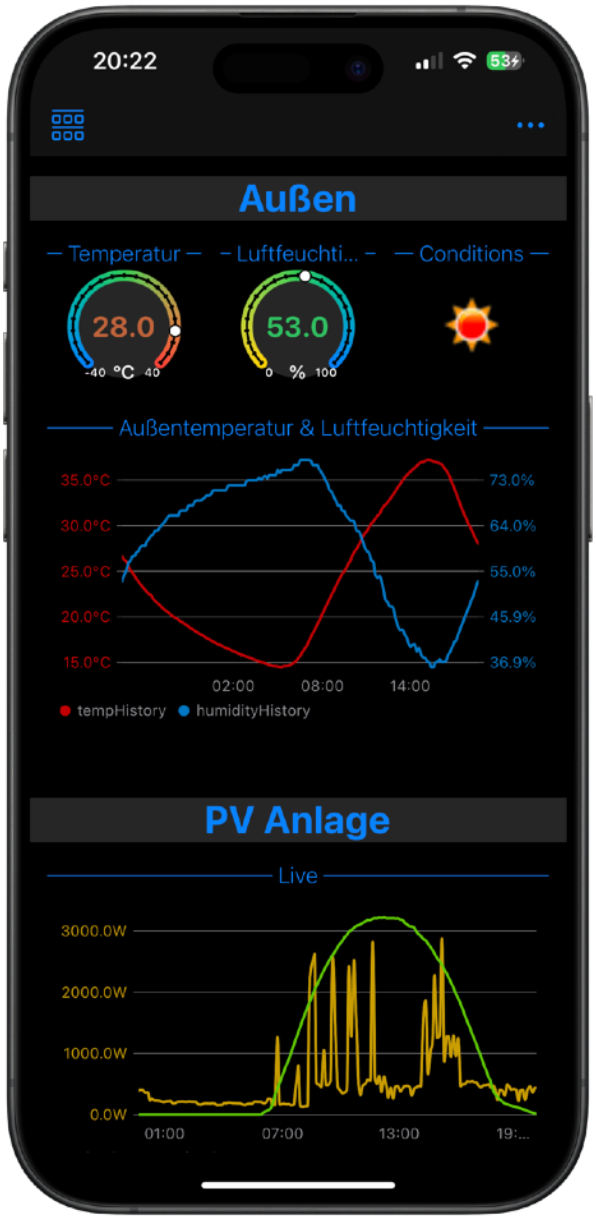
Visual V2.00

SmartHome App für iPhone & iPad



Datenschutzerklärung

Erhobene Daten

Folgende Daten können vom Nutzer in Visual konfiguriert werden um die Funktion der App zu ermöglichen:

- Konfigurationsdaten zu den externen Geräten und Web-Services (z.B. Verbindungsdaten, Benutzernamen, Passwörter)

Diese Daten werden nur auf dem iOS Gerät gespeichert bzw. über dem iCloud Account des Nutzers zwischen iOS Geräten synchronisiert. Die Daten werden weder an den Entwickler noch an Dritte weitergegeben.

Analytics (ab V1.6)

Visual erfasst ab V1.6 optional die Verwendung der App. Hierbei werden **nach Zustimmung des Nutzers** folgende Daten anonym erfasst:

- Verwendete Endpunkte (Anzahl und Typ)
- Verwendete Widgets (Anzahl und Typ)

Es werden keinerlei persönliche Daten erfasst und die Zustimmung kann jederzeit in den App-Einstellungen entzogen (oder gegeben) werden.

Personenbezogene Daten

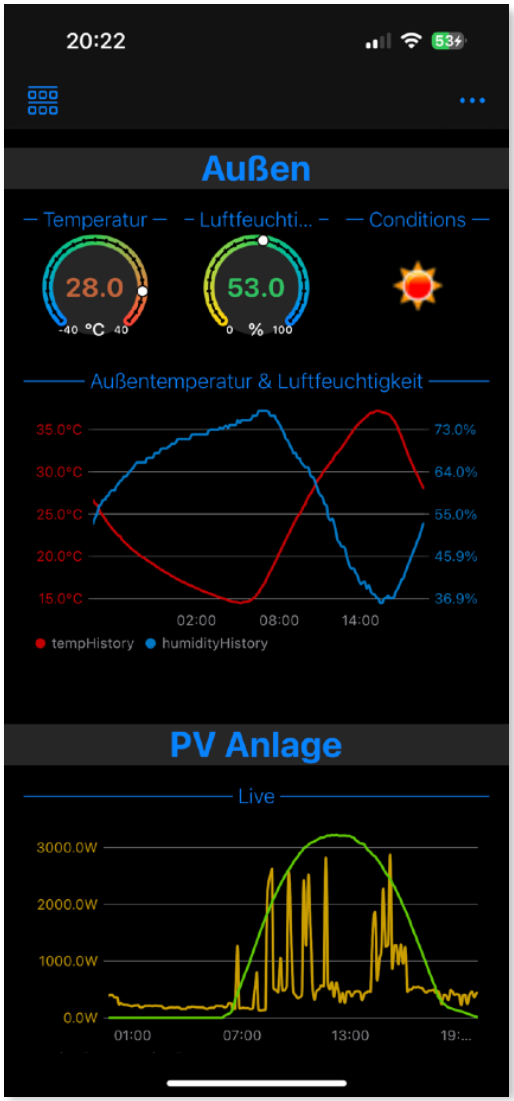
Personenbezogene Daten (z. B. Name, E-Mail-Adresse, Telefonnummer) werden nur dann erhoben, gespeichert und verarbeitet, wenn Sie dem Entwickler diese Daten durch eine explizite E-Mail Anfrage zur Verfügung gestellt werden.
Der Entwickler nutzt diese Daten ausschließlich für die Erfüllung und Abwicklung Ihrer Anfrage oder zur Übermittlung von damit in direktem Zusammenhang stehenden Informationen. Personenbezogenen Daten werde nie an Dritte weitergegeben.

Ansprechpartner

Fragen richten Sie bitte direkt an den Entwickler der App: me@andreas-binner.de

Grundprinzip

Überblick

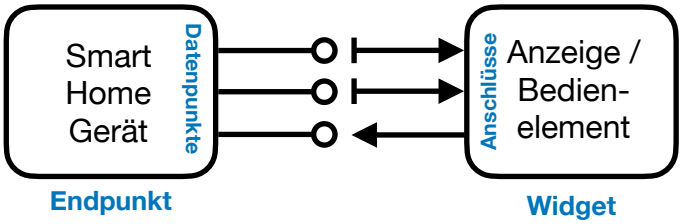


Visual verwaltet ein sog. "*Dashboard*" welches man auch direkt nach dem Start der App sehen kann. Das Dashboard ist eine scrollende Liste von "*Widgets*". Ein Widget kann eine reine Anzeige oder aber auch ein Bedienelement wie zum Beispiel ein Schalter sein.

Das Gegenstück zu den Widgets sind die "*Endpunkte*". Ein Endpunkt repräsentiert eine externe Datenquelle wie z.B. eine SmartHome-Gerät oder einen Web-Dienst. Alle Endpunkte haben die Gemeinsamkeit, dass sie über das Netzwerk erreichbar sind.

Widgets haben *Anschlüsse* und Endpunkte haben einen *Datenpunkte*. Ein Datenpunkt repräsentiert einen dedizierten Datenkanal. Datenpunkt haben eine Richtung ("nur lesen" bzw. "lesen/schreiben") und eine Typ (Bool, Integer, Kommazahl, Prozent, Datenreihe, Farbe, ...). So hat ein Wettersensor u.U. je einen Datenpunkt für Temperatur und Luftfeuchtigkeit ("nur lesen" und Typ "Kommazahl").

Ein Widget wiederum kann ebenfalls genau einen Anschluß haben (z.B. eine einfache Zeigeranzeige) oder auch mehrere Anschlüsse für mehrere Datenreihen (z.B. ein Liniendiagramm). Auch Bedienelemente können mehrere Anschlüsse haben ("lesen/schreiben"), um einen



Wert in unterschiedlichen Repräsentationen auszugeben (z.B. der Farbwähler für RGB und HSV Farbraum) oder einfach um einen Wert am mehrere Endpunkte zu schicken.
Während der Konfiguration werden die Anschlüsse eines Widgets mit Datenpunkten von Endpunkten verknüpft.

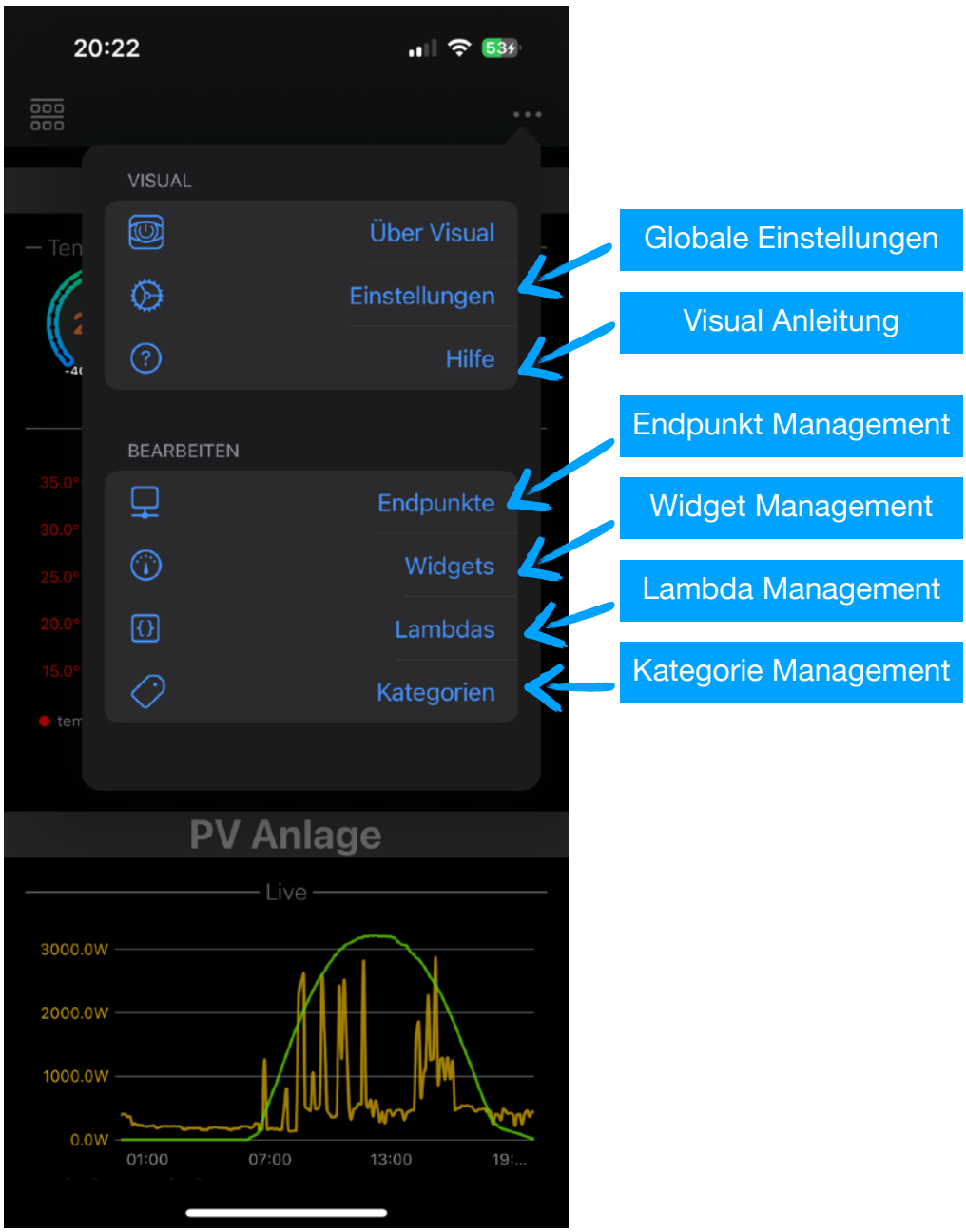
Das Dashboard

Das Dashboard ist die zentrale Ansicht in Visual. Von hier aus erreicht man über die Toolbar am unteren Rand auch alle wichtigen anderen Ansichten:

- In Editiermodus wechseln
- Hauptmenü

Hauptmenü

Über das Hauptmenü erreicht man alle wichtigen Verwaltungs- und Einstellungsmenüs sowie das Benutzerhandbuch (dieser Text). Alle Einstellungen werden im Folgenden im Detail erklärt.



