Wetterstation

# Problemstellung

### Deine Rolle

Du bist Webentwickler bei der Firma Awesome WebDesign GmbH in Imst.

### Die Situation

Ein Hobbygärtner möchte für seine Pflanzenzucht eine Wetterstation mit einem Raspberry Pi und einem Webserver aufbauen. Er hat dazu bereits den RPI konfiguriert und für die Visualisierung einen Wireframe gezeichnet.

### Dein Ziel

Dein Chef hat dich beauftragt einen Prototyp zur Speicherung und Visualisierung der Messdaten zu erstellen. Die Daten werden vom Raspberry Pi viertelstündlich über eine REST-Schnittstelle in die Datenbank gespeichert. Auf der Website sollen diese in Tabellenform und als Liniendiagramm dargestellt werden. Die Anwendung soll mit mehreren Messstationen funktionieren.

### Deine Zielgruppe

Die Applikation soll vom Kunden über einem herkömmlichen Browser verwendet werden können. Die Schnittstelle wird vom Raspberry Pi verwendet, könnte jedoch später auch für eine App verwendet werden.

### Das erwartete Produkt

#### Funktionen

Der Prototyp sollte mindestens folgendes umfassen:

* Startseite
  + Auswahl der Messtation, anschließend Laden der Daten via AJAX
  + Anzeige von Messdaten in Tabellenform und Diagramm mittels Chart.js
* CRUD-GUI für Messstationen
* RUD-GUI für Messwerte
* REST-Schnittstelle für die Messstationen im JSON-Format
  + Name
  + Höhe
  + Ort (Koordinaten)
* REST-Schnittstelle für die Messdaten im JSON-Format
  + Messstation (Fremdschlüssel)
  + Messzeitpunkt (Typ: Timestamp)
  + Temperatur
  + Regenmenge

Zusatzfunktionen des Prototyps:

* Einbindung eines Wetterdienstes (AJAX) und Anzeige der Wetterprognose für den Ort, z.B. <http://openweathermap.org/API>
* „Sicherung“ der Schnittstelle vor unberechtigtem Zugriff mittels KEY,

z.B. [http://localhost/php42/api/measurement/day/2015-12-30&key=12345](http://localhost/php42/api/weather/day/2015-12-30&key=12345)

* Exportfunktion für die Wetterdaten als CSV-Datei
* Darstellung der Messstationen auf einer Karte

# Funktionen der REST-Schnittstelle:

## Einzelne Messstation auslesen (per Messstation- ID)

GET http://localhost/php42/api/station/1

Ergebnis:

{

    "id": 1,

    "name": "HAK Imst Garten",

    "altitude": 827,

    "location": "47.237156,10.739729"

}

## Alle Messstationen auslesen

GET <http://localhost/php42/api/station>

Ergebnis:

[

    {

        "id": 1,

        "name": "HAK Imst Garten",

        "altitude": 827,

        "location": "47.237156,10.739729"

    },

    {

        "id": 2,

        "name": "Karröster Alm",

        "altitude": 1468,

        "location": "47.24817087128975, 10.778006955803807"

    }

]

## Alle Messwerte der Messstation auslesen (per Messstation- ID)

GET http://localhost/php42/api/station/1/measurement

Ergebnis:

[

    {

        "id": 1,

        "time": "2020-03-15 15:45:00",

        "temperature": 0.7,

        "rain": 0,

        "station\_id": 1

    },

    {

        "id": 2,

        "time": "2020-03-15 16:00:00",

        "temperature": -0.1,

        "rain": 0,

        "station\_id": 1

    },

…

]

## Neue Messtation eintragen

POST http://localhost/php42/api/station  
Content-Type: application/json

{"name":"HAK IMST Wetterstation","altitude":870, "location":"47.237156,10.739729"}

## Messtation aktualisieren

PUT http://localhost/php42/api/station/4

Content-Type: application/json

{"name":"HAK IMST Wetterstation","altitude":870, "location":"47.237156,10.739729"}

## Messstation löschen

DELETE http://localhost/php42/api/station/3

## Einzelnen Messwert (inkl. Station) auslesen (per Messwert- ID)

GET http://localhost/php42/api/measurement/1

Ergebnis:

{

    "id": 1,

    "time": "2020-03-15 15:45:00",

    "temperature": 0.7,

    "rain": 0,

 "station\_id": 1,

  "station": {

    "id": 1,

     "name": "HAK Imst Garten",

      "altitude": 827,

    "location": "47.237156,10.739729"

