ELEMENT IoT Handbuch

ZENNER IoT Solutions GmbH

20. Januar 2021

Inhaltsverzeichnis

1 ELEMENT IoT - Einführung

ELEMENT IoT ist eine allgemeine IoT Plattform, das bedeutet, ELEMENT IoT bietet alle Funktionen zur Verwaltung und zum Betrieb von IoT-Geräten.

ELEMENT IoT baut dafür auf 6 grundlegende Bausteinen auf:

- Geräte
- Treiber
- Ordner
- Pakete
- Parser
- Messwerte

1.1 Geräte und Profile

Geräte in ELEMENT bilden physikalische Geräte (Aktoren und Sensoren) ab, welche Daten an ELEMENT senden oder Daten von ELEMENT empfangen. Geräte können einen Ort haben. Beliebige Stammdaten können Geräten über Profile zugeordnet werden.

1.2 Treiber

Treiber beschreiben Schnittstellen, über die Daten von Geräten zu ELEMENT und von ELEMENT zum Gerät kommen. Einem Gerät können mehrere Interfaces zugeordnet werden. Ein Interface beschreibt die Verbindung zwischen Treiber und Gerät.

Als Treiber sind u.a. verfügbar

- LoRaWAN (ZIS LNS, Tracknet LNS, Loriot LNS)
- Sigfox
- MQTT
- LoRaWAN WMBus-Bridge

1.3 Ordner

Geräte in ELEMENT befinden sich immer in mindestens einem Ordner, können aber in mehreren Ordnern enthalten sein. Die Ordenerstruktur ist eine Baumstruktur wie bei Dateisystemen, d.h. Geräte können hierarchisch geordnet werden.

Über Ordner wird außerdem:

- die Berechtigung auf Geräte geregelt
- und Geräte für gemeinsamen Export, API-Zugriff und Regeln zusammengefasst.

1.4 Pakete

Pakete sind zusammenhängende Daten, die als eine Operation an ELEMENT vom Gerät oder zurück geschickt werden.

Es sind damit in gewisser Weise Rohdaten. I.d.R handelt es sich entweder um Binärcodierte Daten oder um JSON.

1.5 Parser und Messwerte

Parser stellen Funktionen bereit, die aufgerufen werden sollen, wenn ein neues Paket eintrifft, um daraus Messwerte zu erhalten. Aus einem Paket können mehrere Messwerte entstehen.

Parser werden als Elixir-Code angegeben und laufen in einer Sandbox. So stehen nur zum Parsen nötige Funktionen bereit, aber keine Funktionen, welche z.B. Zugriff auf Dateien ermöglichen.

Parser stehen teilweise auf Github bereit, können von Nutzern selbst erstellt werden oder von ZRI/ZIS im Auftrag entwickelt werden.

Parser können zur Laufzeit hinzugefügt werden.

1.6 Aufbauende Funktionen

Alle weiteren Kernfunktionen von ELEMENT bauen auf diesen 6 Grundelementen auf, insbesondere:

- Paketstatistiken
- 3 Ansichten für Messwerte (immer pro Gerät oder Ordner)
    - Liste

  - Graph
  - Karte

- Vorlagen (Voreingestellte Profile, Parser usw.)
- Apps (spezielle Sichten)
- Regeln
- Exporte
- API

1.7 Anmelden an der ELEMENT IoT-Plattform

Jede Arbeit mit ELEMENT IoT beginnt mit der Anmeldung eines Nutzers.

Benutzer haben einen Nutzernamen, welcher der E-Mail-Adresse entspricht, und eine Passwort. Angemeldet wird sich immer an einem spezifischen Mandanten.

Nutzer werden nicht zwischen Mandanten geteilt, der selbe Nutzername kann aber ggf. bei mehreren Mandaten genutzt werden. Einstellungen und Passwörter sind aber nicht mandatenübergreifend. Wenn ein Nutzer an einem Mandaten angemeldet ist und z.B. sein Passwort ändert, wird diese Änderung nur für den Mandaten wirksam, an dem der Nutzer aktuell angemeldet ist.

- Öffnen Sie Ihren Browser und geben Sie in die Adresszeile https://element-iot.com ein, die URL kann bei Installationen in Ihrem Rechenzentrum (On-Premise) abweichen
- Es erscheint das Fenster für die Anmeldung an der ELEMENT IoT-Plattform

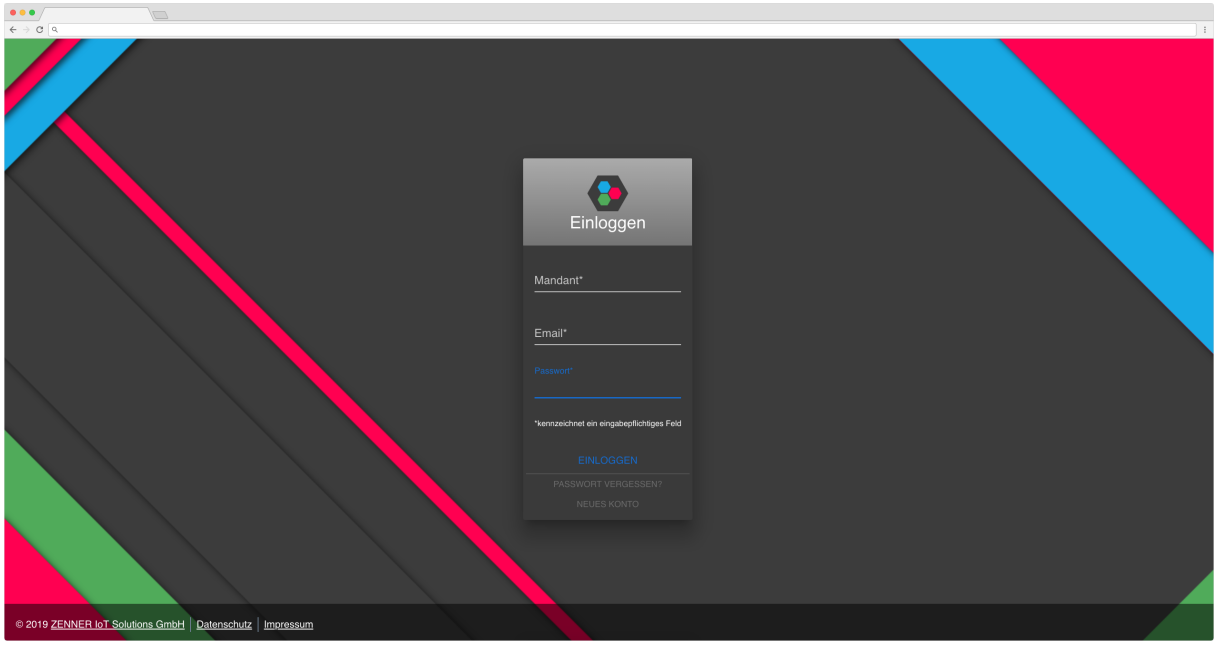


Abbildung 1.1: Screenshot

- Geben Sie nun Ihren Mandaten, Ihre E-Mail-Adresse und Ihr Passwort ein

Sollten Sie ihr Passwort vergessen haben, können Sie dieses selbstständig zurücksetzen .

1.8 Fehleingaben bei der Anmeldung

Sollten Sie ihr Passwort 5 Mal hintereinander falsch eingeben, werden Sie für 10 Minuten ab der letzten Fehlereingabe für die Anmeldung gesperrt.

1.9 Passwort zurücksetzen

Sollten Sie ihr Passwort vergessen haben, können Sie dieses jederzeit über die Anmeldemaske zurücksetzen.

- Öffnen Sie Ihren Browser und geben Sie in die Adresszeile https://element-iot.com ein, die URL kann bei Installationen in Ihrem Rechenzentrum abweichen.
- Es erscheint das Fenster für die Anmeldung an der ELEMENT IoT-Plattform.
- Klicken Sie nun auf den Button **PASSWORT VERGESSEN?**

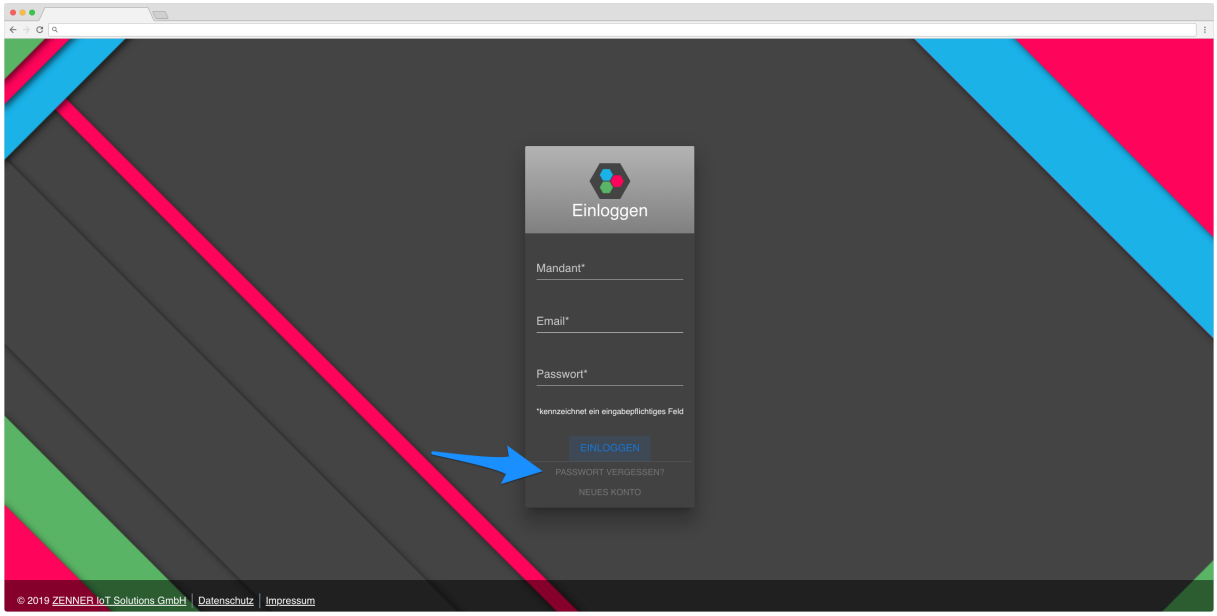


Abbildung 1.2: Screenshot

- In der folgenden Maske geben Sie bitte Ihren Mandanten und Ihre hinterlegte E-Mail-Adresse ein.

Abbildung 1.3: Screenshot

- Nach dem Klicken auf **NEUES PASSWORT ANFORDERN** erhalten Sie eine E-Mail mit einem Link

zum Zurücksetzen Ihres Kennworts

1.10 Benutzerprofil

In Ihrem Benutzerprofil können Sie folgende Einstellungen vornehmen beziehungsweise ändern.

- Sprache der Oberfläche (Englisch oder Deutsch)
- Änderung Ihres Passworts
- Änderung Ihrer hinterlegten E-Mail-Adresse
- Änderung Ihres Namens

Zum Öffnen Ihres Benutzerprofils klicken Sie bitte in der Menüleiste auf Ihren Namen.

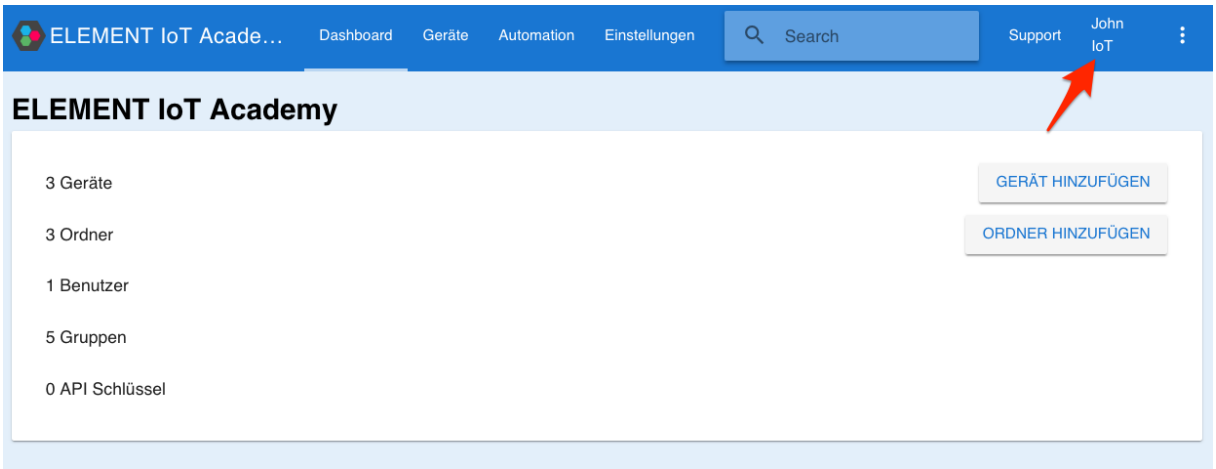


Abbildung 1.4: Screenshot

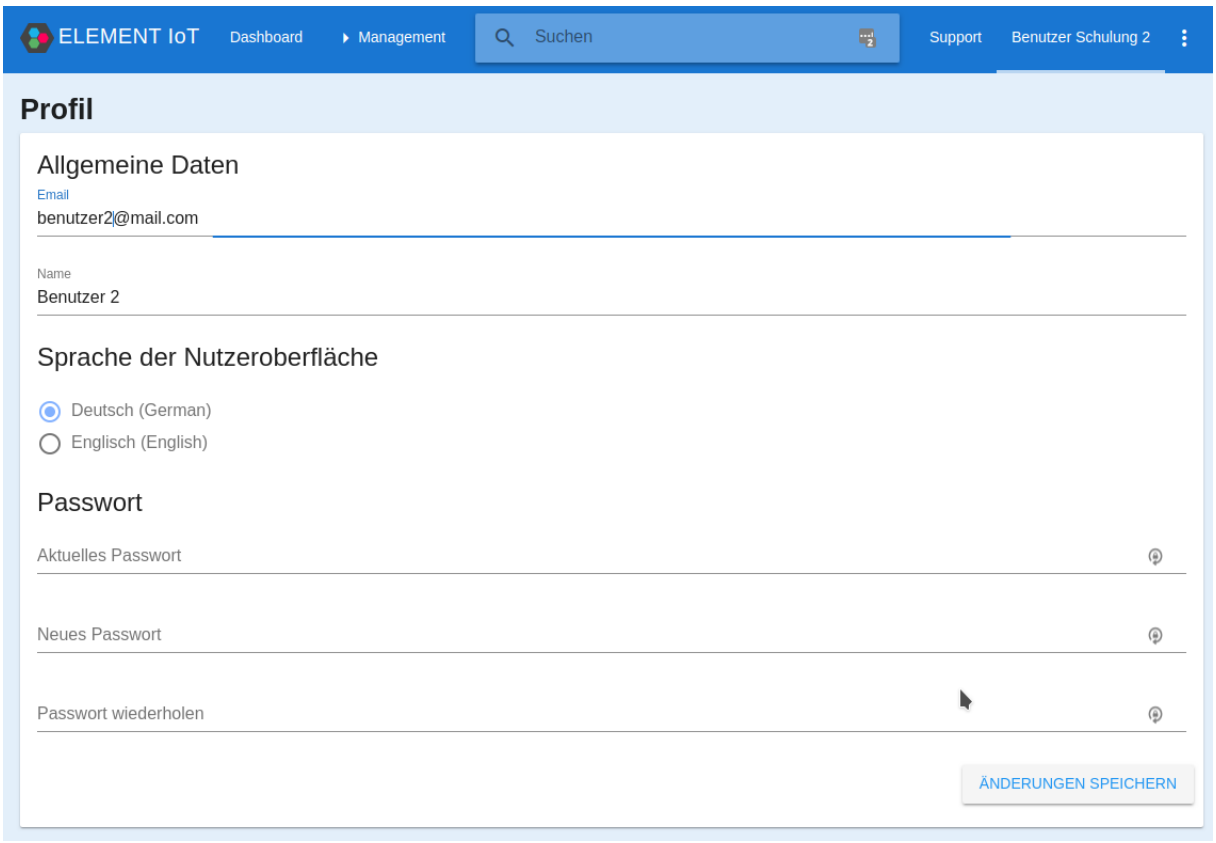


Abbildung 1.5: Screenshot

Bei Änderungen der Sprache kann es notwendig sein, die Seite im Browser neu zu laden.

1.11 Anpassung der Benutzeroberfläche

Sie haben innerhalb der ELEMENT IoT-Plattform - die passende Berechtigung vorausgesetzt - die Möglichkeit, die Farbgestaltung der Oberfläche anzupassen. Die Einstellungen lassen sich für jeden Mandanten gesondert festlegen.

Öffnen Sie den Bereich **EINSTELLUNGEN** und klicken Sie hier auf den Button **FARBSCHEMA ERSTELLEN**.

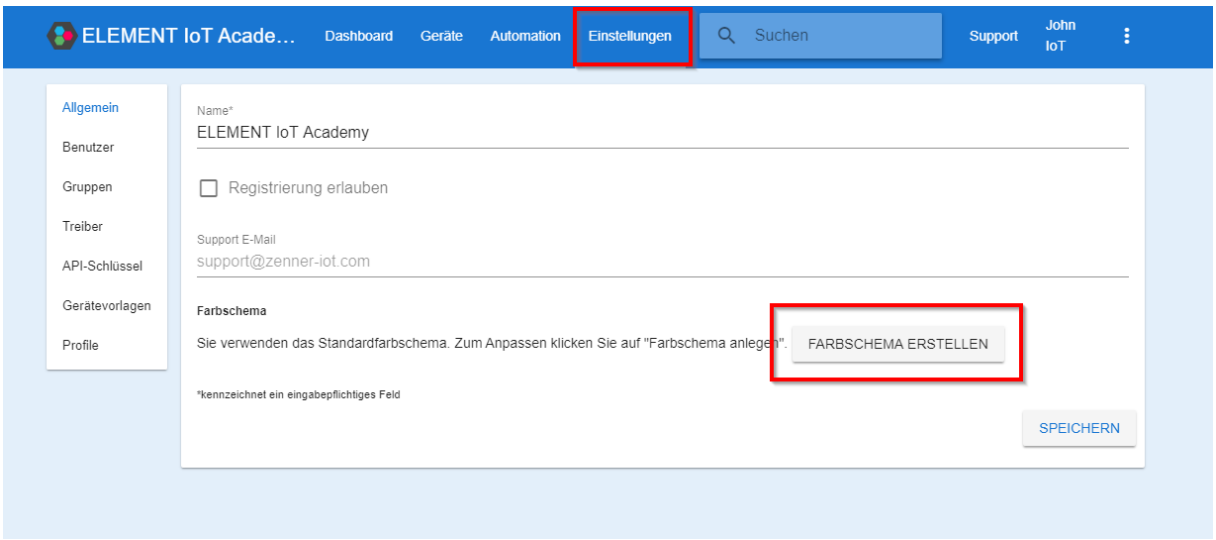


Abbildung 1.6: Screenshot

In der nachfolgenden Maske haben Sie nun die Möglichkeit, individuelle Farbeinstellungen vorzunehmen.

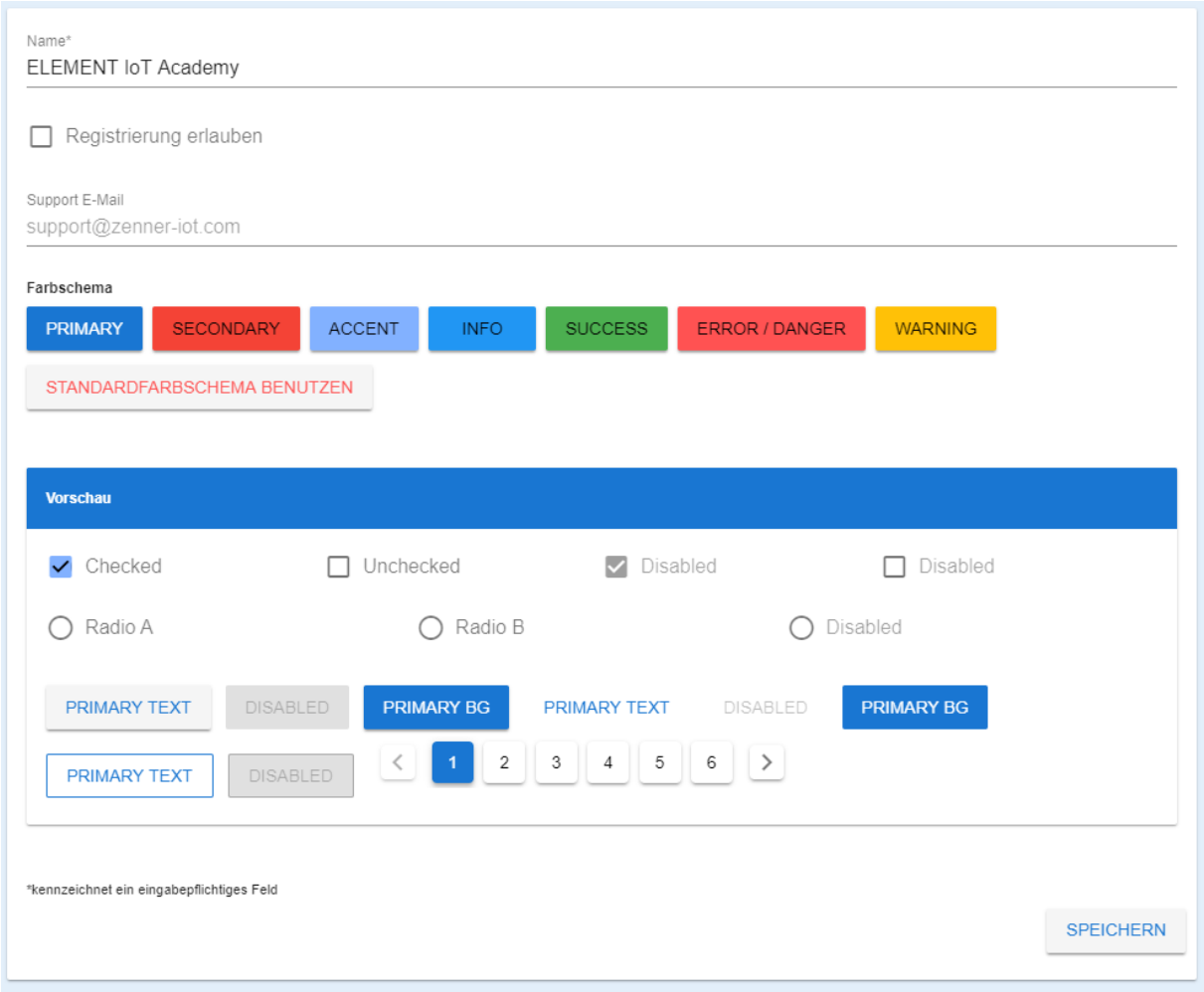


Abbildung 1.7: Screenshot

Möchten Sie zum Beispiel die primäre Farbe ändern, klicken Sie auf **PRIMARY** und wählen Sie aus dem Farbschema die gewünschte Farbe aus. Sie sehen die Änderung direkt an der Oberfläche.

Klicken Sie auf **SPEICHERN**, um die ausgewählten Einstellungen dauerhaft, für alle Nutzer des Mandanten, zu speichern.

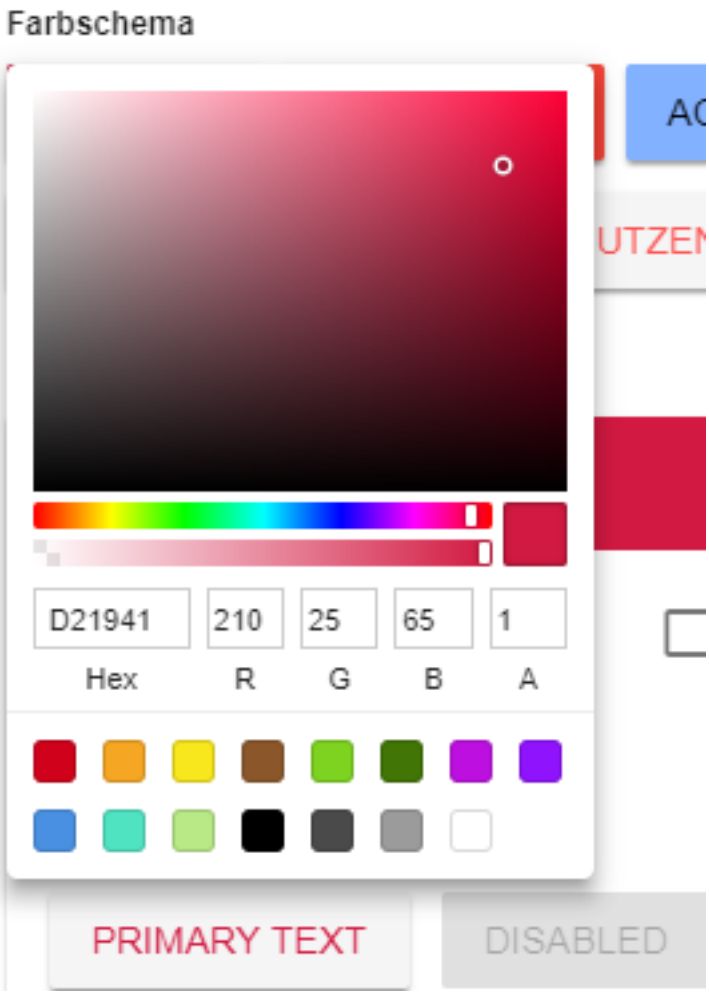


Abbildung 1.8: Screenshot

Sie haben jederzeit die Möglichkeit, die Standardeinstellungen wiederherzustellen. Klicken Sie dafür einfach auf den Button **STANDARDFARBSCHEMA BENUTZEN** und danach auf **SPEICHERN** .

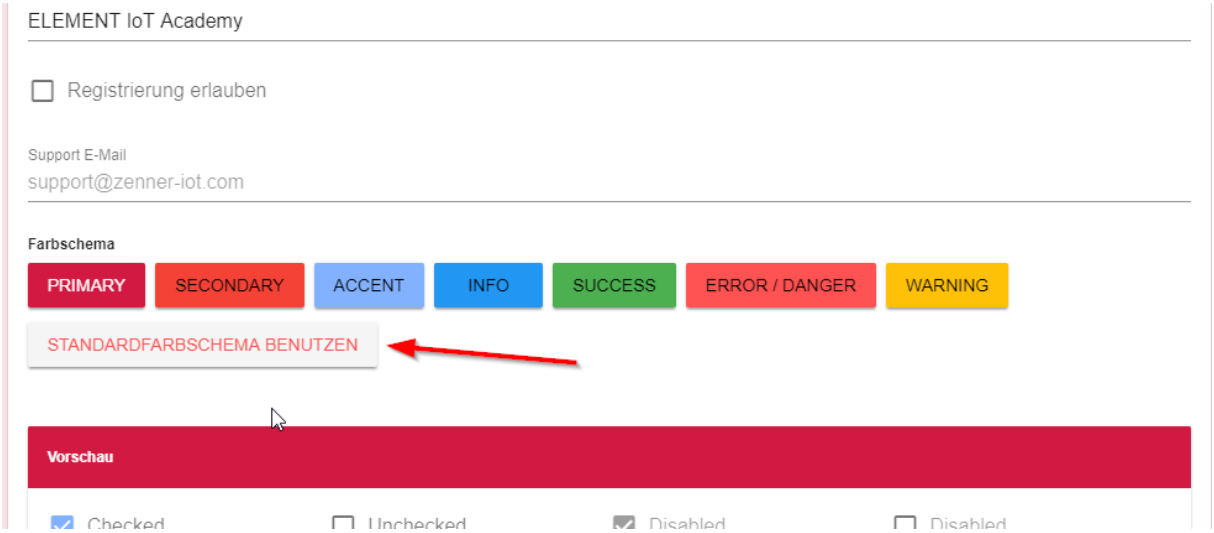


Abbildung 1.9: Screenshot

1.12 Einstellungen für Mandanten ändern

Pro Mandant können auf der ELEMENT IoT-Plattform einige Einstellungen konfiguriert werden, um die Plattform anzupassen. Die Einstellungen finden Sie im Bereich **Einstellungen** - **Allgemein**

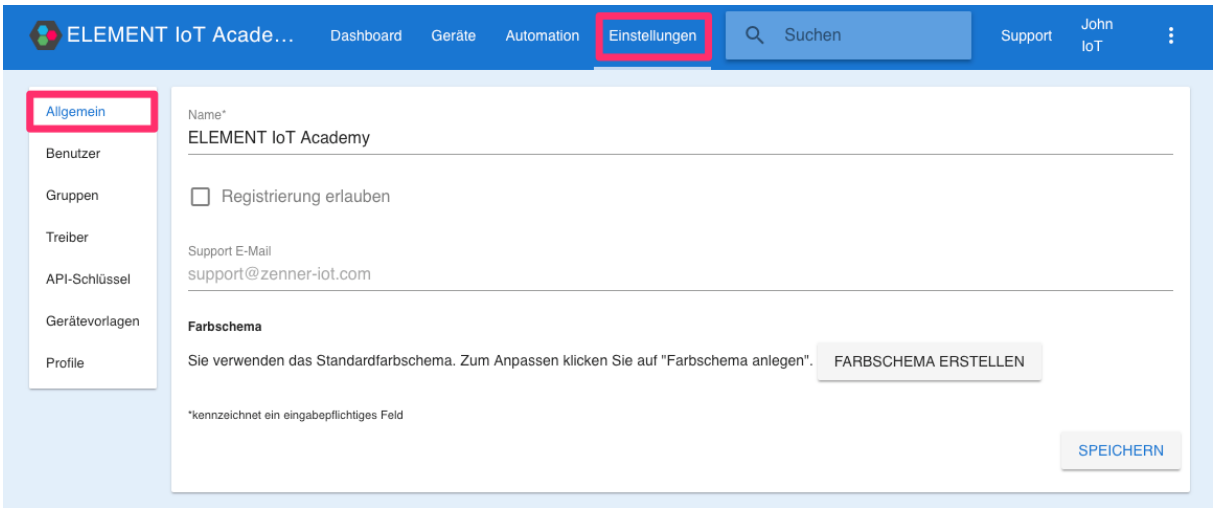


Abbildung 1.10: Screenshot

1.13 Registrierung erlauben / verbieten

Mit der Option Registrierung erlauben haben Sie die Möglichkeit, neuen Benutzern zu gestatten, sich selbständig für einen Mandanten zu registrieren. Diese Option ist sinnvoll, wenn es sich um ein öffentliches oder Testsystem handelt. In der Regel wird diese Option nicht aktiviert.

Abbildung 1.11: Screenshot

1.14 Konfigurieren der Support E-Mail-Adresse

An dieser Stelle können Sie eine E-Mail-Adresse hinterlegen, an welche E-Mails versendet werden sollen, wenn ein Benutzer auf den Link “Support” klickt.

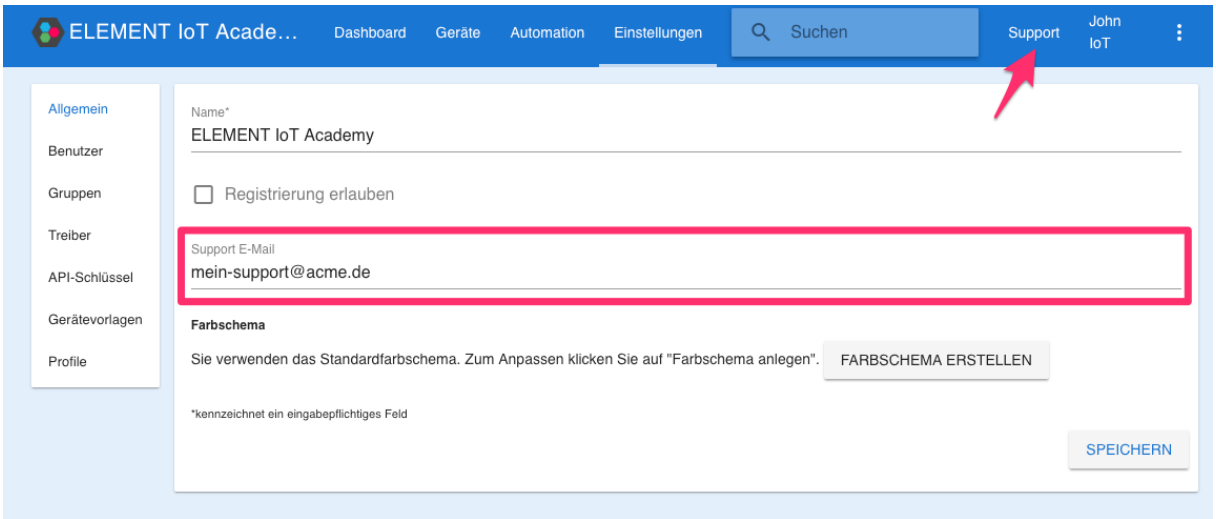


Abbildung 1.12: Screenshot

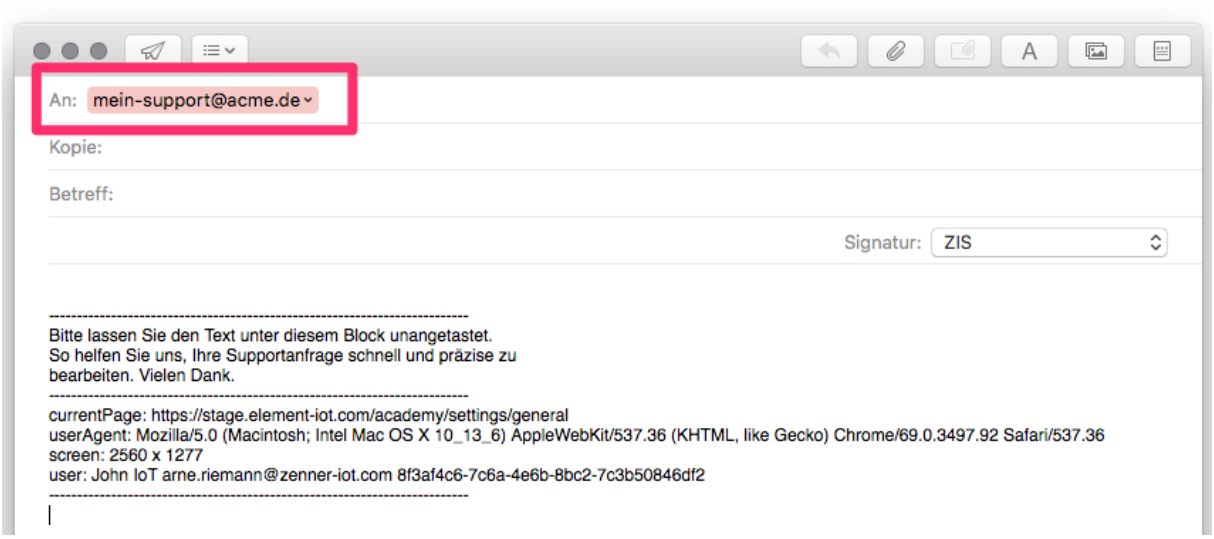


Abbildung 1.13: Screenshot

1.15 Farbschema

Siehe: How To Benutzeroberfläche

1.16 Changelog / Änderungen

Wenn Sie auf die “ ” in der Navigation klicken, erhalten Sie die aktuelle Versionsnummer der Plattform - beim anklicken gelangen Sie zu der Liste der letzten Änderungen an der Plattform (Changelog).

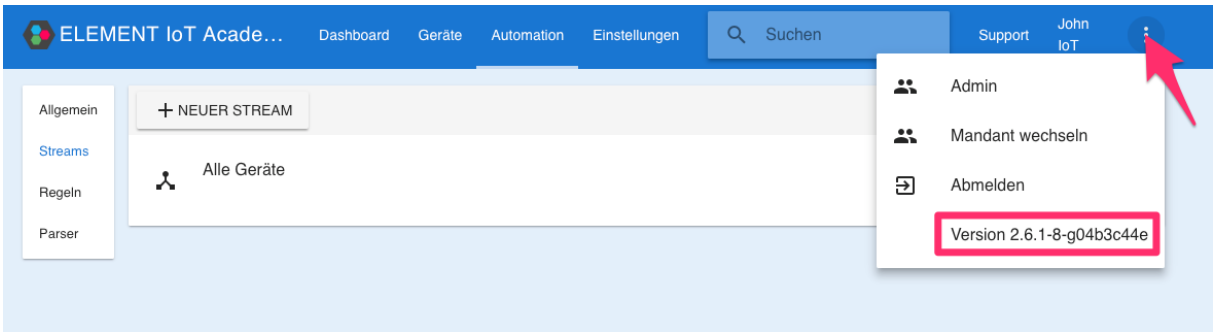


Abbildung 1.14: Screenshot

1.17 Suchen in der ELEMENT IoT-Plattform

Sie haben in der ELEMENT IoT-Plattform ab Version 2.5.x die Möglichkeit, nach folgenden Werten zu suchen:

- Benutzer
- Geräte / Sensoren
- Ordner
- Gerätevorlagen
- Parser
- Regeln
- Streams
- Profile

Zum Suchen geben Sie bitte in das Suchfeld den gewünschten Suchbegriff ein. Ist Ihnen die vollständige Bezeichnung nicht bekannt, können Sie an das Ende des Begriffs ein “*” anhängen.

Möchten Sie zum Beispiel nach einem Gerät mit dem Namen “Türkontaktsensor” suchen, können Sie den Suchbegriff “Türkontakt*” verwenden.

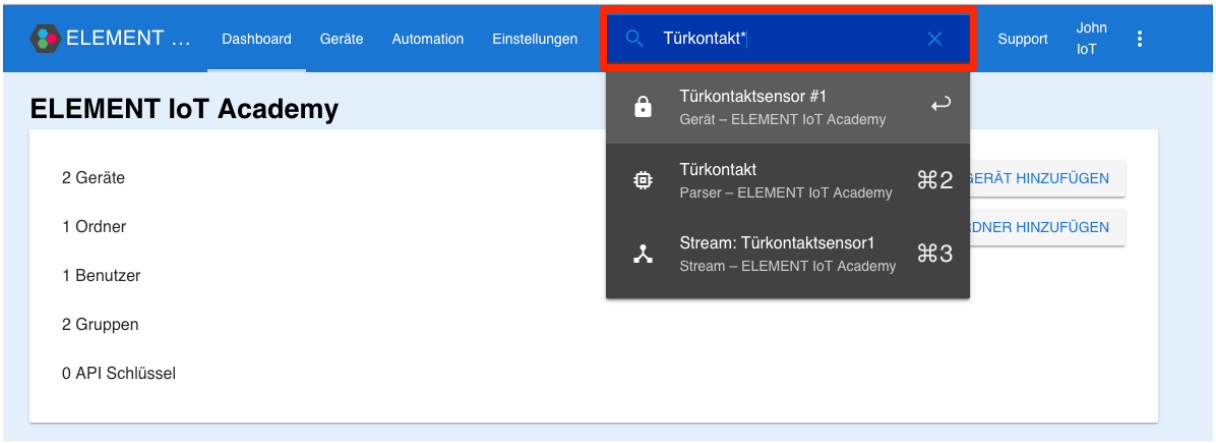


Abbildung 1.15: Suche

Achtung! Es handelt sich um eine Präfixsuche. Das bedeutet, es kann nur nach Begriffen gesucht werden, die mit Ihrer Sucheingabe beginnen; eine suche nach nach “*kontakt”, ist also nicht möglich.

2 Treiber für die Plattform

Treiber sind einer der Grundbausteine von ELEMENT IoT.

Treiber werden genutzt, um unterschiedliche Datenquellen bzw. Geräte mit unterschiedlichen Übertragungstechnologien an die ELEMENT IoT-Plattform anzubinden.

Momentan bietet die Plattform u.a. folgende Treiber an:

- ELEMENT LoRaWAN Network Server
- HTTP REST
- UDP (NB-IoT)
- Gateway Management
- Dummy

In diesem How To wird die Nutzung des Dummy Treibers und des Treibers für den ELEMENT LNS (LoRaWAN Netzwerk Server) behandelt.

2.1 Einrichten eines Dummy Treibers

Der Dummy Treiber kann für Testzwecke genutzt werden. Er ermöglicht das Erzeugen von Paketen in unterschiedlichen Varianten. Es ist möglich einen Festwert, einen Zufallswert, oder eine Sequenz in Schritten zu erzeugen. Auf diese Weise lassen sich zum Beispiel für Simulationen “Dummy LoRa-Pakete” erzeugen.

Als Beispiel erstellen wir einen Dummy Treiber, welcher die Daten eines Temperatursensors simuliert. Dieser soll alle 30 Sekunden einen zufälligen Wert im Bereich 20-36 (Grad) senden.

Zum Anlegen öffnen Sie bitte aus der Menüleiste den Bereich **Einstellungen** und wählen hier die Option **Treiber**.

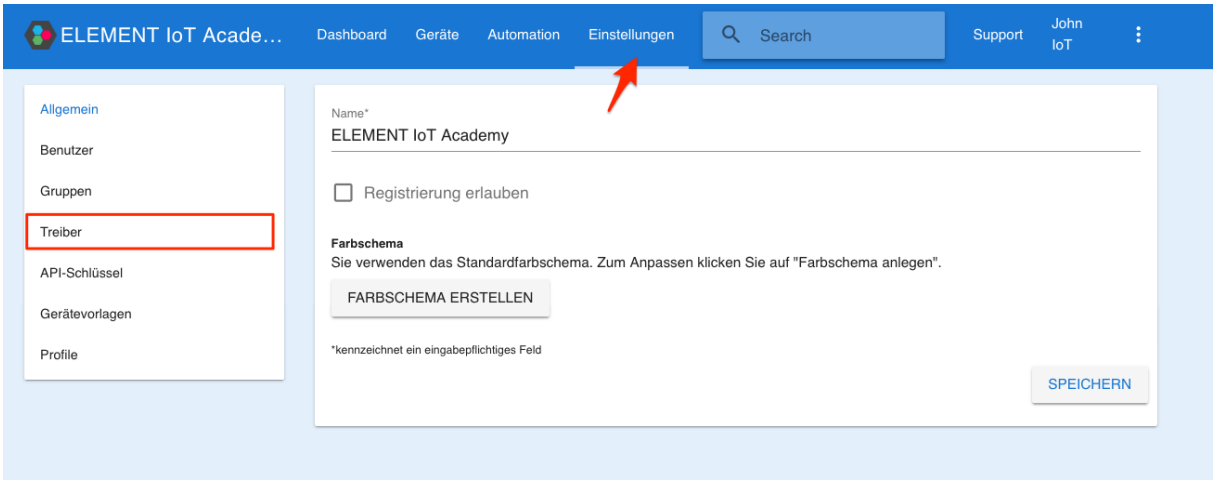


Abbildung 2.1: Screenshot

Klicken Sie in der folgenden Maske auf den Button **NEUEN TREIBER HINZUFÜGEN**.