Raster

Im Dashboard werden die konfigurierten Widgets in einem festen Raster angezeigt. Widgets können eine Größe bis 3 Einheiten Breite und 2 Einheiten Höhe annehmen. Folgende 6 Größen stehen (je nach Widget-Typ) zur Verfügung:

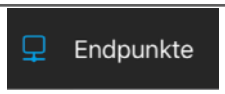
Ausrichtung

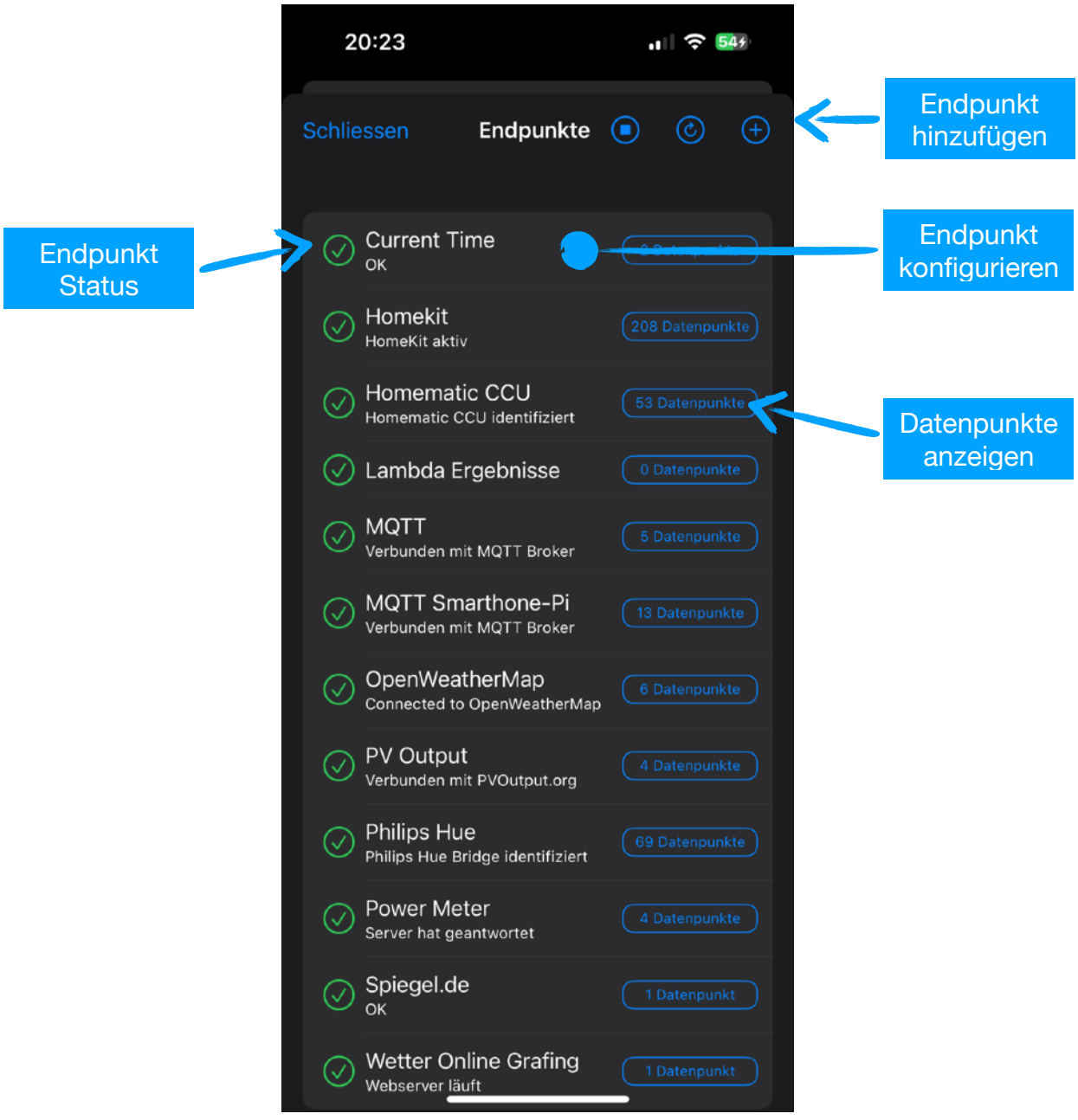
Das Dashboard passt sich automatisch an die Bildschirmgröße- und Ausrichtung an.
Dabei ist die von Apple definierte "Größenklasse" ausschlaggebend. So haben alle iOS Geräte im Hochformat die Größenklasse "Normal". Geräte im Querformat sind entweder "Kompakt" (iPhones) oder "Normal" (iPad, iPhone Plus).

In der Klasse "Kompakt" hat das Dashboard eine Breite von 3 Einheiten. In der Klasse "Normal" verdoppelt sich die Anzahl auf 6 Einheiten.

Visual - Schritt für Schritt

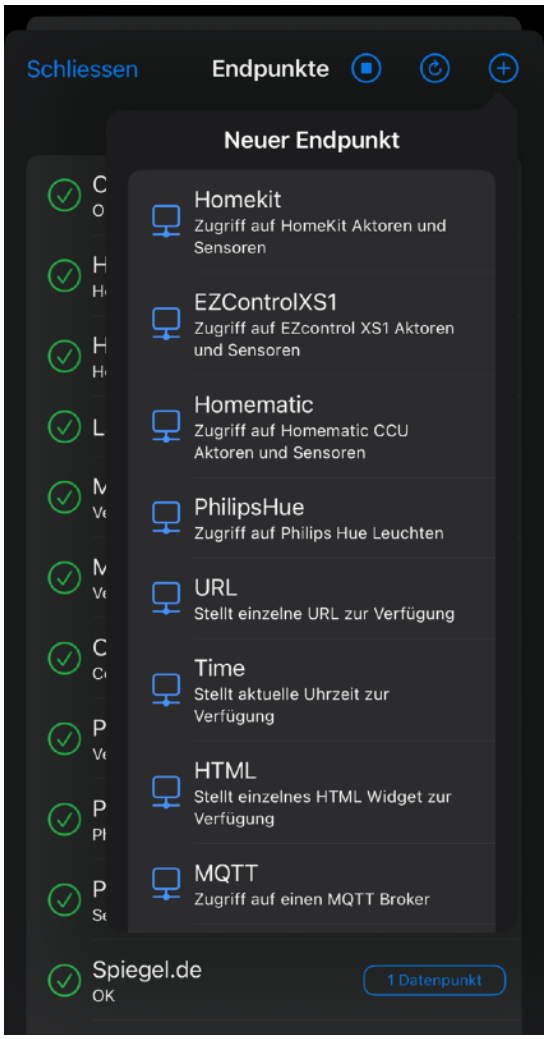
Endpunkt anlegen

Hierzu im Hauptmenü "Endpunkte" wählen: 



Einen Endpunkt fügt man dann einfach hinzu indem man auf den "+" Button tippt und dann den entsprechenden Eintrag in der Liste auswählt. Den Auswahldialog kann man auch ohne Hinzufügen schließen, in dem man außerhalb des Dialogs antippt.

Derzeit unterstützt Visual die folgenden Endpunkte:



Homekit	Zugriff Homekit Geräte und Services
Homematic	Zugriff auf Geräte via Homematic CCU
Philips Hue	Zugriff auf Philips Hue gesteuerte Geräte
URL	Statische URL (im Kombination mit HTML Widget)
Uhrzeit	Liefert aktuelle Uhrzeit
OpenWeatherMap	Zugriff auf OpenWeatherMap Daten
HTML Widget	Stellt HTML Widget zur Verfügung
MQTT Client	Generischer Zugriff auf MQTT Broker (JSON Payload)
HTTP Client	Generischer Zugriff auf HTTP Server (JSON Payload)
UDP Client	Generic UDP Client zum Senden von Daten (Binary, JSON)

Eine detaillierte Beschreibung zu den einzelnen Typen findet sich im Kapitel "Endpunkte".

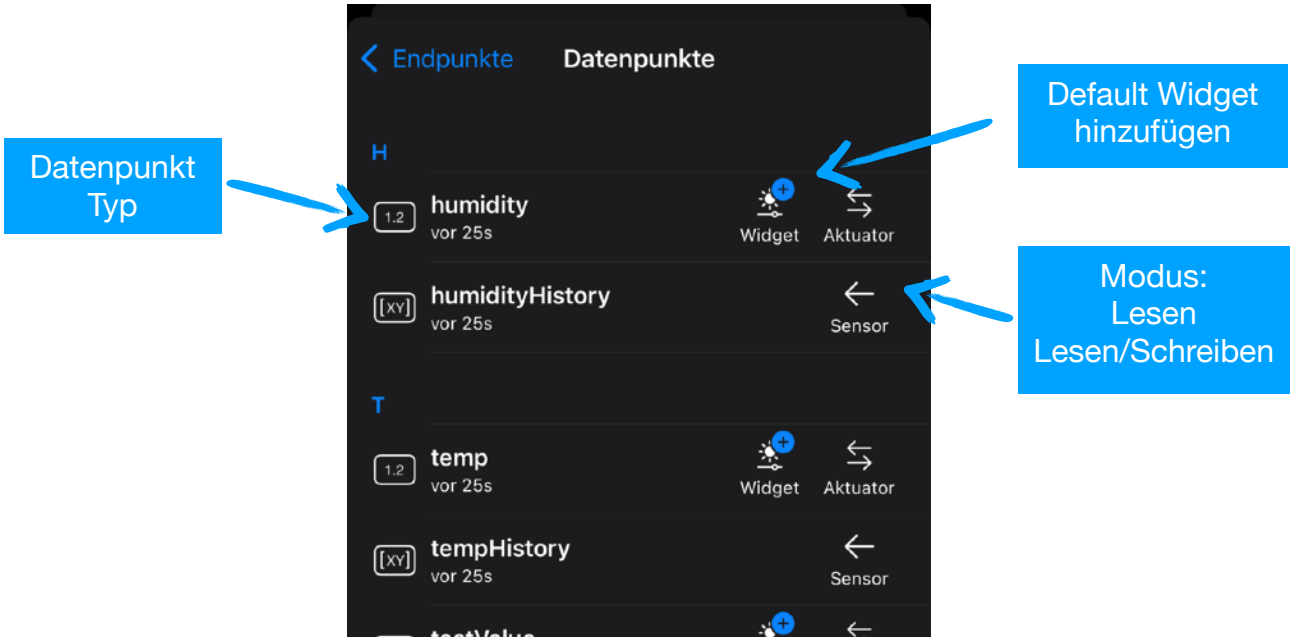
Endpunkte konfigurieren

Um einen Endpunkt zu konfigurieren, einfach auf das Schieberegler Icon des entsprechenden Eintrags tippen.
Je nach Endpunkt-Typ gibt es unterschiedliche Konfigurationsparameter (siehe Kapitel "Endpunkte").
Alle Endpunkte haben aber den Parameter "Name" um den Listeneintrag einen eindeutigen Namen zu geben.

Wichtig: Für alle Endpunkte, die mit einem externen Gerät oder Dienst kommunizieren, ist es mindestens nötig eine URL oder IP-Adresse anzugeben.