  }

}

## Alle Messwerte (aller Stationen sortiert nach Zeitpunkt) auslesen

GET http://localhost/php42/api/measurement

Ergebnis:

[

    {

        "id": 1,

        "time": "2020-03-15 15:45:00",

        "temperature": 0.7,

        "rain": 0,

        "station\_id": 1

    },

    {

        "id": 2047,

        "time": "2020-03-15 15:45:00",

        "temperature": -3.64,

        "rain": 0.7,

        "station\_id": 2

    },

…

]

## Neuen Messwert eintragen

POST http://localhost/php42/api/measurement  
Content-Type: application/json

{"time":"2021-06-01 00:08:00","temperature":"17.58","rain":"0.1","station\_id":"2"}

## Messwert aktualisieren

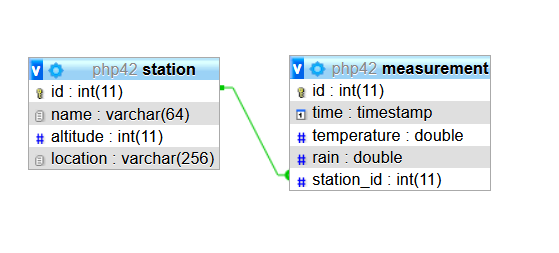
PUT http://localhost/php42/api/measurement/4  
Content-Type: application/json

{"time":"2021-06-01 00:08:00","temperature":"17.58","rain":"0.1","station\_id":"2"}

## Messwert löschen

DELETE http://localhost/php42/api/measurement/3

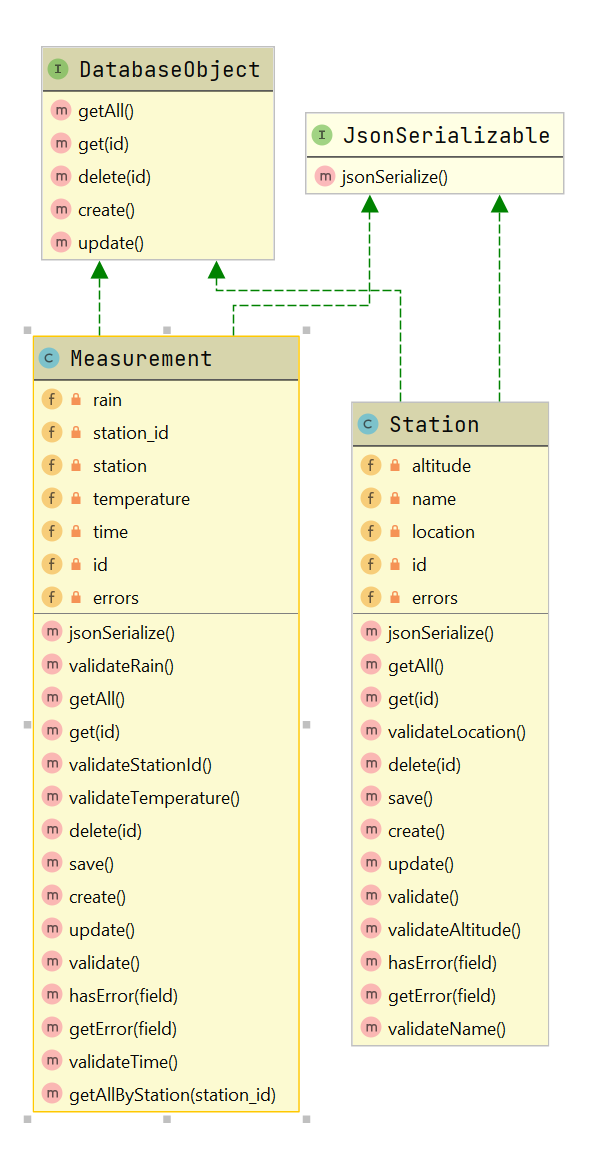
# Entity-Relationship-Diagram / Datenbankschema



# Business- / Modellklassen

## Klasse Measurement:

**getAll ()** liefert ein Array aller Messwerte (Array an Objekten)

**getAllByStation($id)** liefert ein Array aller Messwerte einer Station (Array an Objekten)

**get($id)** liefert einen Messwert (als Objekt)   
inkl. verknüpftem Stations-Objekt

UML-Klassendiagramm

# Mockup / Entwürfe