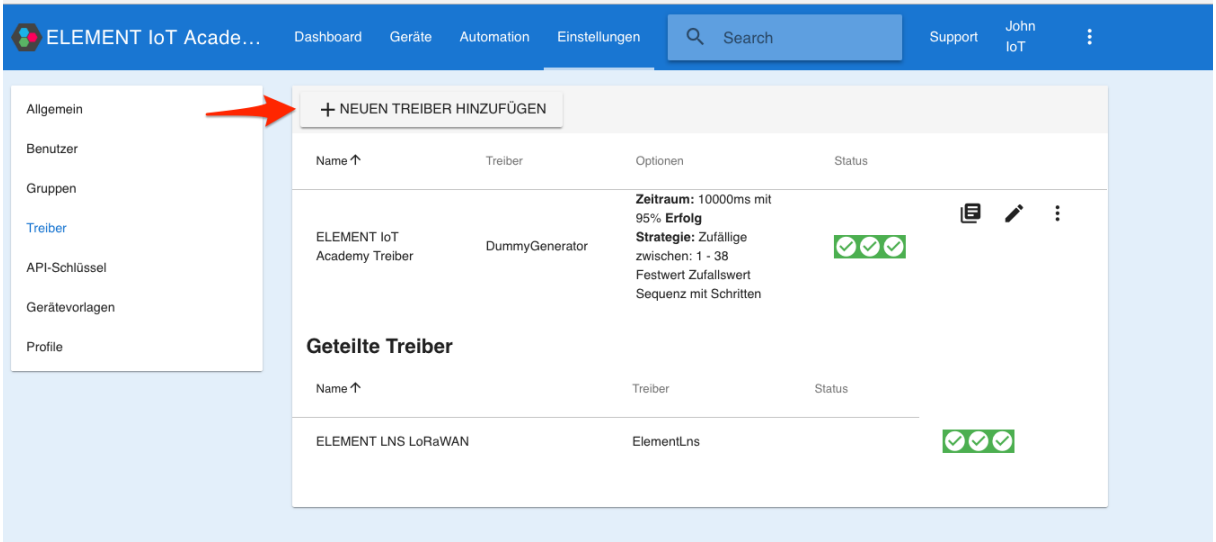


Abbildung 2.2: Screenshot

In der folgenden Maske nehmen Sie bitte folgende Einstellungen vor:

- Name: Dummy - Temperatur
- Dummy Generator (ZENNER IoT Solutions)
- Zufallswert
- Bereichsstart: 20
- Bereichsende: 36

- Zeitraum: 30000 Millisekunden (Umrechnung Millisekunden - Sekunden)
- Faktor: 1 (Faktor 2 würde die erzeugten Werte mit 2 multiplizieren)
- Prozentsatz erfolgreich: 100% (ein kleinerer Wert kann beispielsweise Ausfälle simulieren)

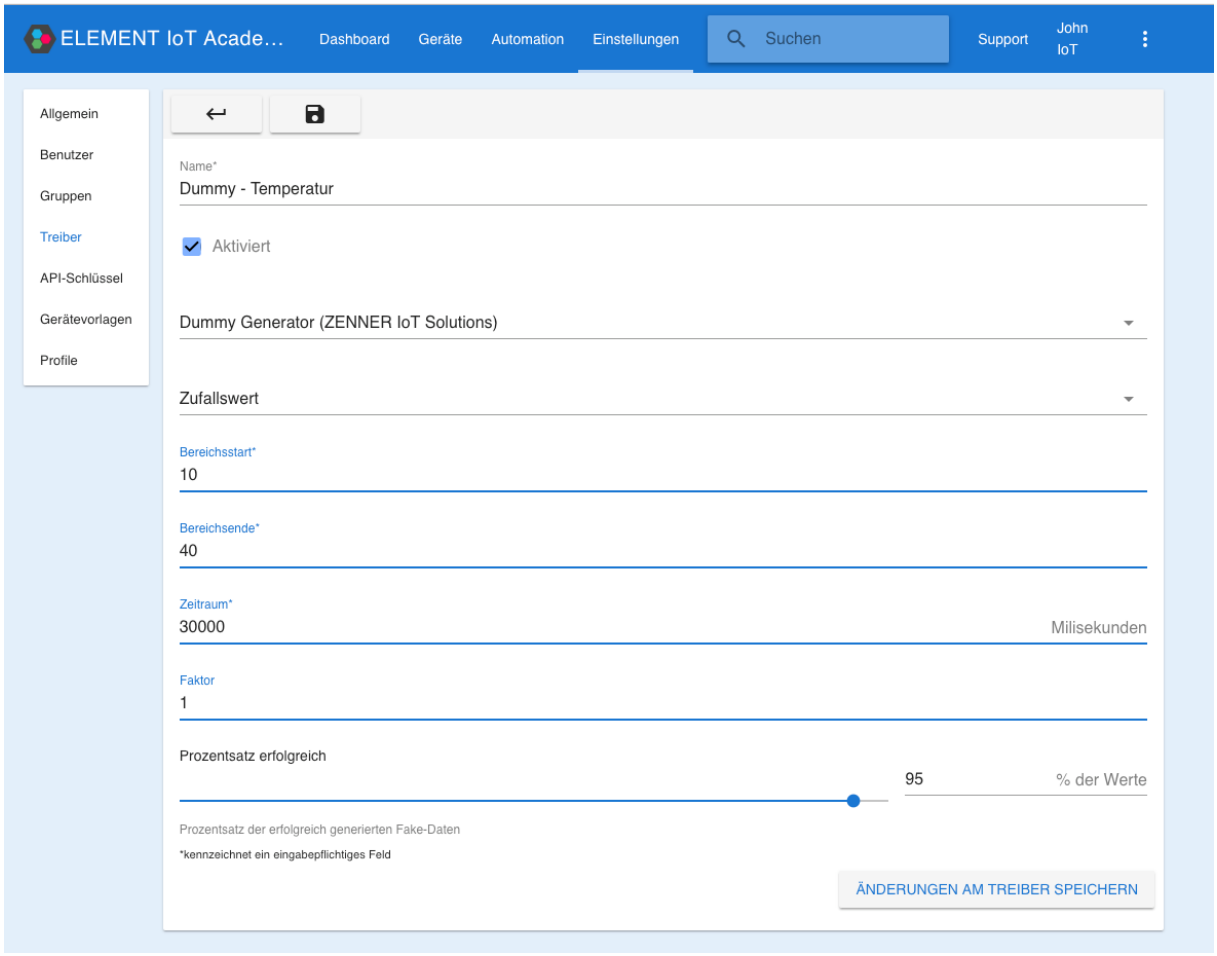


Abbildung 2.3: Screenshot

Klicken Sie nun auf den Button **NEUEN TREIBER HINZUFÜGEN**. In der Übersicht sollten Sie nun den Dummy Treiber sehen.

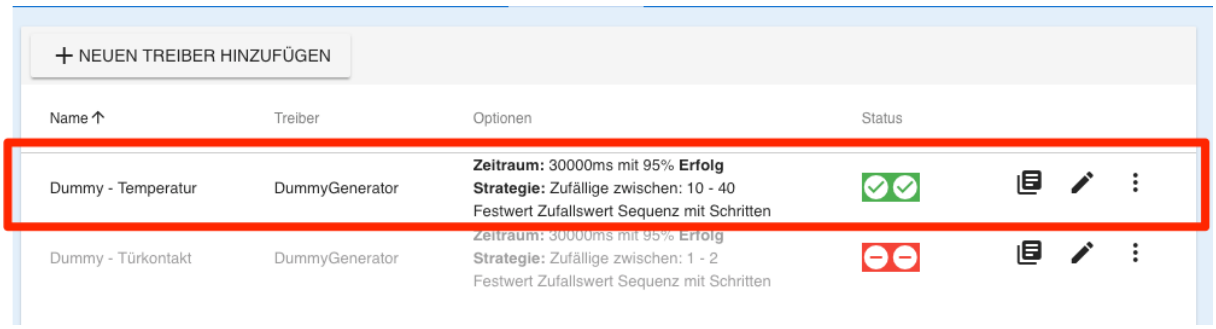


Abbildung 2.4: Screenshot

2.2 Einrichten ELEMENT LNS Treiber

Der ELEMENT LoRaWAN Network Server Treiber wird verwendet, um den gleichnamigen ELEMENT LNS an die ELEMENT IoT-Plattform anzubinden. Für die Einrichtung benötigen Sie folgende Informationen:

- URL der ELEMENT LNS API
- Auth-Token für die Authentifizierung des LNS
- Den Namen der im Message Broker verwendeten Queue

In der Regel ist dieser Treiber bereits angelegt, nur bei einer Neuinstallation ist dieser Prozess notwendig. Die benötigten Informationen erhalten Sie entweder von uns, oder dem entsprechenden verantwortlichen bei Ihnen.

Klicken Sie, wie im Abschnitt Dummy Treiber bereits beschrieben, auf **Einstellungen - Treiber - Neuen Treiber Hinzufügen**. Fügen Sie nun in das Formular die entsprechenden Werte ein.

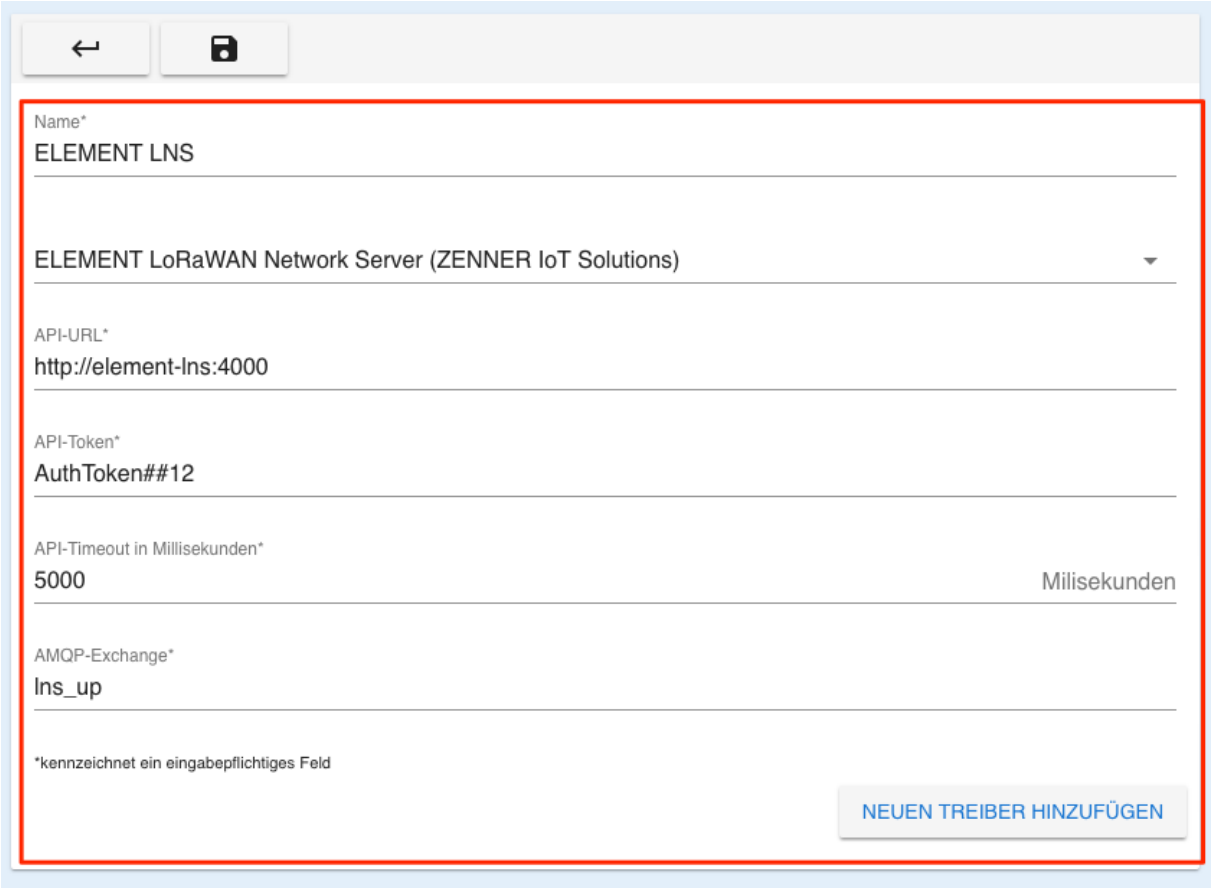


Abbildung 2.5: Screenshot

Nach dem Klicken auf **NEUEN TREIBER HINZUFÜGEN** ist der Treiber aktiv.

2.3 Treiber löschen, neustarten oder deaktivieren

Sie können Treiber über den Bereich **Einstellungen - Treiber** entfernen, neu starten oder vorrübergehend deaktivieren. Ein Neustart kann helfen, wenn Sie ein Fehlverhalten festellen (zum Beispiel wenn keine Pakete mehr eingehen).

Zum Aufrufen der Auswahl klicken Sie bitte auf das “ ⋮ ” Symbol.

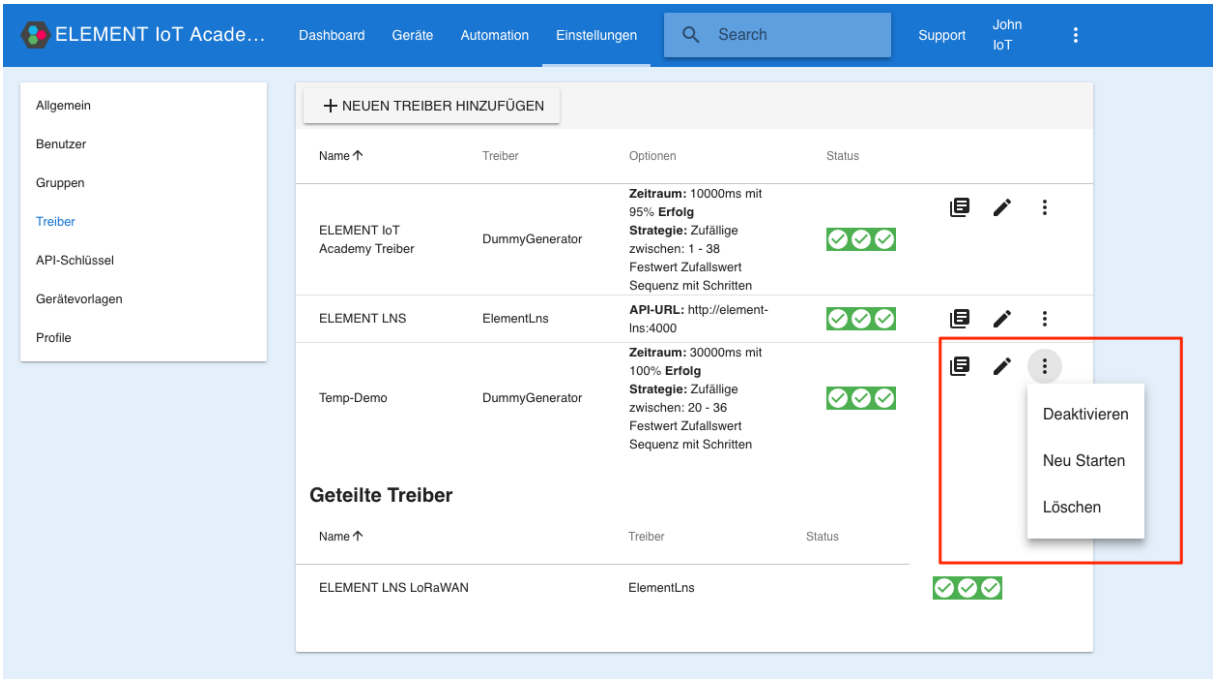


Abbildung 2.6: Screenshot

2.4 Treiber Protokoll anzeigen

Protokoll (Log-Messages) erhalten Sie in der Übersicht der Treiber (**Einstellungen - Treiber**). Klicken Sie hier auf das “ ” Symbol, um die Nachrichten zu sehen.

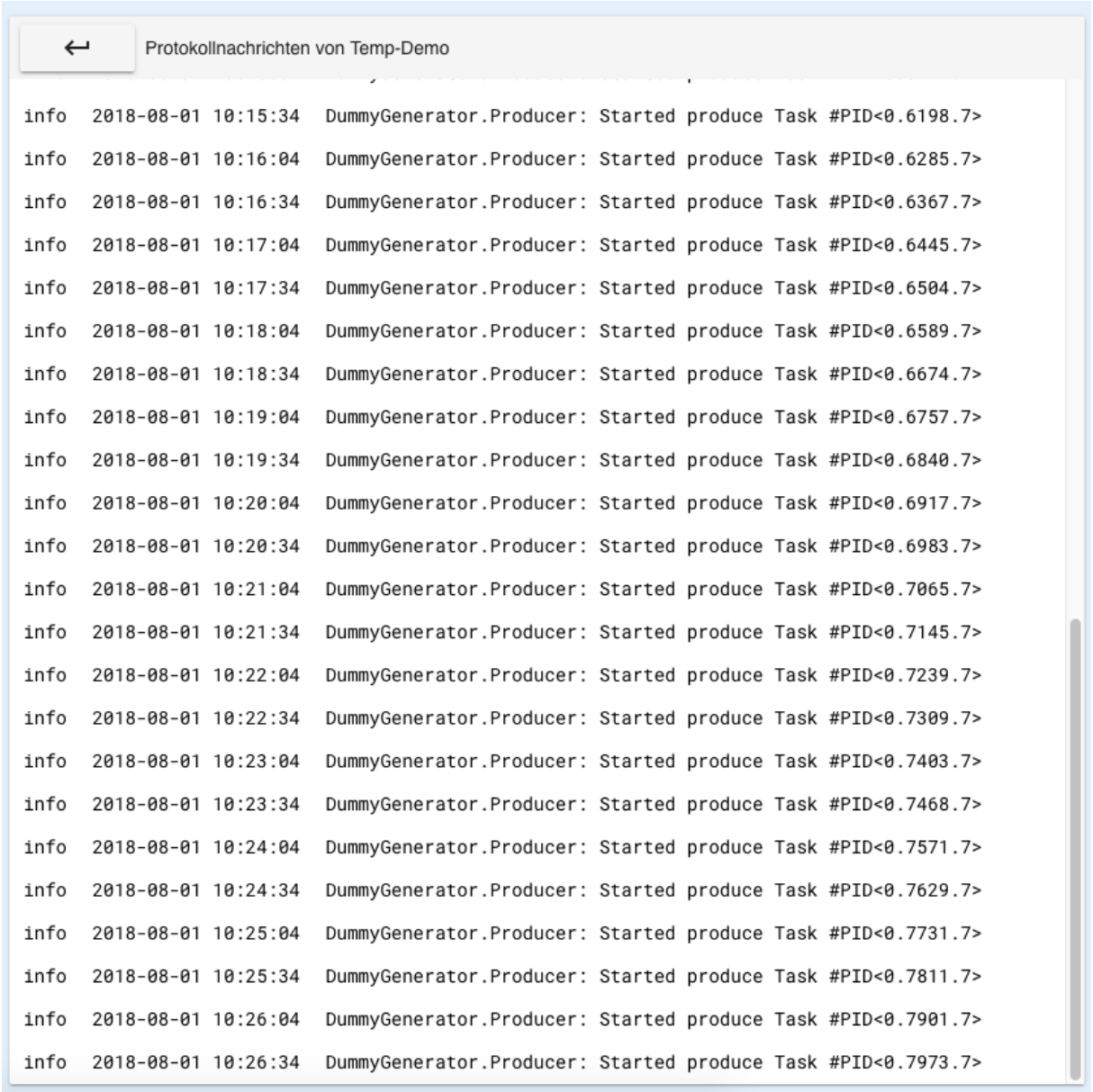


Abbildung 2.7: Screenshot

2.5 Geteilte Treiber

Ein geteilter Treiber kann über mehrere Mandanten genutzt werden. Diese Technik kommt zum Beispiel häufig beim ELEMENT LNS Treiber zum Tragen. Geteilte Treiber können nur direkt in der Datenbank der ELEMENT IoT-Plattform freigeschaltet werden. Dafür wird erst, wie in diesem How To beschrieben, der Treiber angelegt und dieser anschließend durch einen Datenbank-Administrator freigegeben.

3 Parser

Parser werden genutzt, um die von einem Gerät übertragenen Daten in ein für die ELEMENT IoT-Plattform lesbares Format zu übertragen. Um die Ausgestaltung von Parsern möglichst flexibel zu halten, werden diese in der Programmiersprache Elixir entwickelt.

Ein ausführliches Tutorial mit weiteren Details finden Sie unter folgendem Link: Parser Dokumentation. Parser, welche bereits durch ZENNER IoT Solutions für unterschiedliche Sensoren erstellt wurden, befinden sich öffentlich zugänglich bei GitHub. Hier können Sie auch eventuell vorhandene Fehler melden.

3.1 Anlegen eines Parsers

Um einige Beispiele in den vorhandenen How To's durchführen zu können, legen wir in diesem How To einen Beispiel-Parser für einen Temperatursensor an. Der Parser baut auf dem entsprechenden "Dummy-Treiber" aus dem How To für Treiber auf und wird ebenfalls im How To für Regeln und Streams verwendet.

Zum Erzeugen eines neuen Parsers öffnen Sie bitte in der Navigationsleiste den Bereich **AUTOMATION** und hier den Link **Parser**. Zum Anlegen klicken Sie bitte auf **NEUEN PARSER HINZUFÜGEN**

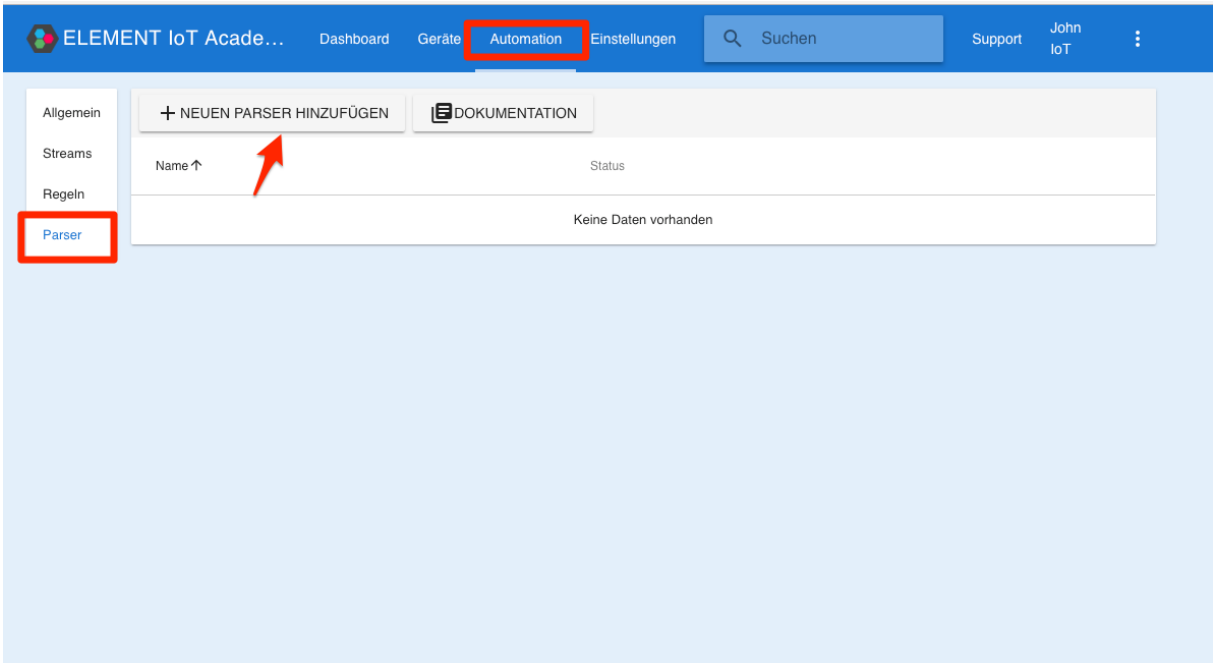


Abbildung 3.1: Screenshot

3.2 Beispiel Parser Temperatursensor

Geben Sie als Namen “Temperatursensor” in das Feld “Name” ein, und kopieren Sie folgenden Parser-Code in den Bereich “Code”.

```
defmodule Parser do
  use Platform.Parsing.Behaviour

  def parse(event, _meta) do
    %{temperatur: get_in(event, ["payload"])}
  end

  def fields do
    [
      %{
        field: "temperatur",
        display: "Temperatur",
        unit: "C"
      }
    ]
  end
end
```

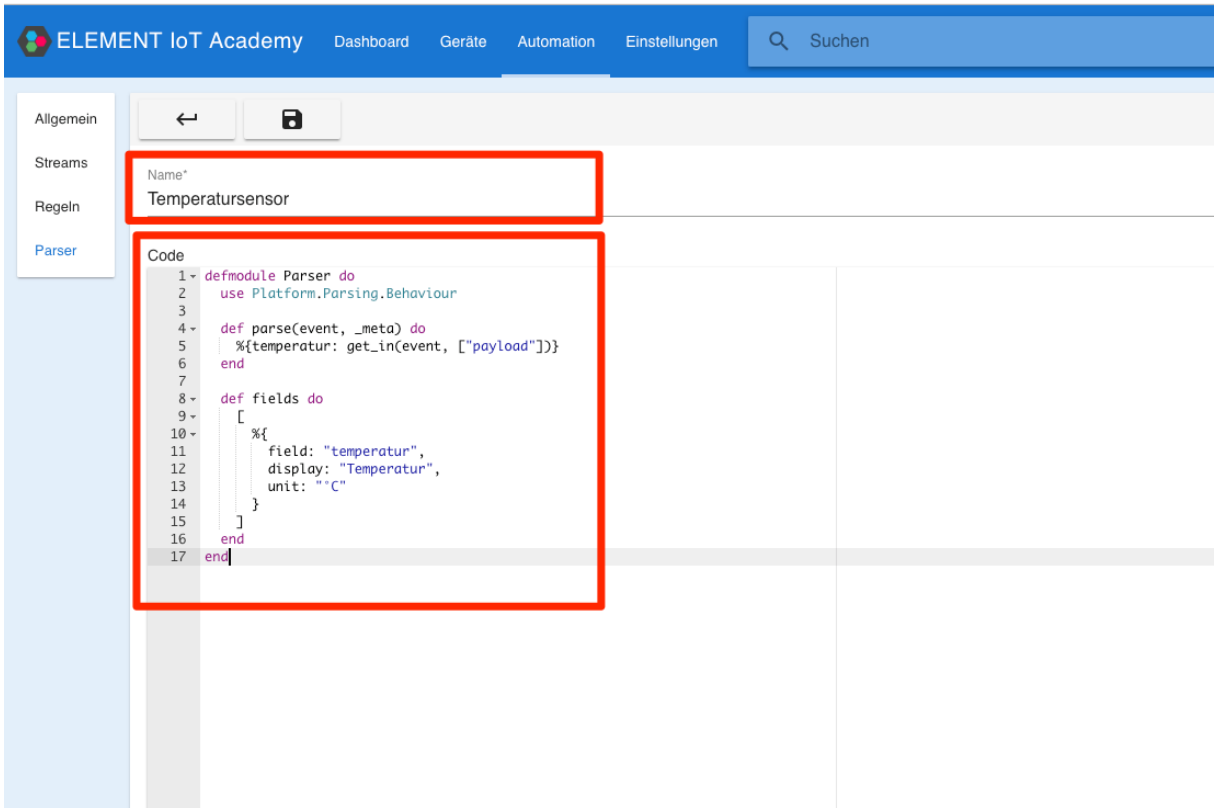


Abbildung 3.2: Screenshot

Der Parser nimmt die in diesem Fall vom “Temperatur Dummy-Treiber” übermittelten Daten und erzeugt daraus ein lesbares Format mit dem Feld “Temperatur” und Maßeinheit “Celsius”. Zum Speichern des Parsers, klicken Sie bitte auf **NEUEN PARSER HINZUFÜGEN**.

3.3 Löschen von Parsern

Um einen Parser zu löschen öffnen Sie die Übersicht und klicken Sie auf das Papierkorb-Symbol.

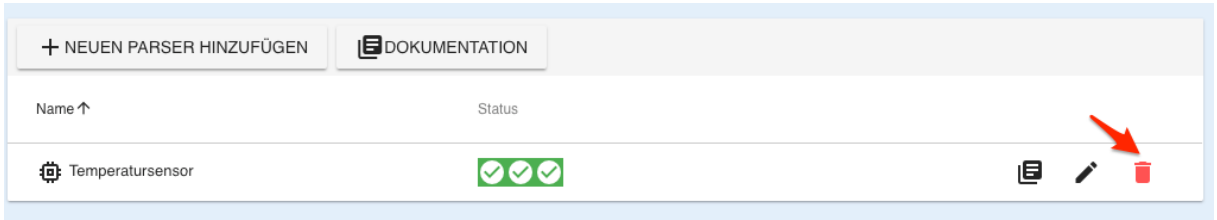


Abbildung 3.3: Screenshot

3.4 Protokoll anzeigen

Um zu kontrollieren ob ein Parser richtig funktioniert, können Sie in der Übersicht das Protokoll (Logfile) des Parsers ansehen, klicken Sie dafür auf das entsprechende Symbol.

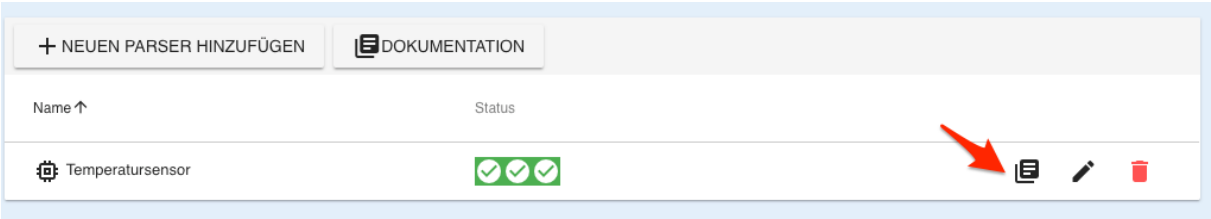
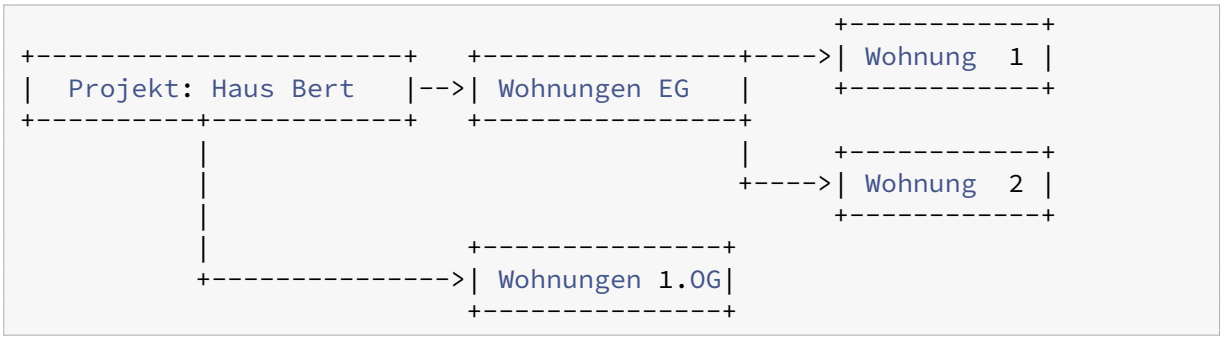


Abbildung 3.4: Screenshot

4 Einführung Anlegen von Ordnerstrukturen

Die ELEMENT IoT-Plattform erlaubt das Anlegen von Ordnerstrukten, ähnlich wie Sie es von einem Dateisystem kennen. So haben Sie die Möglichkeit, Ihre Geräte und Gateways zu strukturieren. Ordner spielen auch für die Vergabe von Berechtigungen eine entscheidende Rolle, so können Sie beispielsweise Benutzer nur auf Geräte in einem Ordner berechtigen.

In diesem How To werden wir eine mehrstufige Ordnerstruktur erzeugen. Die Zielstruktur sieht wie folgt aus:



4.1 Anlegen Ordner Ebene 1

Öffnen Sie den Bereich **Geräte** und klicken Sie hier auf das **Plus**-Zeichen, um einen neuen Ordner zu erzeugen.

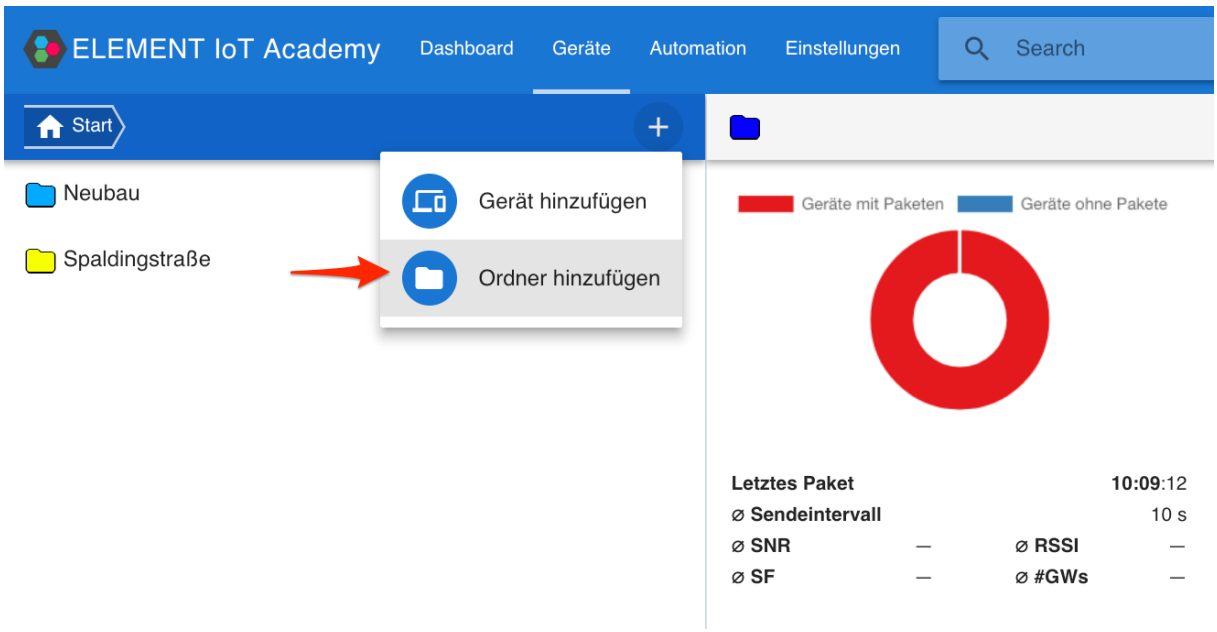


Abbildung 4.1: Screenshot

In folgenden Maske vergeben Sie bitte einen Namen (Projekt: Haus Bert) und eine Beschreibung für den neuen Ordner. Sie können zusätzlich eine beliebige Farbe wählen, in welcher der Ordner in der Auswahl dargestellt wird. Das Feld **Profile** wird an dieser Stelle nicht benötigt und in einem weiteren How To näher erklärt. Klicken Sie zum Abschluss auf den Button **ORDNER ERSTELLEN**.

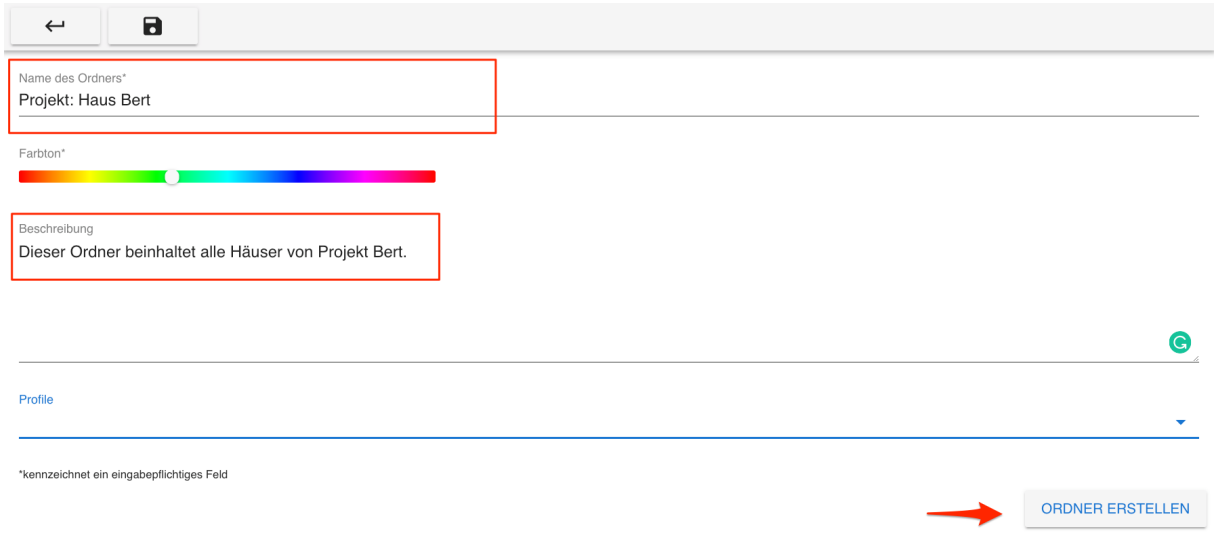


Abbildung 4.2: Screenshot

4.2 Anlegen Ordner Ebene 2 + 3

Haben Sie den Ordner auf der obersten Ebene angelegt, können Sie mit der zweiten Ebene beginnen. Öffnen Sie den neuen Ordner “Projekt: Haus Bert” und klicken Sie hier erneut auf das **Plus**-Zeichen, um einen Ordner innerhalb des bestehenden Ordners zu erzeugen.

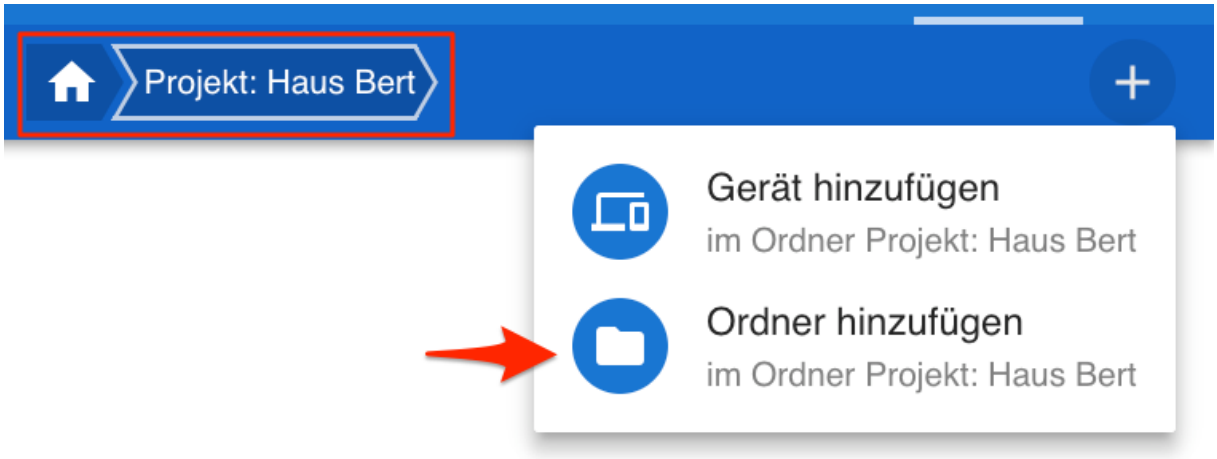


Abbildung 4.3: Screenshot

Erzeugen Sie hier 2 weitere Ordner mit den Bezeichnungen “Wohnungen EG” und “Wohnungen 1.OG”. Innerhalb des Ordners “Wohnungen EG” können Sie nun eine weitere Ebene für die Wohnungen erzeugen.