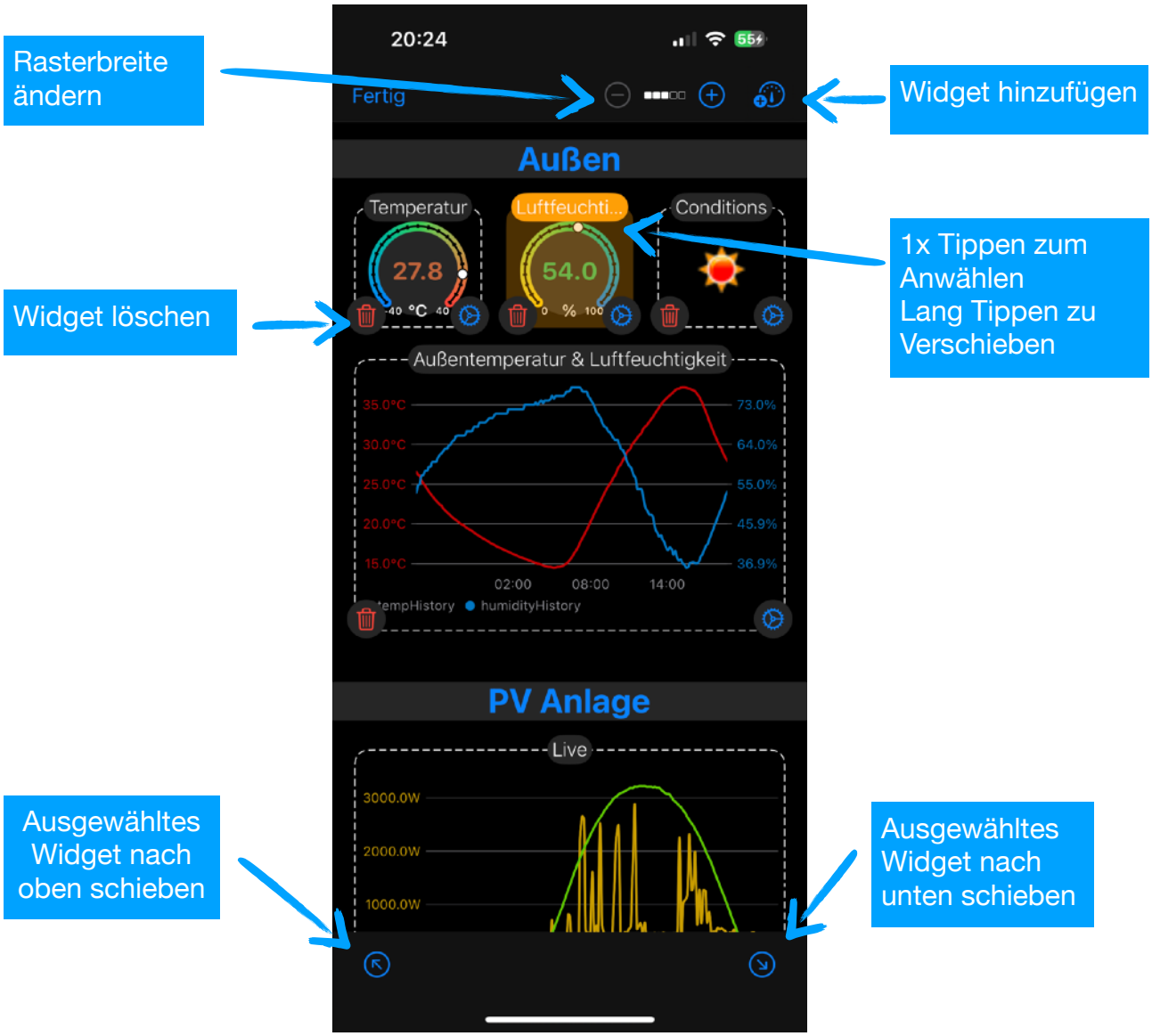
Datenpunkte

Nachdem ein Endpunkt eine Verbindung hergestellt hat, kann man die Datenpunkte des Endpunkts anzeigen. Dazu einfach auf den Datenpunkte Button tippen.



Widgets hinzufügen

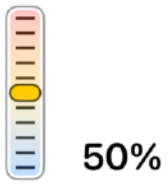
Widgets fügt man im "Editiermodus" an. Dazu auf das Icon links, unten in der Toolbar antippen. Im "Editiermodus" wackeln alle Widgets!

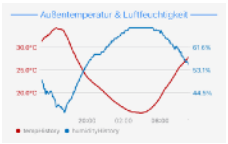


Nun kann man auf das "Widget hinzufügen" Icon in der jeweiligen Kategorie tippen. Im erscheinenden Dialog dann auf den entsprechenden Eintrag in der Liste tippen. Den Dialog kann man auch ohne Hinzufügen schließen, in dem man außerhalb des Dialogs antippt.

Hinweis: In einem leeren Dashboard erscheint das "Widget hinzufügen" Icon unten in der Werkzeugleiste!

Folgende Widget Typen stehen zur Verfügung:

Zeigeranzeige	Anzeige eines einzelnen Werts als Zeigerdiagram (mit Einheit)	
Level	Anzeige eines einzelnen Werts als Fortschrittsbalken	
Statusanzeige	Anzeige eines binären Zustands (z.B. 0/1 oder an/aus) oder eines Farbwerts	
Text	Anzeige einer einzelner Textzeile	Hallo
WebView	Anzeige von HTML Inhalten (z.B. ein HTML Widget oder einen externe Internetseite)	
Auswahl	Auswahl einer einzelnen Option aus einer Liste	Rot Grün Blau
Schalter	Einfacher An/Aus Schalter	
Tastenmatrix	Anzeige von (bis zu 6) zustandslosen Tastern	1 2 3 4
Regler	Horizontaler Schieberegler	0 100 50%

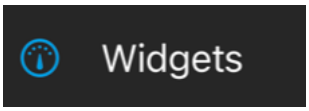
Farbregler	Farbanzeige und -wähler	
Taster	Zustandsloser Taster	
Bild	Anzeige eines Bilds (als Daten empfangen oder von URL geladen)	
Liniendiagramm	Anzeige von (bis zu 8) Datenreihen. **Hinweis:** Die Anschlüsse 1-4 und 5-8 teilen sich jeweils eine y-Achse, d.h. es werden zwei unabhängige Wertebereiche unterstützt.	

Widget-Reihenfolge innerhalb einer Kategorie beeinflussen

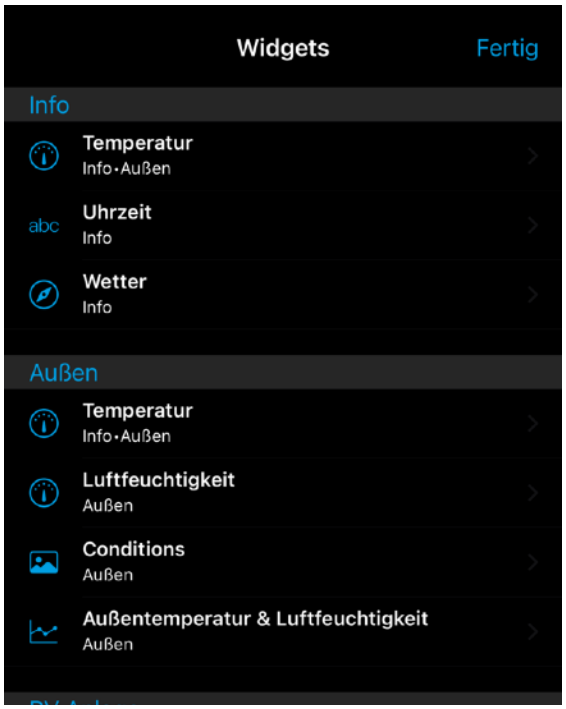
Dazu Widget auswählen indem man einmal auf das Widget tippt. Nun kann man mit den Pfeil-Tasten (unten in der Werkzeugleiste) die Position des Widgets verändern. Wichtig: Visual versucht die Widgets innerhalb einer Kategorie immer möglichst kompakt anzuordnen (mit wenigen Lücken). Daher lässt sich die Position nicht völlig frei wählen!

Widget Verwaltung

Alternativ zum Dashboard Editiermodus, kann man die Widgets auch über die Widget-Verwaltung bearbeiten. Hierzu im Hauptmenü auf "Widgets" tippen.



Hier sind alle Widgets in einer einfachen Liste zu sehen. Die Reihenfolge der Widgets kann einfach per Drag&Drop verändert werden - auch zwischen Kategorien kann verschoben werden.



Durch Antippen eines Listeneintrags öffnen sich die Einstellung für das Widget.

Widget konfigurieren

Um ein Widget zu konfigurieren gibt es zwei Wege:

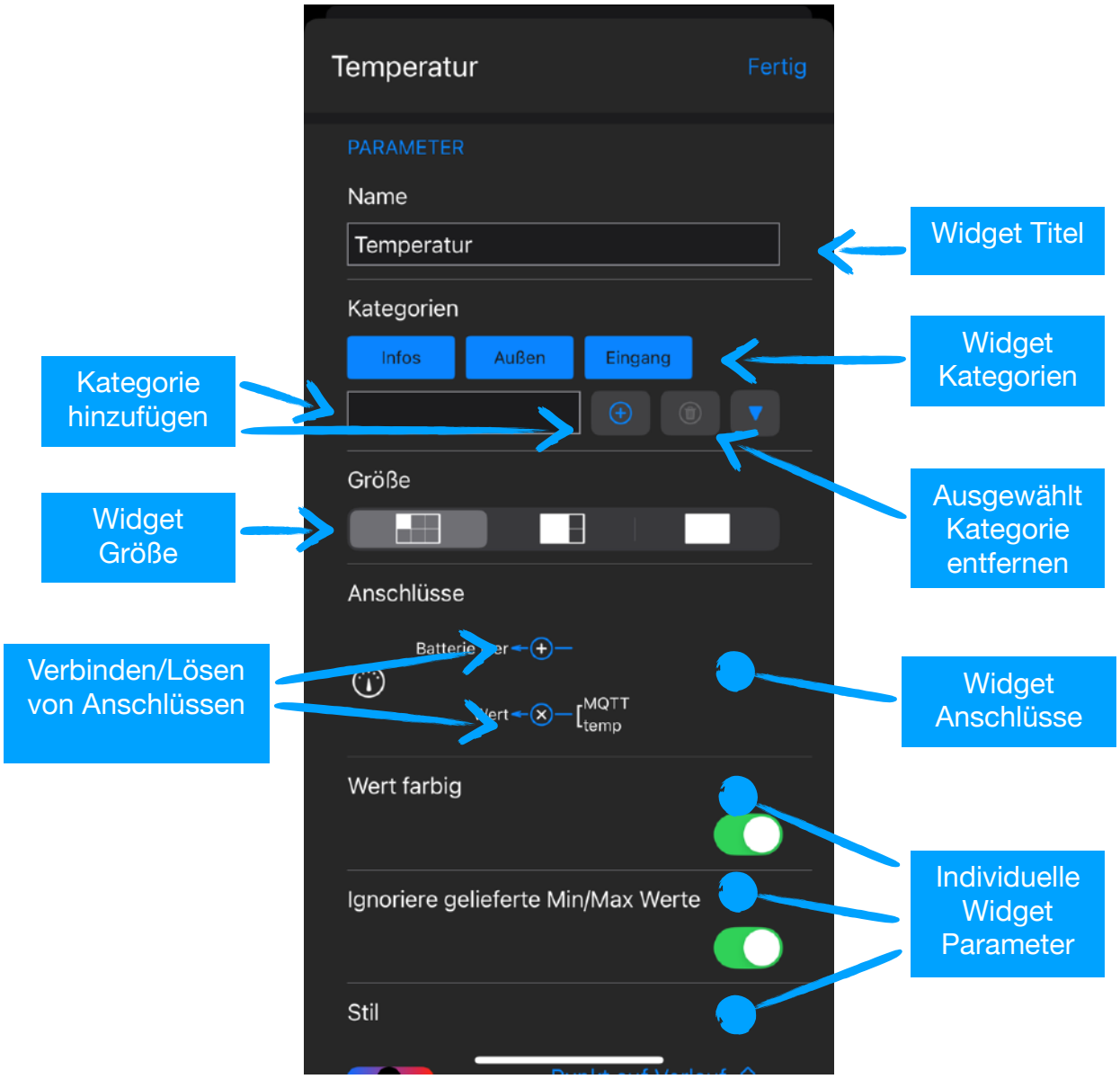
1. Im Editiermodus des Dashboard
   Tippen auf den "Zahnrad" Button
2. Im Hauptmenü "Widgets" auswählen
   Auf die entsprechende Widget-Zeile tippen

Folgende Parameter gibt es bei allen Widget-Typen:

Name	Der Titel des Widgets
Size	Ein Widget kann eine von 6 Größen annehmen (nicht alle Größen sind bei allen Widgets erlaubt!) 

