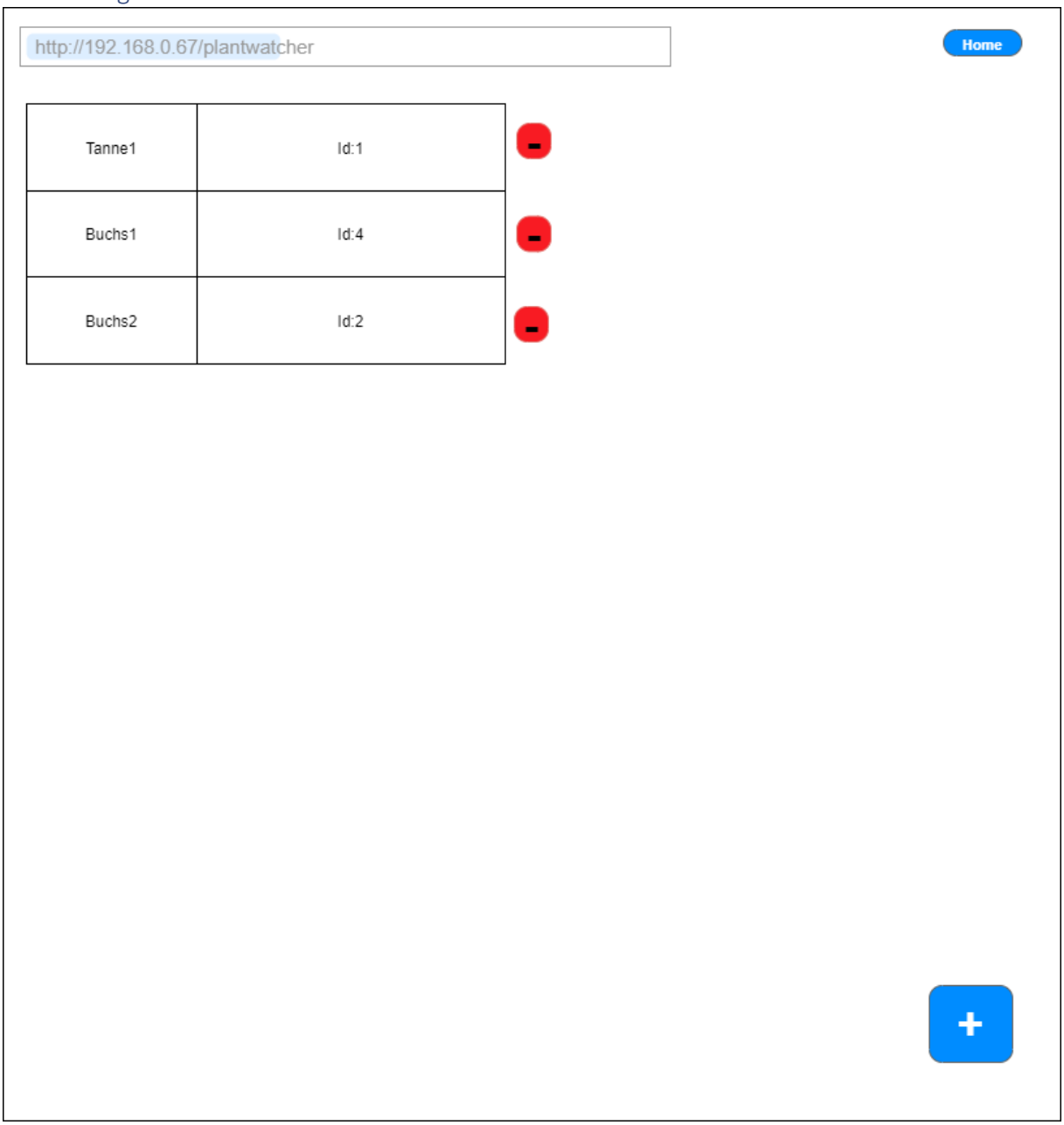


**Abbildung 7: Mokup Settings**

# 8 Zielübersicht

| Zielkatalog | | |
|---|---|---|
| **Nodeserver für Applikationshosting erstellt** | Technik | (M) |
| **Komplette Umgebung in Dockercontainern** | Technik | (K) |
| **Nutzer kann Pflanzen umnennen, erfassen und sortieren** | Funktion | (M) |
| **Backend erhält Messdaten via Wlan** | Funktion | (M) |
| **Applikation ist mobile-friendly** | Funktion | (M) |
| **Die Sensorik agiert unabhängig vom Stromnetz und ist witterungsbeständig** | Technik | (M) |
| **Daten können auch über Trello-Bot abgerufen werden** | Funktion | (K) |
| **Statt CSS wird SCSS verwendet** | Technik | (K) |
| **Das Frontend aktualisiert sich ohne Reload bei Updates** | Technik | (K) |
| **Die statischen Dateien, CSS, Bilder, Javascript werden über «Webpack» bereit gestellt** | Technik | (K) |