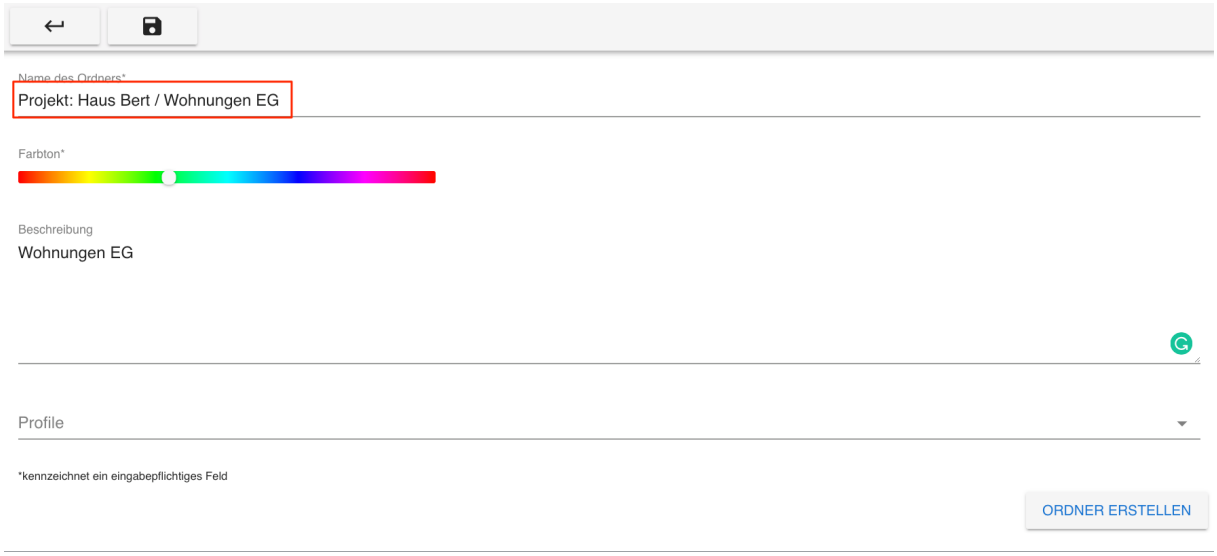


Abbildung 4.4: Screenshot

5 Geräte anlegen und verwalten

In diesem How To wird Ihnen Schritt für Schritt gezeigt, wie Sie neue Geräte in der ELEMENT IoT-Plattform anlegen können. Das How To baut auf folgenden weiteren How To's auf:

- Dummy-Treiber Temperatur (siehe How To: Treiber)
- Parser für Temperatursensor (siehe How To: Parser erstellen)
- Ordnerstruktur für Wohneinheiten (siehe How To: Anlegen von Ordnerstrukturen)

5.1 Ein neues Gerät anlegen

Um ein neues Gerät anzulegen, öffnen Sie bitte in der Navigationsleiste den Bereich **Geräte** und navigieren Sie in einen Ordner. Klicken Sie hier nun auf das **Plus**-Zeichen, um ein neues Gerät anzulegen.

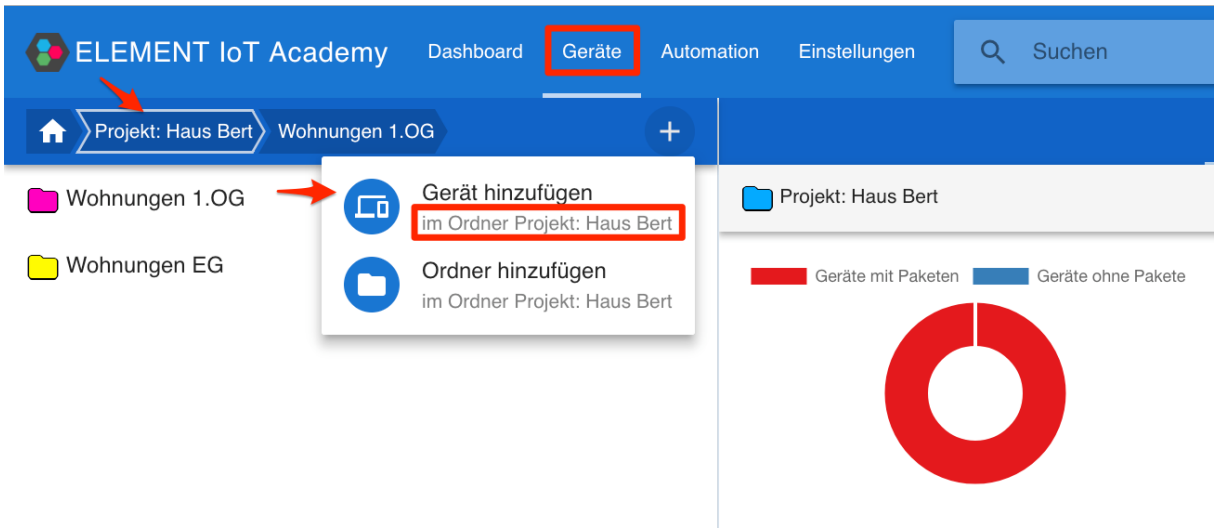


Abbildung 5.1: Screenshot

Das Anlegen von neuen Geräten erfolgt in mehreren Schritten. Bitte vergeben Sie im ersten Schritt einen eindeutigen Namen, wählen Sie den entsprechenden Ordner, in welchem das Gerät angelegt werden soll sowie den entsprechenden Parser.

Wenn gewünscht haben Sie weiterhin die Möglichkeit, ein Symbol für die bessere Identifizierung des Gerätetyps auszuwählen.

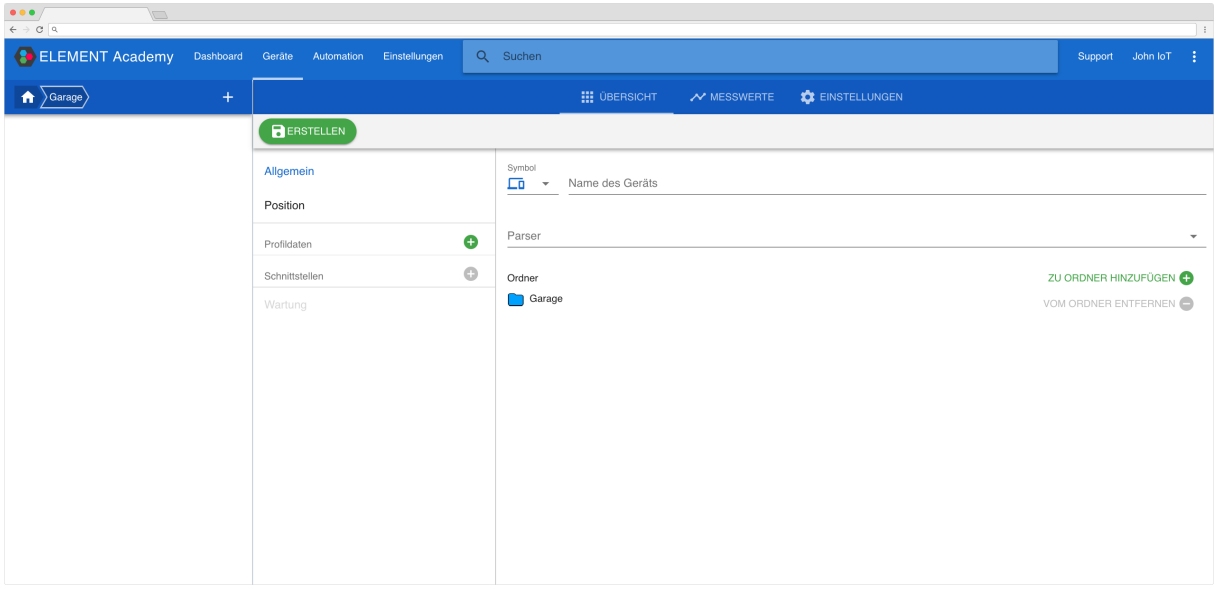


Abbildung 5.2: Screenshot

Bestätigen Sie diesen Schritt über den Button 

.

Im nächsten Schritt können Sie die Position des Gerätes festlegen. Klicken Sie dafür auf den gewünschten Punkt auf der Karte.

Möchten Sie verhindern, dass die Position durch etwaige GPS Daten des Gerätes überschrieben wird, können Sie dieses durch das betätigen des “  ” Symbols erreichen.

Zum Speichern der Position, klicken Sie bitte auf den Button “SPEICHERN”.

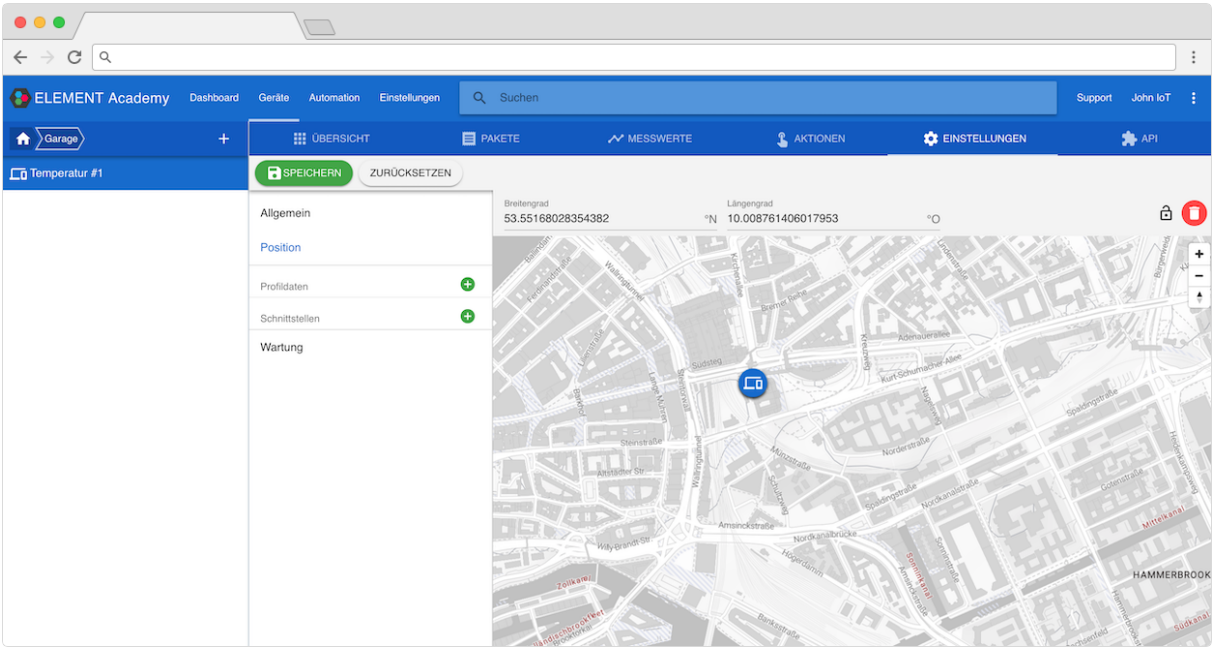


Abbildung 5.3: Screenshot

Haben Sie bereits Profile für Geräte angelegt, können Sie diese im nächsten Schritt zuweisen.

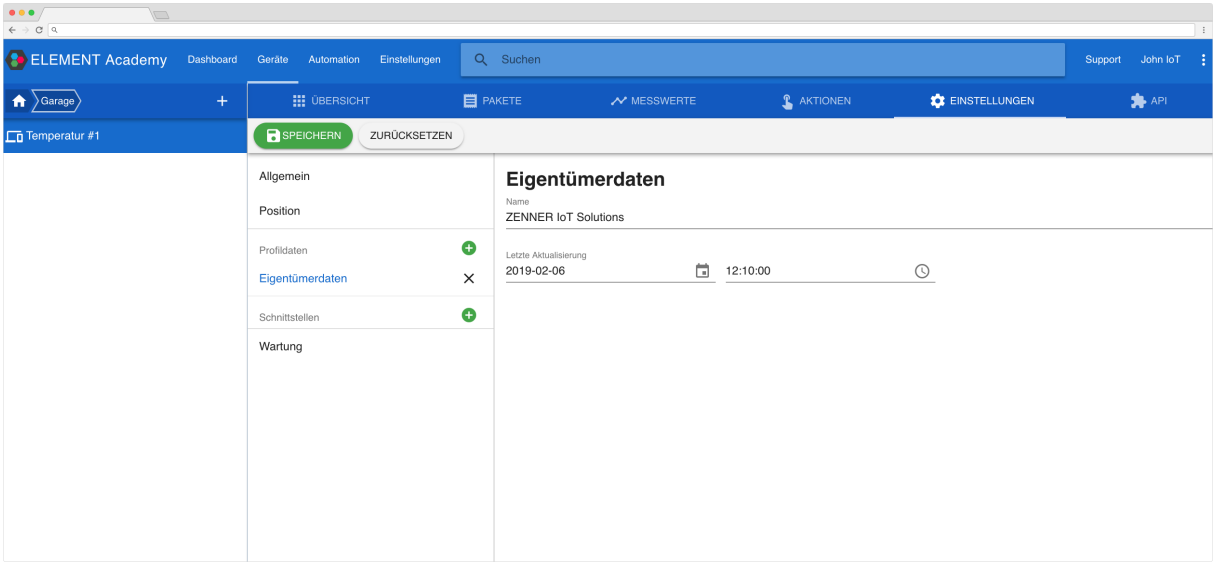


Abbildung 5.4: Screenshot

Im letzen Schritt können Sie dem Gerät eine entsprechende Schnittstelle zuordnen.

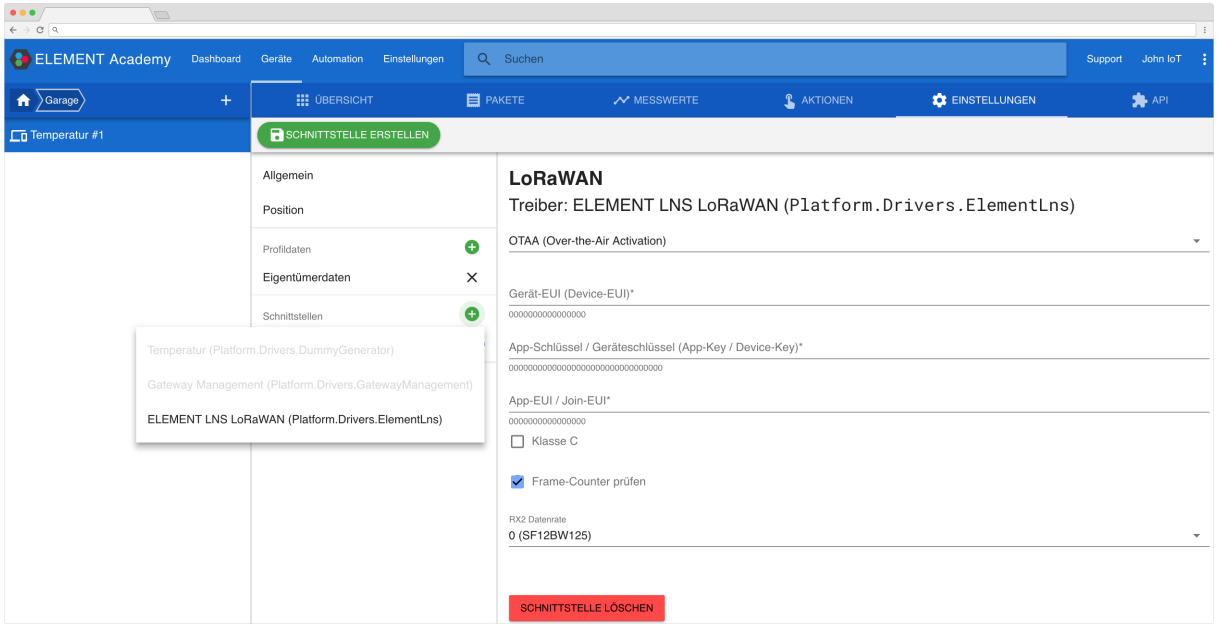


Abbildung 5.5: Screenshot

5.2 Übersicht

Direkt nach dem Anlegen des Gerätes gelangen Sie auf die Übersichtsseite. Hier erhalten Sie einen ersten Überblick über das erstellte Gerät.

- Lokation, wenn eingestellt oder GPS Daten gesendet werden
- Statistiken über eingehende Pakete
- Ordner, in welchem sich das Gerät befindet
- Konfigurierte Schnittstellen

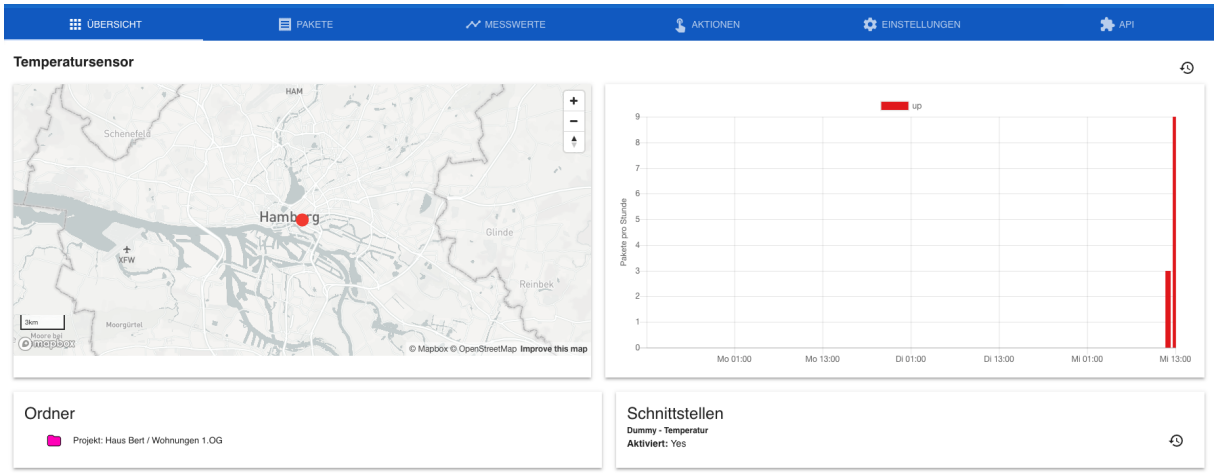


Abbildung 5.6: Screenshot

5.3 Pakete

Im Abschnitt **Pakete** sehen Sie alle vom Gerät empfangenen Pakete, in diesem Beispiel sind dies Daten vom Dummy-Treiber. Sie können sich hier durch das Klicken auf die entsprechenden Symbole weitere Details zu den einzelnen Paketen anzeigen lassen.

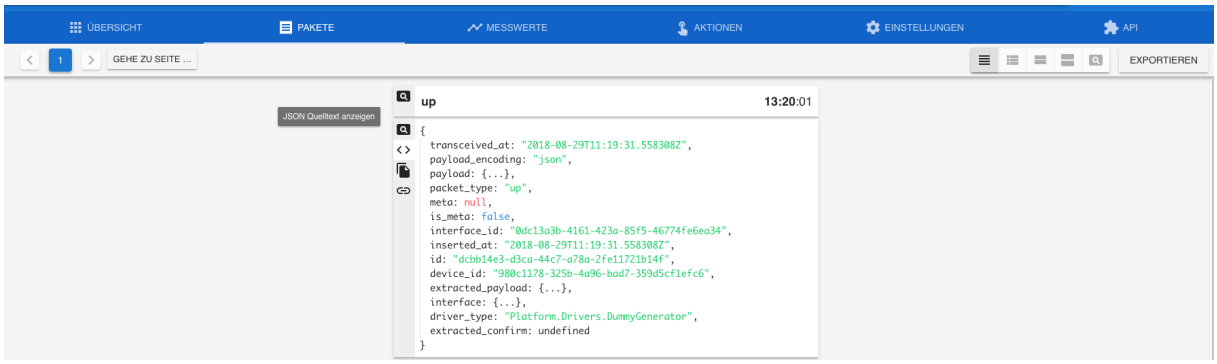


Abbildung 5.7: Screenshot

Sie haben hier auch die Möglichkeit sich das Paket im “JSON-Format” direkt in die Zwischenablage zu kopieren.

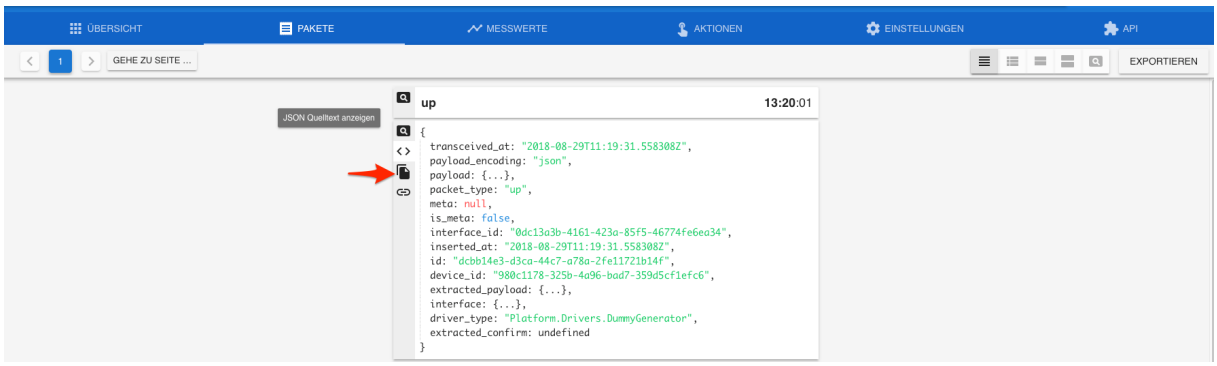


Abbildung 5.8: Screenshot

Möchten Sie den Detailgrad der Liste ändern, finden Sie hierfür am oberen Rand der Ansicht eine Vielzahl von Möglichkeite, sowie die Option zum Exportieren der Pakete in eine CSV-Datei (siehe How To: Export von Paketen).

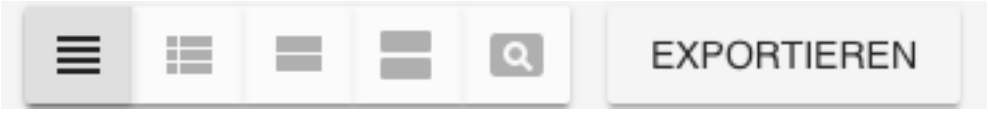


Abbildung 5.9: Screenshot

5.4 Messwerte

Im Bereich Messwerte finden Sie die aus den Pakten durch den Parser bearbeiteten Messwerte des Geräts. Hier haben Sie folgenden Möglichkeiten der Darstellung:

5.5 Graphen

Darstellungen der Messwerte als Graph. Standardmäßig werden folgende Werte dargestellt:

- Average
- Median
- 95th perenctille
- 97th perenctille
- Total

Abbildung 5.10: Screenshot

Sie haben die Möglichkeit, die Graphen Ihren Wünschen entsprechend anzupassen. Möchten Sie zum Beispiel alle Graphen bis auf den Average/Durchschnitt entfernen, klicken Sie auf den Button **BEARBEITEN** und entfernen Sie die nicht gewünschten Graphen.

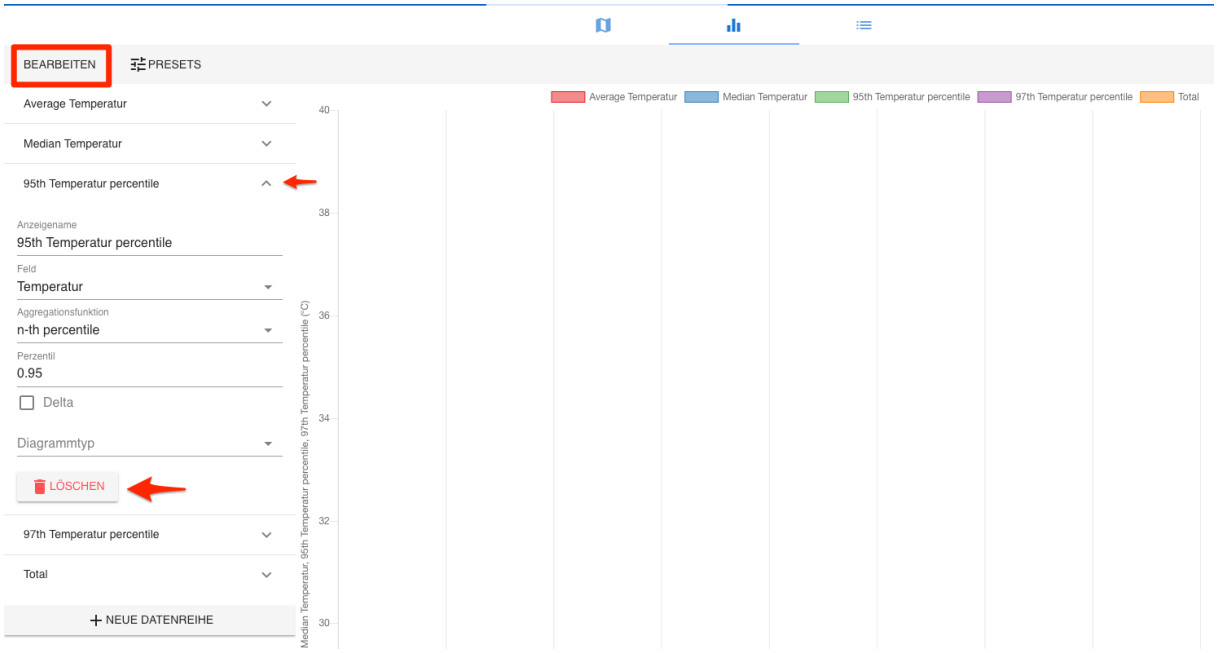


Abbildung 5.11: Screenshot

Nun können Sie diese Konfiguration als neues Preset speichern und bei Bedarf abrufen.

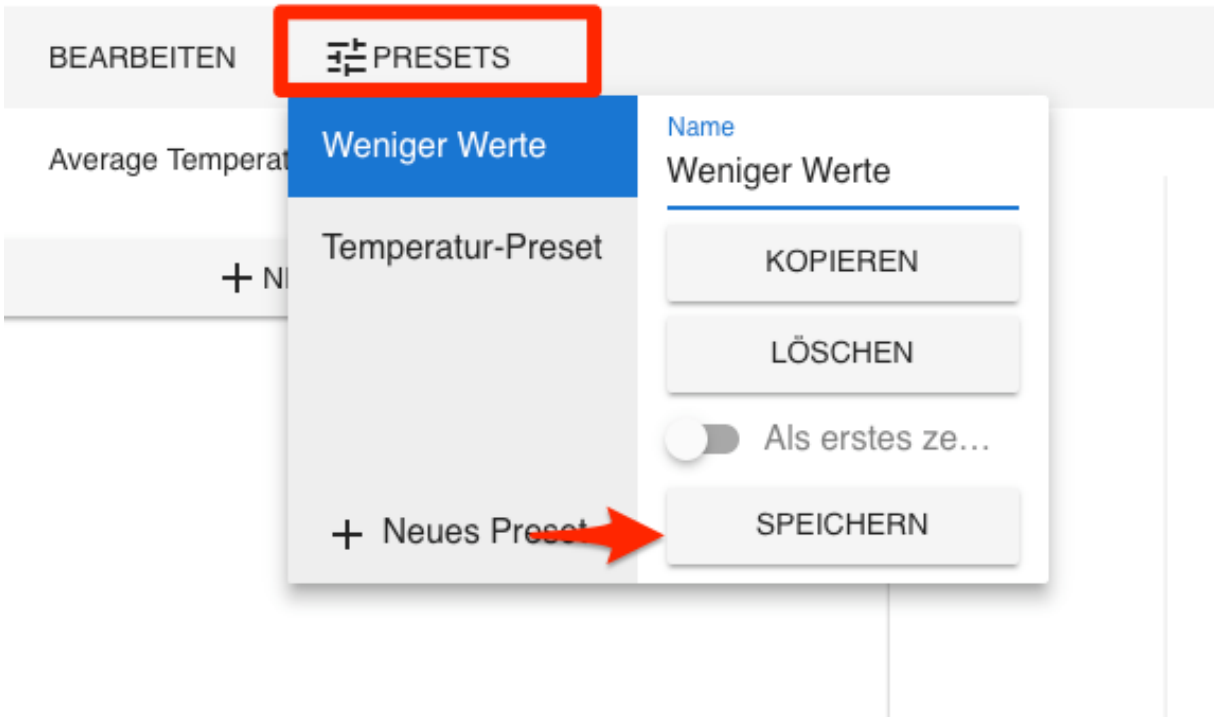


Abbildung 5.12: Screenshot

Über den Button **ZEITBEREICH** können Sie die Darstellung auf einen Zeitraum eingrenzen. Neben bereits vordefinierten Zeitspannen können Sie auch selber einen Zeitraum festlegen.

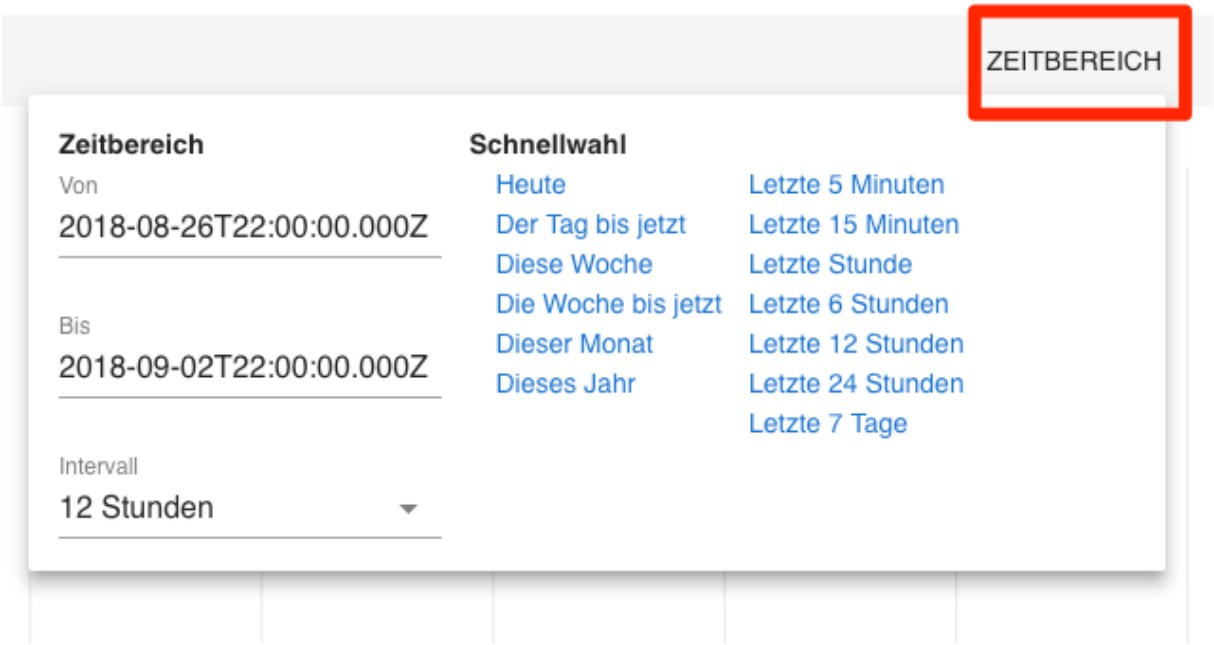


Abbildung 5.13: Screenshot

5.6 Karte

Im Bereich Karte sehen Sie den Standort des Gerätes, wenn dieses eine GPS Position sendet oder Sie den Standort beim Anlegen in den Einstellungen hinterlegt haben.

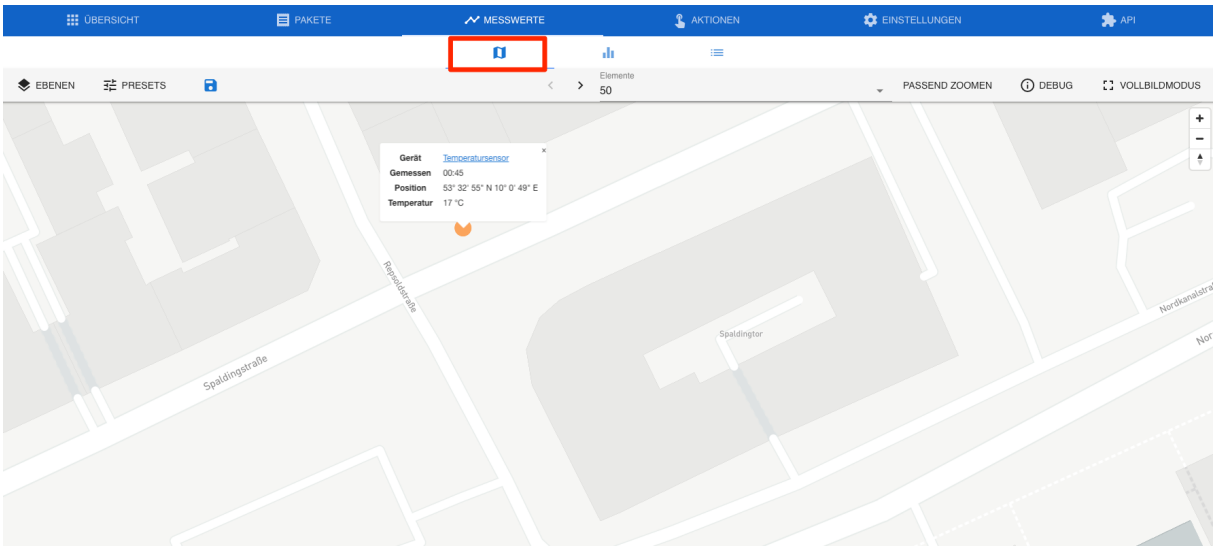


Abbildung 5.14: Screenshot

Ähnlich wie bei den Graphen haben Sie die Möglichkeit, die Darstellung der Karte individuell anzupassen.

5.7 Liste

Die Liste ist quasi die klassische Ansicht für Messwerte, hier sehen Sie alle Werte, welche durch den Parser definiert wurden. Wie auch bei den Paketen haben Sie die Möglichkeit, die Messwerte in eine CSV-Datei zu exportieren (siehe How To: Export von Messwerten).

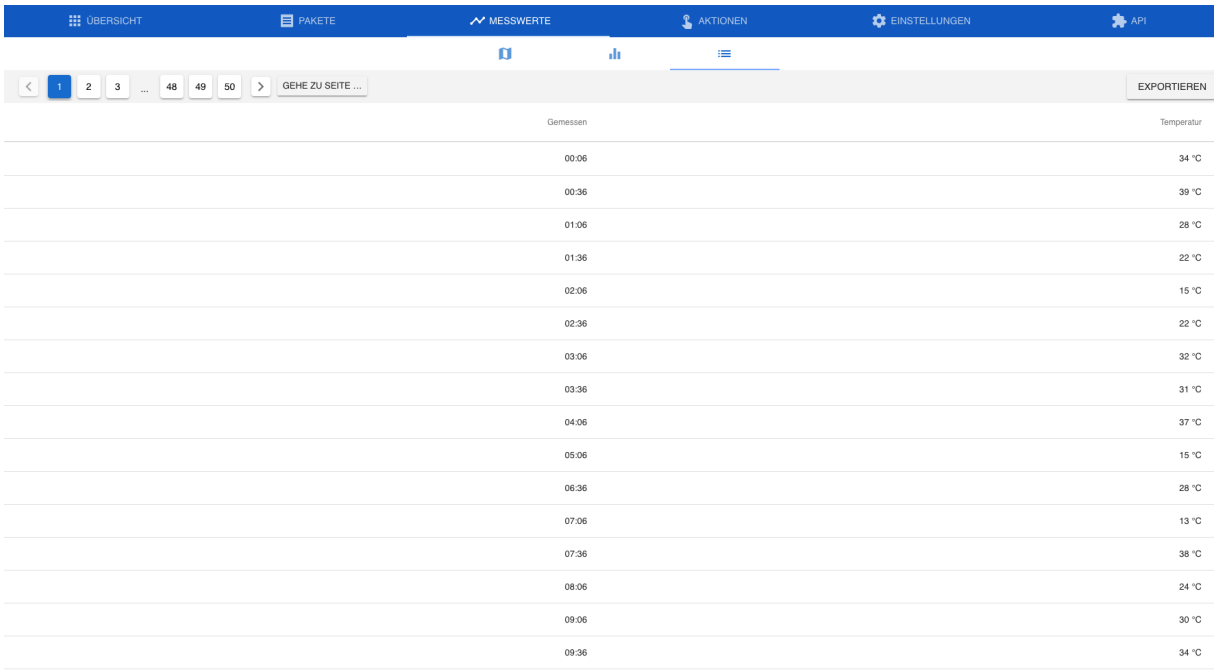


Abbildung 5.15: Screenshot

5.8 Einstellungen / Gerät löschen

Im Bereich **Einstellungen** finden Sie die Konfiguration für das Gerät. Es handelt sich hier um die gleichen Optionen, welche Sie auch beim Anlegen verwendet haben. Möchten Sie ein Gerät vollständig löschen öffnen Sie bitte den Abschnitt “Wartung”.

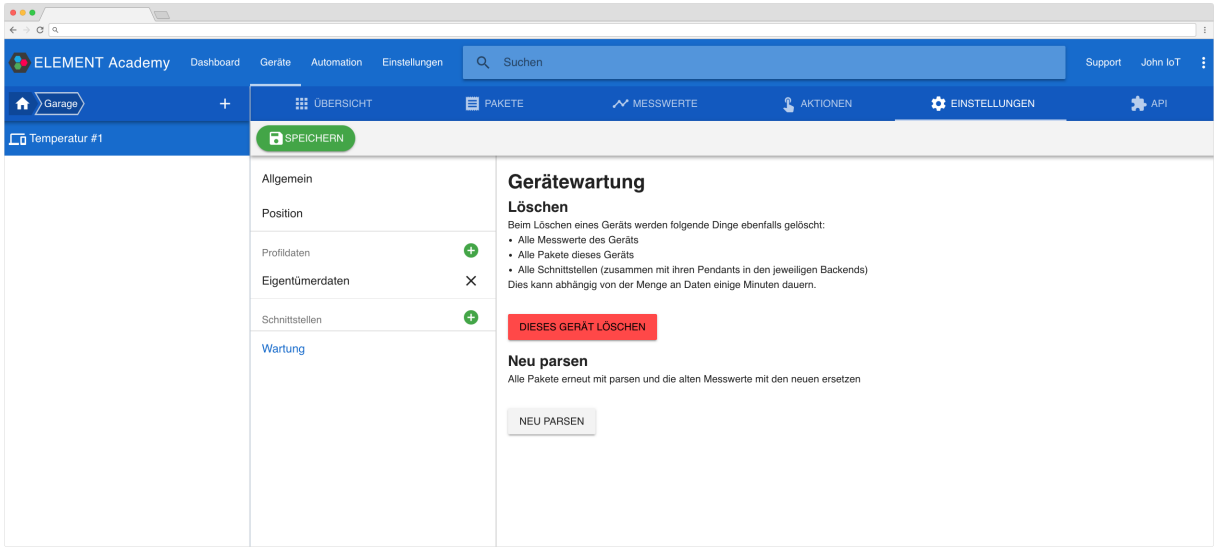


Abbildung 5.16: Screenshot

Über den Button “DIESES GERÄT LÖSCHEN” wird das Gerät gelöscht (bei Geräten mit vielen Daten, kann der Vorgang mehrere Minuten dauern).

5.9 Neu parsen von Paketen

Im Bereich **Einstellungen** / **Wartung** haben Sie die Möglichkeit vorhandene Pakete erneut zu parsen. Dieses kann sinnvoll sein, wenn Sie einen Parser angepasst haben nun auch die bereits vorhanden Messwerte ersetzen möchten.

5.10 API

Im Bereich **API** finden Sie Beispiele für die Nutzung der API. Haben Sie bereits einen API-Schlüssel angelegt (Siehe How To: API-Schlüssel verwalten), können Sie die vorgegeben Aufrufe direkt nutzen.

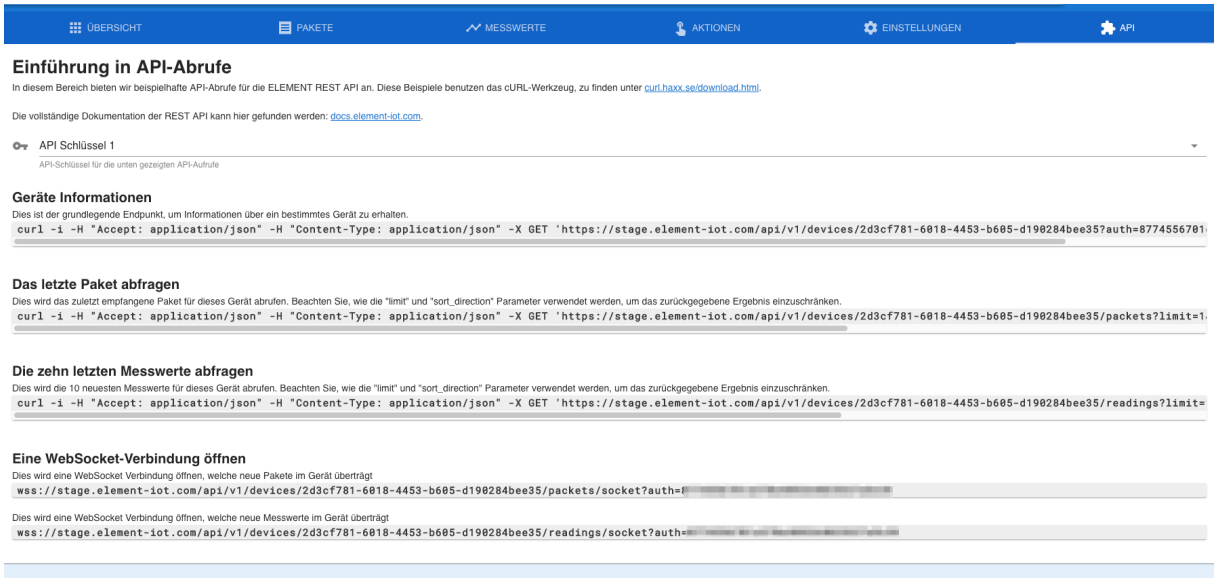


Abbildung 5.17: Screenshot

6 Profile für Geräte und Ordner

Über Profile haben Sie die Möglichkeit, Geräten und Ordnern weitere Datenfelder hinzuzufügen. So können Sie einem angelegten Gerät zusätzliche wichtige Informationen, beispielsweise Eigentümer oder eine Anschrift, mitgeben. Profile können über eine Eltern/Kind-Struktur in Abhängigkeit gebracht werden. Ein gängiges Vorgehen ist es zum Beispiel, einem Ordner das Profil “Wohnhaus” mitzugeben und allen Unterordnern das Profil “Wohnung”, um in jedem Ordner Informationen über die entsprechende Wohneinheit speichern zu können. Einige Beispiel-Szenarien werden Sie in diesem How To kennenlernen.