Categories	Widgets können einer oder mehreren Kategorien zugeordnet sein. Widgets tauchen im Dashboard in allen zugeordneten Kategorien auf. **Achtung: Ein Widget das keiner Kategorie mehr angehört wird gelöscht!**
Ports	Hier sind alle Anschlüsse des Widgets gelistet und es kann eine Verknüpfung zu einem Endpunkt/Datenpunkt hergestellt (oder gelöscht)

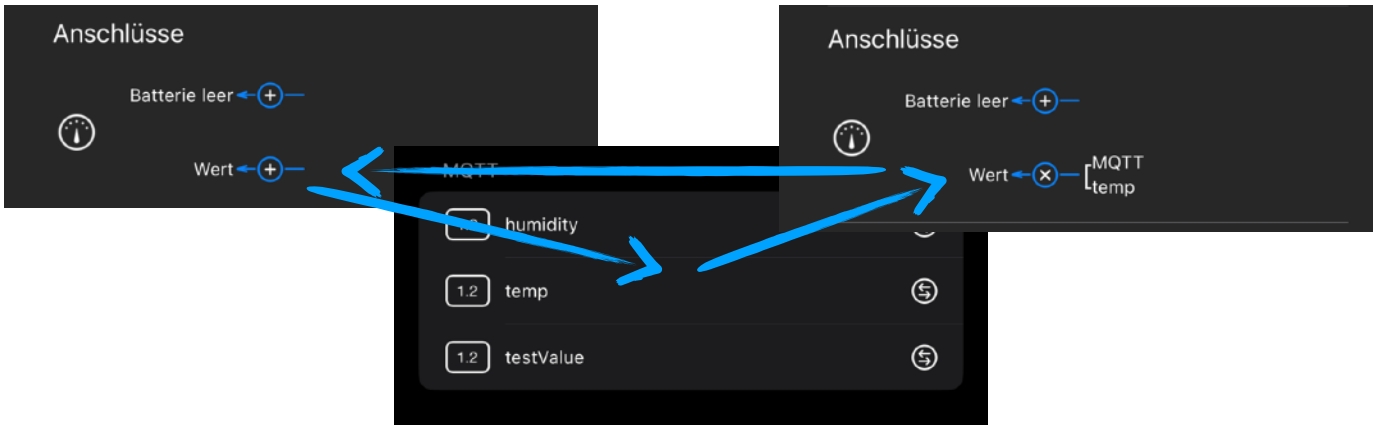
Hinweis: Nur kompatible Datenpunkte werden zur Auswahl angeboten!



Port Connections

Auf ⊕ tippen um eine Anschluß mit einem Datenpunkt zu verbinden.

Hinweis: Nur kompatible Datenpunkte werden zur Auswahl angeboten!
Auf ⊗ tippen um die Verbindung zu lösen.



Individuelle Widget Parameter

Jeder Widget-Typ hat noch spezielle Parameter mit denen man das Aussehen und Verhalten beeinflussen kann.

Zeigeranzeige	Ignoriere gelieferte Min/Max Werte	**An**: Standardwerte (abgeleitet aus der Einheit) für Maximal- und Minimalwert verwendet **Aus**: Es wird der "Min" und "Max" Wert des Datenpunkts verwendet
Level	Anzeigetyp	• Normal: Grauer Balken • Grün➔Rot: Verlauf von Grün nach Rot • Rot➔Grün: Verlauf von Rot nach Grün • Peak Rot: Oberhalb von 80% Rot
Statusanzeige	Aus Text	Text der im "Aus" Zustand angezeigt wird**
	An Text	Text der im "An" Zustand angezeigt wird**
	An Farbe	Farbe im "An" Zustand (wird ignoriert wenn der Anschluß "Farbwert" verwendet wird!)
Text	Schriftgröße	Schriftgröße in Punkt

	Monospace Schrift verwenden	Es wird die nichtproportionaler Systemschrift verwendet
Auswahl	Auswahlliste	Liste der möglichen Werte *Hinweis: Je nach Anschlußtyp wird ein Auswahlindex (0...5) oder der umgewandelte Wert aus der Beschriftung (z.B. 18°C -> 18.0) gesendet.*
Schalter	Aus Text	Text der im "Aus" Zustand angezeigt wird**
	An Text	Text der im "An" Zustand angezeigt wird**
Tastenmatrix	Auswahlliste	Liste der Beschriftung der Tasten. *Hinweis: Je nach Anschlußtyp wird ein Tastenindex (0...5) oder der umgewandelte Wert aus der Beschriftung (z.B. 18°C -> 18.0) gesendet.*
Regler	Verzögerung in ms	Minimale Zeit zwischen zwei gesendeten neuen Werten
	Werte anzeigen	Aktueller Wert wird angezeigt
	Anzahl der Unterteilungen	Schrittweite des Reglers
Farbregler	Verzögerung in ms	Minimale Zeit zwischen zwei gesendeten neuen Werten
Taster	Wert zum Senden	Wert der beim Auslösen des Taster gesendet wird: 0/0.0/0% oder 1/1.0/100% (je nach Anschlußtyp)
	Tastertext	Text der im Taster angezeigt wird**
Liniendiagramm	Interpolation	• Linear: Verbindung über eine Gerade • Stepped: Verbindung über Stufen • BezierCubic: Verbindung über Bezierkurven

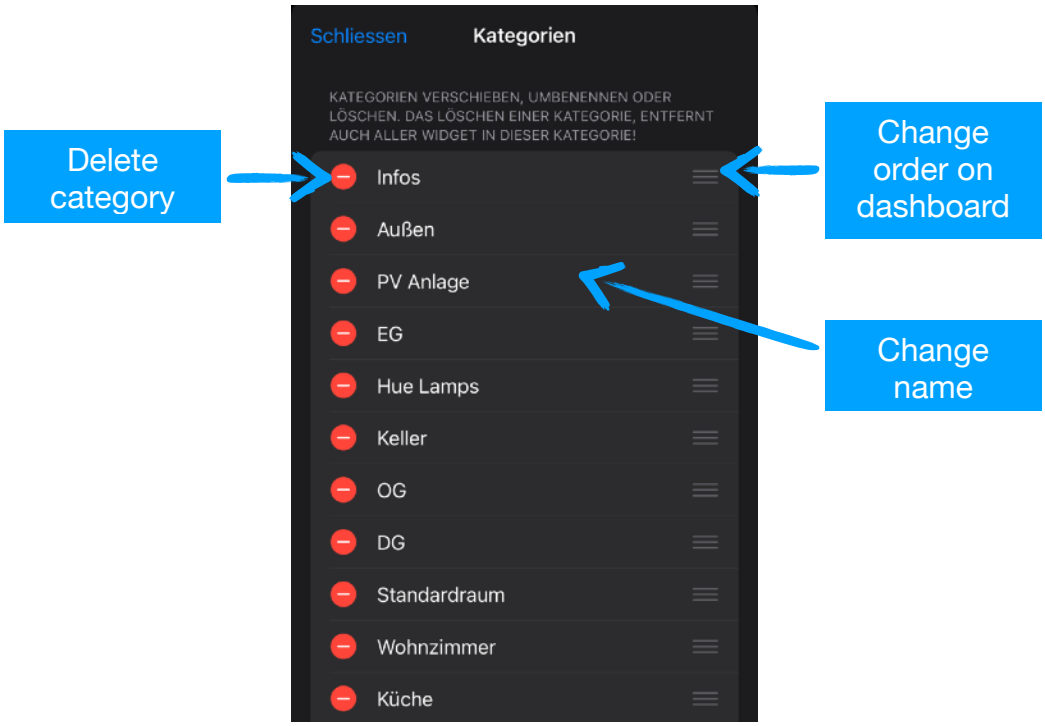
	Y-Achse autom. skalieren	**An**: Skaliert die Y-Achse(n) basierend auf dem minimalen und maximalen Wert der Reihe(n) **Aus**: Es wird der "Min" und "Max" Wert des Datenpunkts verwendet.
	Ausgefüllt	Fläche unter der Kurve wird ausgefüllt
	Punkte anzeigen	Zeigt einzelne Werte als Punkte
	Farbe der ersten Reihe	Farbe der ersten Wertereihe
	Farbabstand	Bestimmt Farben der folgenden Reihen
	Format X-Achse	• Keine: Keine X-Achse • Zeit: x-Werte als Zeit anzeigen • Datum: x-Werte als Datum anzeigen • Minuten: x-Werte als Minuten seit 00:00 Uhr interpretieren

** SF-Symbole werden unterstützt. Dazu statt dem Text den Namen des SF-Symbols mit eine vorangestellten '#' angeben. Zum Beispiel *#power* für zur Darstellung folgenden Symbols: ⏻

Dashboard organisieren

Es gibt verschiedene Möglichkeiten, das Aussehen des Dashboards zu beeinflussen:

- **Konfiguration des einzelnen Widgets ändern**
  Manche Widgets haben Anzeigeoptionen die man in der Widget-Konfiguration findet (*siehe oben "Widget Konfiguration"*)
- **Die Reihenfolge der Kategorien ändern**
  Dazu im Hauptmenü "Kategorien" auswählen und im dem Dialog durch Drag&Drop in der Liste die Reihenfolge ändern. Hier können die Kategorien auch umbenannt werden.



Endpunkte (Einfach)

Visual unterstützt eine Reihe von Endpunkten, die im Folgenden hier beschrieben werden.

Homekit

Stellt alle Homekit Geräte und deren "Services/Characteristics" in Visual als einen Endpunkt zur Verfügung.

Wichtig: Die Verwendung dieses Endpunkts setzt voraus, dass Sie Visual die Berechtigung zum Zugriff auf Homekit Geräte gegeben haben!

Homematic

Mit diesem Endpunkt kann man auf Homematic Geräte zugreifen. Dazu muss ein CCU oder CCU2 Gateway mit installiertem XML-API Patch vorhanden sein. Folgende Parameter sind zu konfigurieren:

URL	URL der CCU
Login	Login Name (falls vergeben)
Passwort	Passwort (falls vergeben)
Zeitintervall zum Neuladen	Die Daten aus der CCU werden periodisch in dem angegebene Zeitintervall (in Sekunden) ausgelesen

Nicht lesbare Datenpunkte verbergen	Homematic Datenpunkte, die nicht lesbar sind werden ignoriert

Philips Hue

Mit diesem Endpunkt kann man auf Philips Hue Lampen zugreifen. Dazu muss ein Philips Hue Gateway vorhanden sein. Folgende Parameter sind zu konfigurieren:

URL	URL des Hue Gateways
Zeitintervall zum Neuladen	Die Daten aus dem Gateway werden periodisch in dem angegebene Zeitintervall (in Sekunden) ausgelesen

Wichtig: Beim ersten Start werden Sie aufgefordert die App mit dem Hue Gateway zu koppeln. Bitte folgen Sie der Anweisung und drücken dazu den Koppeln-Knopf auf dem Hue Gateway.

URL

Dieser Endpunkt stellt nur einen Anschluß zur Verfügung. Dieser Anschluß liefert eine konfigurierbare URL. Diese kann in Verbindung mit dem WebView-Widget verwendet werden, um eine beliebige Webseite im Dashboard anzuzeigen.