6.1 Ordner mit weiteren Datenfeldern

Um für einen Ordner weitere Datenfelder anzulegen, öffnen Sie bitte den Bereich **EINSTELLUNGEN** und hier den Abschnitt **Profile**. Klicken Sie nun auf **PROFIL HINZUFÜGEN**, um ein neues Profil zu erzeugen.

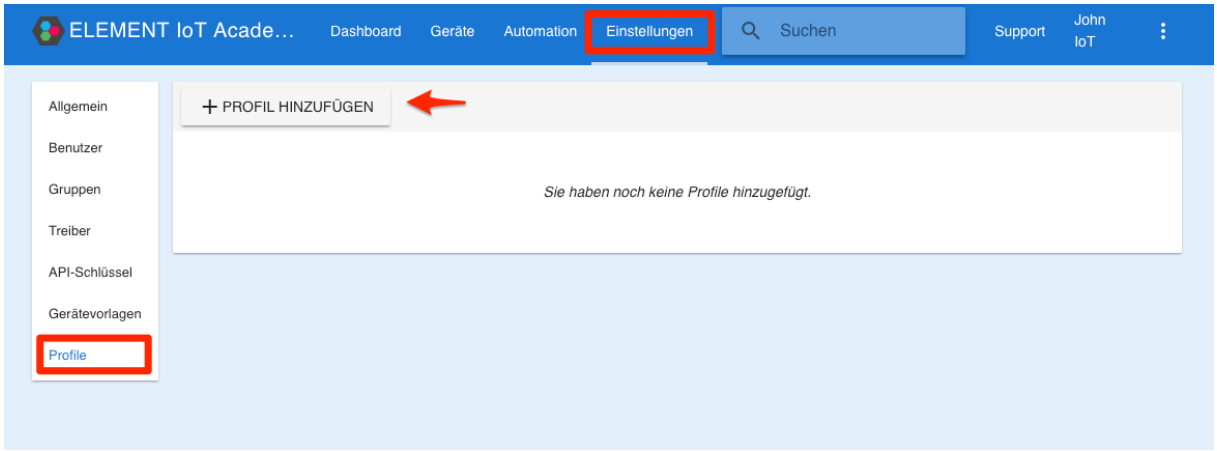


Abbildung 6.1: Screenshot

Im folgenden Formular vergeben Sie bitte als Anzeigenamen “Wohnung” mit einer Beschränkung auf “Ordner”. Der technische Name wird vom System vorbelegt, und findet insbesondere bei der Nutzung der ELEMENT IoT API Verwendung.

Als weiteres Feld legen Sie bitte ein Feld “Eigentümer” an. Klicken Sie hierfür auf **“+ FELD HINZUFÜGEN”**, wählen Sie als Anzeigename “Eigentümer” und als Datentyp “Zeichenkette”

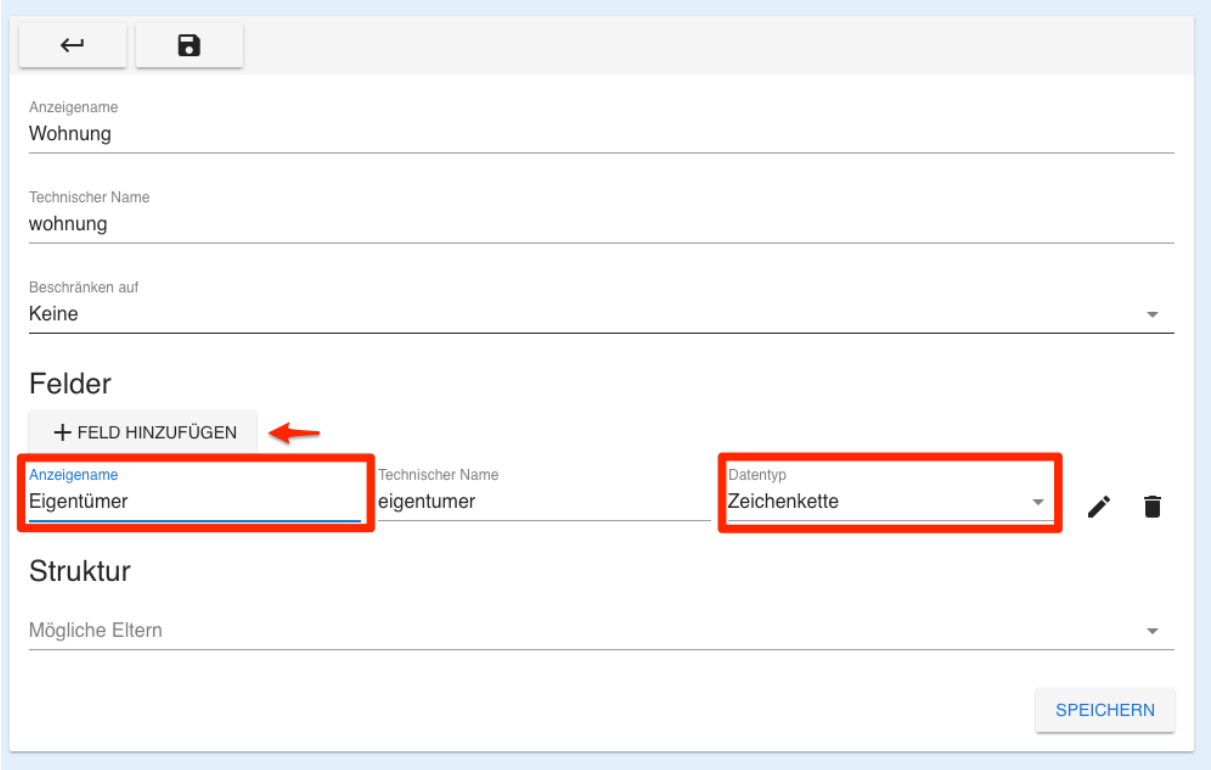


Abbildung 6.2: Screenshot

Klicken Sie anschließend auf **SPEICHERN**, um das Profil zu sichern.

Öffnen Sie nun den Bereich **Geräte** und wählen Sie einen Ordner aus. In den **Einstellungen** des Ordners können Sie nun das Profil “Wohnung” zuweisen, tippen Sie dazu einfach die Anfangsbuchstaben des Profilnamens in das entsprechende Feld und wählen das vorgeschlagene Profil durch anklicken aus. Speichern Sie anschließend die Einstellungen.

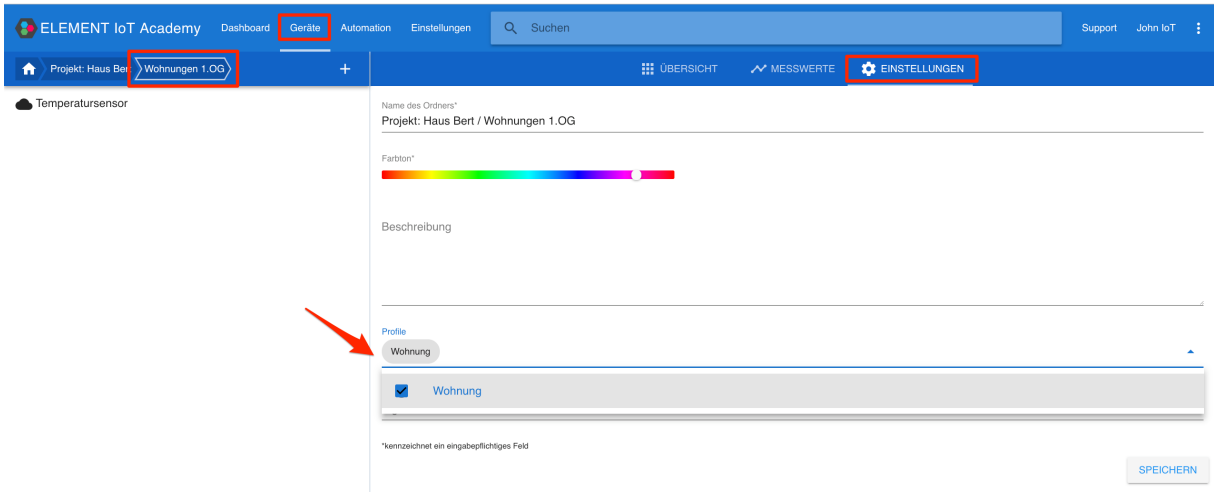


Abbildung 6.3: Screenshot

In den Einstellungen finden Sie nun ein neues Feld “Eigentümer”, welches sie mit eigenen Daten füllen können.

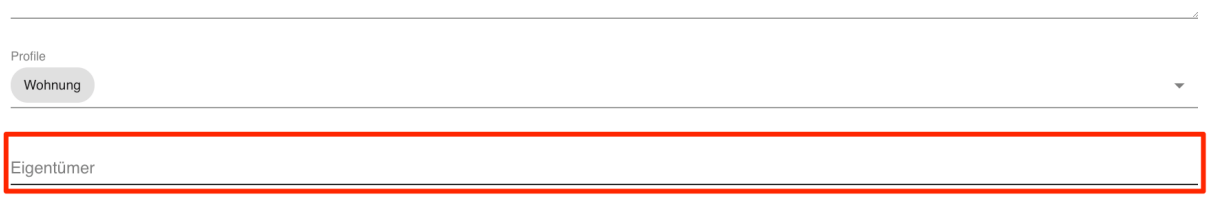


Abbildung 6.4: Screenshot

6.2 Gerät mit weiteren Datenfeldern

Das Vorgehen, um einem Gerät weitere Datenfelder hinzuzufügen, ist sehr ähnlich wie bei Ordnern. Nehmen wir an Sie möchten für Geräte das Feld “Typ” hinzufügen. Öffnen Sie auch hierfür den Bereich **Einstellungen** und hier den Abschnitt **Profile**. Klicken Sie nun auf **PROFIL HINZUFÜGEN**, um ein neues Profil zu erzeugen.

Verwenden Sie als Anzeigename zum Beispiel “Metadaten”.

Als weiteres Feld legen Sie bitte ein Feld “Typ” an. Klicken Sie hierfür auf **“+ FELD HINZUFÜGEN”**, wählen Sie als Anzeigename “Typ”, als Datentyp “Zeichenkette” und klicken anschließend auf **SPEICHERN**.

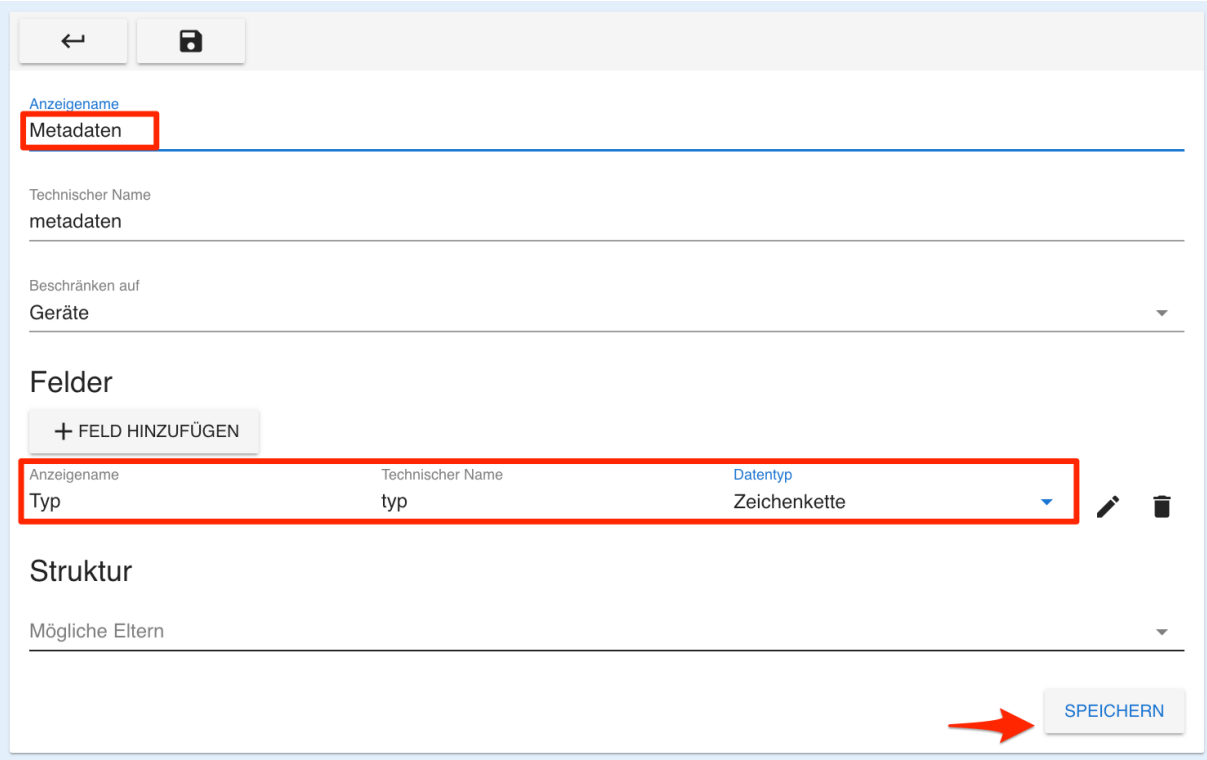


Abbildung 6.5: Screenshot

Öffnen Sie nun den Bereich **Geräte** und wählen Sie ein beliebiges Gerät aus. In den Einstellungen des Geräts können Sie nun das Profil "Metadaten" zuweisen, tippen Sie dazu einfach die Anfangsbuchstaben des Profilnamens in das entsprechende Feld und wählen das vorgeschlagene Profil durch anklicken aus. Speichern Sie anschließend die Einstellungen.

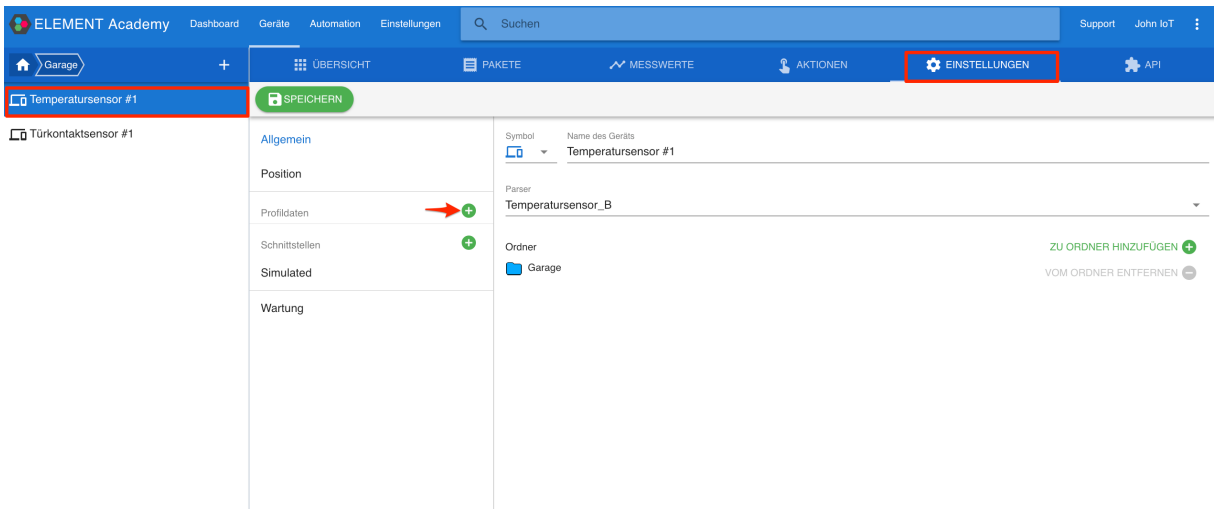


Abbildung 6.6: Screenshot

In den Einstellungen finden Sie nun ein neues Feld “Typ”, welches sie mit eigenen Daten füllen können.

6.3 Eltern/Kind-Struktur anhand von Ordnern

Sie haben die Möglichkeit festzulegen, welche Ordner oder Geräte ein bestimmtes Profil bekommen dürfen. Nehmen wir als Beispiel eine Wohnung. Sie können über eine Eltern/Kind-Beziehung festlegen, das alle Geräte in dem Ordner Wohnung nur das Profil Zimmer erhalten können.

Das Profil “Wohnung” haben Sie bereits im oberen Abschnitt dieses How To angelegt, legen Sie nun ein weiteres Profil mit der Bezeichnung “Zimmer” an.

- Beschränken auf: Ordner

Legen Sie als zusätzliches Feld “Anzahl Fenster” mit dem Datentyp Ganze Zahl an. Unter Struktur/Mögliche Eltern wählen Sie “Wohnung” aus. Durch das Eingeben der ersten Buchstaben wird Ihnen dieses Profil automatisch vorgeschlagen.

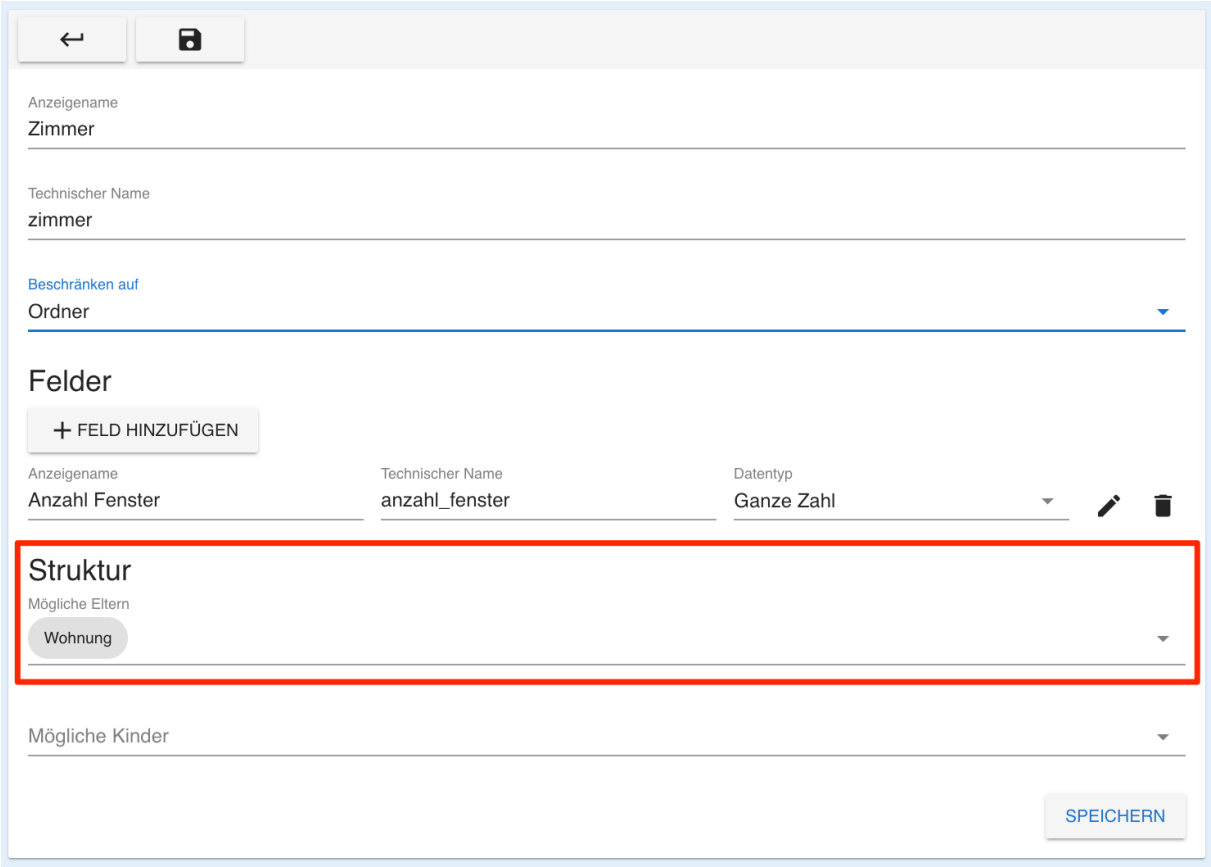


Abbildung 6.7: Screenshot

Nun können Ordner, welche sich innerhalb eines Ordners mit dem Profil “Wohnung” befinden, nur noch das Profil Zimmer erhalten.

6.4 Bilder zu Geräte erfassen

Zur Nutzung in der Kartendarstellung lassen sich externe Bild-URLs zu Geräten hinterlegen. Dazu ist es nötig, ein Profil mit dem Namen “Images” und einem Feld, ebenfalls mit dem Namen “Images” und dem Typ “Zeichenkette” anzulegen:

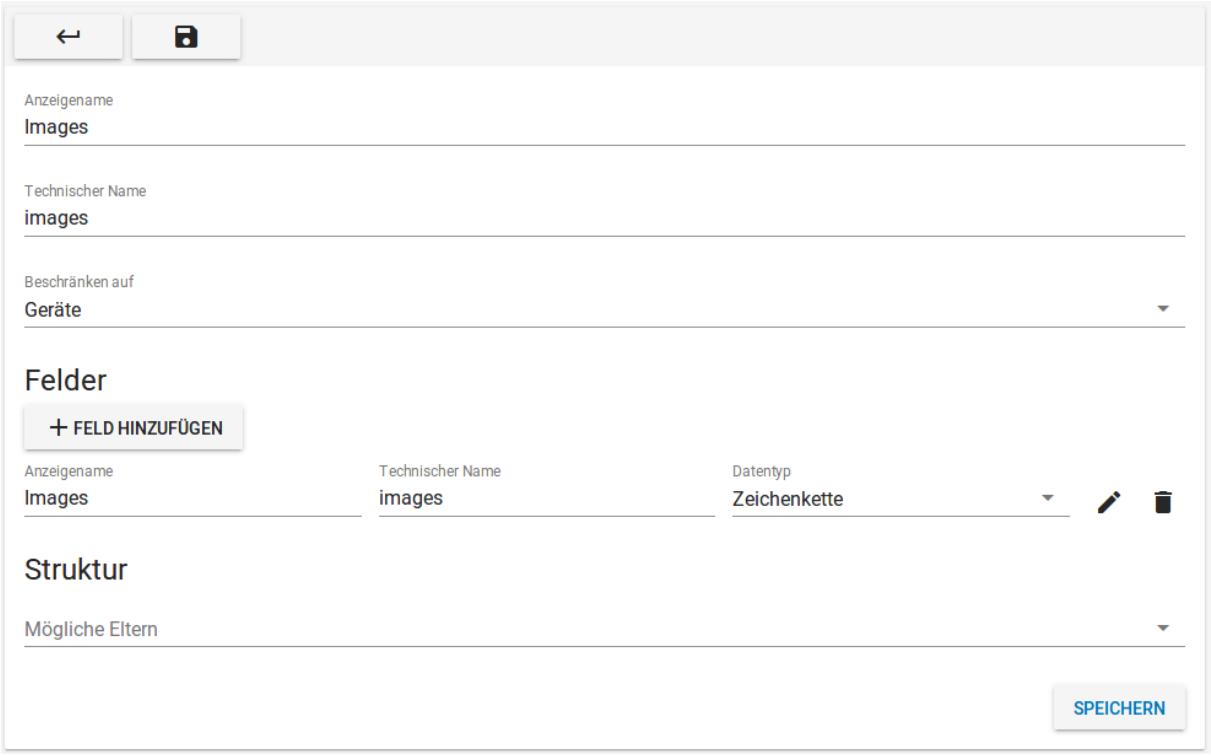


Abbildung 6.8: Screenshot

In Geräten, welche dieses Profil zugeordnet bekommen, können dann mehrere URLS in Form eines JSON-Arrays mit JSON-Strings (https://json.org/) angegeben werden, z.B. `["https://i.imgur.com/O8ScJNq.jpg"]`

7 Gerätevorlagen

Vorlagen können genutzt werden, wenn Sie häufig identische Geräte anlegen. Sie haben die Möglichkeit, folgende Werte festzulegen. Diese Werte werden dann beim Anlegen eines neuen Gerätes automatisch ausgefüllt.

- Name für das Gerät
- Symbol für das Gerät
- Ordner für das Gerät
- Profil
- Parser
- Schnittstelle
- Position

In diesem How To legen wir eine Vorlage für einen Temperatursensor an und fügen die Vorlage dem “Plus”-Button auf der Oberfläche hinzu.

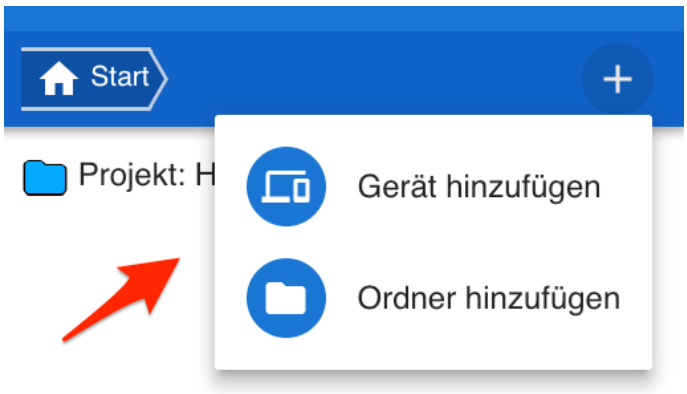


Abbildung 7.1: Screenshot

7.1 Eine neue Gerätevorlage anlegen

Öffnen Sie in der oberen Navigationsleiste den Bereich **Einstellungen** und klicken Sie anschließend auf **Gerätevorlagen**. In der folgenden Übersicht sehen Sie bereits angelegte Vorlagen und haben die

Möglichkeit über **GERÄTEVORLAGE HINZUFÜGEN** eine neue Vorlage anzulegen.

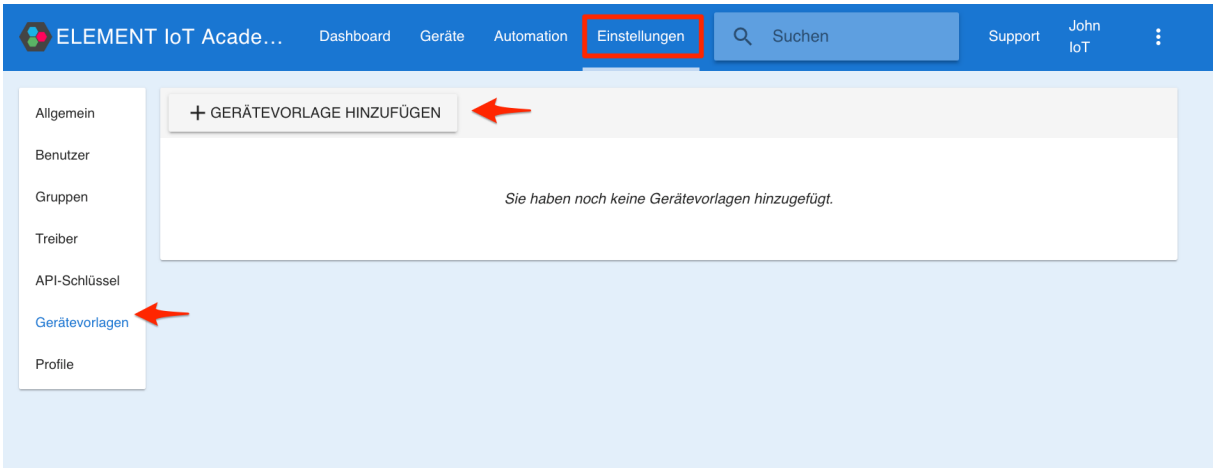


Abbildung 7.2: Screenshot

Vergeben Sie als Namen für die Vorlage “Temperatursensoren” und befüllen Sie die weiteren Felder wie im folgenden Screenshot abgebildet. Anschließend klicken Sie bitte auf **SPEICHERN**, um die Vorlage anzulegen.

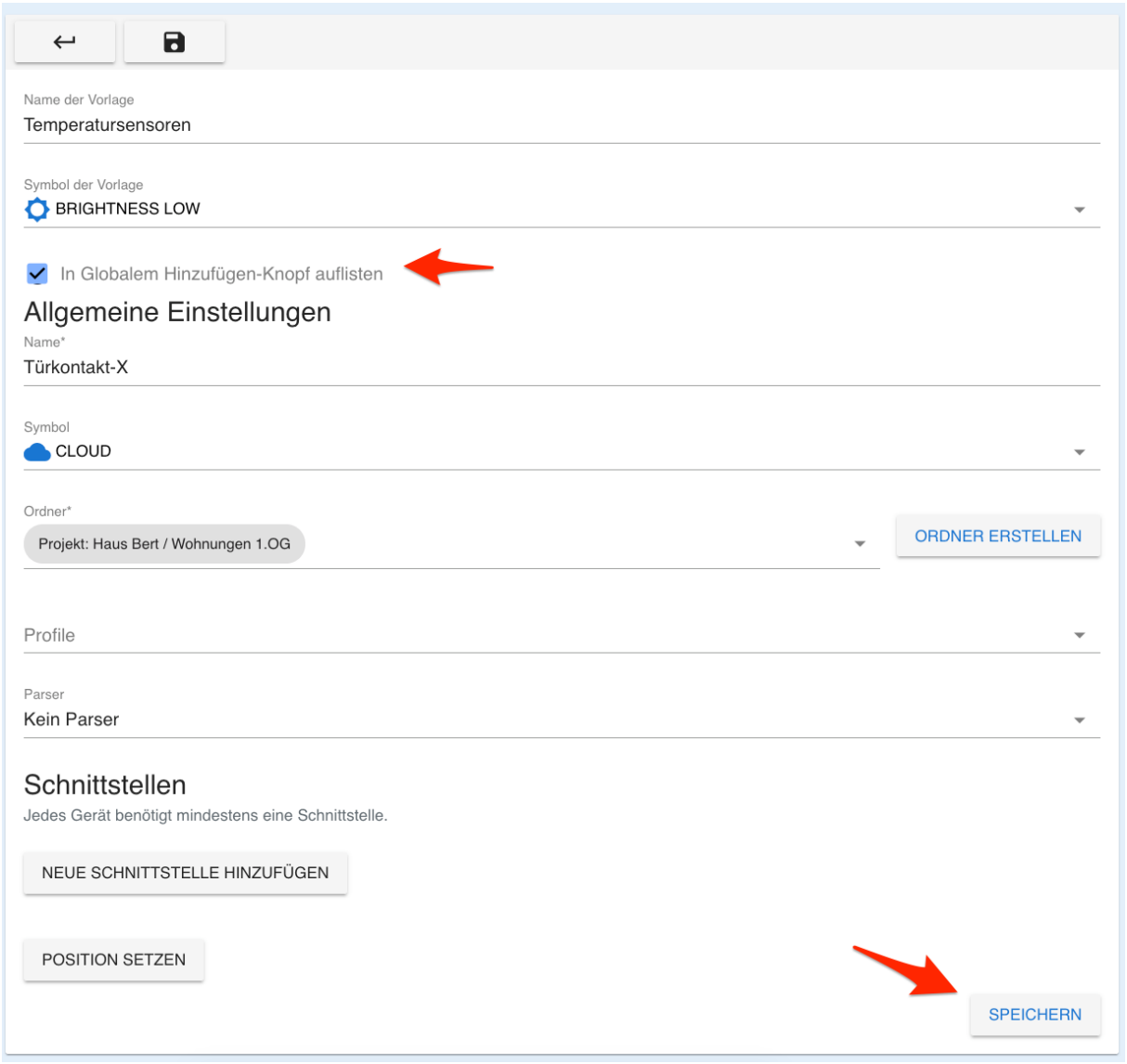


Abbildung 7.3: Screenshot

7.2 Geräte über eine Vorlage anlegen

Um ein neues Gerät anhand einer vorhandenen Vorlage anzulegen, haben Sie mehrere Möglichkeiten.

- Sie können direkt in der Übersicht aller vorhanden Vorlagen den “+” Button nutzen.

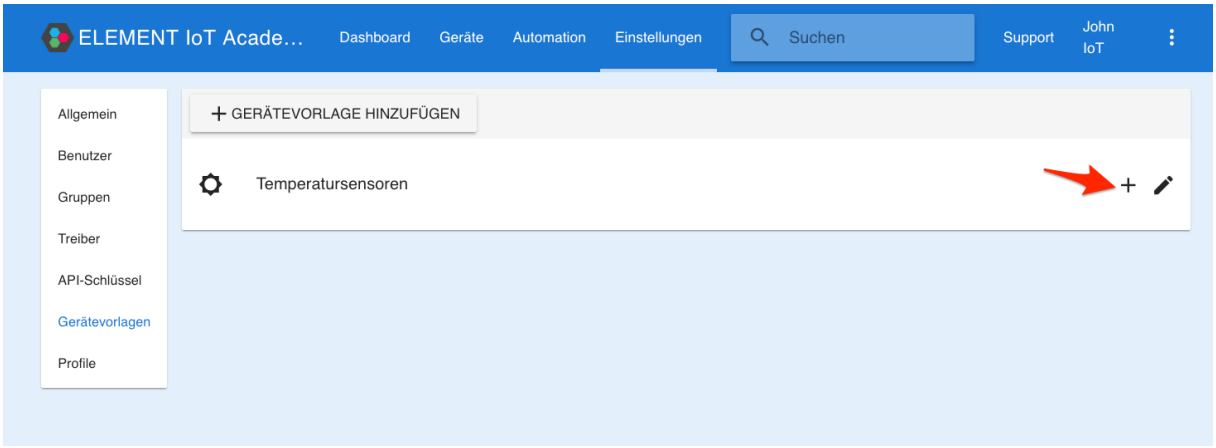


Abbildung 7.4: Screenshot

- Sie können den “+” Button im Bereich Geräte nutzen, wenn Sie die entsprechende Option beim Anlegen der Vorlage gesetzt haben (“In Globalem Hinzufügen-Knopf auflisten”).

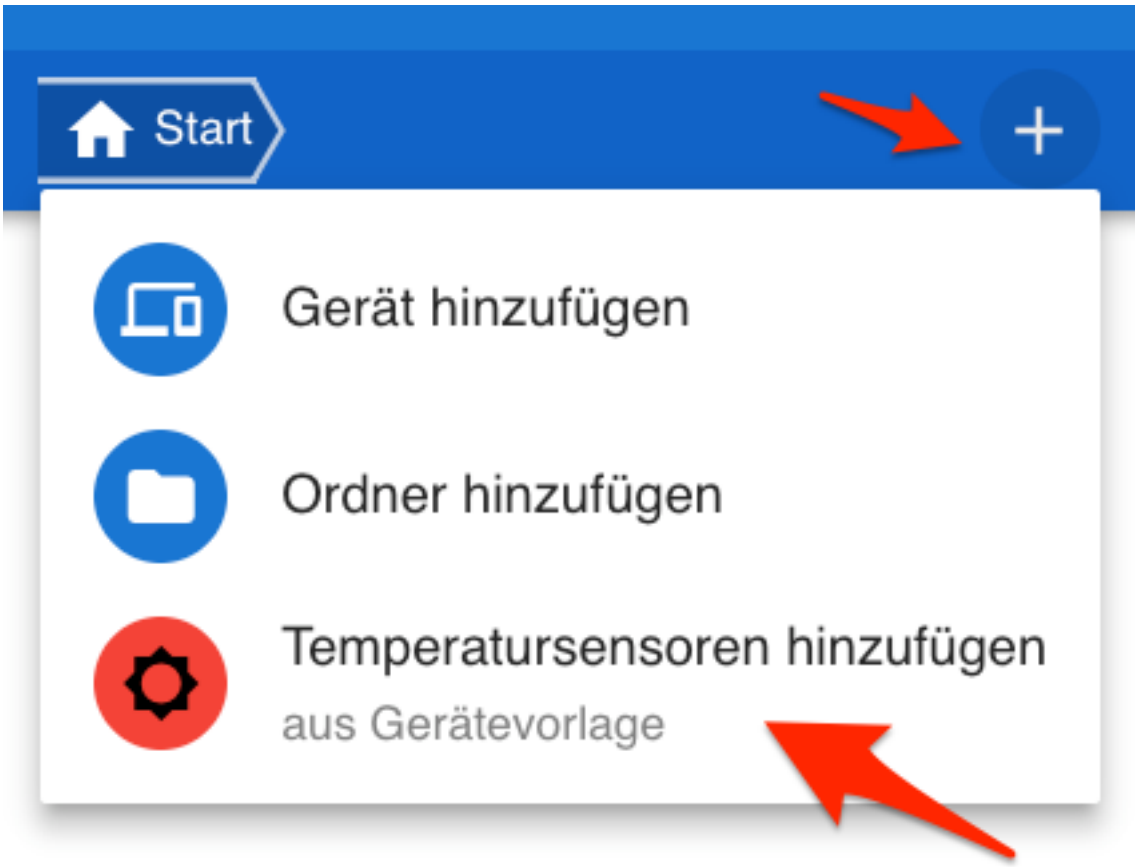


Abbildung 7.5: Screenshot

- Sie können beim Anlegen eines Gerätes die entsprechende Vorlage auswählen.

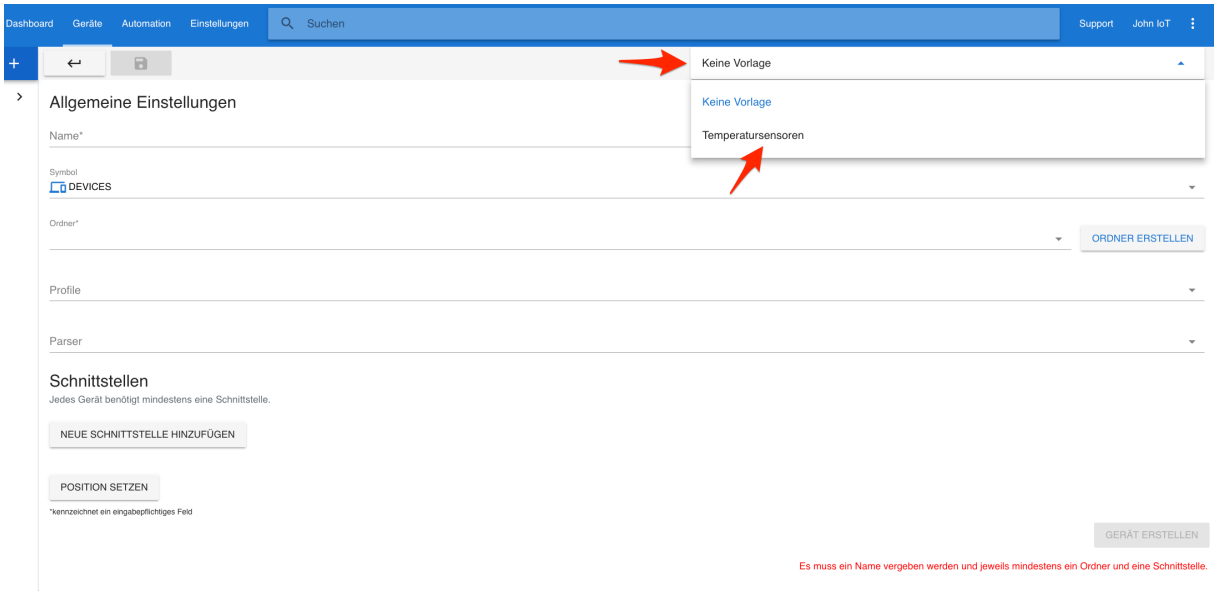


Abbildung 7.6: Screenshot

7.3 Vorlagen löschen und bearbeiten

Um eine bestehende Gerätevorlage zu bearbeiten oder zu löschen, öffnen Sie den Bereich **Einstellungen** und Klicken Sie auf **Gerätevorlagen**. Neben dem Namen der Vorlage finden Sie ein “Stift”-Symbol, klicken Sie dieses an um die Vorlage zu ändern oder zu löschen.

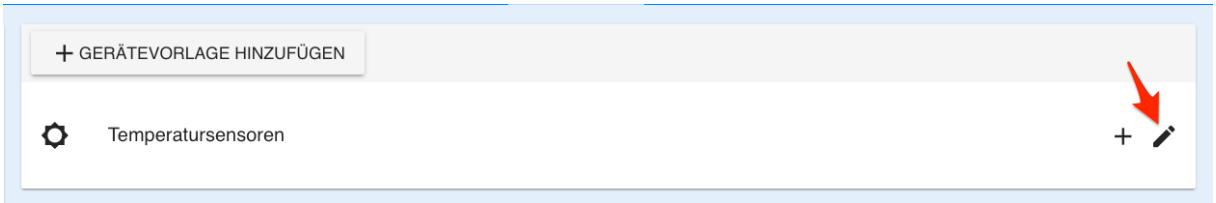


Abbildung 7.7: Screenshot

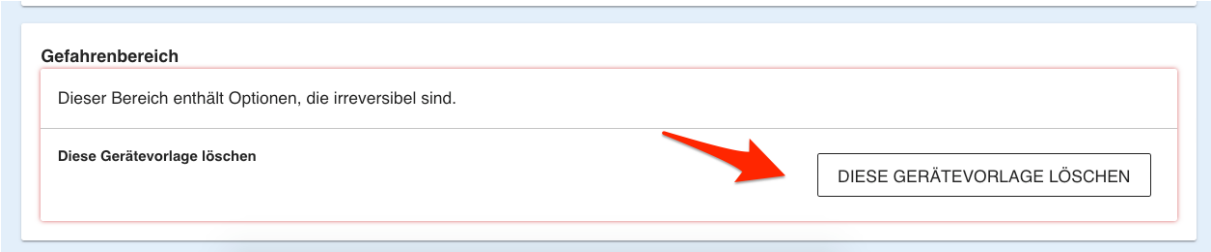


Abbildung 7.8: Screenshot

8 Benutzerverwaltung

8.1 Anlegen neuer Benutzer

Mit der Benutzerrolle “Admin” haben Sie die Möglichkeit, neue Benutzer für Ihren Mandaten zu registrieren. Bitte öffnen Sie dafür den Bereich **Einstellungen**.

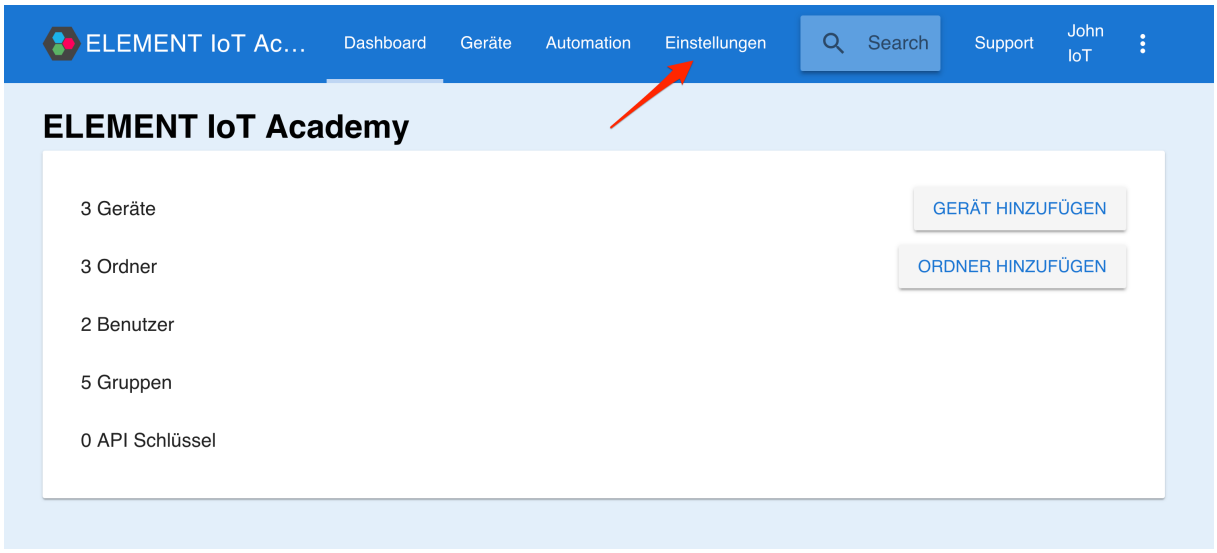


Abbildung 8.1: Screenshot

In der folgenden Übersicht wählen Sie bitte die Option Benutzer und klicken anschließend auf den Button **NUTZER HINZUFÜGEN**.

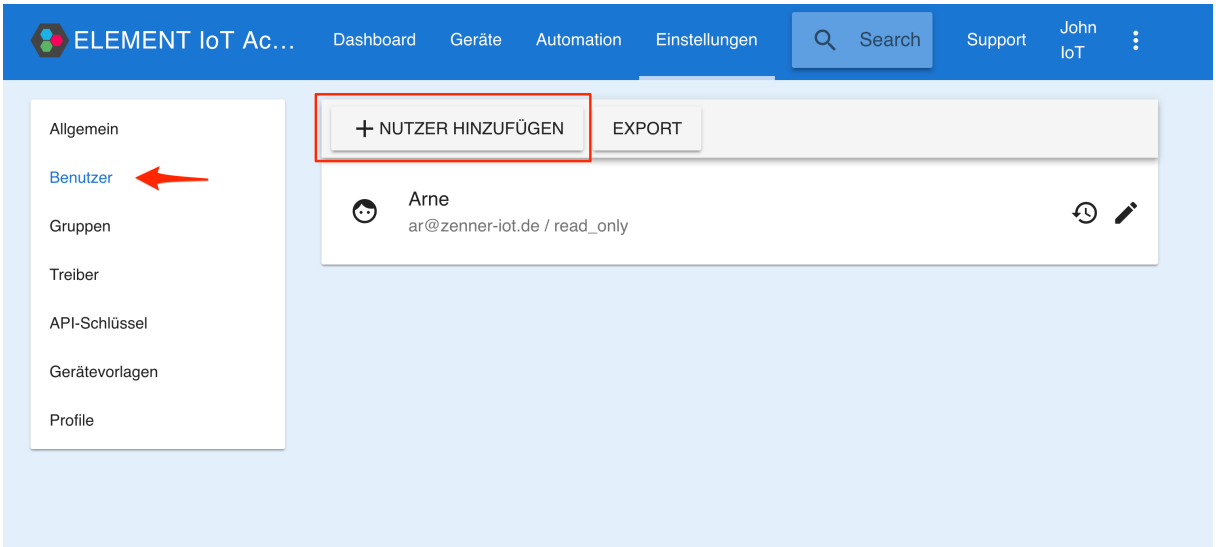


Abbildung 8.2: Screenshot

In der Maske vergeben Sie bitte einen Namen für den Benutzer - dieser wird innerhalb der ELEMENT IoT-Plattform angezeigt - eine E-Mail-Adresse und ein sicheres Passwort aus mindestens 8 Zeichen. Die E-Mail-Adresse und das Passwort werden für die Anmeldung an der Plattform benötigt. Neue Benutzer sollten ihr Passwort nach der ersten Anmeldung ändern.

Als Berechtigungen stehen Ihnen folgende Möglichkeiten zur Verfügung:

- **Nur lesen**: Der neue Benutzer kann alle Daten, Geräte, Einstellungen etc. in der Plattform sehen, aber nicht bearbeiten.
- **Benutzer**: Wenn Sie diese Berechtigung vergeben, können Sie dem Benutzer individuell erstellte Berechtigungsgruppen zuordnen (siehe How To: @TODO).
- **Administrator**: Der Benutzer hat die volle Berechtigung auf die Plattform, dies beinhaltet das Anlegen neuer Geräte, Ordnerstrukturen, Benutzer usw., das Ändern von sämtlichen Einstellungen sowie das Löschen.

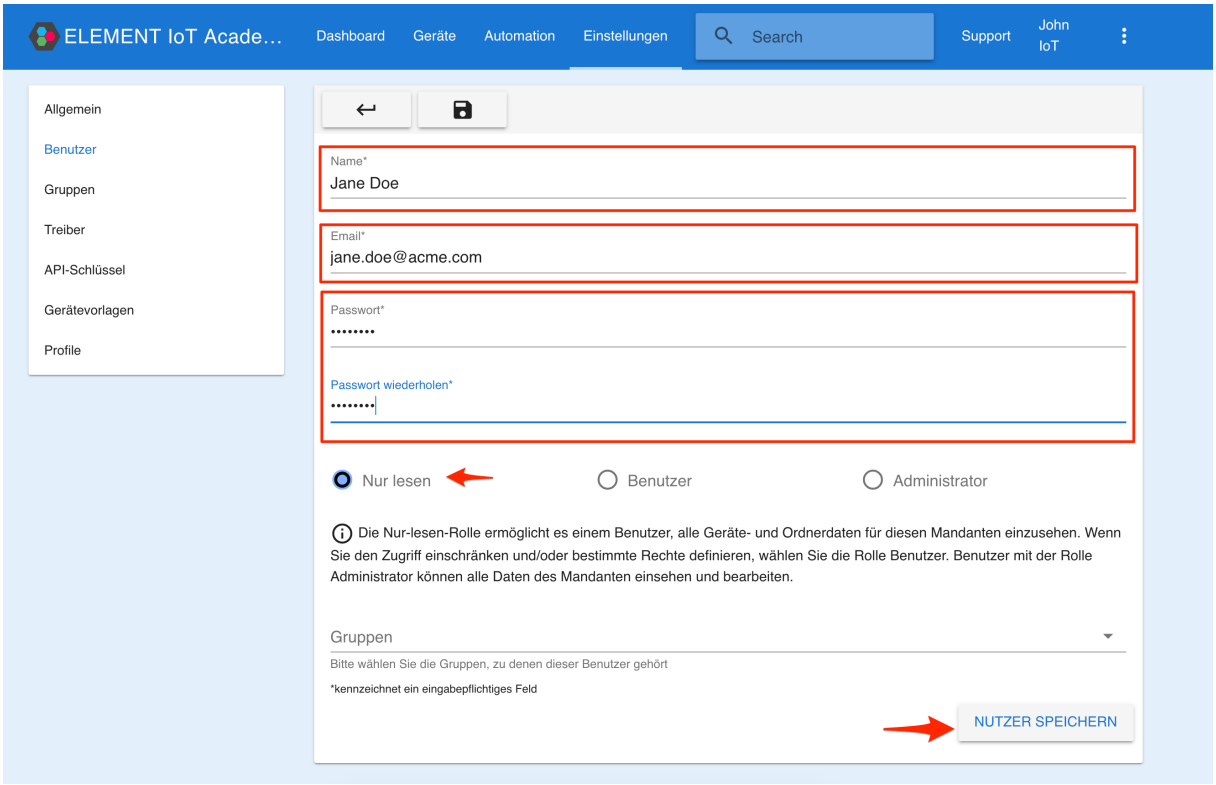


Abbildung 8.3: Screenshot

Klicken Sie zum Abschluss auf **NUTZER SPEICHERN**, der neue Benutzer kann sich jetzt mit dem entsprechenden Mandanten an der Plattform anmelden.

8.2 Bearbeiten und Löschen von Benutzern

Zum Bearbeiten von bereits bestehenden Benutzern öffnen Sie bitte, wie bereits beim Anlegen neuer Benutzer, den Bereich **Einstellungen** und hier die Option **Benutzer**.

In der folgenden Übersicht können Sie existierende Benutzer über das Stift-Symbol ändern oder löschen.

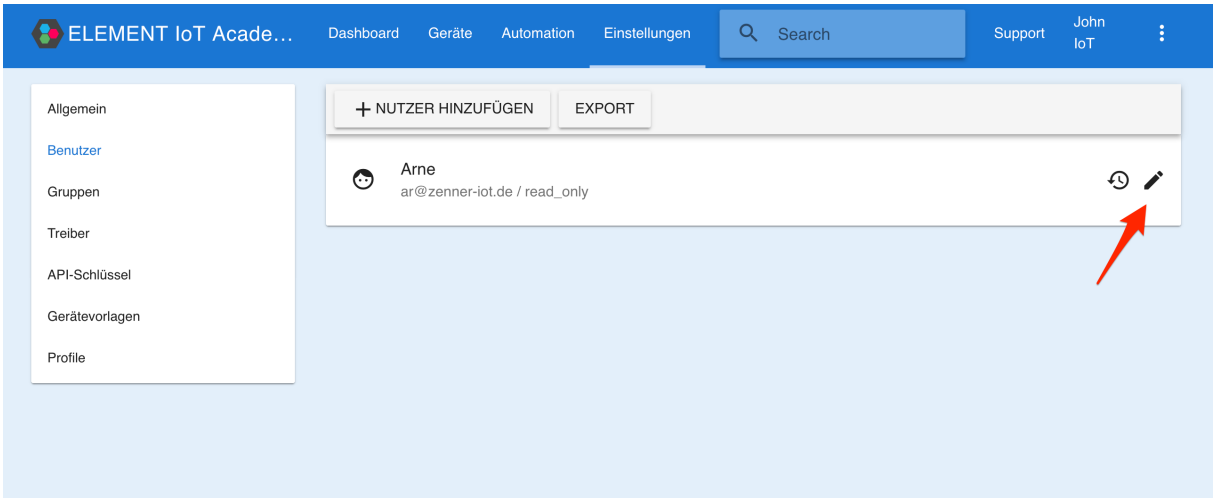


Abbildung 8.4: Screenshot

Im folgenden Formular können Sie nun einen neuen Namen vergeben, die E-Mail-Adresse oder die Berechtigungen ändern. Bitte beachten Sie: Das Ändern eines Passworts ist an dieser Stelle **nicht** möglich und sollte durch den Benutzer selber vorgenommen werden.

Zum Löschen eines Benutzers klicken Sie bitte auf den Button **DIESEN BENUTZER LÖSCHEN**, nach einer Sicherheitsabfrage wird der Benutzer dauerhaft aus der Plattform gelöscht.

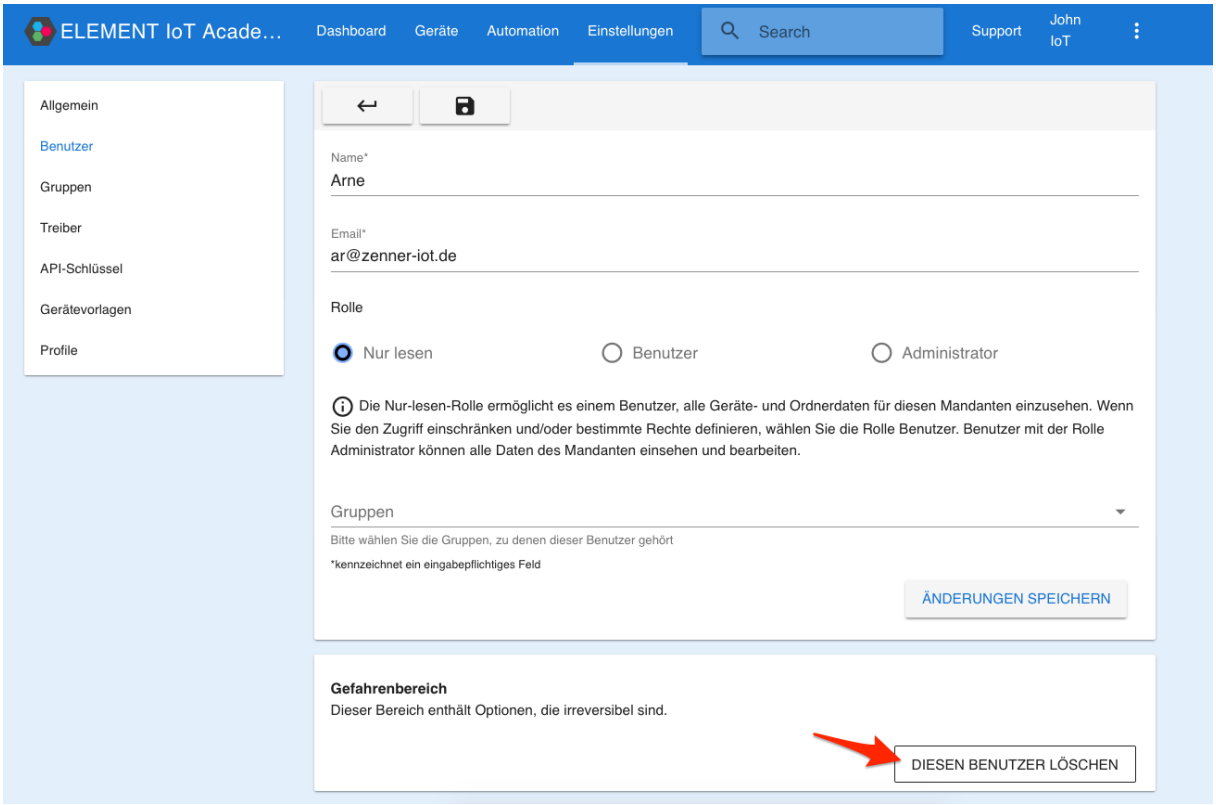


Abbildung 8.5: Screenshot

9 Berechtigungsverwaltung

Die ELEMENT IoT-Plattform bietet ein umfangreiches Berechtigungssystem. Sie haben die Möglichkeit, Gruppen anzulegen und diese dann entsprechenden Benutzern zuzuordnen.

9.1 Rollen

Grundsätzlich gibt es in ELEMENT IoT drei grundlegende Arten von Nutzerrollen (*Normaler Nutzer*, *"Nur-Lesen"-Nutzer* und *Mandantenadministratoren*) und eine weitere ("*Superadmin*"), die nur dem Betreiber der ELEMENT IoT-Instanz zugänglich ist - im Falle von element-iot.com der ZENNER International GmbH und der ZENNER IoT Solutions GmbH.

Normale Nutzer gehören zu einer Gruppe, in der die Berechtigungen der Gruppenmitglieder im Detail definiert sind.

"Nur-Lesen"-Nutzer gehören zu einem Mandanten und dürfen in diesem Mandanten alles sehen oder lesen, aber nichts ändern.

Mandantenadministratoren gehören zu einem Mandanten und dürfen in diesem alle sehen, anlegen, ändern und löschen. Diese Nutzergruppe ist besonders dafür einzusetzen, die weiteren Nutzer eines Mandanten zu verwalten.

Superadministratoren stehen nur dem Instanzbetreiber zur Verfügung und wird vorwiegend zum Verwalten der Mandanten verwendet. Zu Supportzwecken kann sich ein Nutzer dieser Rolle auch in jedem Mandanten wie ein Mandantenadministrator bewegen - Einschränkungen, die für den Mandaten hinterlegt sind, greifen dabei nicht.

Die Berechtigungen der drei Rollen *"Nur-Lesen"-Nutzer*, *Mandantenadministrator* und *Superadministrator* können nicht weiter konfiguriert werden. Die Rolle *normaler Nutzer* hingegen wird über die zugeordnete Gruppe feingranular eingestellt.

9.2 Gruppen & Berechtigungen

Der Rolle *normaler Nutzer* wird immer eine Gruppe zugeordnet. In dieser Gruppe können jeweils die Berechtigungen für *sehen*, *ändern*, *anlegen* und *löschen* für die folgenden Bereiche festgelegt werden:

- Ordner
- Benutzer
- Parser
- Treiber
- API-Schlüssel
- Streams
- Regeln
- Gerätevorlagen
- Profile
- Ansichten
- Jobs
- Apps

In diesem How To lernen Sie einige oft genutzte Gruppenkonzepte anhand von Beispielen kennen.

9.3 Wo findet man die Einstellungen für Gruppen

Die Einstellungen für Gruppen finden Sie unter **Einstellungen** - **Gruppen**

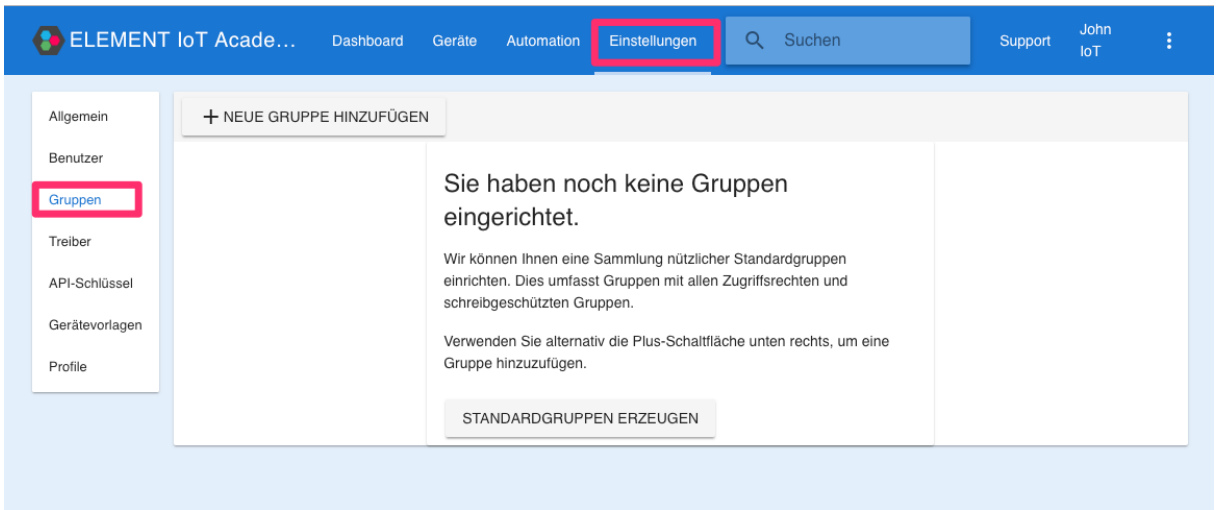


Abbildung 9.1: Screenshot

9.4 Die Standardgruppen

Wenn Sie noch keine Gruppen angelegt haben, erhalten Sie beim Öffnen der Gruppenübersicht die Möglichkeit, drei Standardgruppen durch die Plattform erzeugen zu lassen. Nach dem Klicken auf den Button **STANDARDGRUPPEN ERZEUGEN** erhalten Sie die folgenden Gruppen:

Vollzugriff Mitglieder dieser Gruppe erhalten, wie der Name der Gruppe schon andeutet, vollen Zugriff auf die Plattform (den Mandanten). Ohne weitere Anpassungen haben Nutzer mit dieser Gruppe praktisch die selben Rechte wie Nutzer mit der Rolle *Mandantenadministrator*!

Lesender Zugriff Mitglieder dieser Gruppe erhalten einen lesenden Zugriff auf alle Inhalte der Plattform, können aber keinerlei Änderungen durchführen. Dies Gruppe dient als Ausgangsgruppe für Nutzer mit überwiegend lesendem Zugriff. Ohne weitere Anpassung haben Nutzer damit praktisch die selben Rechte wie mit der Rolle *“Nur-Lesen”*!

Normaler Nutzer Dies Gruppe dient als Ausgangsgruppe für Nutzer, die mit Ausnahme von Nutzer- und API-Key-Verwaltung alle Rechte in einem Mandaten haben.

Alle diese Gruppen dienen als Beispiel bzw. Startpunkt für eine individuelle Einstellung der Berechtigungen. Beachten Sie insbesondere die Hinweise zu der Gruppe *Vollzugriff*.

Abbildung 9.2: Screenshot

Abbildung 9.3: Screenshot

9.5 Einem Benutzer eine Gruppe zuweisen

Um den Einstieg in das Berechtigungssystem zu erleichtern, erzeugen wir uns im ersten Schritt einen weiteren Benutzer zum Testen der Gruppen. Wie Sie neue Benutzer anlegen, erfahren Sie im How To Benutzerverwaltung.

Für die Beispiele in diesem How To verwenden wir einen Benutzer mit dem Namen “Gruppen Test”, der E-Mail Adresse test@element.com und einem beliebigen Passwort. Um gleich die Standardgruppen auszuprobieren, wählen Sie bitte direkt beim Anlegen die Option “Benutzer” und die Gruppe “Lesender Zugriff”

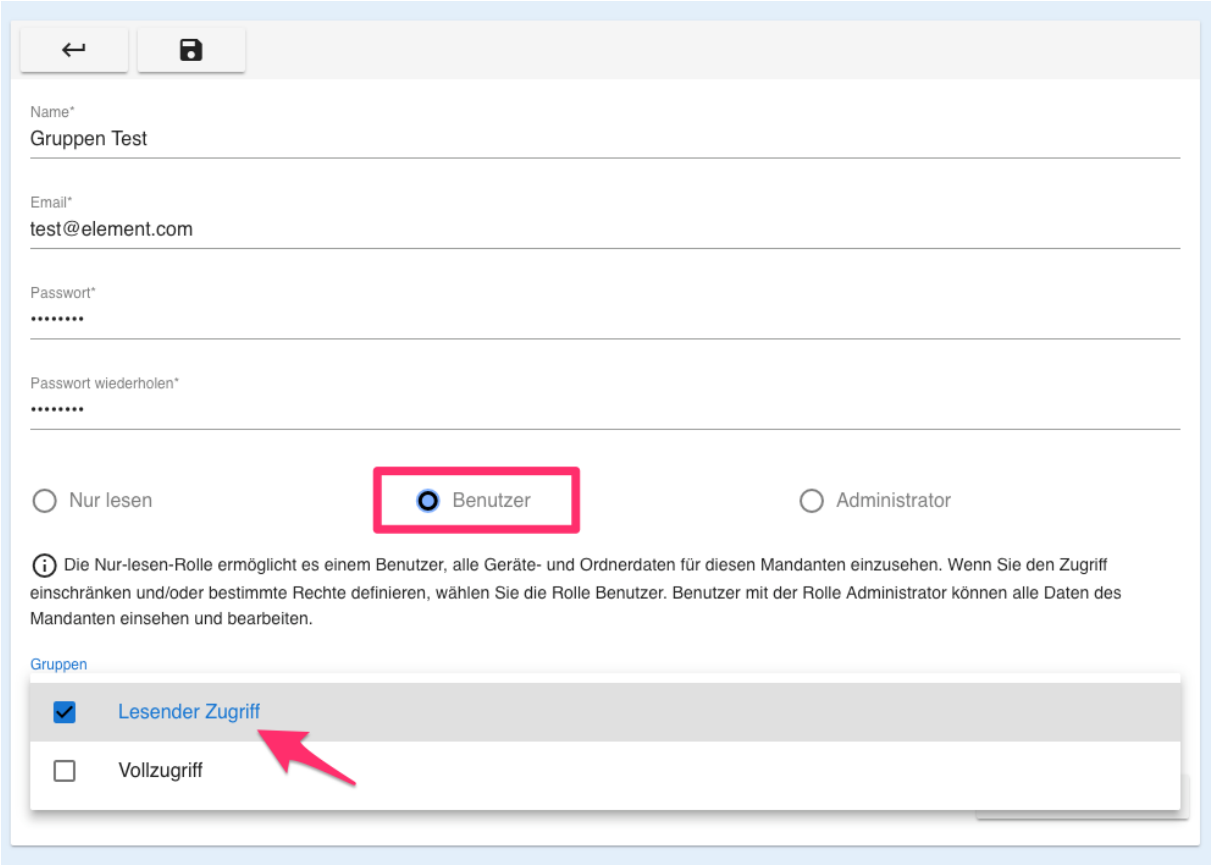


Abbildung 9.4: Screenshot

Zum Testen öffnen Sie bitte einen weiteren Browser und melden Sie sich mit dem gerade erstellten Benutzer an. Falls Sie den Google Chrome Browser verwenden, können Sie anstelle eines anderen Browsers den Inkognito-Modus verwenden.

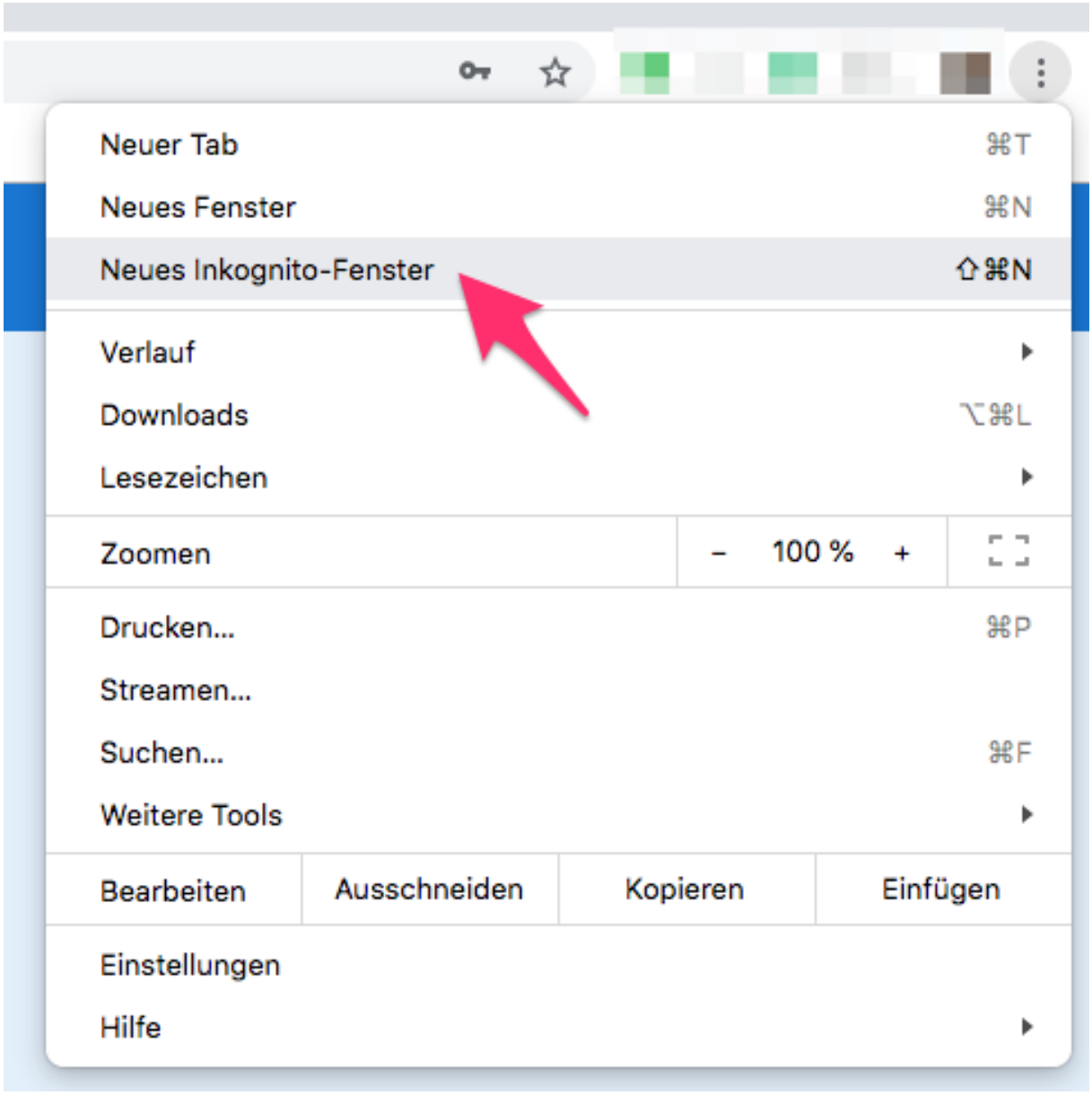


Abbildung 9.5: Screenshot

Öffnen Sie nun zum Beispiel den Bereich **Geräte** und versuchen Sie Einstellungen zu ändern. Da der Benutzer in der Gruppe “Lesender Zugriff” ist, erhalten Sie beim Versuch, die Änderungen zu speichern, eine entsprechende Fehlermeldung.

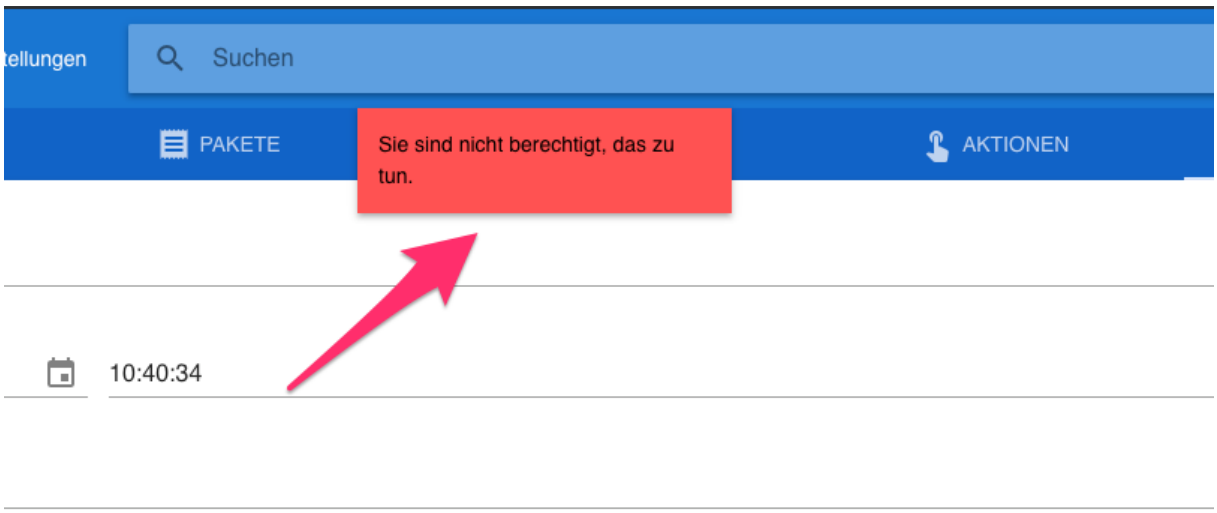


Abbildung 9.6: Screenshot

9.6 Leseberechtigungen auf einen Ordner

Als nächstes werden wir eine Gruppe erstellen, welche den lesenden Zugriff auf nur einen Ordner erlaubt. Sollten Sie momentan nur einen Ordner in ihrem Mandanten haben, legen Sie bitte einen weiteren an und fügen Sie ein Gerät hinzu. Für dieses Beispiel haben wir einen Ordner “Projekt: Haus Sven” mit dem Gerät “Temperatursensor - Dach” angelegt.

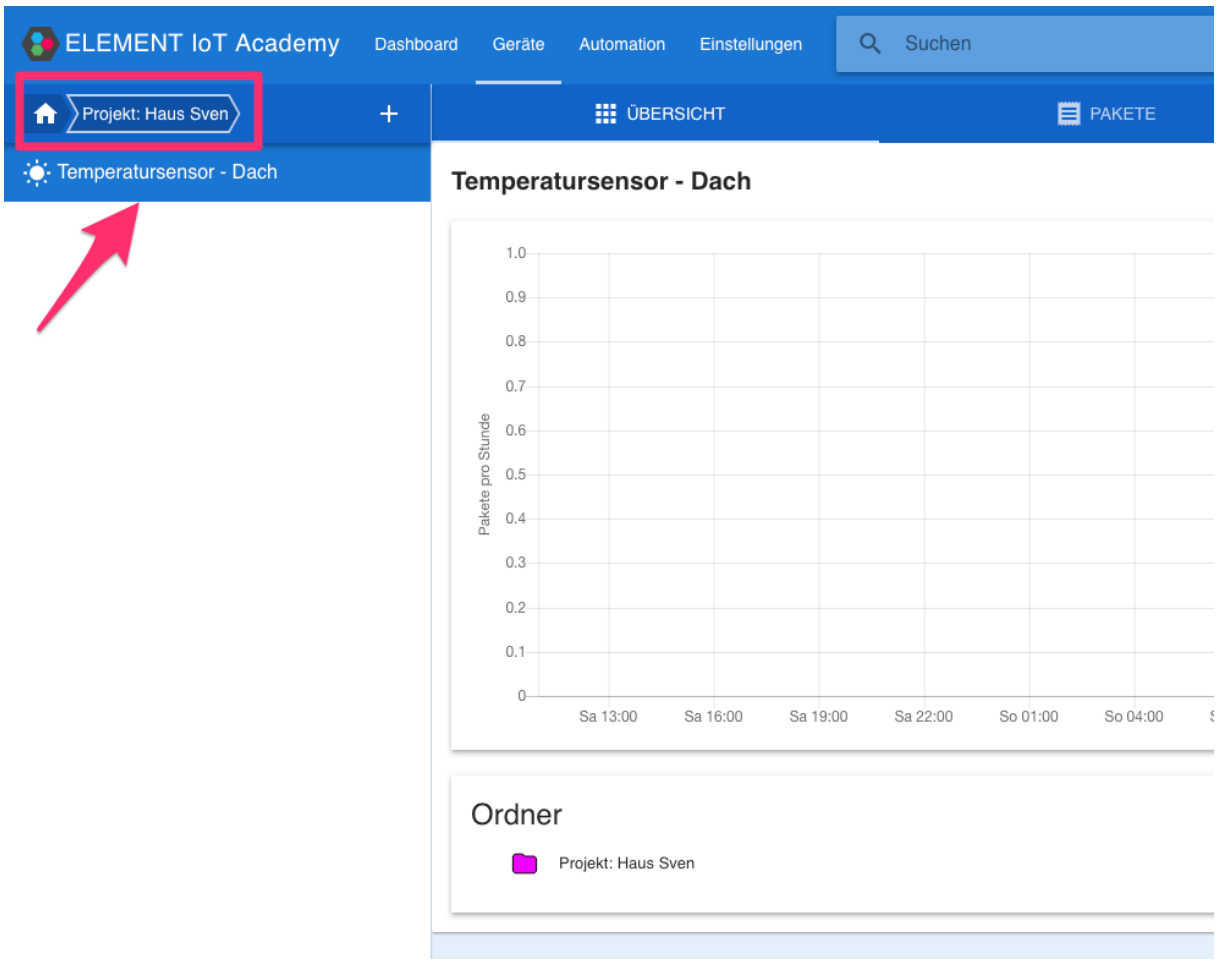


Abbildung 9.7: Screenshot

Legen Sie nun über **Einstellungen** - **Gruppen** eine neue Gruppe mit dem Namen: Zugriff lesend - Haus Sven.

Als Berechtigungen wählen Sie bitte:

- Ordner
- Treiber (notwendig um Geräte sehen zu können)
- Parser (notwendig um Geräte sehen zu können)

Im Feld Ordnerbaum geben Sie bitte folgendes ein:

```
Projekt: Haus Sven/**
```

Der Ausdruck besagt, dass die Gruppe alles unterhalb des Ordners “Projekt: Haus Sven” sehen darf. Klicken Sie anschließend auf **ÄNDERUNGEN SPEICHERN**. Die ** stehen für: Alle weiteren Elemente unterhalb des Ordners.

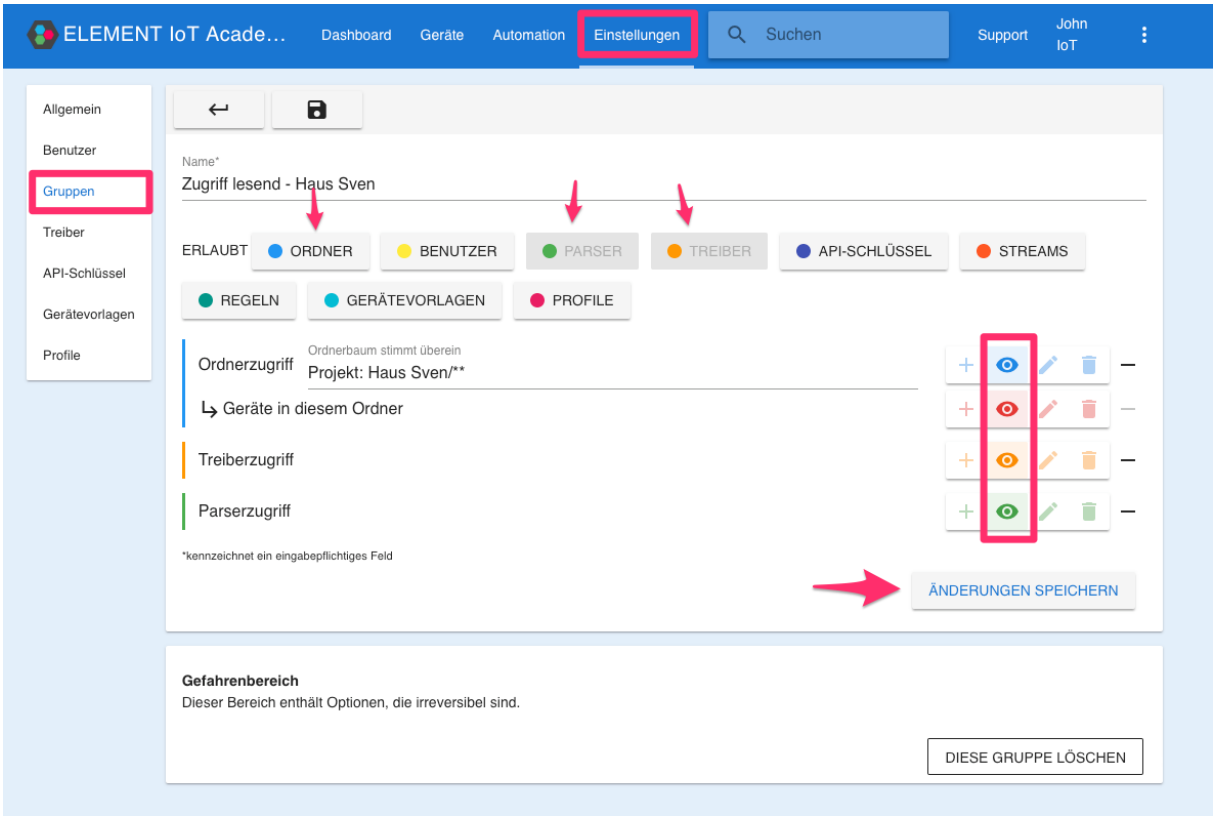


Abbildung 9.8: Screenshot

Im nächsten Schritt müssen Sie die Gruppe noch dem Test-Benutzer zuweisen, öffnen Sie hierfür den Bereich **Einstellungen** - **Benutzer** und wählen Sie die entsprechende Gruppe.

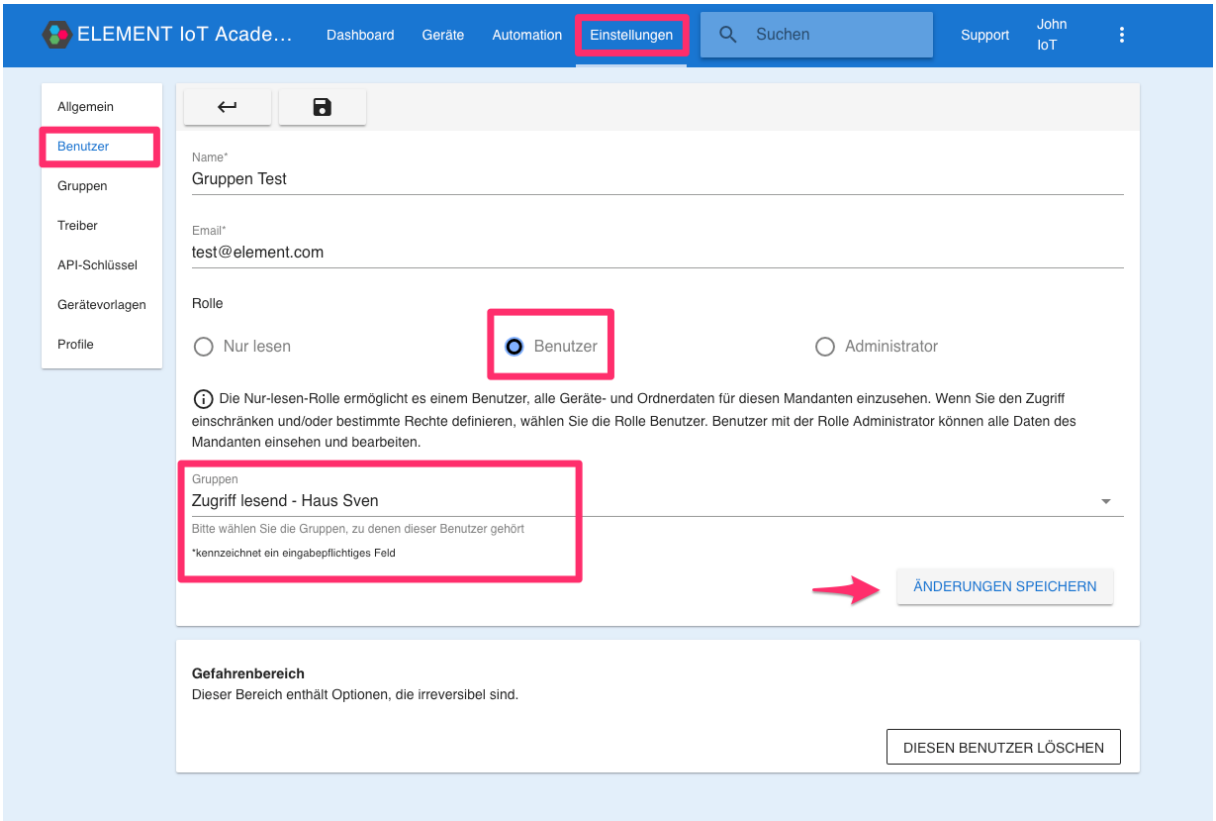


Abbildung 9.9: Screenshot

Wenn Sie sich nun mit dem Test-Benutzer anmelden, sehen Sie im Bereich Geräte nur noch das “Projekt: Haus Sven” mit dem entsprechenden Gerät.