Uhrzeit

Dieser Endpunkt stellt nur einen Anschluß zur Verfügung. Dieser Anschluß liefert die aktuelle Uhrzeit als Text. Derzeit einziger sinnvoller Einsatz ist derzeit die Verbindung mit dem Text-Widget, um die Uhrzeit im Dashboard anzuzeigen

OpenWeatherMap

Der OpenWeatherMap Endpunkt liefert Anschluß für die aktuelle Temperatur, Luftfeuchtigkeit, Windstärke und Windrichtung. In der Endpunkt-Konfiguration kann man den Ort für die aktuellen Wetterdaten spezifizieren und den OpenWeatherMap App-Key eingeben. **Wichtig: Zuerst bitte auf OpenWeatherMap (kostenlos) registrieren und einen App-Key generieren!**

HTML Widget

In diesem Endpunkt kann man ein HTML Widget hinterlegen. HTML Widgets sind normalerweise zum Einbetten auf eine Webseite gedacht. Man findet diese im Internet z.B. auch auf vielen Wetterseiten. Der HTML Code wird einfach per "Copy/Paste" in das dafür vorgesehene Feld in der Endpunkt-Konfiguration kopiert.
Der Anschluß stellt dann eine lokale URL zur Verfügung, dass dann mit einem WebView Widget verbunden werden kann.

PVOutput.org

Dieser Endpunkt liest Daten vom Photovoltaik Web-Service pvoutput.org

Zeitintervall zum Neuladen	Die Daten periodisch in dem angegebene Zeitintervall (in Sekunden) angefordert
System-ID	System-ID (aus dem persönlichen PVOutput.org Profil übernehmen)
API-ID	API-ID (aus dem persönlichen PVOutput.org Profil übernehmen)

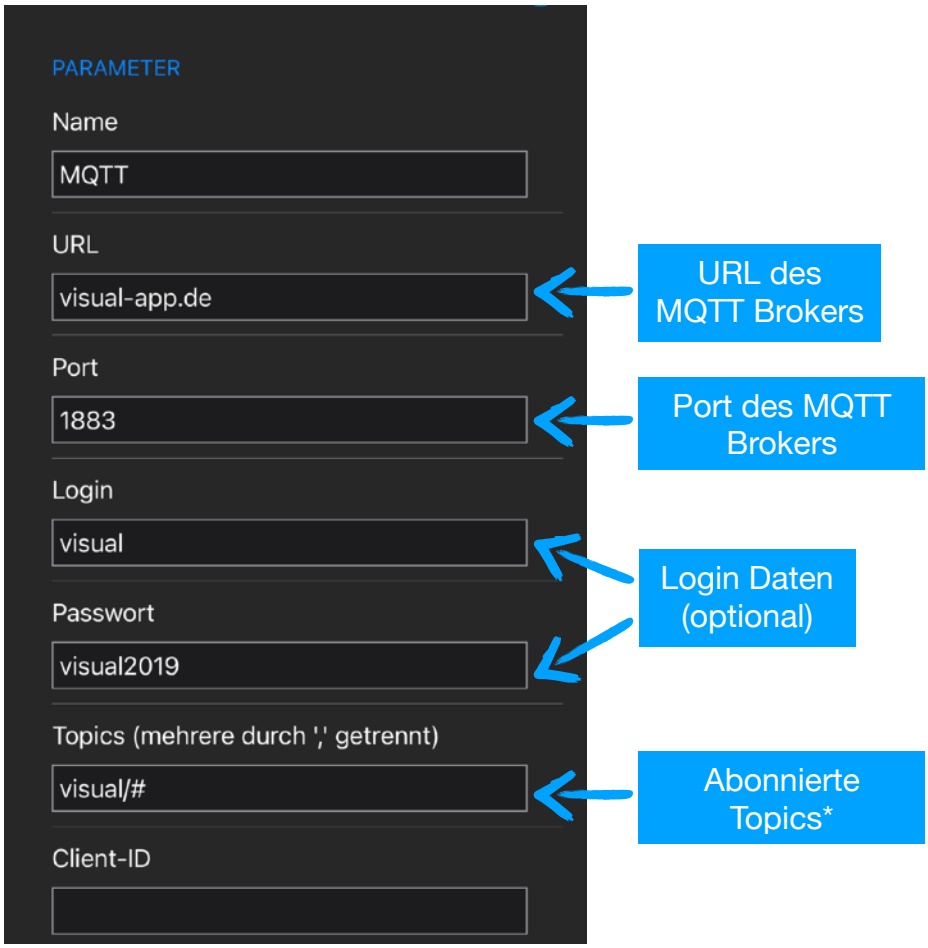
Endpunkte (Experten)

MQTT Client

MQTT CLIENT KONFIGURATION

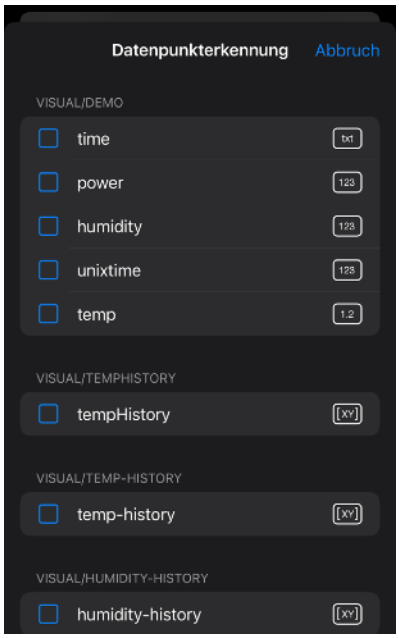
Dieser Endpunkt ermöglicht es Daten von einem MQTT Broker zu abonnieren ("Subscribe") oder zu veröffentlichen ("Publish")
Wichtig: Es wird vorausgesetzt, dass die MQTT Topics als JSON Objekt formatiert sind!

Grundkonfiguration



** Mehrere Topics werden durch Komma getrennt. Es ist erlaubt Wildcards ("#" oder "+") zu verwenden*
In der Menüleiste wird durch ein Icon angezeigt, ob die Verbindung korrekt zustande gekommen ist. Nur in diesem Fall, ist auch der Datenpunkt Wizard anwählbar!

DATENPUNKT WIZARD



Der Wizard versucht die Datenpunkt Konfiguration halb-automatisch auszufüllen. Dazu "hört" der Wizard auf alle abonnierten Topics und analysiert die Payload. Dann wird eine Liste von möglichen Datenpunkten angezeigt. Hier die Datenpunkte selektieren die erzeugt werden sollen. Danach auf "Anwenden" tippen.

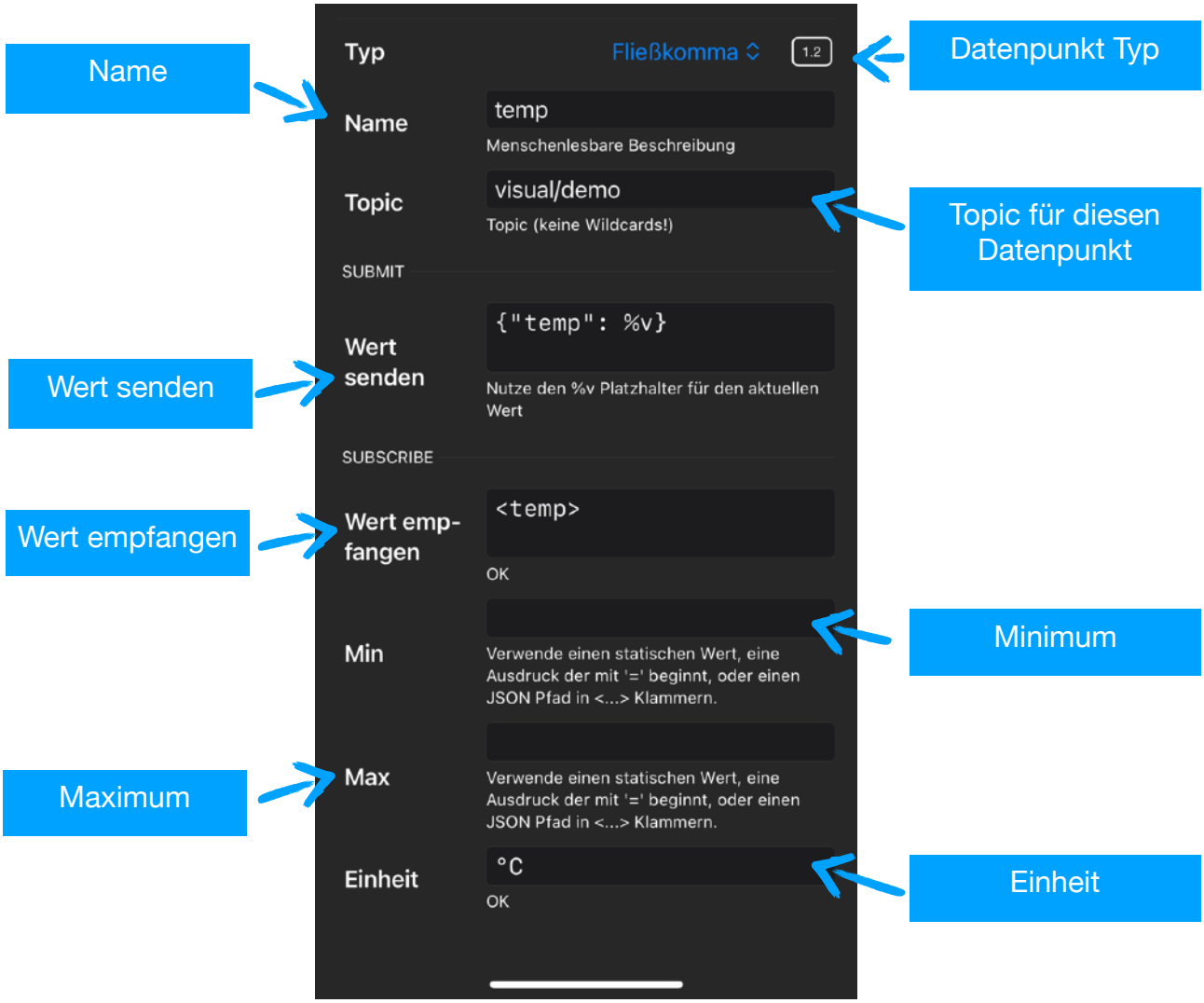
Wichtige Hinweise damit der Wizard funktionieren kann:
•Die Payload muss im JSON Format sein!
•Vorher im "Topics" Feld alle Topic Namen eintragen auf die der Wizard reagieren soll!
•Es müssen Daten empfangen werden, also sicherstellen, dass die relevanten Topics publiziert werden währen der Wizard läuft!