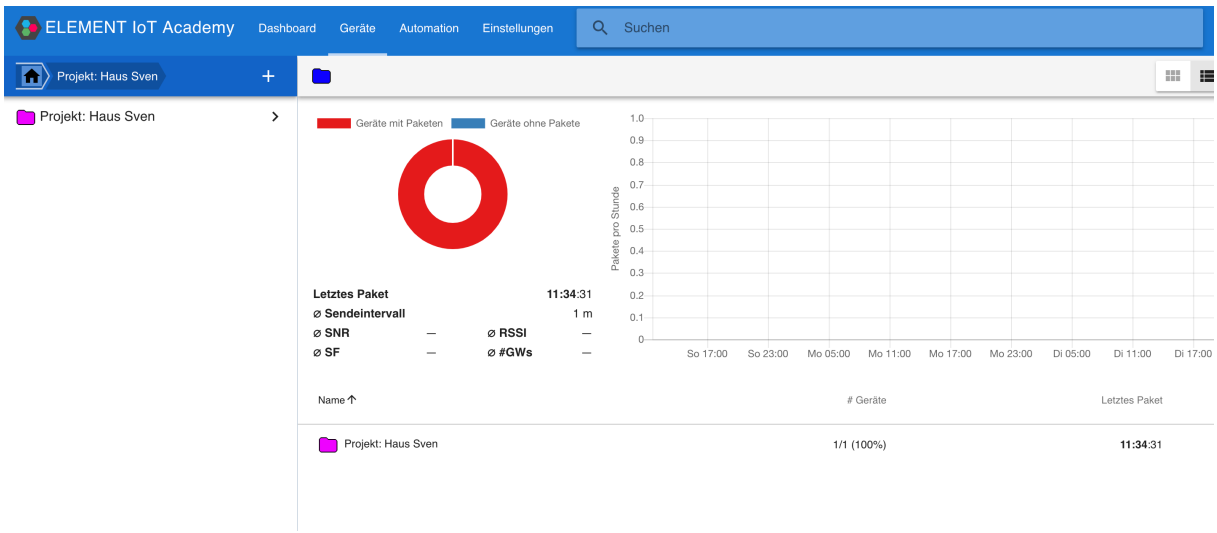


Abbildung 9.10: Screenshot

9.7 Berechtigung auf einen Ordner und einen Unterordner

Es ist möglich, auch Berechtigungen auf einzelne Unterordner zu vergeben. Um dieses Beispiel nachvollziehen zu können, legen Sie bitte einen neuen Ordner mit der Bezeichnung “Projekt: Haus Alex” und 2 weiteren Ordnern innerhalb dieses Ordners an: “1. OG” und “2.OG”. Legen Sie für Testzwecke im Ordner “2.OG” ebenfalls ein Gerät an.

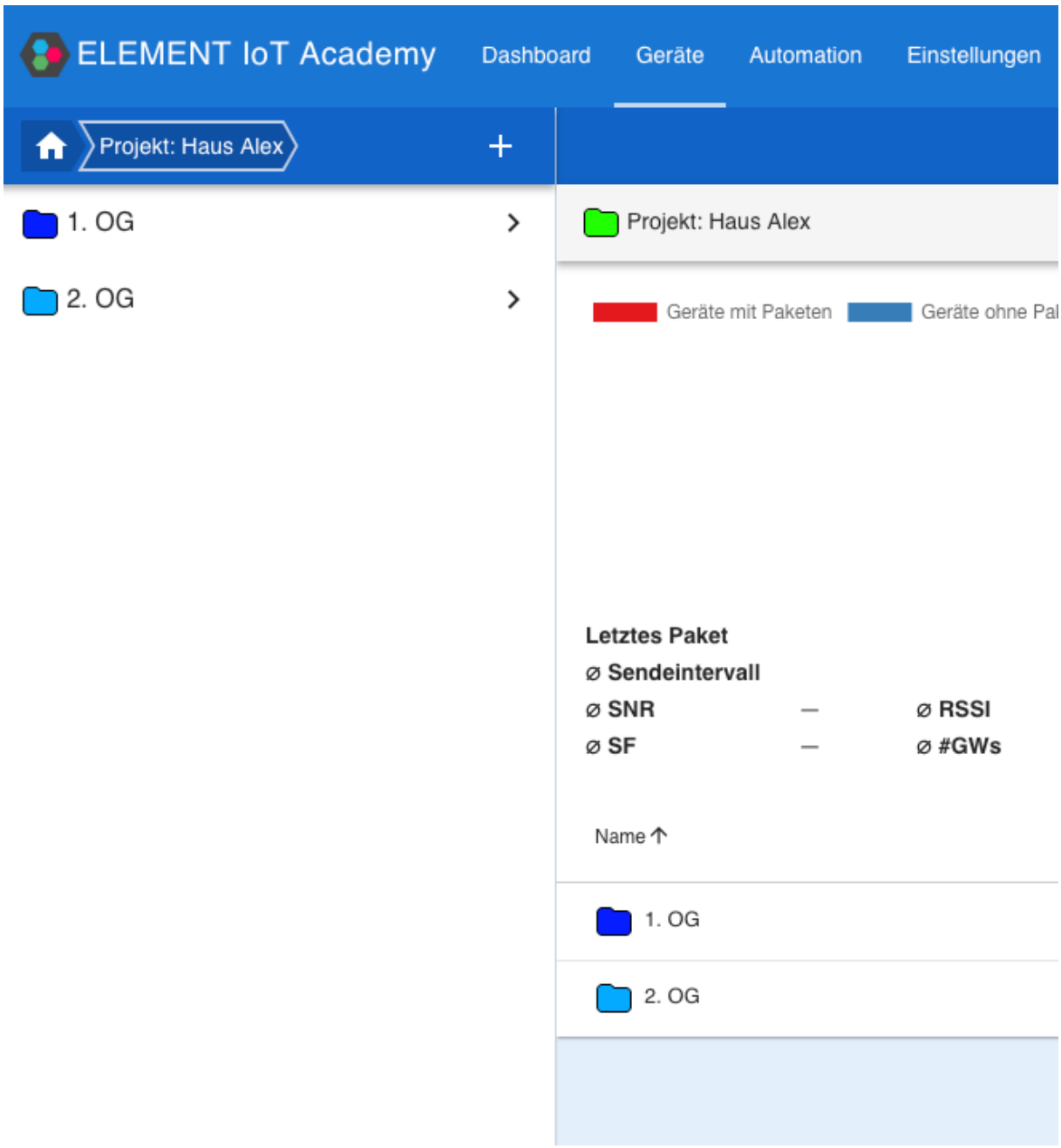


Abbildung 9.11: Screenshot

Das Ziel ist es, dem Test-Benutzer nur Berechtigungen auf den Ordner “2.OG” zu gewähren (Vollzugriff).

Legen Sie nun eine neue Gruppe mit dem Namen “Zugriff 2.OG” und folgenden Berechtigungen an:

- Ordner

- Treiber
- Parser

Im Feld “Ordnerbaum” geben Sie bitte folgenden Ausdruck an: Projekt: Haus Alex/2. OG/**

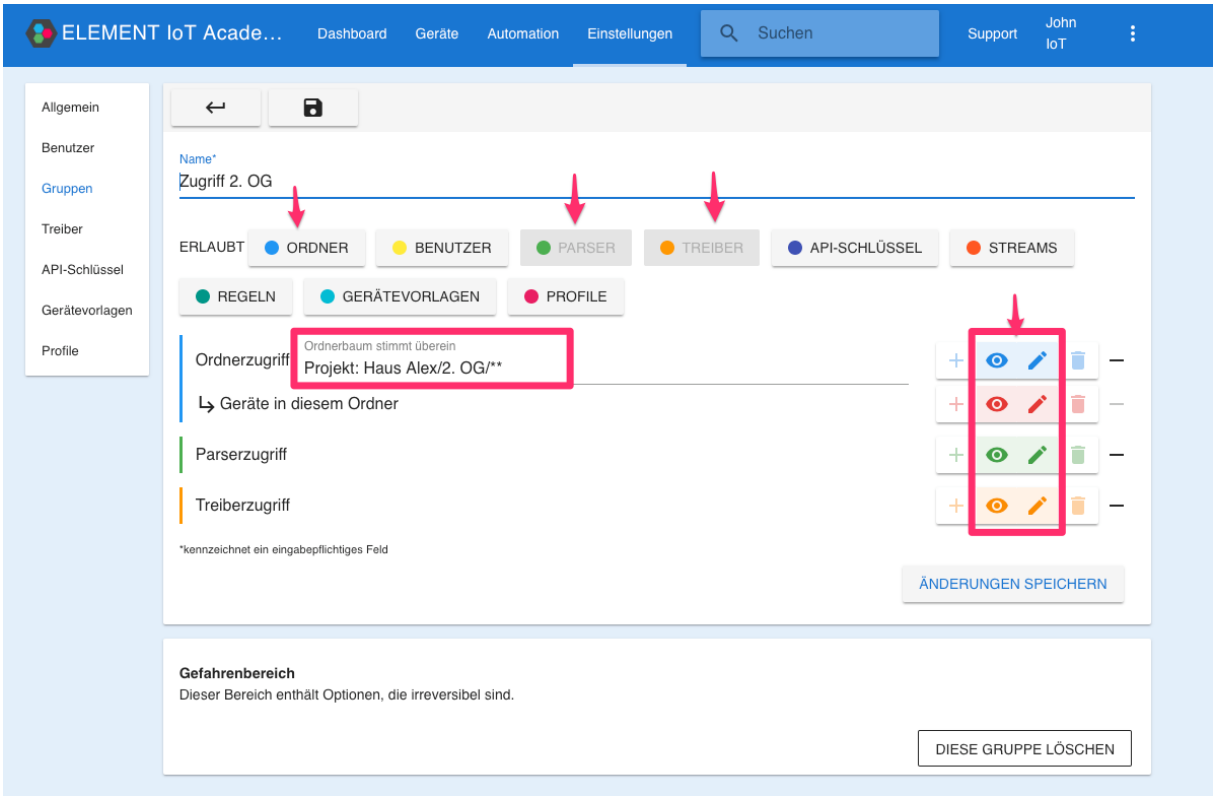


Abbildung 9.12: Screenshot

Mit diesen Einstellungen erhält die Gruppe Zugriff auf den Ordner “Projekt: Haus Alex/2.OG”. Die Mitglieder dieser Gruppe dürfen Änderungen durchführen aber nicht löschen. Soll die Gruppe auch die Berechtigung zum Löschen erhalten, muss die entsprechende Option (Papierkorb) ebenfalls ausgewählt werden.

9.8 Gruppe für Geräteadministratoren

Möchten Sie einem Personenkreis erweiterte Berechtigungen geben, können Sie entweder die Gruppe “Vollzugriff” nutzen, welche automatisch durch die ELEMENT IoT-Plattform erstellt wird, oder eine eigene Gruppe erzeugen. Mit solch einer Gruppe könnten Sie zum Beispiel eine Administratoren-Gruppe nur für Geräte im Ordner “Projekt: Haus Alex” anlegen.

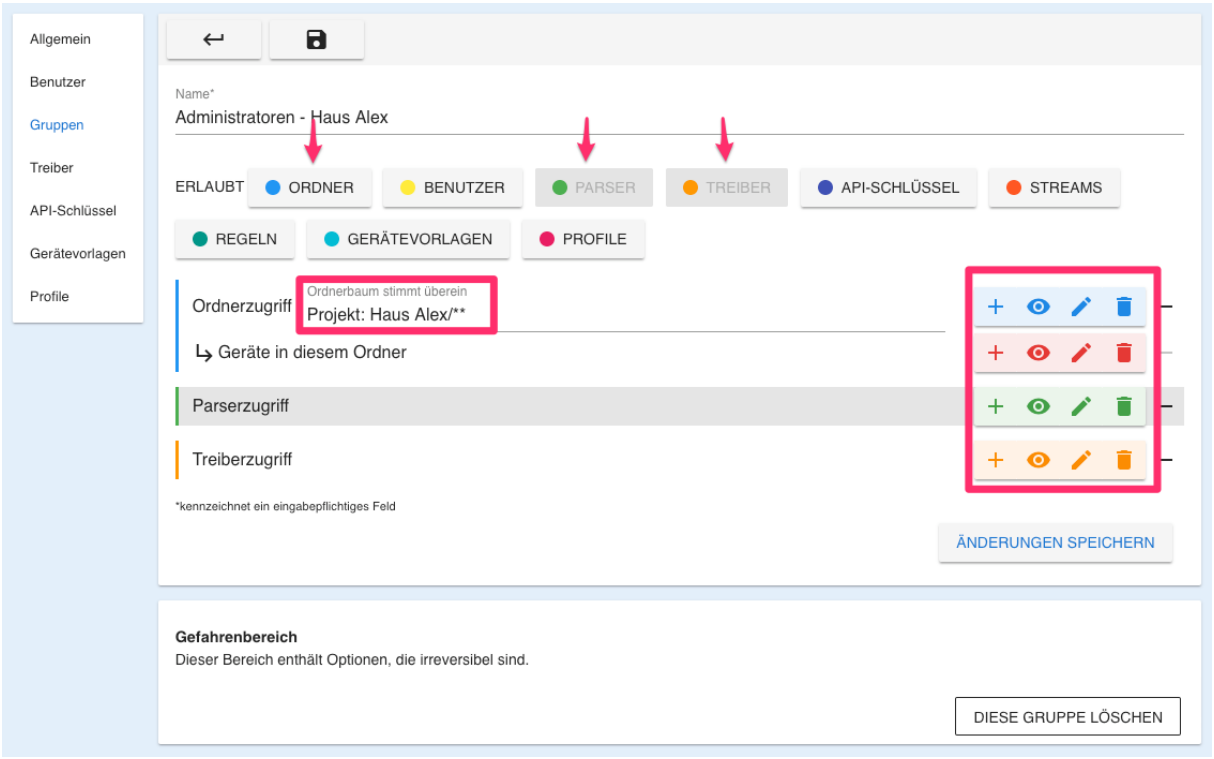


Abbildung 9.13: Screenshot

9.9 Gruppe für Entwickler (Parser)

In gewissen Konstellationen kann es sinnvoll sein, explizite Gruppen für zum Beispiel Entwickler zu erstellen. So könnten Sie Entwicklern nur Zugriff zum Erstellen von Parsern geben, ohne dass diese Geräte oder andere Objekte in der Plattform sehen oder bearbeiten können.

Legen Sie dafür eine neue Gruppe mit dem Namen “Entwickler” an, und vergeben Sie folgende Berechtigungen:

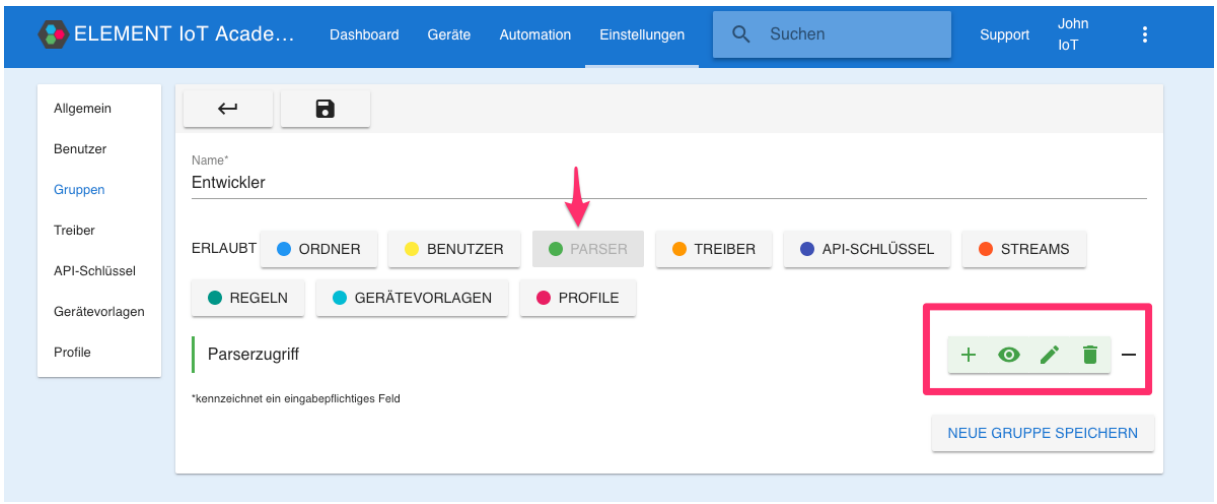


Abbildung 9.14: Screenshot

9.10 Gruppen und die ELEMENT API

Gruppen können nicht nur Benutzern zugeordnet werden, Sie haben ebenfalls die Möglichkeit, API-Schlüssel mittels Gruppen einzuschränken. Öffnen Sie dafür den Bereich **Eintellungen** - **API Schlüssel** und bearbeiten Sie den entsprechenden API-Schlüssel.

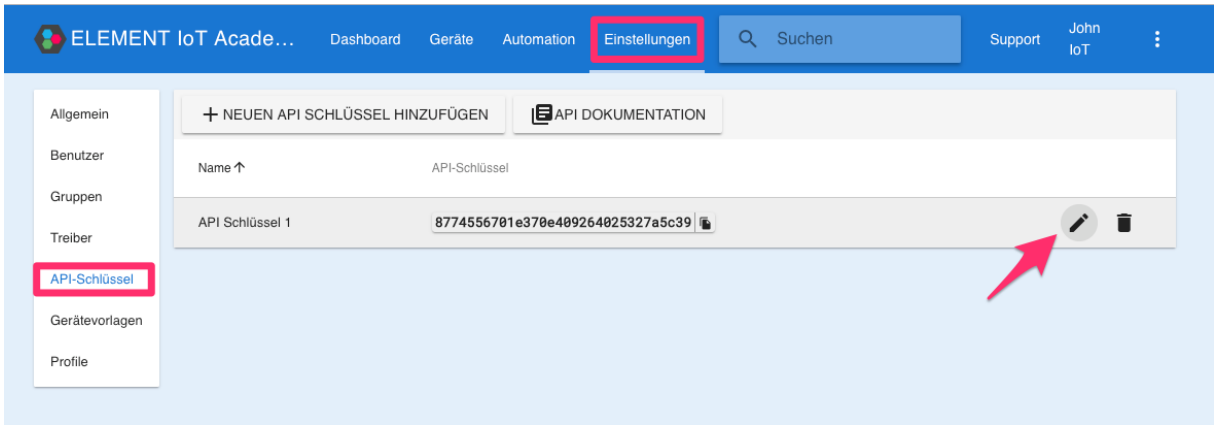


Abbildung 9.15: Screenshot

In den Einstellungen zu dem API-Schlüssel können Sie nun eine entsprechende Gruppe zu diesem hinzufügen.

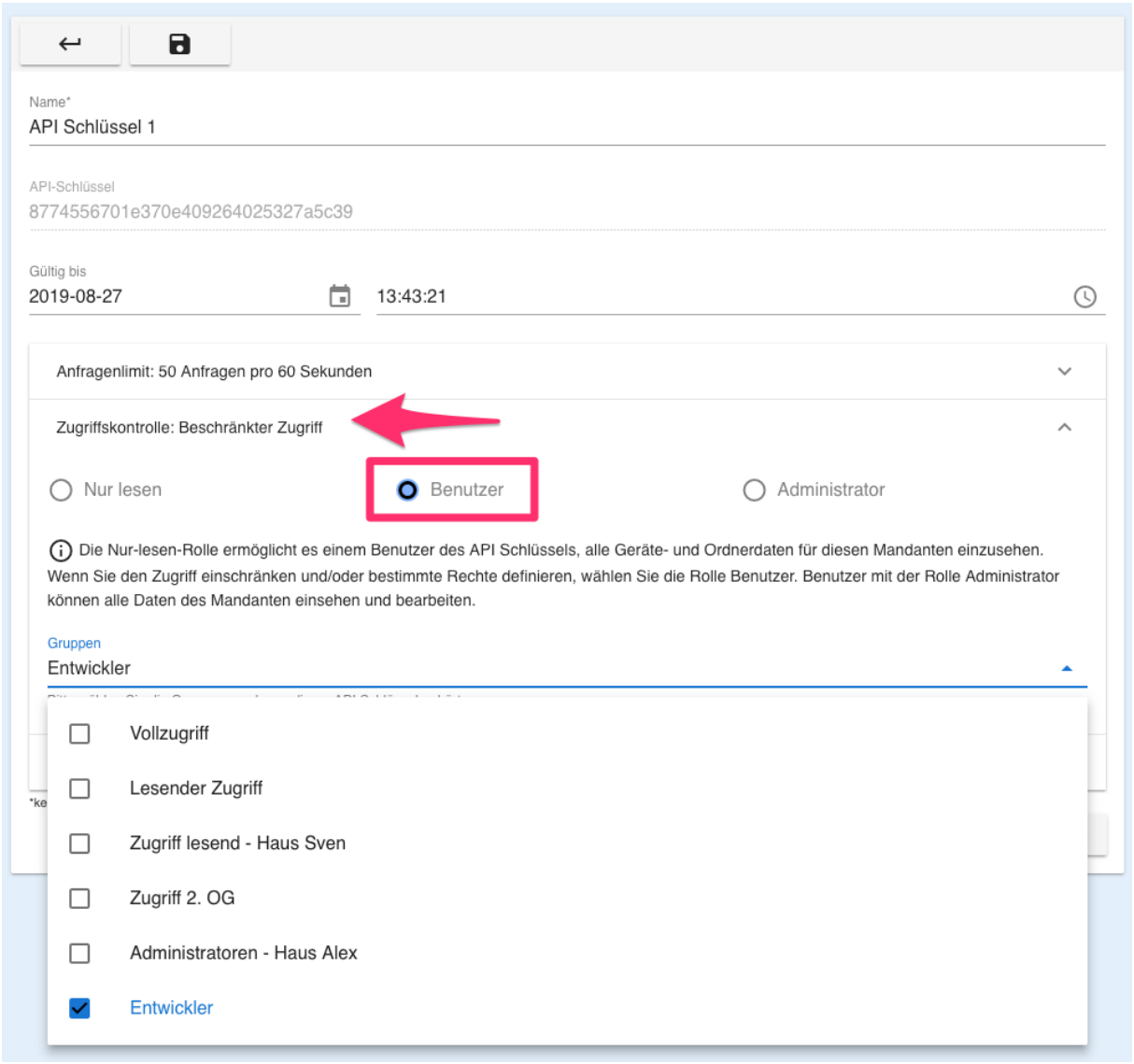


Abbildung 9.16: Screenshot

10 Berechtigungen API

10.1 Verwaltung von API-Schlüsseln

API-Schlüssel werden benötigt, um Zugriff auf die ELEMENT API zu erlangen. Sie können die vorher definierte Anzahl (wird beim Erzeugen des Mandanten festgelegt) an API-Schlüsseln vergeben.

Um einen neuen API-Schlüssel zu erzeugen, öffnen Sie den Bereich **EINSTELLUNGEN** und dort die Einstellung für **API-Schlüssel**.

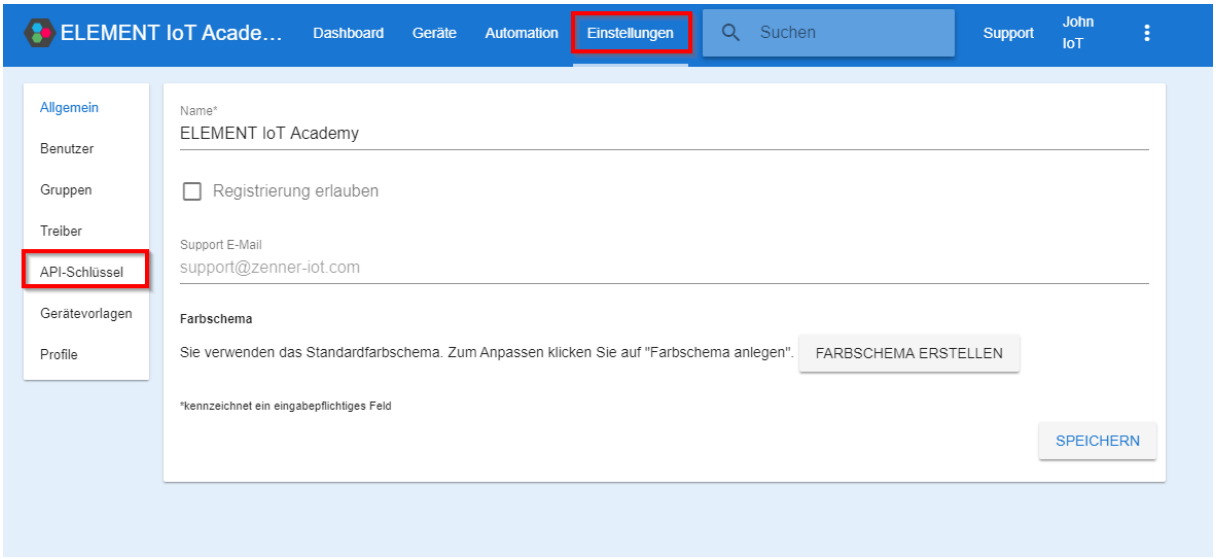


Abbildung 10.1: Screenshot

In der folgenden Maske klicken Sie bitte auf **NEUEN API-SCHLÜSSEL HINZUFÜGEN**. Vergeben Sie einen eindeutigen Namen und eine Laufzeit für den API-Schlüssel. Nach dem Ablaufen der Laufzeit wird der API-Schlüssel automatisch deaktiviert.

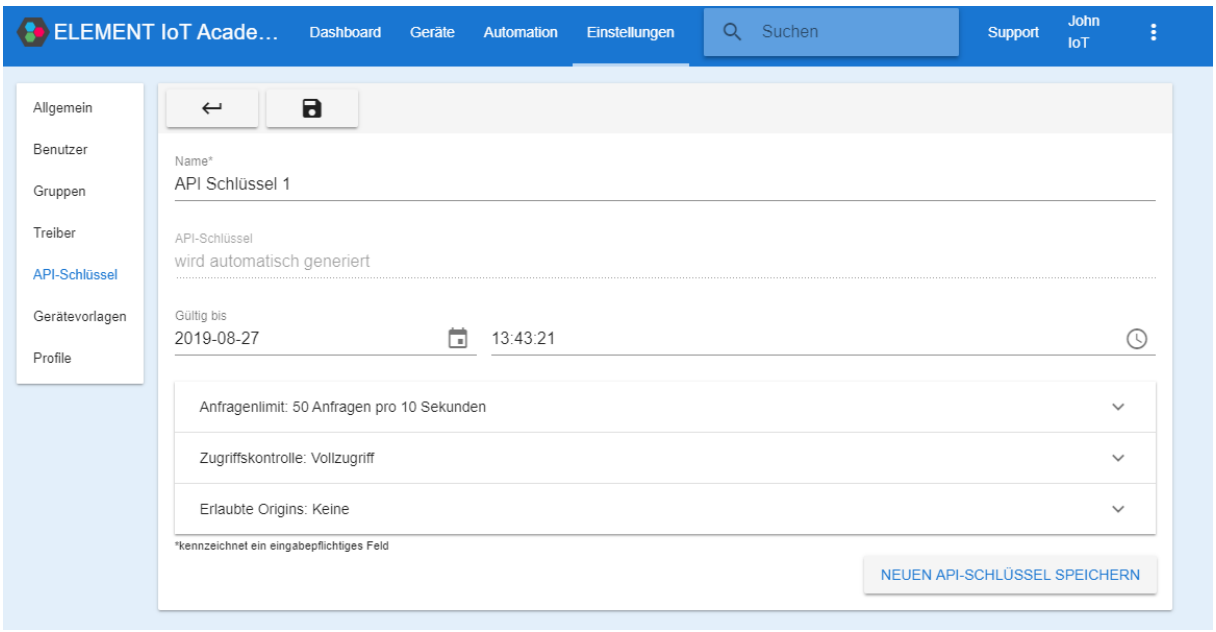


Abbildung 10.2: Screenshot

Der eigentliche API-Schlüssel wird automatisch erzeugt.

Neben diesen Einstellungen haben Sie noch die Möglichkeit, Limits und Berechtigungen festzulegen.

10.2 Anfragenlimit

Mit Hilfe eines Anfragenlimits können Sie die maximalen Zugriffe des API-Schlüssels über einen Zeitraum festlegen.

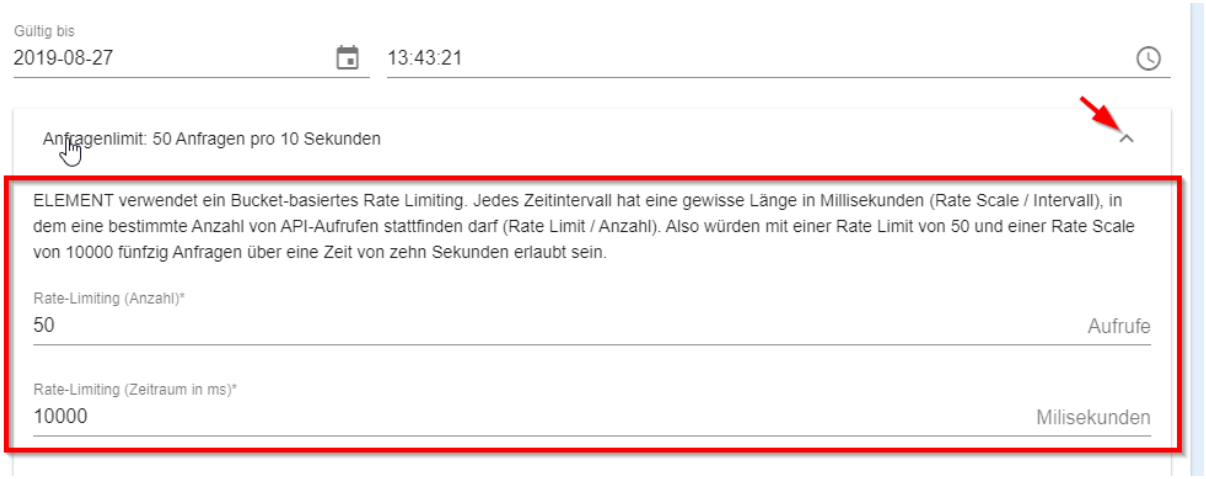


Abbildung 10.3: Screenshot

Möchten Sie zum Beispiel 10 Zugriffe pro Minute erlauben (was sehr wenig ist), würden die Werte wie folgt aussehen:

- Rate-Limiting (Anzahl): 10
- Rate-Limiting (Zeitraum in ms): 60000

Die Werte für den Zeitraum müssen in Millisekunden angegeben werden. Für das einfache Umrechnen können Sie Google nutzen, geben Sie dafür einfach in das Suchfeld “60 Sekunden in Millisekunden” ein.

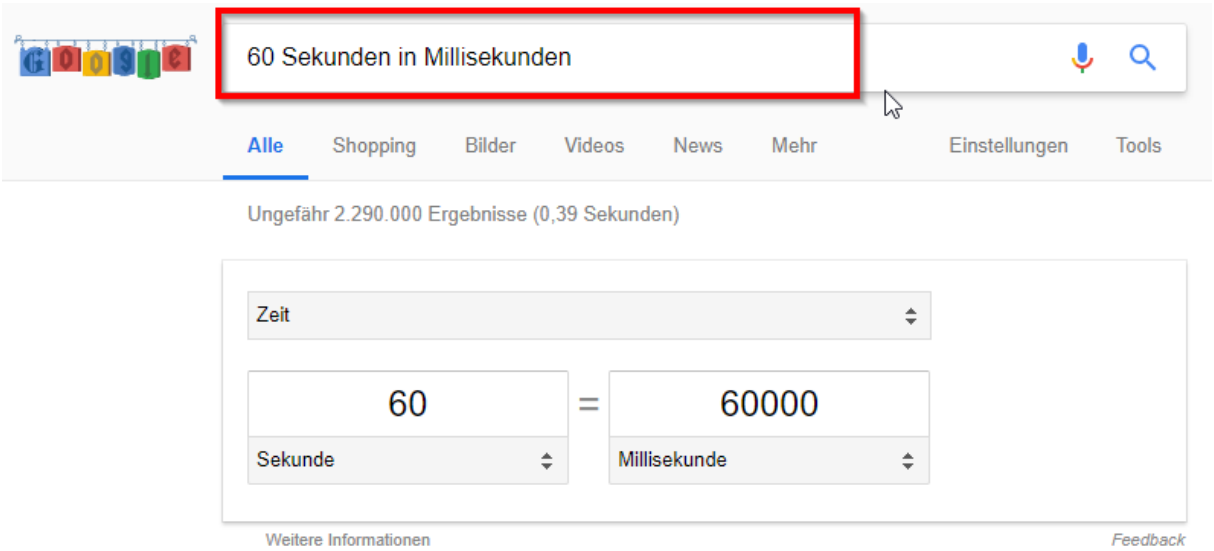


Abbildung 10.4: Screenshot

10.3 Zugriffskontrolle

Über diese Einstellung können Sie die Berechtigungen für einen API-Schlüssel einschränken. Es können dieselben Berechtigungen wie auch für Benutzer gesetzt werden. Siehe dazu: How To: Berechtigungen

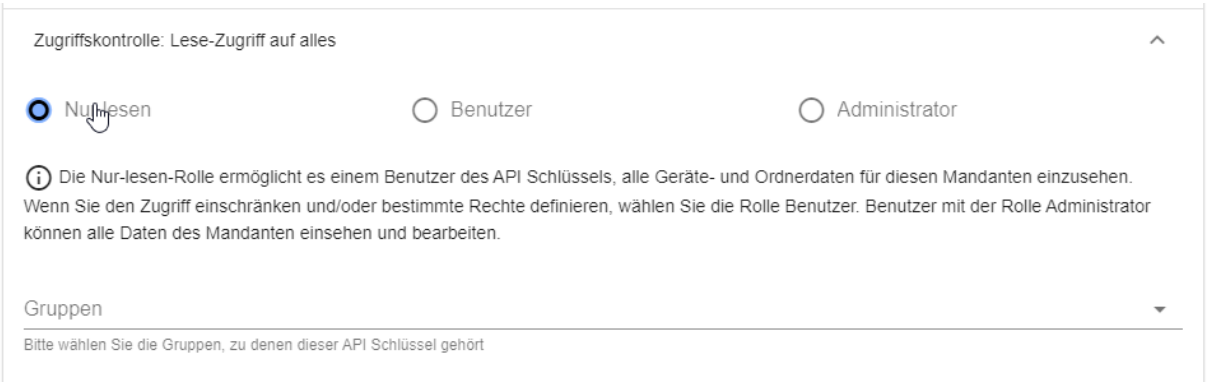


Abbildung 10.5: Screenshot

10.4 Erlaubte Origins

Mit dieser Einstellung können Sie festlegen, welche Quellen Zugriff auf die API erhalten. Haben Sie zum Beispiel eine Webanwendung, welche die API nutzen soll, muss die entsprechende Quell-URL eingetragen werden.

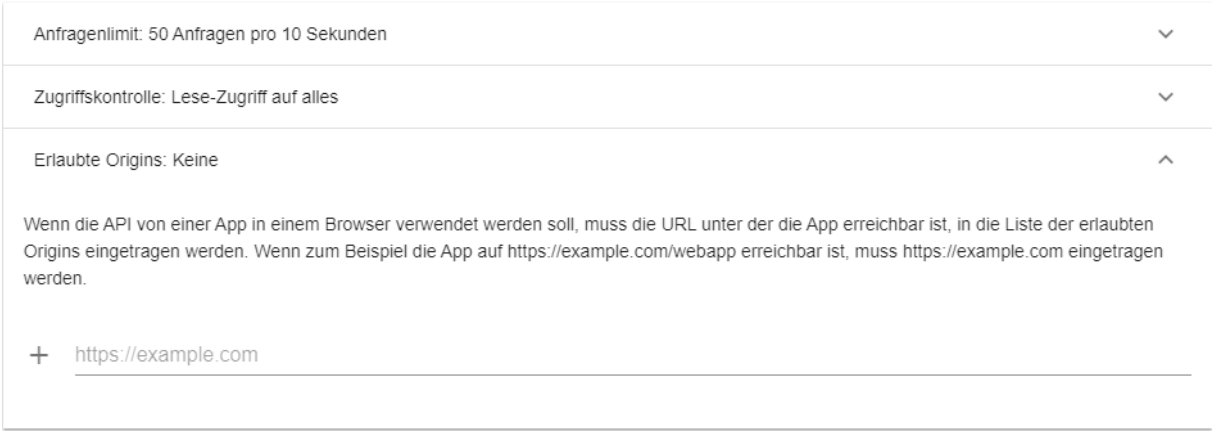


Abbildung 10.6: Screenshot

10.5 Bearbeiten und Löschen von API-Schlüsseln

Um die Einstellungen für einen API-Schlüssel zu ändern, öffnen Sie erneut den Bereich **EINSTELLUNGEN - API-Schlüssel**. In der Übersicht können Sie den Schlüssel über das Stift-Symbol bearbeiten oder über das Papierkorb-Symbol löschen. Bitte beachten Sie, dass nach dem Löschen kein Zugriff mehr auf die API mit dem entfernen API-Schlüssel möglich ist.

11 Datenstrukturen in ELEMENT IoT

Um effektiv in Filtern, Regeln, Visualisierungen und über die API durch die Datenstrukturen von ELEMENT IoT zu navigieren, ist es wichtig, zunächst die zugrundelegende Datenstruktur zu verstehen. Im Folgenden werden die wichtigsten Entitäten in ELEMENT IoT beschrieben und ihre Verknüpfung untereinander dargestellt.

11.1 Geräte

ELEMENT IoT hat als zentrale Entität das Gerät (Device https://docs.element-iot.com/data_structure/devices/).

Jedes Gerät hat die folgenden Attribute (hier nur die wichtigsten für Regeln und Visualisierungen):

- **name**

  Der Name des Geräte, so wie er in der Benutzeroberfläche angezeigt wird

- **slug**

  Der “technische” Name eines Gerätes. Er wird aus dem Namen gebildet, der dem Gerät beim Anlegen gegeben wurde, indem Leerzeichen uns Sonderzeichen ersetzt werden. Er kann später nicht mehr geändert werden, daher eignet er sich gut, um darüber in der API das Gerät zu referenzieren.

- **location**

  Sofern dem Gerät ein Ort zugeordnet wurde, enthält dieses Attribut die Latitude und Longitude.

Geräte haben darüber hinaus Schnittstellen und Profile und befinden sich in Ordnern.