MANUELLE ANSCHLUSS KONFIGURATION

Name	Anzeigename
Topic	Topic mit dem dieser Anschluss verbunden ist. *Hinweis: Hier ist kein Wildcard erlaubt!*
Wert senden	Falls der Anschluss zum Publizieren verwendet werden soll muß hier die Payload eingegeben werden. Hierbei wird der Platzhalter %v durch den eigentlichen Wert ersetzt. Beispiel: { "value": %v }
Wert empfangen	Hier ist spezifiziert wo in der Payload der Anschluss-Wert zu finden ist. Hierzu spezifiziert man einen "JSONPath" Ausdruck in spitzen Klammern ("< >") - siehe Beispiele unten.
Einheit	Einheit des Anschlusses - Statisch oder per JSONPath aus der Topic Payload
Min	Minimalwert des Anschlusses - Statisch oder per JSONPath aus der Topic Payload*

Max	Maximalwert des Anschlusses - Statisch oder per JSONPath aus der Topic Payload*

* *Immer beide "Min" und "Max" zusammen angeben!*



WERT AUS EMPFANGENEN DATEN EXTRAHIEREN

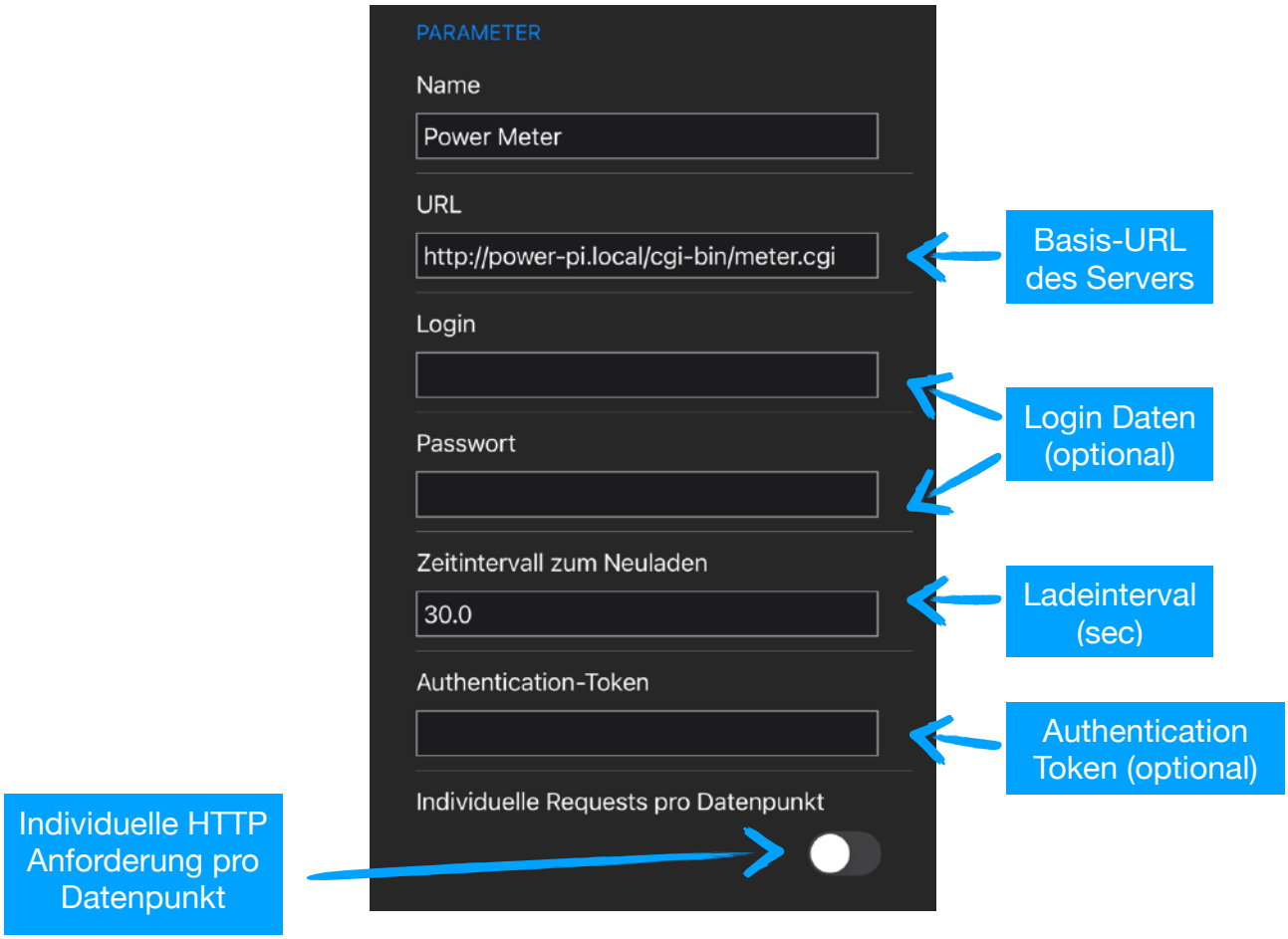
Statische Werte können direkt in die Felder eingegeben werden (meist für Min, Max und Einheit). In den meisten Fällen ist es notwendig, den Wert dynamisch aus der JSON-Nutzlast des Brokers abzurufen. Dies kann über eine vereinfachte JSONPath-Notation auf einzelne JSON-Eigenschaften zugegriffen werden (Details zur Syntax finden Sie in Anhang A). Der JSON-Pfad muss in "<...>"-Klammern stehen!

ANSCHLUSS LÖSCHEN

Auf dem Anschluss nach links wischen, dann auf den "Löschen" Button tippen. Hinweis: Ein Anschluss muss immer konfiguriert werden!

HTTP Client

Dieser Endpunkttyp ist vom Prinzip sehr ähnlich zu dem MQTT Client. Nur wird hier als Transportprotokoll HTTP anstatt MQTT verwendet.



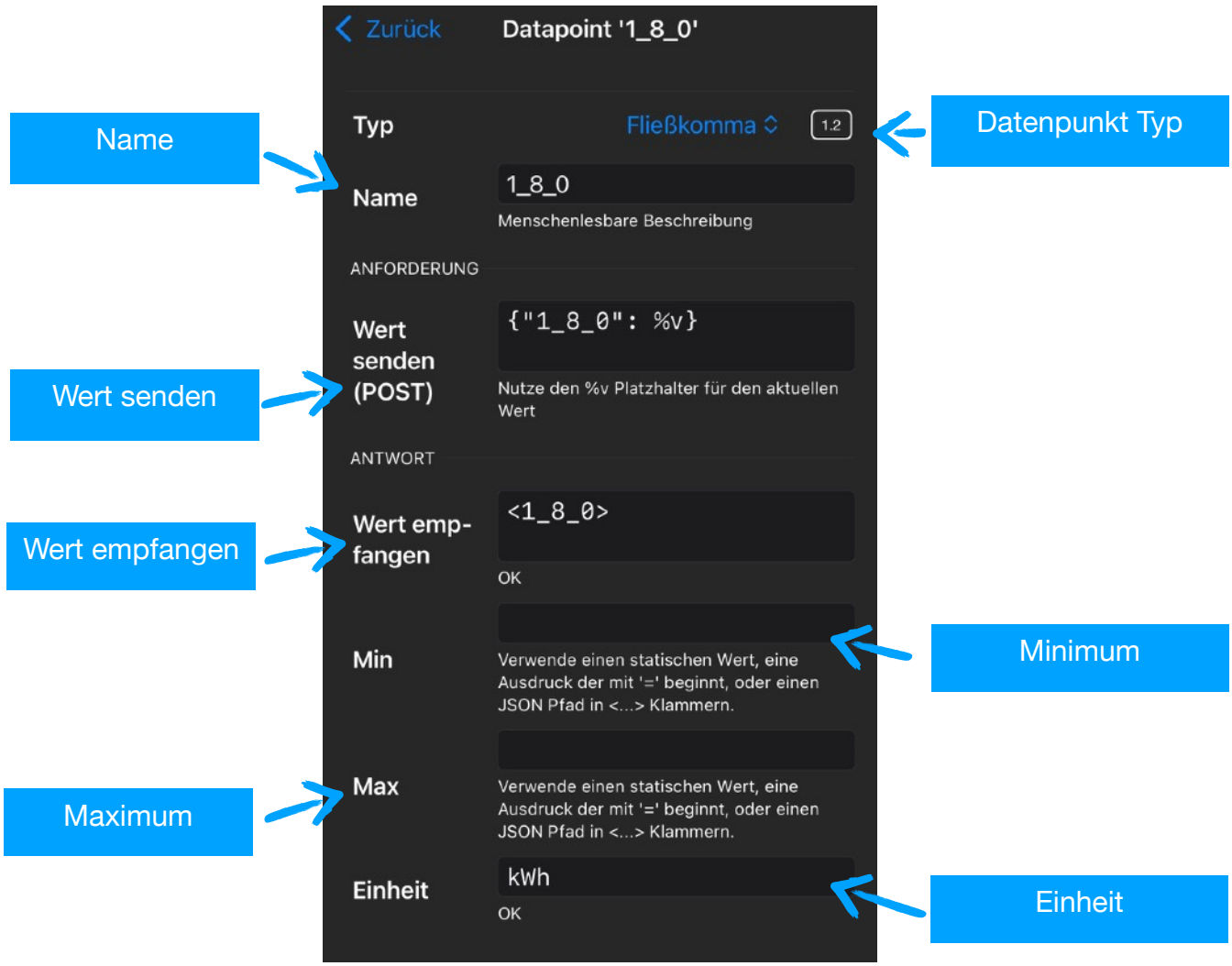
DATENPUNKT KONFIGURATION

Wie schon beim MQTT Client können die einzelnen Datenpunkt manuell konfiguriert werden:

Name	Anzeigename
Pfad	Wenn die Option „Individuelle Requests pro Datenpunkt“ eingeschaltet ist wird dieser Wert an die globale URL angehängt. Wenn die globale URL nicht mit eine "/" endet, wird vor dem Anhängen die letzte Pfadkomponente entfernt. In diesem Feld kann der Platzhalter "%v" für den Wert und "%i" für die eindeutige Identifikation des Datenpunkt verwendet werden.

Wert schreiben	Falls der Anschluss zum Senden verwendet werden soll muß hier die Request Payload eingegeben werden. Hierbei wird der Platzhalter %v durch den eigentlichen Wert ersetzt. Beispiel: { "value": %v }
Wert empfangen	Hier ist spezifiziert wo in der Response Payload der Anschluss-Wert zu finden ist. Hierzu spezifiziert man einen "JSONPath" Ausdruck in spitzen Klammern ("< >") - siehe Beispiele beim MQTT Client.
Einheit	Einheit des Anschlusses - Statisch oder per JSONPath aus der Response Payload
Min	Minimalwert des Anschlusses - Statisch oder per JSONPath aus der Response Payload*
Max	Maximalwert des Anschlusses - Statisch oder per JSONPath aus der Response Payload*

** Immer beide "Min" und "Max" zusammen angeben!*



Daten lesen: Dieser Endpunkt benutzt die HTTP GET Methode um Daten vom Server zu lesen. Die Serverantwort auf den Request muß im JSON Format sein, damit Visual die Daten auswerten kann!
Daten senden: Um Aktionen auf dem Server auszulösen wird die HTTP POST Methode verwendet. Die Payload des Requests kann für jeden Datenpunkt getrennt definiert werden. Dabei wir der Platzhalter "%v" durch den tatsächlichen Wert des Widget-Anschlusses ersetzt. Alternativ ist es möglich HTTP GET zu verwenden. Hierzu muß im Pfad-Feld der "%v" Platzhalter in der verwendet werden.