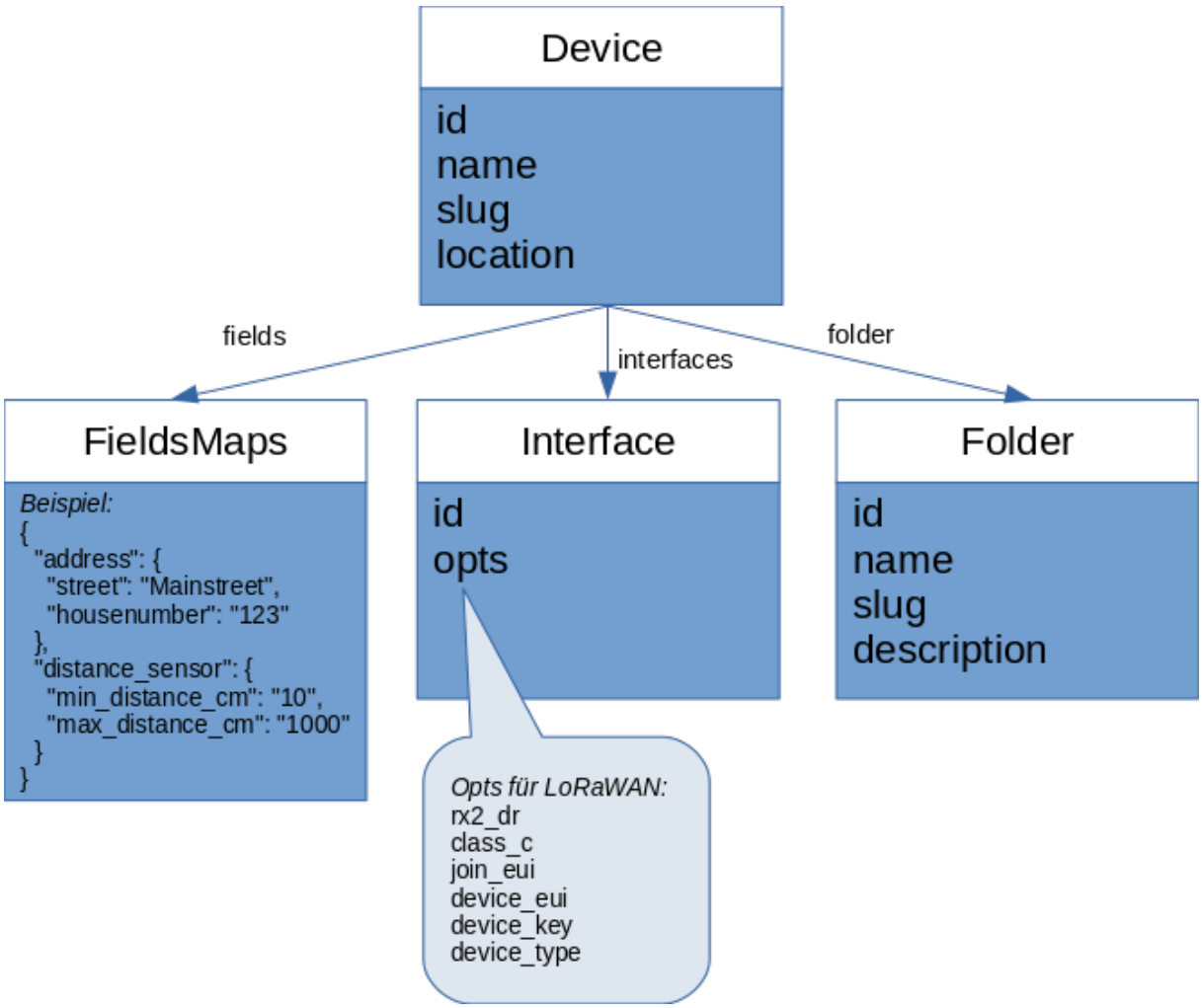


Abbildung 11.1: Geräte und angrenzende Entitäten

Komplette Liste der Assoziationen

Assoziation	Kardinalität	Entität	Beschreibung
`mandate`	one	Mandate	Mandant, in dem sich das Gerät befindet
`parser`	one	Parser	der zugeordnete Parser
`tags`	many	Folder	Liste der Ordner, in denen sich das Gerät befindet
`fields`	virtual, one	Fields Map	virtuelle Assoziation zum vereinfachten Zugriff auf Profildaten

Assoziation	Kardinalität	Entität	Beschreibung
`profiles`	many	Profile	Liste der Profile, die das Gerät hat
`profile_data`	many	Profile Data	Profildaten, einfacher über `fields` zugreifbar
`interfaces`	many	Interface	alle Interfaces, die das Gerät hat

11.2 Schnittstellen

Schnittstellen beschreiben, auf welchem Weg Daten zu einem Gerät empfangen werden. LoRaWAN-Geräte haben entsprechend eine Schnittstelle zu einem LoRaWAN-Netzwerkserver (in der Regel ELEMENT LNS). In der Schnittstelle werden die Schnittstellen-spezifischen Einstellungen vorgenommen.

Da ein Gerät mehr als eine Schnittstelle haben kann, ist die Assoziation *Interfaces* eine Liste, deren Elemente über den Array-Syntax mit eckigen Klammern zugegriffen werden können. Z.B. `interfaces[0]` für das erste Interface (in den meisten Fällen auch das Einzige).

LoRaWAN-Geräte mit dem ELEMENT LNS haben eine Schnittstelle, deren `opts` Feld die LoRaWAN Daten enthält:

Feldname	Typ	Beschreibung
`rx2_dr`	`string`	Datenrate von Rx2
`class_c`	**`boolean`**	Unterstützt Class c
`join_eui`	`string` (hex)	Join/App-EUI
`device_eui`	`string` (hex)	Dev-EUI
`device_key`	`string` (hex)	App-Key

11.3 Profile

Einem Geräte können mehrere Profile zugeordnet werden. Profile beschreiben Stammdatenfelder die für ein Gerät gefüllt werden können oder müssen. Ein Profil hat einen technischen Namen, über den von einem Gerät aus die definierten Felder erreicht werden können. Der Einfachheit halber, kann über die virtuelle Relation `fields` eine Schlüssel-Wertpaar-Datenstruktur zugegriffen werden. Jeder technische Name eines Profils, welches dem Gerät zugeordnet ist, ist ein Schlüssel in dieser Datenstruktur.

Der grundsätzliche Aufbau ist wie folgt:

```
{
  "<profile 1 technical name>": {
    "<field 1 technical name>": "<field 1 data>",
    "<field 2 technical name>": "<field 2 data>"
  },
  "<profile 2 technical name>": {
    "<field 1 technical name>": "<field 1 data>",
    "<field 2 technical name>": "<field 2 data>"
  }
}
```

Ein Beispiel für eine Gerät mit den Profilen `address` und `distance_sensor` würde so aussehen:

```
{
  "address": {
    "street": "Mainstreet",
    "housenumber": "123"
  },
  "distance_sensor": {
    "min_distance_cm": "10",
    "max_distance_cm": "1000"
  }
}
```

Die Felder in den Profilen können die folgenden Typen haben:

- Zeichenkette
- Ganze Zahl
- Fliesskommazahl
- Boolescher Wert
- Datum und Uhrzeit
- Datum
- Uhrzeit

11.4 Ordner

Geräte sind immer mindestens einem und optional mehreren Ordnern zugeordnet. Ordner haben die folgenden Attribute:

- **name**

  Der Name bzw. Pfad des Ordners. Der Pfad kann `/`-Zeichen enthalten, was bedeute, dass der Ordner ein Unterordner ist. Zum Beispiel bedeutet `Hauptordner / Unterordner / Kind`,

dass der Ordner mit dem Namen `Kind` in dem Ordner `Unterordner` liegt, der wiederum im Ordner `Hauptordner` liegt.

- **slug**

  Der technische Name des Ordners, analog zum *slug* bei Geräten

- **describtion**

  Ein Beschreibungstext für den Ordner

11.5 Navigation durch die Felder und Assoziationen

Der Zugriff auf Felder und Assoziationen in z.B. Regeln, Filtern und Visualisierung erfolgt mittels der `.`-Notation. Ist das Gerät das aktuelle Root-Element (z.B. in der Filteransicht eines Ordners), kann mit `fields.address.street` auf das Profil mit dem technischen Namen `address` und dadrin auf das Feld `street` zugegriffen werden.

In Fällen, wo eine 1-zu-n-Beziehung besteht, wird die Listennotation mit eckigen Klammern verwendet, z.B. `interfaces[0].opts.device_eui` um auf die Device-EUI eines LoRaWAN-Gerätes zuzugreifen (sofern das erste Interface das LNS-Interfae ist).

11.6 Pakete

Packete sind Rohdaten von einem Gerät. Die eigentlichen Daten sind dabei im Feld `payload` enthalten, desen Inhalt treiberabhängig ist.

Im Fall von LoRaWAN-Paketen enthält `payload` die entschlüsselte binäre Payload als *hex*.

Felder

Feld	Typ	Beschreibung
`id`	`uuid`	ID des Pakets
`payload`	`string` oder `json`	Abhängig von Feld `payload_encoding`, Hexadecimal-codierte binäre Payload, JSON oder UTF8-String
`packet_type`	`string`	Treiberanhängiger Wert, der die Codierung des Feldes Payload angibt

Feld	Typ	Beschreibung
`meta`	`map`	Metadata, treiberabhängig. Im Fall von LoRaWAN enthalt das Feld u.a. den Spreizfaktor
`transceived_at`	`ISO8601`	Vom Treiber gesetzte Zeit des Empfangs, Quelle der Zeitmessung ist treiberabhängig
`inserted_at`	`ISO8601`	Zeit, zu der das Paket in der Datenbank gespeichet wurde
`is_meta`	**`boolean`**	Gibt an, ob es sich um ein Paket handelt, das nicht geparsed werden soll, z.B. Join-Requests

Assoziationen

Assoziationen	Kardinalität	Referenziert auf	Beschreibung
`interface`	Eins	Schnittstellen	Interface, über das das Paket gekommen ist
`device`	Eins	Geräte	Gerät, zu dem das Paket gehört
`gateway_stats`	Mehrere	Gateway Statistiken	Bei LoRaWAN-Geräten die Liste der Gateways mit SNR und RSSI

11.7 Meta-Feld von Paketen

Das Meta-Feld von Paketen besitzt bei LoRaWAN unter anderem die Felder:

- `data_rate`
- `frequency`
- `frame_count_up`
- `frame_port`
- `confirm`

Beispiel für den Inhalt des `meta`-Felds bei LoRaWAN mit ELEMENT LNS

```
{
  "rx": null,
  "ack": false,
  "chan": 6,
```

```
  "codr": "4/5",
  "datr": "SF7BW125",
  "ipol": null,
  "modu": "LORA",
  "powe": null,
  "rfch": 1,
  "size": 21,
  "stat": 1,
  "region": "eu863",
  "confirm": false,
  "data_rate": 5,
  "dev_nonce": null,
  "frequency": 868.3,
  "frame_port": 1,
  "adr_ack_req": false,
  "region_meta": {
    "code": "eu863",
    "name": "EU863",
    "bitrate": 5470,
    "datarate": 5,
    "bandwidth": 125,
    "spreadingfactor": 7
  },
  "mac_commands": [],
  "gateway_stats": [
    {
      "snr": 9,
      "rssi": -71,
      "tmst": 4018682596,
      "router_id": 123656386774257,
      "router_id_hex": "00007076FF0204F1"
    }
  ],
  "frame_count_up": 24310,
  "lorawan_toa_ms": 56.57600000000001
}
```


11.8 Messwerte

Messwerte (Readings) entstehen durch den dem Gerät zugeordnete Parser, welcher die Pakete des Geräts verarbeitet. Die Struktur des Messwerte ist daher von der Implementierung des Parsers abhängig. Konkret ist die Rückgabe der `parse`-Funktion ausschlaggeben siehe Parserentwicklung.

Allen Messwerten sind die folgenden Felder und Assoziationen gemein:

Felder

Felder	Typ	Beschreibung
`id`	`uuid`	Id des Messwerts
`data`	`map`	enthält eine JSON-Struktur bzw. ELIXIR-Map, die der Parser festgelegt hat
`measured_at`	`ISO8601`	Zeit, zu dem der Messwert gemessen wurde. Parser- und Treiberabhängig
`inserted_at`	`ISO8601`	Zeit, zu der der Messwert geschrieben wurde
`location`	`Geo.Point`	Ort, an dem der Messwert gemessen wurde, ggf. leer. Parser- und Treiber-Abhängig

Assoziationen

Assoziationen	Kardinalität	Referenziert auf	Beschreibung
`packet`	one	Paket	Paket, aus dem der Messwert gebildet wurde
`device`	one	Geräte	Gerät, zu dem der Messwert gehört
`parser`	one	Parser	Parser, der den Messwert gebildet hat

11.9 Beispiel für den Zugriff auf Messdaten

Für den folgenen Beispielparser kann mittels `data.temperatur` auf den Messwert zugegriffen werden:

```
defmodule Parser do
  use Platform.Parsing.Behaviour

  def parse(<<temp::16>>, _meta) do
    %{temperatur: temp - 30)}
  end

end
```

12 Abacus

12.1 Einführung

An Stellen, wo in ELEMENT Terme eingegeben werden, u.a. Filter, Regel und Spalteninhalte bei Messwertlisten, werden *Abacus*-Terme verwendet.

12.2 Pfadausdrücke

Mit Hilfe der `.`-Notation kann auf Felder einer Entität und Assoziationen zugegriffen werden (siehe [Datenstrukturen]). Zum Beispiel `device.parser.name`, wenn das Root-Element `reading` ist.

Mit der *Eckige-Klammer*-Notation (`[n]`) kann auf das n-te assozierte Element zugegriffen werden (Zählung startet bei 0, 0 ist also das erste Element). Zum Beispiel `device.interfaces[0].opts.device_eui`, wenn das Root-Element `reading` ist und das erste Interface des Gerätes ein LoRaWAN-Interface ist.

12.3 Schlüsselwörter und Konstanten

In Abacus können die folgenden Schlüsselwörter genutzt werden:

- **`null`** - für einen nicht existenten Wert
- **`false`** und **`true`** - die beiden Boolschen Werte wahr und falsch

Beispiele:

- parser_id == null
- meta.confirm == false

In Abacus können außerdem Konstanten in den Ausdrücken verwendet werden, beispielsweise:

- `"ich bin ein string"` - für Textkonstanten
- `3` - die Zahl 3
- `3.0` - die Fließkommzahl 3.0

12.4 Vergleiche

Um Felder, auf welche mittels Pfadausdrücken zugegriffen wird, zu vergleichen, stehen die folgenden Vergleichsoperatoren zur Verfügung:

- == für Gleichheit
- != für Ungleichheit
- > für Größer
- < für Kleiner
- >=, <=, Für Größer- und Kleinergleich

Beispiele:

- `data.temperature >= 35`
- `parser.name == "Super Sensor 3000"`

12.5 Boolsche Operatoren

Um mehrere Bedingungen zu verknüfen, stehen boolsche Operatoren zur Verfügung (`a` und `b` können Abacus-Terme sein):

- `a && b` - nur wahr (**`true`**), wenn a und b **`true`** sind
- `a || b` - ahr (**`true`**), wenn a oder b, oder beide **`true`** sind
- `not a` - negiert den Wert von `a`

Beispiel:

- `data.door_open && data.brightness < 100`

12.6 Mathmatische Operatoren

- +, −, * und / für Addition, Subtraktion, Multiplikation und Division
- `a^b` für a hoch b
- `!n` für n-Fakultät

12.7 Funktionen

Zusätzlich zu den *Cast*-Funktionen, welche im nächsten Kapitel beschrieben werden, gibt es die folgenden Funktionen in Abacus:

Datetime functions:

- `date_trunc`, Beispiel: `date_trunc("day", measured_at)` extrahiert den Tag aus dem Datum
- `now`, Beispiel: `now()` gibt aktuelles Datum und Zeit zurück

String functions:

- `like` und `ilike`, Stringvergleich mit Platzhaltern (%), Beispiel: `ilike(name, "%gateway%")`
- `concat`, Zusammenfügen von Strings, Beispiel: `concat(name, "- ", parser.name)`
- `substr`, gibt den gewünschten Teil eines Strings zurück , Beispiel: `substr ( '1234567', -2 )`
- `to_hex`, Zahl hexcodieren, Beispiel: `to_hex(511)== "1ff"`
- `encode`, Binärpayload zu Hex codieren, Beispiel: `encode(packet.payload, "hex")`

12.8 Cast

Daten aus der Datenbank, die in Termen verwendet werden, müssen in einigen Fällen in einen anderen Datentyp überführt werden (*cast*). Dies ist inbesondere dann der Fall, wenn auf Daten zugegriffen und diese in Funktionen verwendet werden sollen, die in der Datenbank als JSON hinterlegt sind. Das ist bei Feldern in `reading.data` und Profildaten der Fall.

Dafür hat Abacus die folgenden Funktionen:

Grundtypen:

- numeric - Konvertiert JSON oder Text in nummerische Werte
- integer - Konvertiert JSON oder Text in ganze Zahlen
- float - Konvertiert JSON oder Text in Fließkommawerte
- text - Konvertiert JSON, Text oder boolsche Werte in Text
- boolean - Konvertiert JSON oder Text in boolsche Werte

Beispiele:

- numeric(“3”) - 3
- text(3) - “3”
- boolean(0) - false

Datums- und Zeittypen

- timestamp and timestamptz - Konvertiert Datums-Zeittypen oder Strings in Zeitstempel mit oder ohne Zeitzone (Endung `tz` für *mit Zeitzone*)

- date - Konvertiert Datums-Zeittypen oder Strings in ein Datum ohne Uhrzeit
- time and timetz - Konvertiert Datums-Zeittypen oder Strings in Uhrzeiten mit oder ohne Zeitzone (Endung `tz` für *mit Zeitzone*)
- interval - Konvertiert einen Text zu einem Interval

Beispiel für die Verwendung von `interval`:

- `measured_at >= now()- interval("3 months")` -> Filter für Messwerte aus den letzten drei Monaten

Beispiele:

- timestamptz(“2018-05-02 19:30 Europe/Berlin”) - 2018-05-02 17:30:00+00
- timestamp(“2018-05-02 19:30 Europe/Berlin”) - 2018-05-02 19:30:00
- timetz(“15:30 MESZ”) - 15:30:00+02
- time(“15:30 MESZ”) - 15:30:00
- date(“2018-03-19”) - 2018-03-19

13 Streams und Regeln

13.1 Streams

Streams sind eine Möglichkeit, Geräte zu Gruppen zusammenzufassen, so dass Daten von diesen gemeinsam über einen Websocket abgefragt werden können. Streams sind eine Gruppe von Ordnern, wahlweise mit oder ohne Unterordner und einzeln gewählte Geräte.

Außerdem dienen Streams dazu, dass Regeln nicht nur geräte- oder ordnerweise definiert werden können.

13.2 Regeln

Mittels Streams und Regeln lassen sich Aktionen ausführen, welche unter definierten Bedingungen ausgelöst werden. So ist es zum Beispiel möglich, eine E-Mail zu versenden, wenn ein Türkontaktsensor eine Öffnung feststellt, oder eine URL aufzurufen, wenn die Temperatur in einem Raum über 36 Grad steigt.

In diesem How To werden wir zwei Beispiele umsetzen.

- Versenden einer E-Mail, wenn ein neues Paket eines Temperatursensors eintrifft. Die E-Mail wird die aktuelle Temperatur als Text enthalten.
- Aufrufen einer URL mittels POST Request und Übertragung der Daten im JSON Format an einen Test-Endpunkt.

13.3 Anlegen Treiber für Testzwecke

Zur Simulation der eingehenden Daten werden zwei Dummy-Treiber angelegt.

- Dummy - Temperatur (Werte 10-40, zufälliger Wert, alle 30 Sekunden)
- Dummy - Türkontakt (Werte 1-2, zufälliger Wert, alle 30 Sekunden), 1 = Die Tür ist geschlossen, 2 = Die Tür ist geöffnet

Wie Sie Treiber (Dummy-Treiber) anlegen können, erfahren Sie im Tutorial für Treiber.

Treiber Temperatur

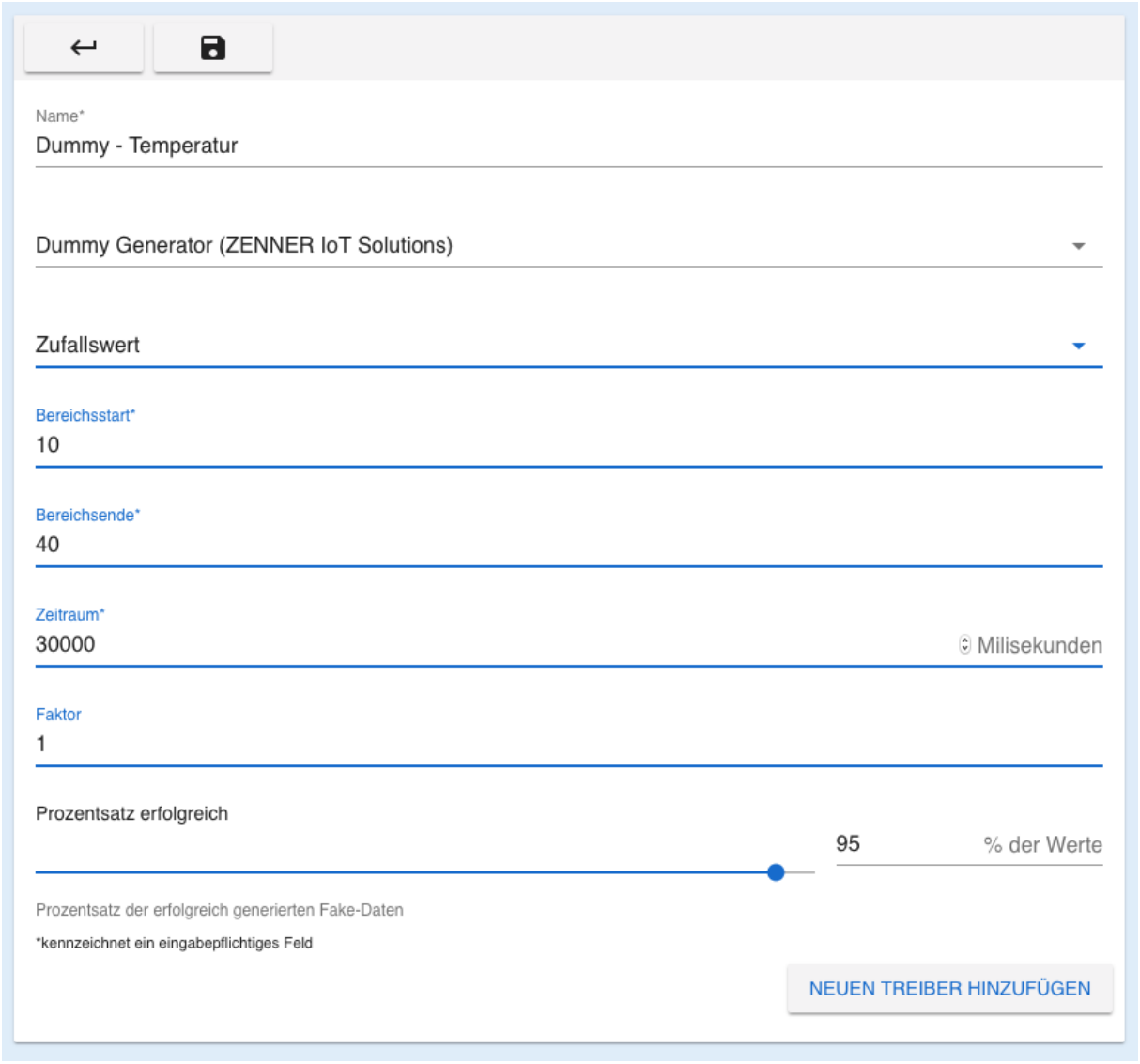


Abbildung 13.1: Treiber

Treiber Türkontakt

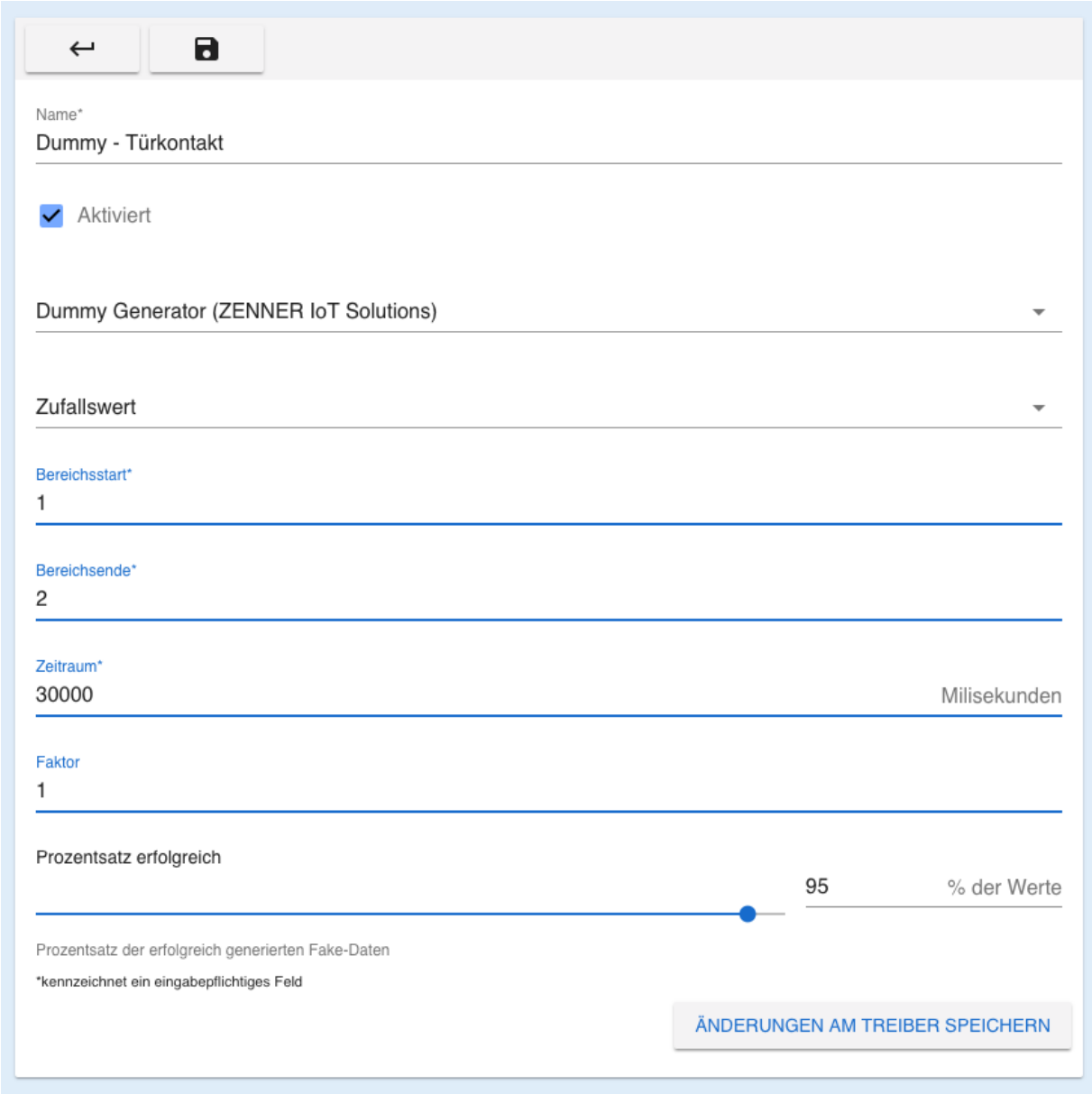


Abbildung 13.2: Treiber

13.4 Anlegen Parser für Temperatur und Türkontaktsensor

Für die Messwerte werden 2 Parser benötigt (Details zum Anlegen von Parsern finden Sie im How To: Parser).

- Temperatursensor

```
defmodule Parser do
  use Platform.Parsing.Behaviour

  def parse(event, _meta) do
    %{temperatur: get_in(event, ["payload"])}
  end

  def fields do
    [
      %{
        field: "temperatur",
        display: "Temperatur",
        unit: "C"
      }
    ]
  end
end
```

- Türkontakt

```
defmodule Parser do
  use Platform.Parsing.Behaviour

  def parse(event, _meta) do
    %{status: get_in(event, ["payload"])}
  end

  def fields do
    [
      %{
        field: "status",
        display: "Status" # 1 open, 0 closed
      }
    ]
  end
end
```

13.5 Anlegen Geräte für Testzwecke

Wir legen 2 Geräte in einem beliebigen Ordner an.

- Temperatursensor #1 unter verwendung des Dummy - Temperatur Treibers
- Türkontaktsensor #1 unter verwendung des Dummy - Türkontakt Treibers

Wie Sie Geräte anlegen können, erfahren Sie im entsprechenden How To für die Geräteverwaltung.

Gerät Türkontaktsensor #1

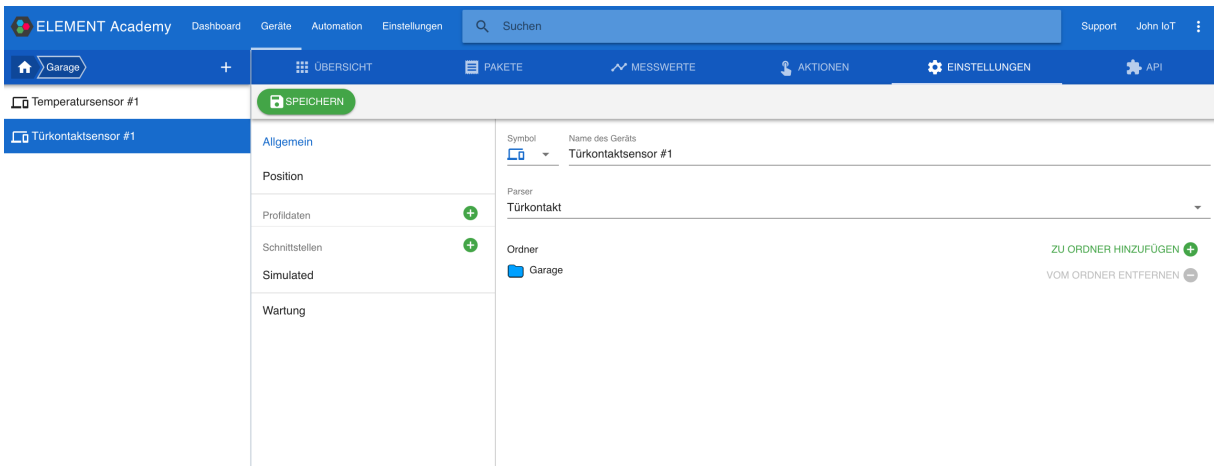


Abbildung 13.3: Geräte

Gerät Temperatursensor #1

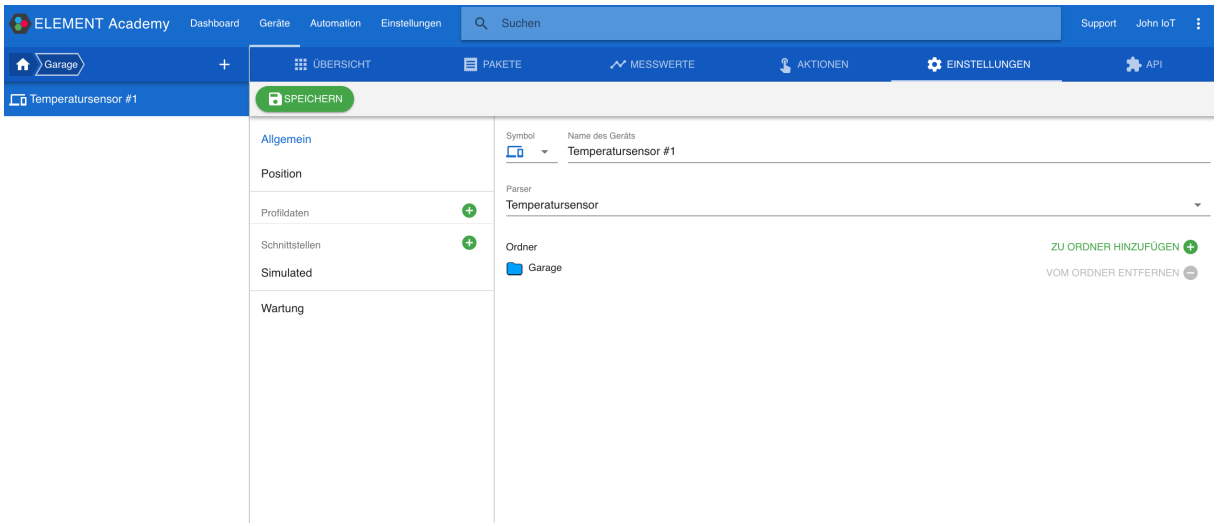


Abbildung 13.4: Geräte

13.6 Konfiguration der Streams

Für die Nutzung von Regeln werden im ersten Schritt 2 neue Streams benötigt. Ein Stream für das Gerät “Türkontaktsensor” sowie ein Stream für das Gerät “Temperatursensor”. Neben einzelnen Geräten können über Streams auch Geräte, welche sich in einem Ordner befinden, gruppiert werden. Ebenfalls ist es möglich, einen Stream für alle im Mandanten angelegten Geräte zu erstellen.

Zum Erstellen der Streams klicken Sie bitte in der Navigation auf **Automation** und anschließend auf **Regeln & Streams** und nutzen Sie die Option **STREAM ERSTELLEN**.

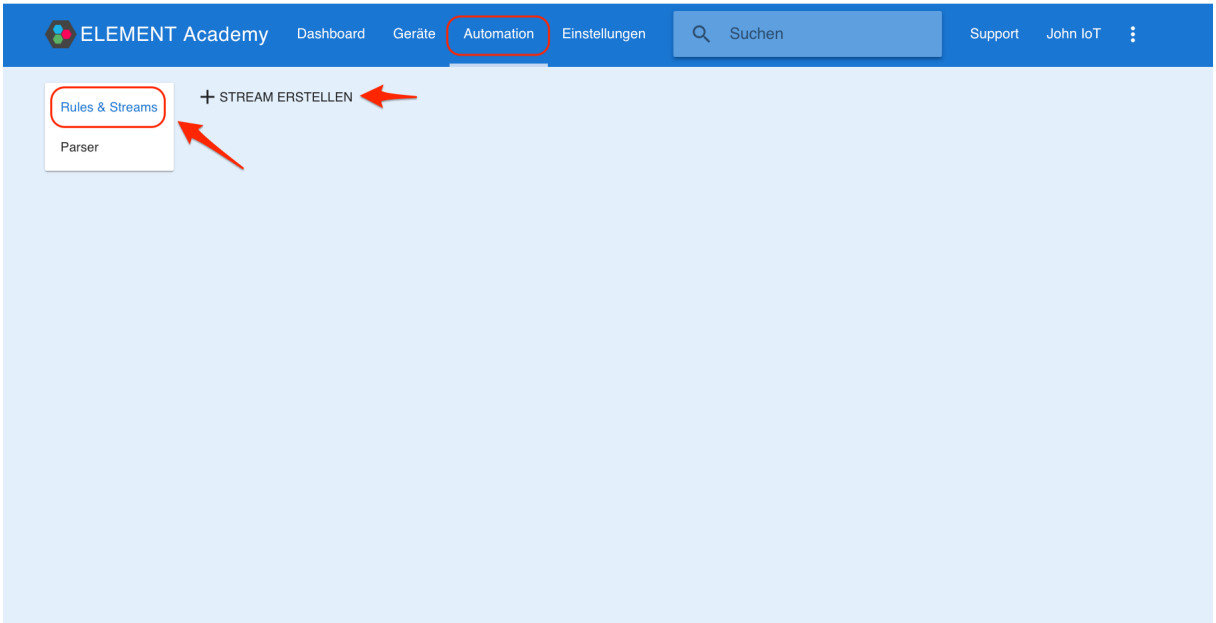


Abbildung 13.5: Streams

In der folgenden Maske vergeben Sie einen Namen, wählen bei Quell-Typ “Gerät” und als Gerät den Türkontaktsensor (Sie können einfach in das Feld ein “T” tippen, die Plattform schlägt dann mögliche Geräte vor).

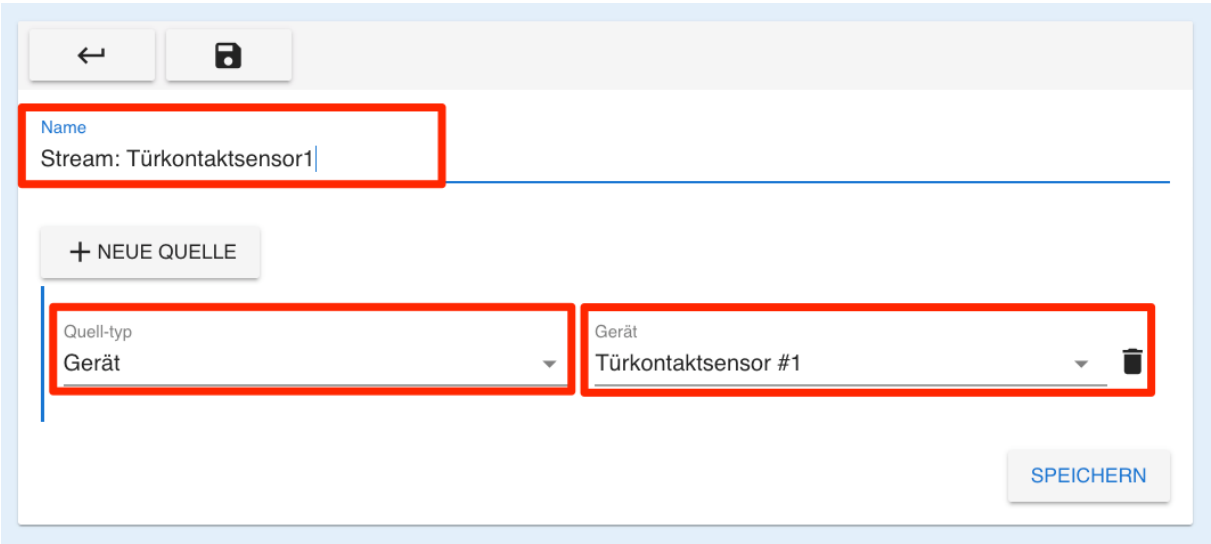


Abbildung 13.6: Streams

Klicken Sie anschließend auf **Speichern** und wiederholen Sie die Schritte für den Temperatursensor.

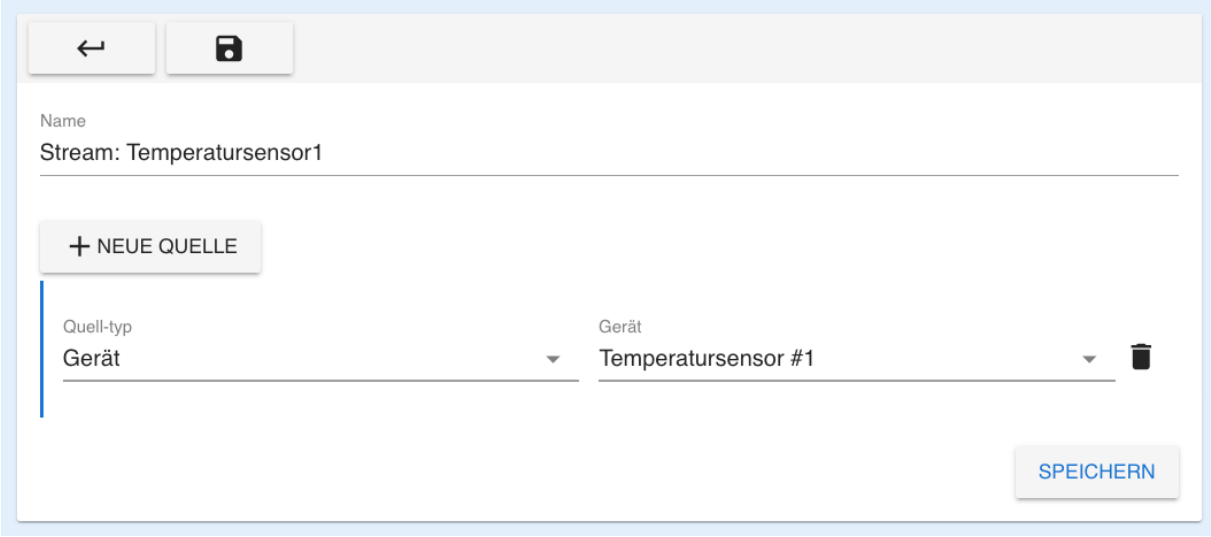


Abbildung 13.7: Streams

Haben Sie beide Streams angelegt, sollte die Übersicht folgendermaßen aussehen. Über das Stift-Symbol können Sie die Streams bearbeiten oder löschen. Das Löschen ist nur möglich, wenn der Stream von keiner Regel genutzt wird.

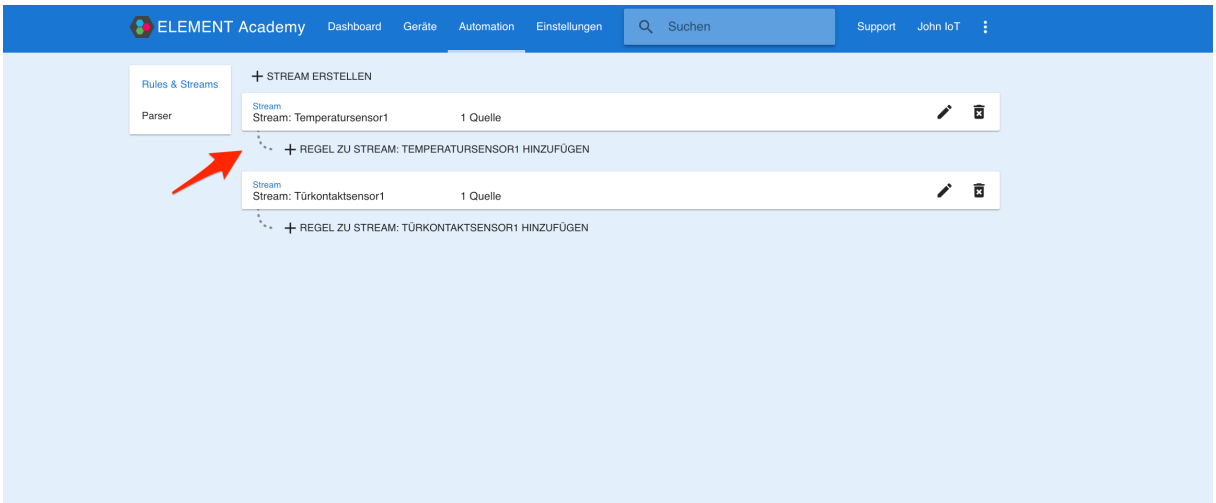


Abbildung 13.8: Streams

13.7 Anlegen der Regel (E-Mail)

Die Regeln werden in der selben Ansicht erstellt. Klicken Sie zum Erzeugen einer neuen Regel auf das “+” Symbol unter dem entsprechenden Stream.