Beispiele:

Globale URL: http://myserver/api/

Temperatur lesen via HTTP GET von "getTemperature"
Server-Antwort in JSON: { "temp": 22.0, "unit": "°C" }

Path	getTemperature
Value (in)	<temp>
Unit	<unit>

```
➜ GET http://myserver/api/getTemperature
```

Spezifische Temperatur lesen via HTTP GET von "getTemperature"
Server-Antwort in JSON: { "temp": 22.0, "unit": "°C" }

Ident	Kitchen
Path	getTemperature&ident=%i
Value (in)	<temp>
Unit	<unit>

```
➜ GET http://myserver/api/getTemperature?ident=Kitchen
```

Temperatur schreiben via HTTP GET von "setTemperature"
Widget Wert ist 20.0

Path	setTemperature?value=%v
Value (out)	

```
→ GET http://myserver/api/setTemperature?value=20.0
```

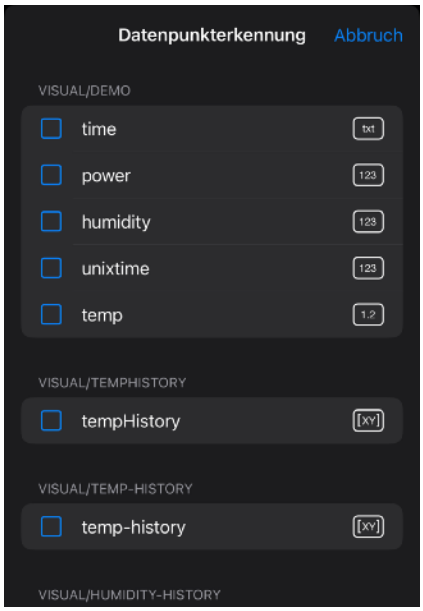
Temperatur schreiben via HTTP POST von "setTemperature"
Payload in JSON, Widget Wert ist 20.0

Path	setTemperature
Value (out)	{ "value": %v }

```
→ POST http://myserver/api/setTemperature
Payload:{ "value":20.0 }
```

DATENPUNKT WIZARD

Alternativ kann man den Wizard verwenden. Hier wird die globale URL verwendet um Daten beim Server per HTTP GET abzurufen

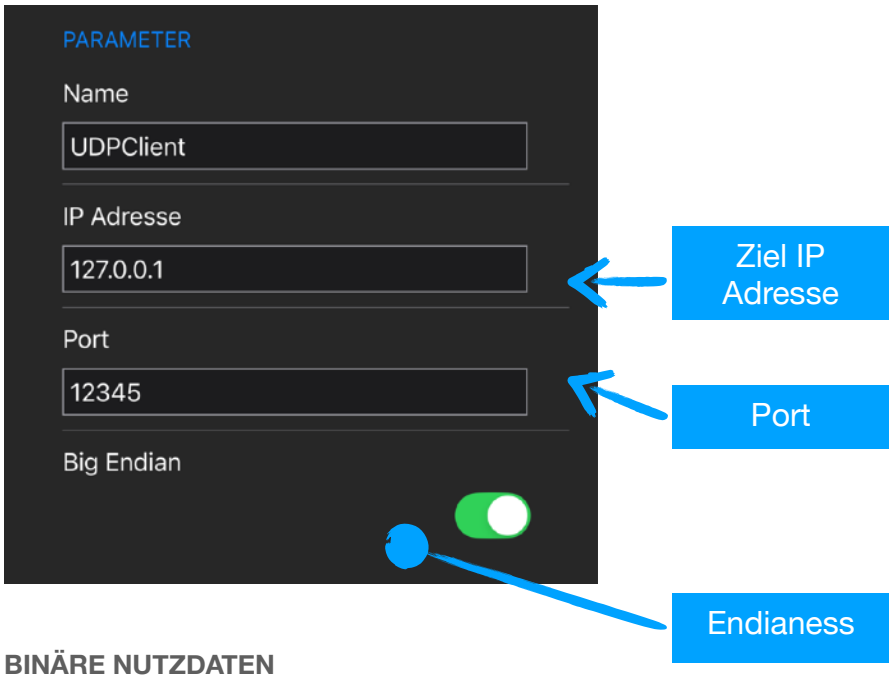


UDP Client

Dieser generische Endpunkt ermöglicht das Senden von UDP-Datagrammen. Die Datagramme können so konfiguriert werden, dass sie textbasierte Nutzdaten (z. B. JSON) oder binäre Daten-Playloads senden.

Hinweis: Dieser Endpunkt unterstütz keinen Empfang von Daten!

DATENPUNKT KONFIGURATION



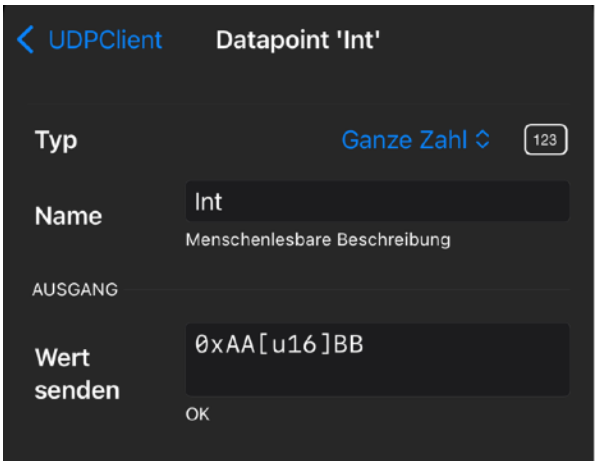
BINÄRE NUTZDATEN

Um eine binäre Nutzlast zu senden, verwenden Sie einfach das Präfix "0x" in der Konfiguration "Wert senden". Um den aktuellen Wert des Datenpunkts einzubetten, sind mehrere Platzhalter zulässig:

Platzhalter	Typ
`[b]`	1-Byte Boolean
`[u8], [u16], [u32]`	Unsigned 8, 16 oder 32 bit Integer
`[i8], [i16], [i32]`	Signed 8, 16 oder 32 bit Integer
`[f], [d]`	Einfache oder doppelte Präzision Fließkommazahl
`[s]`	UTF8 String

Für alle Werttypen, die länger als ein Byte sind, wird die globale Einstellung „Endianess“ verwendet, um den Wert im Binärpuffer zu bilden!

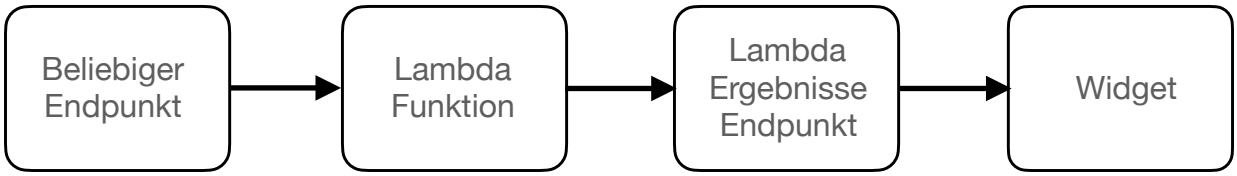
Das folgende Beispiel fügt den aktuellen Integer-Datenpunkt als 16-Bit-Wert in



eine binäre Nutzlast ein. Die Nutzlast beginnt mit dem Byte „0xAA“, gefolgt vom 16-Bit-Wert und endet mit dem Byte „0xBB“:

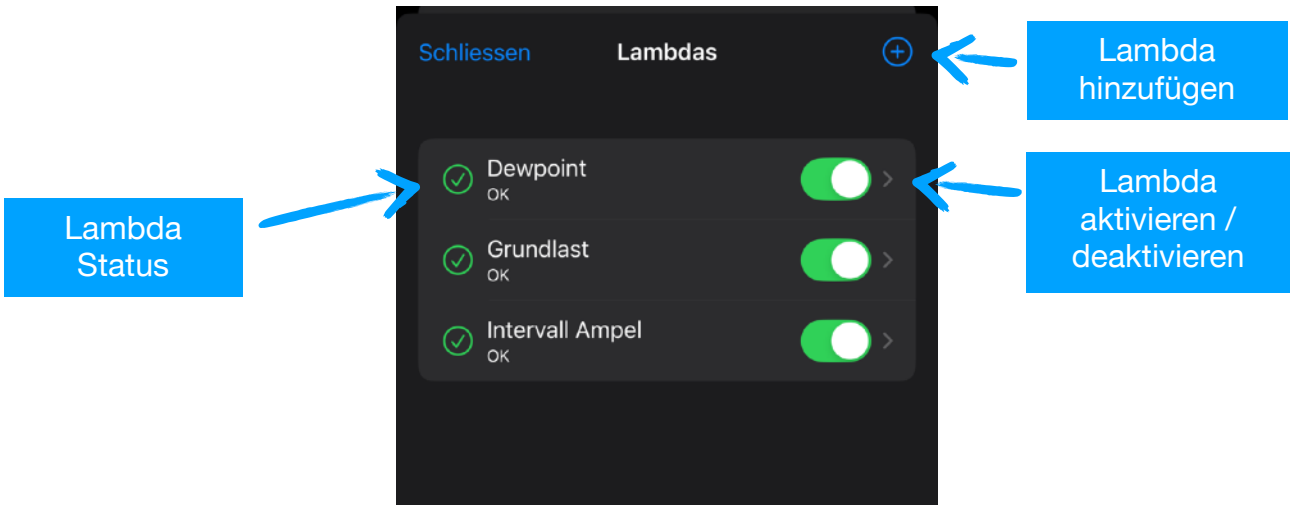
Lambda Funktionen (Erweiterte Funktion)

Lambdas sind kleine in Javascript geschrieben Funktionen, die Daten von Endpunkten verarbeiten und die Ergebnisse wieder zur dem Rest des Systems zur Verfügung stellen können.
Wichtig: Dies ist eine Expertenfunktion, denn sie setzt Programmierkenntnisse in JavaScript voraus!

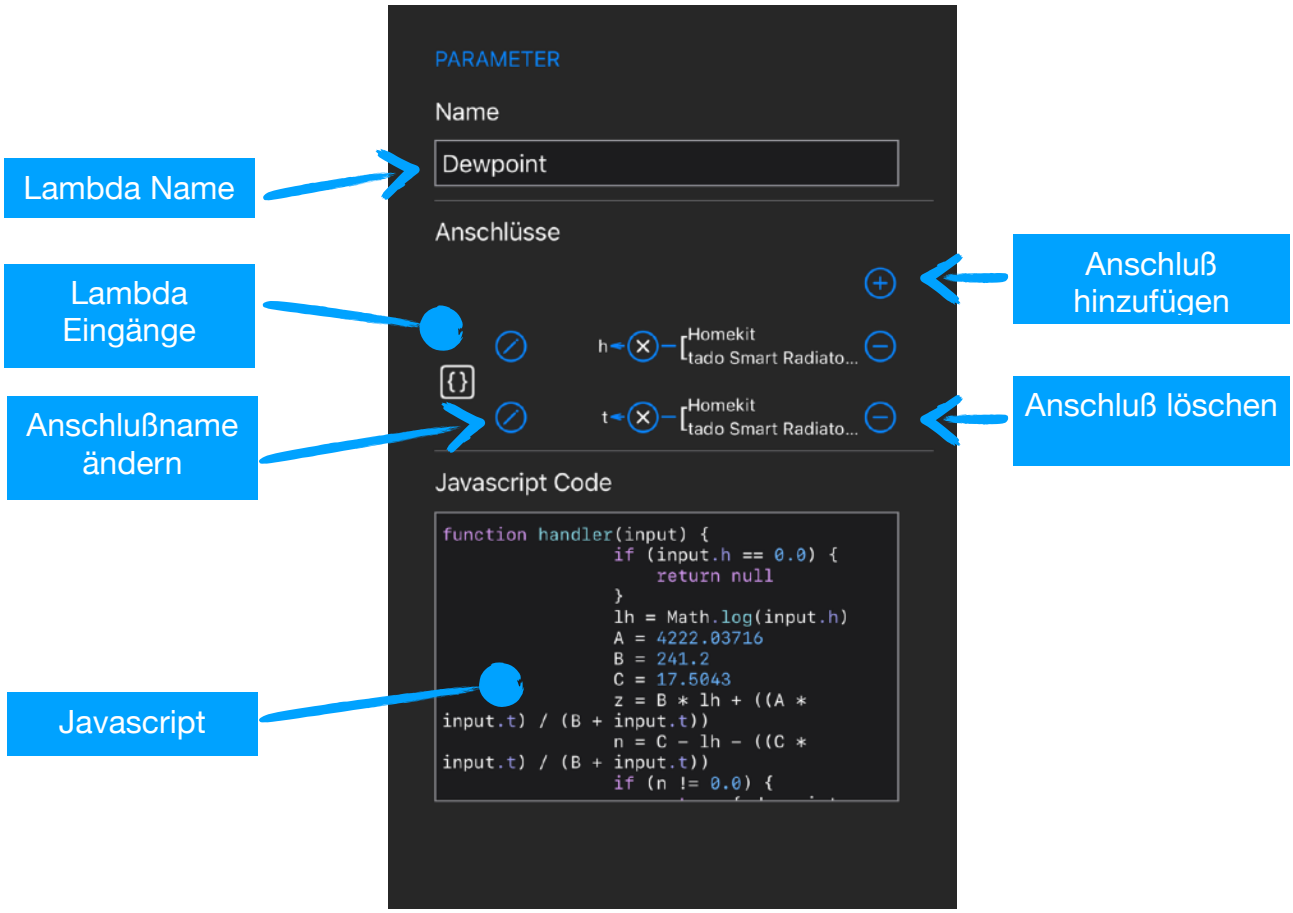


Vom Hauptmenü aus aus kommt man über den Eintrag "Lambdas" in den Lambda-Verwaltungsmodus:

Durch Wischen von Rechts nach Links, erscheint der "Löschen" Button.
Durch Antippen einer Zeile kommt man zur Lambda-Konfiguration.



Lambda Konfiguration



Eine Lambda besteht aus genau einer Javascript Funktion mit dem Namen "handler". Diese Funktion erhält ein Javascript Objekt als Übergabeargument.

Daten-Input

Ein Lambda Funktion kann beliebig viele Eingangsanschlüsse haben. Dazu einfach mit dem "+" Icon neue Anschlüsse hinzufügen und mit beliebigen Endpunkten verbinden (derzeit werden nur die Typen "Integer", "Float", "Bool", "Index" und "Prozent" unterstützt!),
Alle Anschlusswerte werden in eine Übergabeobjekt verpackt:

```
{
   name: wert,
   name: wert,
   allValues: [wert, wert, ...]
}
```

Zusätzlich zu den einzelnen Werten, werden auch alle Werte in einem Array mit dem Namen "allValues" übermittelt (Achtung: Die Werte in diesem Array sind sortiert nach dem Anschlußnamen!) In unserem Beispiel könnte das so aussehen

```
{
  h: 78.3,
  t: 18.7,
  allValues: [78.3, 18.7]
}
```

Die Property-Namen im Objekt entsprechen gewählten Anschlußnamen!

Die Lambda Funktion wird immer dann aufgerufen, wenn sich einer der Eingangswerte verändert hat. Falls es mehrere Eingangswerte gibt, werden deren letzter bekannter Wert übergeben, d.h. auch dass beim ersten Start jeder Eingangswert mindestens einmal empfangen worden sein muss, bevor die Lambda Funktion das erste Mal aufgerufen wird!

Daten-Output

Die Lambda Funktion kann mehrere Rückgabewerte haben. Auch dies ist wieder in ein Javascript Objekt mit dem folgenden Schema:

```
{
   name: { value: wert, unit: unit, min: min, max: max },
   name: { value: wert, unit: unit, min: min, max: max },
   ...
}
```

wobei "unit", "min" und "max" optional sind. In unserem Beispiel wieder

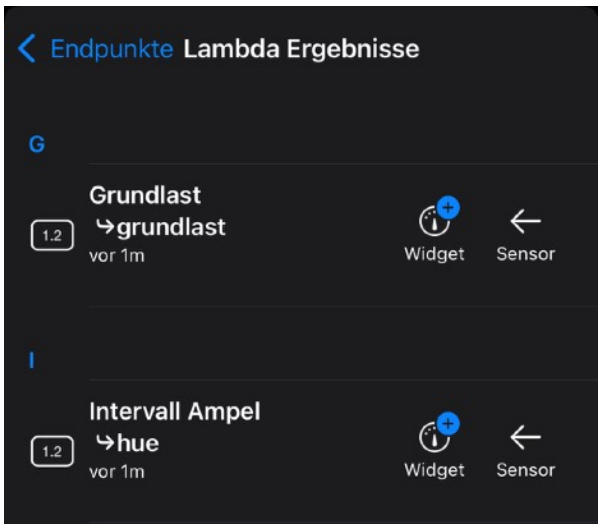
```
{
   dewpoint: { value: 13.4, unit: "°C", min: 0.0, max: 40.0 }
```

```
}
```

Hinweis: Da JavaScript keinen Integer-Typ hat, kann man das per "forceInt: true" erzwingen, z.B.

```
{
   ganzzahl: { value: 42, forceInt: true }
}
```

Jeder von der Funktion zurückgegebene Wert wird über den speziellen Endpunkt "Lambda Ergebnisse" veröffentlicht und kann nun mit einem Widget zur Anzeige verbunden werden:

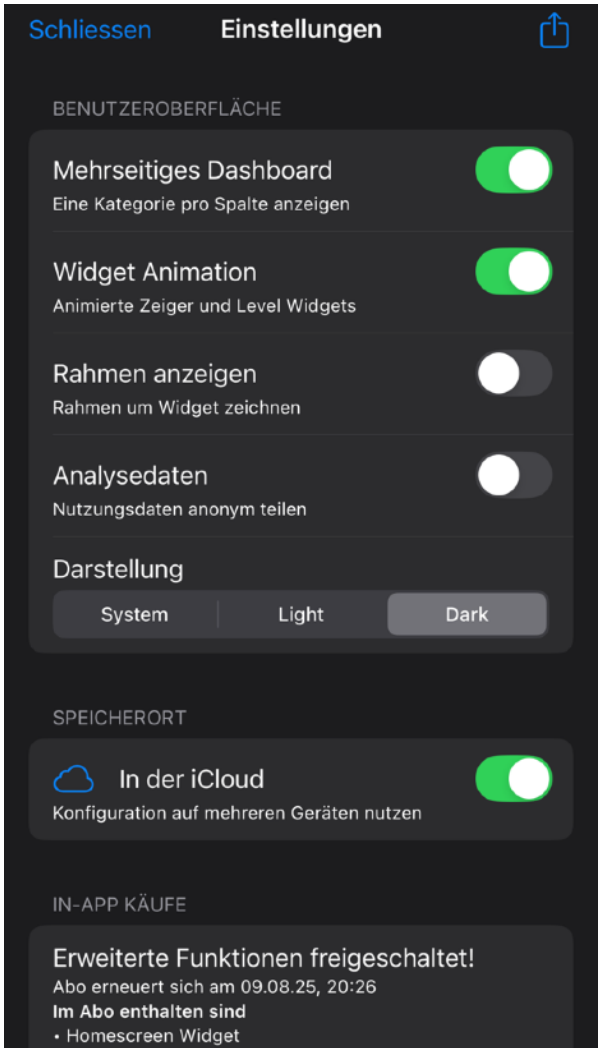


Der Rückrabewert "null" wird entsprechend ignoriert und kein Wert veröffentlicht.
Hinweis: Das Ergebnis von einer Lambda Funktion kann wieder Input einer anderen Funktion (aber nicht sich selbst) sein!
Das hier als Beispiel verwendete Lambda berechnet den Taupunkt aus Temperatur und Luftfeuchtigkeit mittels der Magnus-Formel:

```
function handler(input) {
  if (input.h == 0.0) {
    return null
  }
  lh = Math.log(input.h / 100.0)
  A = 4222.03716
  B = 241.2
  C = 17.5043
  z = B * lh + ((A * input.t) / (B + input.t))
  n = C - lh - ((C * input.t) / (B + input.t))
  if (n != 0.0) {
    return { dewpoint: { value: z/n, unit: "°C" } }
  }
  return null
}
```

Globale Einstellungen

Die globalen Einstellung beinhalten derzeit



- **Mehrspaltiges Dashboard** (In-App Feature)
  Pro Kategorie wird eine eigene Spalte im Dashboard angezeigt
- **Widget Animationen**
  Zeiger und Level Widgets werden animiert
- **Rahmen anzeigen**
  Zeichnet einen Rahmen um die Widgets
- **Analysedaten**
  Beim ersten App-Start wir diese Einstellung separate abgefragt und kann hier aber jederzeit verändert werden. Wenn die Option aktiviert ist werden folgende Informationen **anonym** mit dem App-Autor geteilt:
    - Art und Anzahl der verwendeten Widgets
    - Art und Anzahl der verwendeten Endpunkte

- **Darstellung**
  System: Darstellung folgt der Systemeinstellung (Hell/Dunkel)
  Hell: Immer helle Darstellung (weißer Hintergrund)
  Dunkel: Immer dunkle Darstellung (schwarzer Hintergrund)
- **Speicherort**
  Konfiguration wir in der iCloud gespeichert und kann so über mehrere Geräte hinweg synchron gehalten werden
- **In-App Käufe**
  Per In-App Kauf kann man hier für 1,49€ im Monat erweiterte Funktionen freischalten. Für Nutzer die schon ein Werbefrei-Abo abgeschlossen haben, sind diese Funktionen auch freigeschaltet!
  Die Funktionen richten sich vor allem an Nutzer mit sehr großen SmartHome Setups:
    - Homescreen Widgets (ab V1.7)
    - Javascript Lambda Funktion (ab V1.8)
    - Mehrspaltiges Dashboard
    - Datenpunkte lassen sich durchsuchen und filtern
    - Szenen lassen sich importieren (von ausgewählten Endpunkten, z.B. Philips Hue)
    - Debug-Logging ist für alle Endpunkte möglich um Probleme aufzuspüren

Bitte unterstützt Visual mit einem In-App Kauf für den Unkostenbeitrag von weniger als einen Cappuccino im Monat! Nur so kann Visual weiterentwickelt (neue Features) und gepflegt (neue iOS Versionen) werden. Danke!

Hinweis: Abos können jederzeit in den persönlichen iTunes Einstellungen beendet werden!

iCloud Sync

Die gesamte Konfiguration von Visual wird lokal in der App-Sandbox gespeichert und automatisch iCloud Account des Nutzers synchronisiert. D.h. jede Änderung in Visual wird automatisch auf allen anderen iOS Geräten in Visual sichtbar - sofern man auf den Geräten mit dem gleiche iCloud Account angemeldet ist!
Wenn man auf mehreren Geräten gleichzeitig Änderungen vornimmt, erhält man eine Konfliktwarnung und man muss sich manuell für eine Version entscheiden.

Anhang A: Vereinfachter JSONPath Syntax

In die Felder können direkt statische Werte eingetragen. In den meisten Fällen ist es aber nötig den Wert dynamisch aus der JSON Payload des Brokers zu erfragen. Dazu kann man mit einer vereinfachten JSONPath Notation auf einzelne JSON-Properties zugreifen. **Der JSONPath Ausdruck ist entsprechend in spitze Klammern (<...>) einzutragen**.

Beispiele:

```
{
   "Name": "Mustermann",
   "Vorname": "Max"
}
```

<Name>	Mustermann
<Vorname>	Max

```
{
   "color": { "red": 1.0, "green": 0.2, "blue": 0.5 }
}
```

<color.red>	1.0
<color.blue>	0.5
<color.@keys>	red, green, blue

```
{
   "Fahrzeuge": [
       {"Typ": "PKW", "Räder": 4},
       {"Typ": "Fahrrad", "Räder": 2}
   ]
}
```

<Fahrzeuge.[0].Räder>	4
<Fahrzeuge.[1].Typ>	Fahrrad
<Fahrzeuge.*.Typ>	PKW, Fahrrad
<Fahrzeuge.@count>	2

Zusätzlich können mathematische Ausdrücke in den Feldern verwendet werden. Dazu den Ausdruck mit "=" beginnen z.B.:

```
{
   "color": { "red": 1.0, "green": 0.2, "blue": 0.5 }
}
```

=<color.blue>*100	50.0
=<color.blue>*<color.green>*100	10.0

Anhang B: Datenformat Für Diagramme

Typ	JSON Format	JSON Path
[XY]	`[ {"x": x0, "y": y0}, {"x": x1, "y": y1}, … ]`	`<[*]>`
	`{ "data": [ {"x": x0, "y": y0}, {"x": x1, "y": y1}, … ] }`	`<data.[*]>`
[1.2]	`[ y0, y1, … ]`	`<[*]>`
	`{ "data": [ y0, y1, … ] }`	`<data.[*]>`

[XY] Die "x" Werte können einfach ein Index oder ein Datum/Uhrzeit im Unit-Time Format (Sekunden sein 01.01.1970) sein.

[1.2] Die "x" Werte werden automatisch erzeugt (0...n)

Kontakt

me@andreas-binner.de