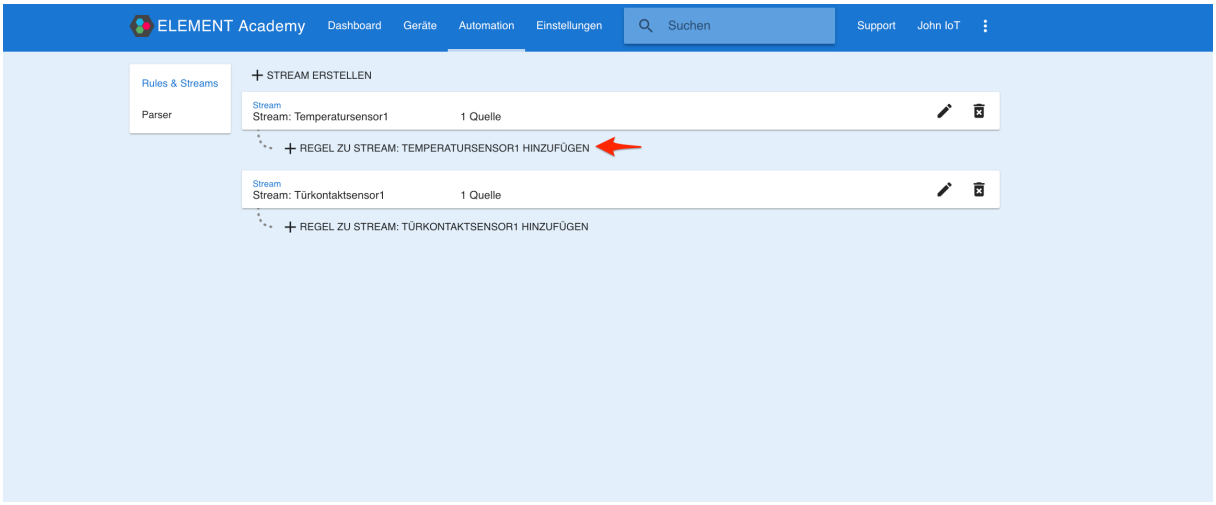


Abbildung 13.9: Regeln

Sie gelangen Sie in das Formular zur Konfiguration der Regel. Vergeben Sie hier als ersten Schritt einen eindeutigen Namen.

Weitere Einstellungen:

- Aktion ausführen: … sofort
- Stream: Temperatursensor1
- Ergeinis: Messwert hinzugefügt
- Bedingung: `data.temperatur > 36` (das Feld Temperatur wird durch den erstellten Parser bereitgestellt)
- Aktion: Benachrichtigung senden
- E-Mail Adresse des Empfängers: Ihre E-Mail-Adresse
- E-Mail Betreff: Temperatur übersteigt 36 Grad
- E-Mail Text: `Die Temperatur beträgt: {{ data.temperatur }}`

Sie können in den Bedingungen und dem Text auf alle Felder (siehe Abschnitt Platzhalter) zugreifen, welche durch den Parser befüllt werden. So lassen sich auch komplexe Szenarien realisieren.

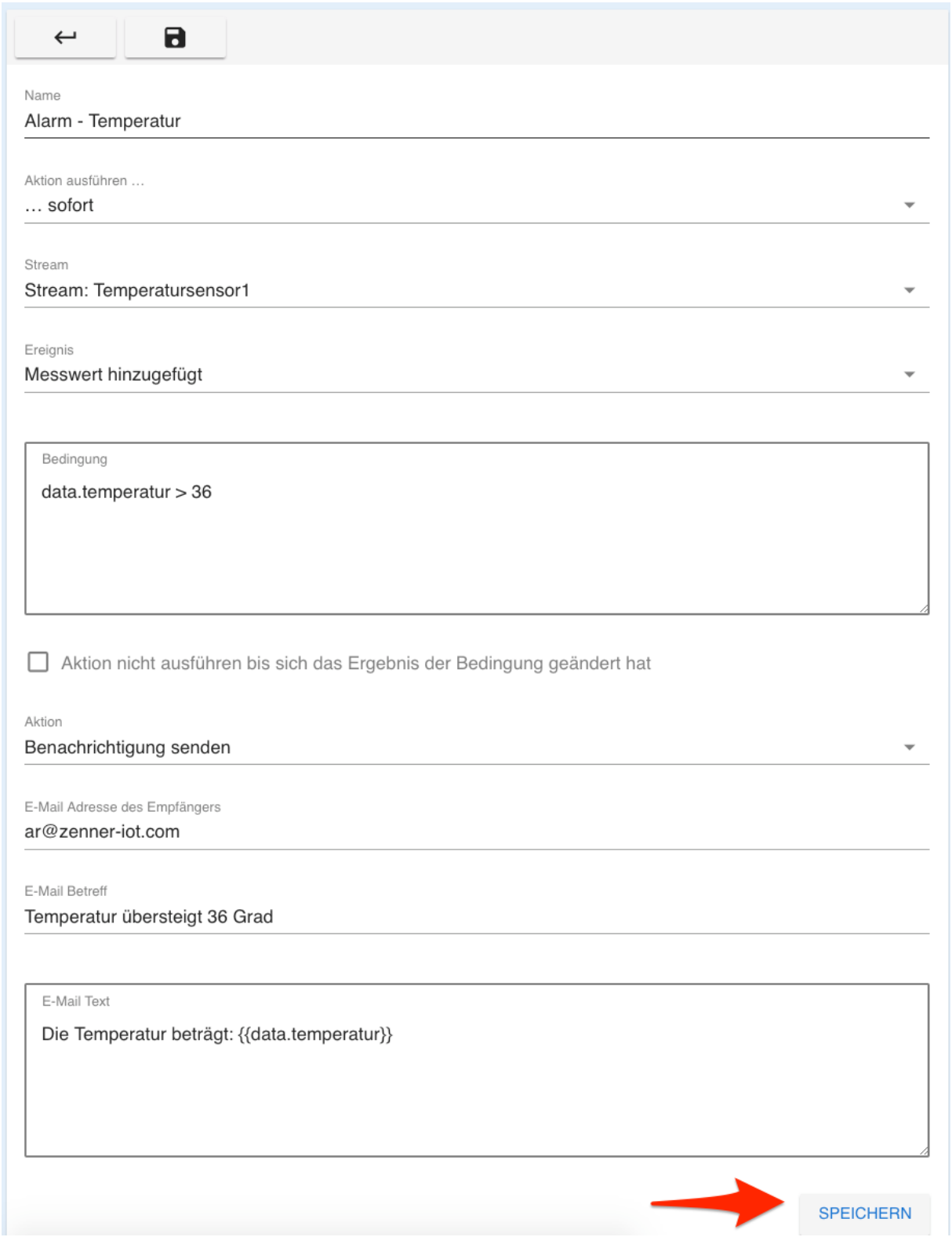


Abbildung 13.10: Regeln

Speichern Sie die Regel über den Button **SPEICHERN**. Wenn ein Messwert >36°C eintrifft erhalten Sie eine E-Mail, bis die Regel gelöscht oder die Bedingung geändert wird.

13.8 Anlegen der Regel (Web-Endpunkt)

Öffnen Sie, wie bereits bei der E-Mail Regel vorgestellt, **Automation** / **Regeln** und klicken Sie auf **REGEL HINZUFÜGEN**.

Vergeben Sie einen aussagekräftigen Namen und nehmen Sie folgende Einstellungen vor:

- Aktion ausführen: … sofort
- Stream: Türkontaktsensor1
- Ergeinis: Messwert hinzugefügt
- Bedingung: `true` (die Regel wird bei jedem eintreffenden Messwert ausgeführt)
- Aktion: HTTP Anfrage senden
- HTTP Methode: POST
- URL: Ein Endpunkt zum Testen, z.B. via req0.de
- Markieren Sie die Option “Ganzes Ereignis als JSON senden”

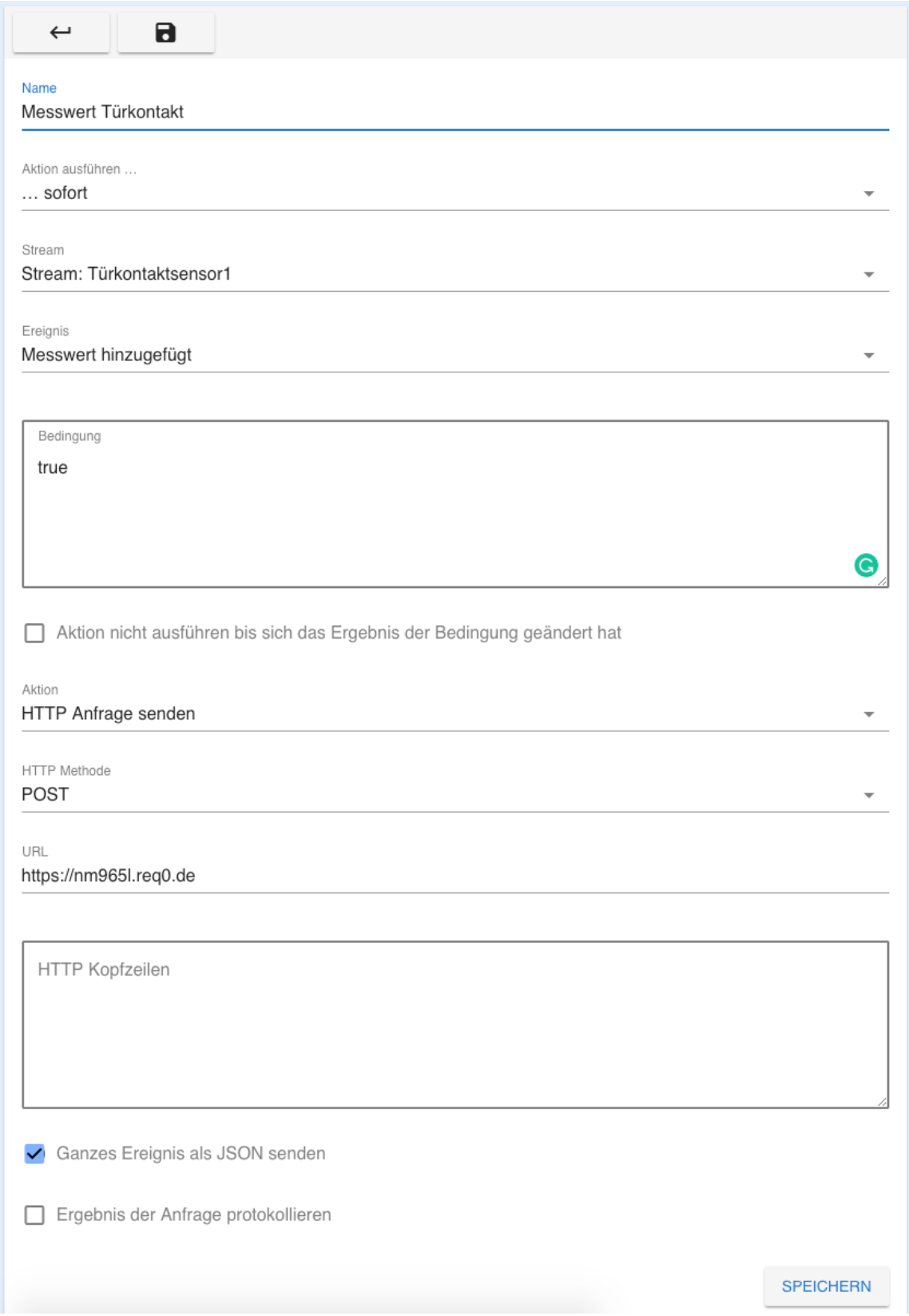


Abbildung 13.11: Regeln

Speichern Sie die Regel, anschließend werden die Messwerte an den Endpunkt, welchen Sie konfiguriert haben, gesendet.

Abbildung 13.12: Regeln

13.9 Fehlersuche bei Regeln und Streams

Jede Regel schreibt ein eigenes Protokoll. Öffnen Sie dafür die Übersicht der Regeln und klicken Sie auf das Symbol für das Protkoll. Die Meldungen werden in Echtzeit aktualisiert.

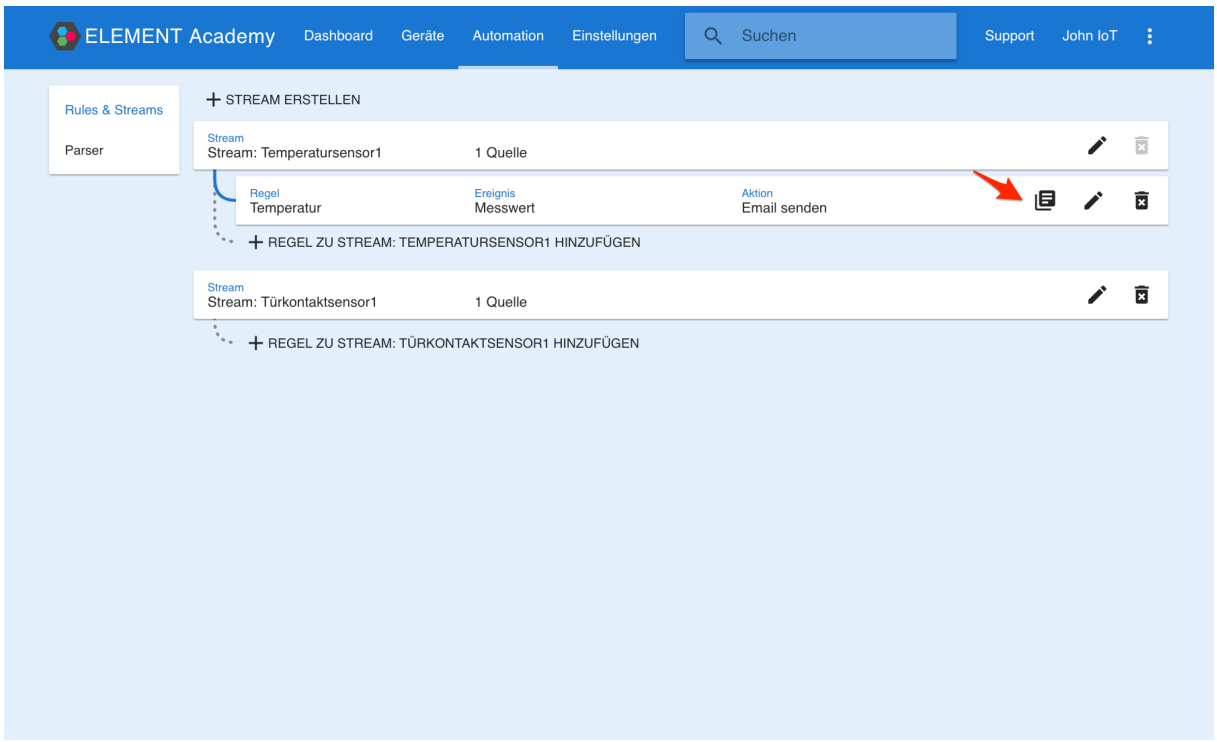


Abbildung 13.13: Regeln

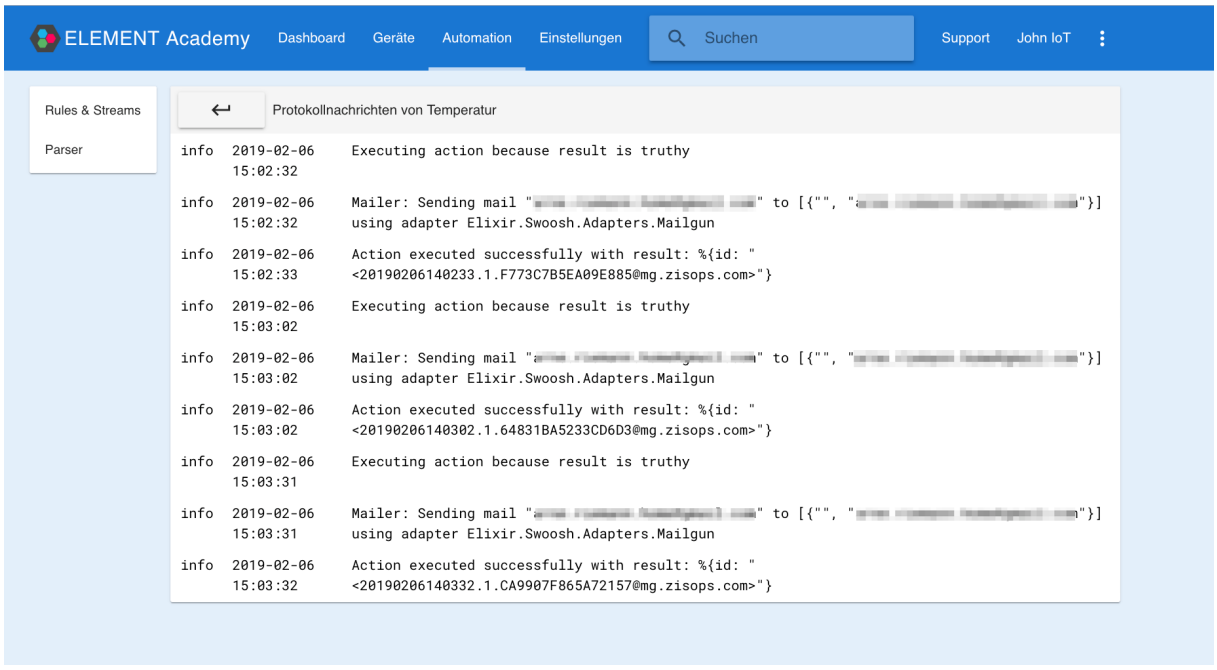


Abbildung 13.14: Regeln

13.10 Mögliche Platzhalter bei der Definition von Regeln

Folgende Platzhalter sind in den Feldern “Bedingung”, “URL”, “HTTP-Header” und “HTTP-Body” aktuell möglich.

Platzhalter für Pakete

- `id` Eindeutige Kennung des Pakets (Beispiel: “58399fe7-6947-43b7-8855-0147b0e04a0a”)
- `device_id` Eindeutige Kennung des Geräts (Beispiel: “abc18c70-b8ff-4b23-98b3-4c7ea03982aa”)
- `device.name` Name des Geräts zu dem Paket
- `interface_id` Eindeutige Kennung für das Interface (Treiber), über den das Paket empfangen wurde (Beispiel: “f393271d-7671-4dab-bdab-e9e462c41bd5”)
- `transceived_at` Zeitstempel, an dem das Paket empfangen wurde (Beispiel: “2018-02-05T15:14:24.472455Z”) Hinweis: Zeitzone ist UTC
- `payload_encoding` Art des Pakets, kann “binary” oder “json” sein
- `payload` Nutzdaten des Pakets, abhängig von payload_encoding
- `packet_type` Gibt die Art des Pakets an. Ist Abhängig vom verwendeten Treiber (Beispiele: “up”, “down”, join_req”, “join_accept”)

Platzhalter für Messwerte

- `id` Eindeutige Kennung eines Messwerts (Beispiel: “58399fe7-6947-43b7-8123-0147b0e04a0a”)
- `parser_id` Eindeutige Kennung des verwendeten Parsers (Beispiel: “61518c70-b8ff-4123-98b3-4c7ea03982aa”)
- `packet_id` Eindeutige Kennung des dazugehörigen Pakets (Beispiel: “8481c148-04c6-4123-bc57-a0fd41f88a15”)
- `device_id` Eindeutige Kennung des dazugehörigen Geräts (Beispiel: “078e38f5-67c0-4123-b4a2-472662b5be21”)
- `measured_at` Zeitstempel, an dem der Messwert erzeugt wurde (Beispiel: “2018-02-05T15:11:54.275972Z”) Hinweis: Zeitzone ist UTC
- `data.xxx` Einzelner Wert des Messwerts. Statt `xxx` ist eine durch den Parser erzeugter Schlüssel zu verwenden, wie beispielsweise “value”
- `device.name` Name des Geräts zu dem Paket
- `device.id` Eindeutige Kennung des Geräts (Beispiel: “078e38f5-67c0-44aa-1337-472662b5be21”)

13.11 Fortgeschritten: Regel zum Überwachen von Gateways

Sie haben mit Regeln auch die Möglichkeit zu überwachen, ob Gateways regelmäßig senden. Sie können sich zum Beispiel eine E-Mail senden lassen, wenn Gateways keine Pakete mehr senden.

Wichtig ist, dass sich die Gateways, die überwacht werden sollen, in einem Ordner befinden, in unserem Beispiel im Ordner “Gateways”.

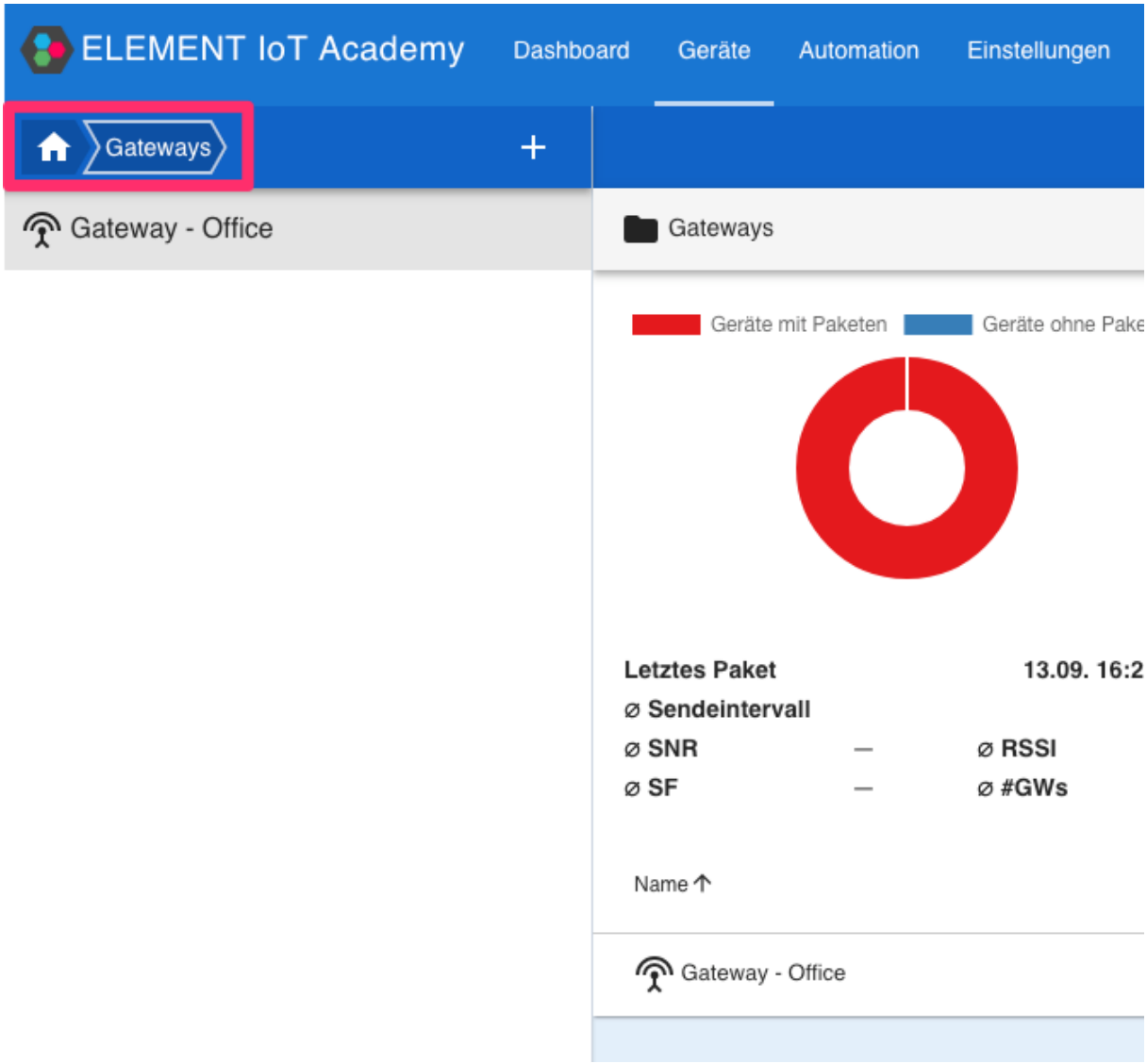


Abbildung 13.15: Regeln

Erzeugen Sie nun einen neuen Stream für alle Geräte im Ordner *Gateways*.

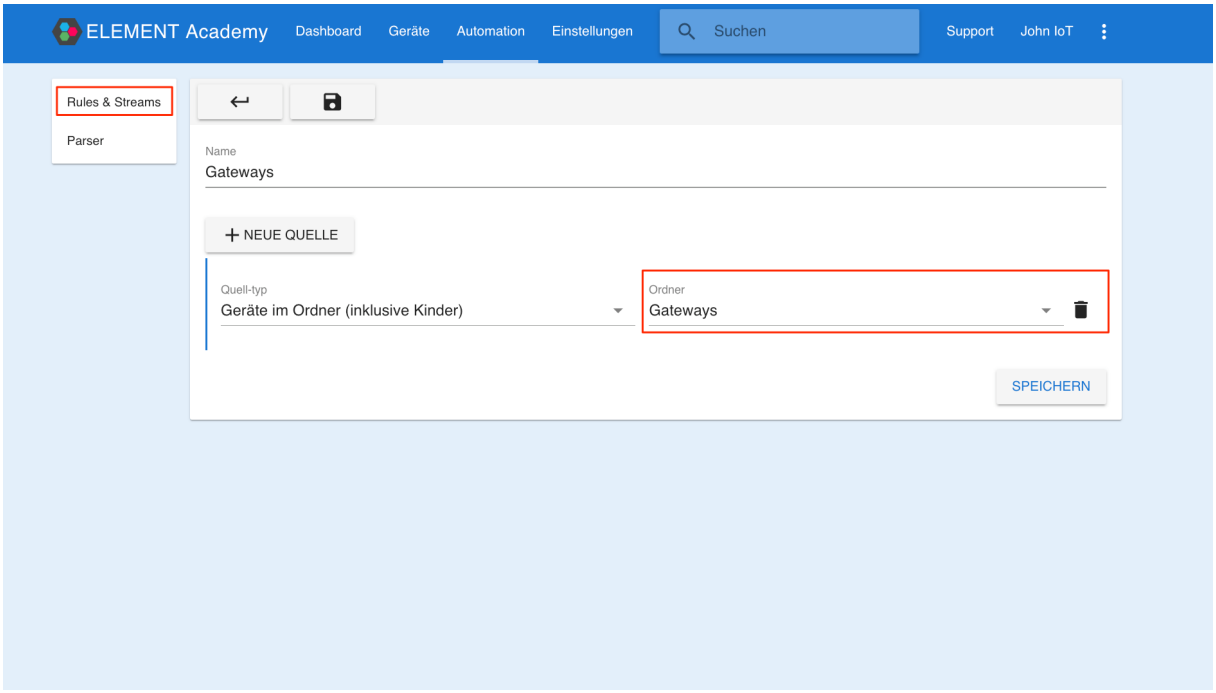


Abbildung 13.16: Regeln

Als nächstes folgt die entsprechende Regel zum Prüfen der Gateways.

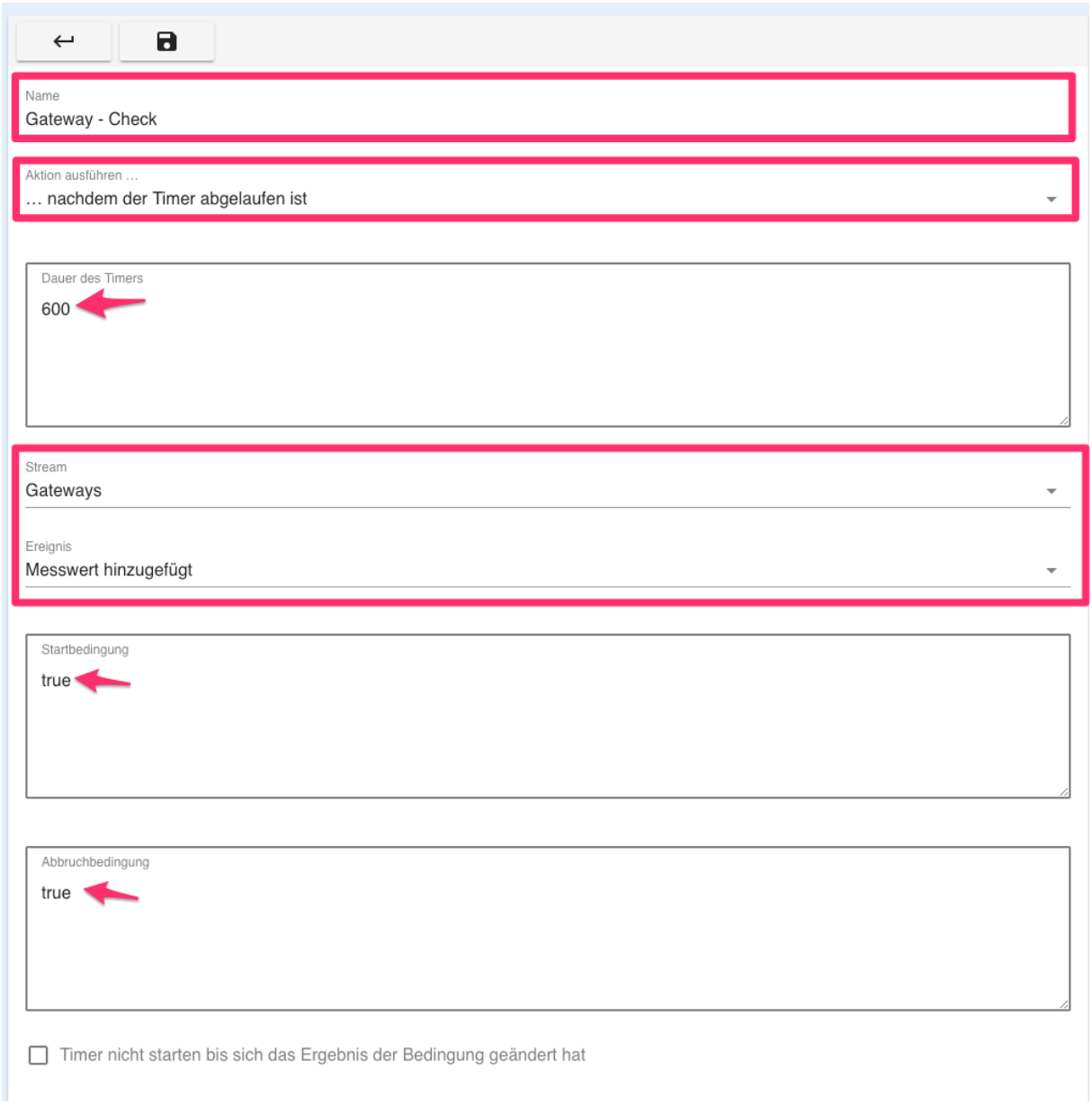


Abbildung 13.17: Regeln

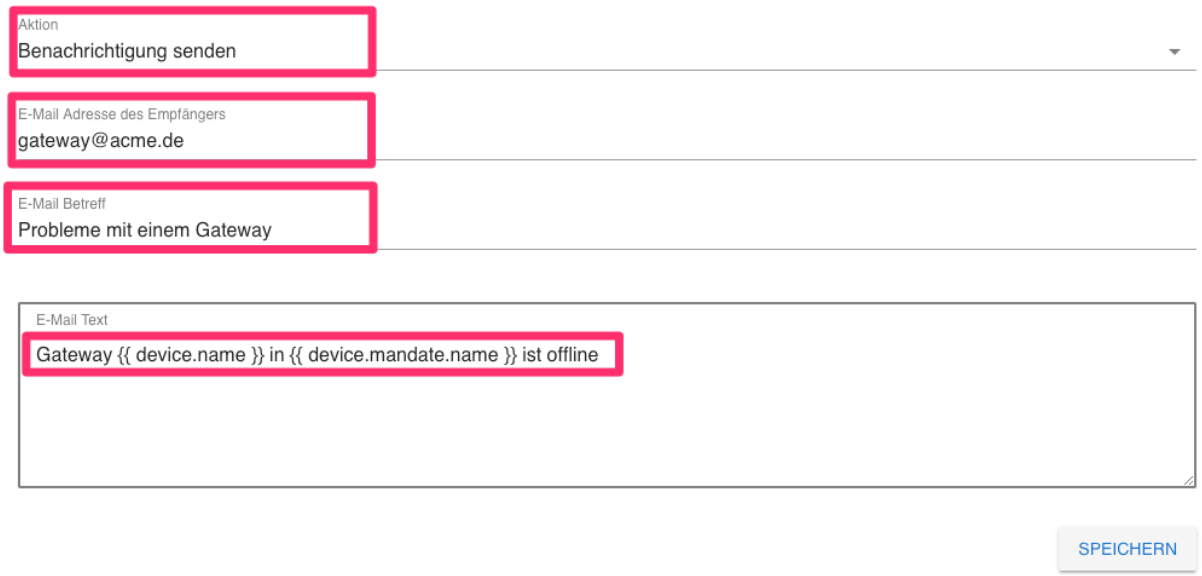


Abbildung 13.18: Regeln

Mit dieser Regel wird eine E-Mail versendet, wenn ein Gateway in dem Stream sich in 10 Minuten nicht gemeldet hat.

Bitte beachten Sie: Die Überwachung von Gateways ist nur möglich, wenn auf den Gateways das ZENNER IoT Gateway Management System installiert ist!

14 Visualisierung

14.1 Einführung

ELEMENT IoT kann nicht nur als Middleware eingesetzt werden, sondern viele Anwendungsfälle lassen sich direkt mit Hilfe der eingebauten Visualisierungsfunktionen umsetzen.

Für eine effektive Nutzung dieser Funktionen ist ein Verständnis der Datenstrukturen wichtig. Abhängig von der jeweiligen Funktion ist der Root-Element, von dem aus auf Felder und Assoziationen zugegriffen wird, unterschiedlich. In der Beschreibung der jeweiligen Funktion wird das Root-Element ausgewiesen.

Ist z.B. das Root-Element ein Gerät, steht u.a. das Feld `name` zur Verfügung.

14.2 Ordner

Ist ein Ordner ausgewählt, stehen für alle Messwerte von allen Geräten, die direkt diesem Ordner zugeordnet sind, die drei Visualisierungsfunktionen zur Verfügung:

1. Listen und Sichten
2. Graphenpresets
3. Kartenpresets

14.3 Einzelne Geräte

Ist eine Gerät ausgewält, gibt es die gleichen drei Visualisierungsfunktionen wie bei Ordnern:

1. Listen und Sichten
2. Graphenpresets
3. Kartenpresets

14.4 Listen und Sichten

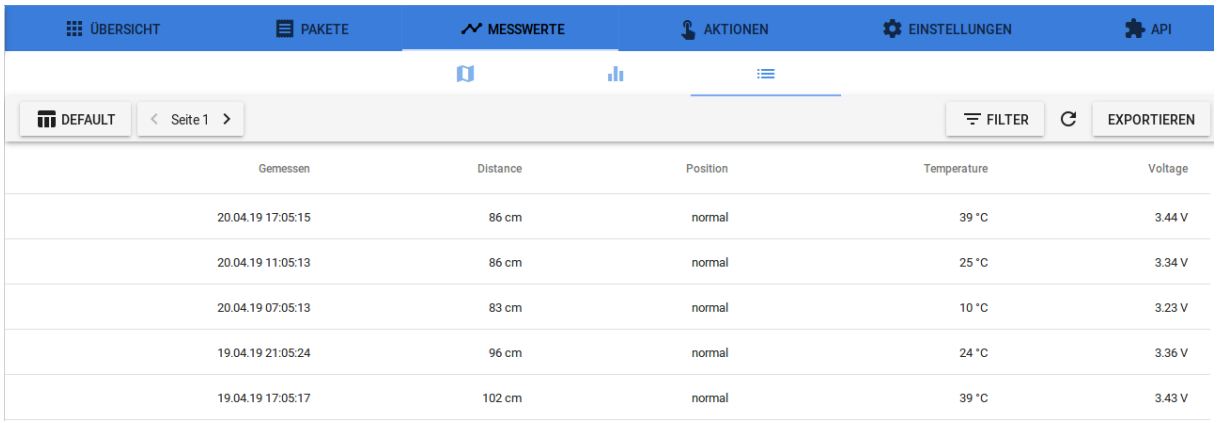


Abbildung 14.1: Messwertliste in der Standardansicht

Über den Filter kann diese Liste der Messwerte eingegrenzt werden. Das Root-Element ist dabei der Messwert (Reading). In der Regel wird also mit `data.<feldname>` auf den Messwert zugegriffen. Es sind dann die Vergleichsoperatoren von Abacus(#abacus) verwendbar.

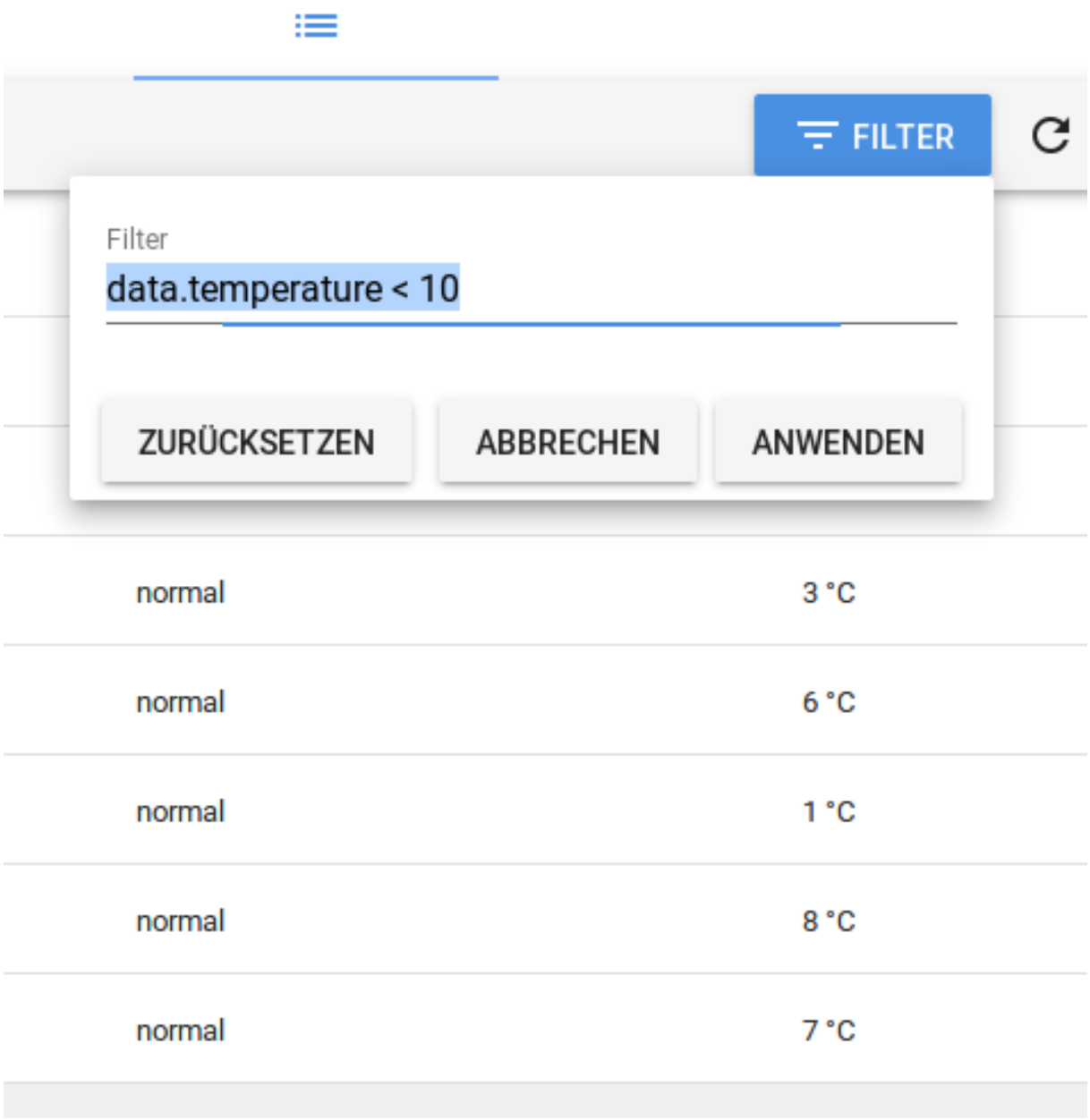


Abbildung 14.2: Beispiel für einen Filter

Über Sichten können eigene Listenansichten definiert werden, indem Spalten und eine Spaltenreihenfolge selbst definiert werden.

Es ist möglich:

- Die Spaltennamen zu definieren
- Für den Spalteninhalt auf das Root-Element Messwert (Reading) zuzugreifen

- Für den Spalteninhalt Abacus-Rechenausdrücke zu verwenden
- Im Fall von Datums- oder Zeitangeben selbst ein Format für die Anzeige festzulegen
- Einen Filterausdruck festzulegen
- Die Liste in der definierten Sicht zu exportieren

Zum Anlegen einer neuen Sicht muss der Sichtenauswahlknopf links oben gewält werden (“Default” oder “Standard”, wenn noch keine Sicht gewählt wurde).

Dort kann eine vorhandene Sicht ausgewählt werden oder durch Klick auf “Neues Preset” eine neue Sicht definert werden. Hier kann außerdem festgelegt werden, ob eine Sicht standardmäßig angezeigt werden soll, wenn die Messwertliste für diesen Ordner oder dieses Geräte angewählt wird.

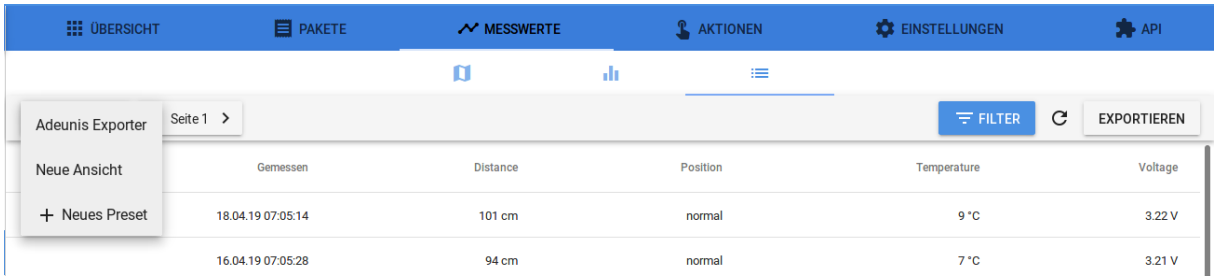


Abbildung 14.3: Anlegen von Sichten

Über *Spalte Hinzufügen* werden neue Spalten hinzugefügt.

Die Spalten-Beschriftung kann in dem entsprechenden Feld eingetragen werden.

Über das *Mülleimer*-Symbol kann eine Spalte wieder gelöscht werden.

Der Name der Ansicht kann in den Einstellungen festgelegt werden.

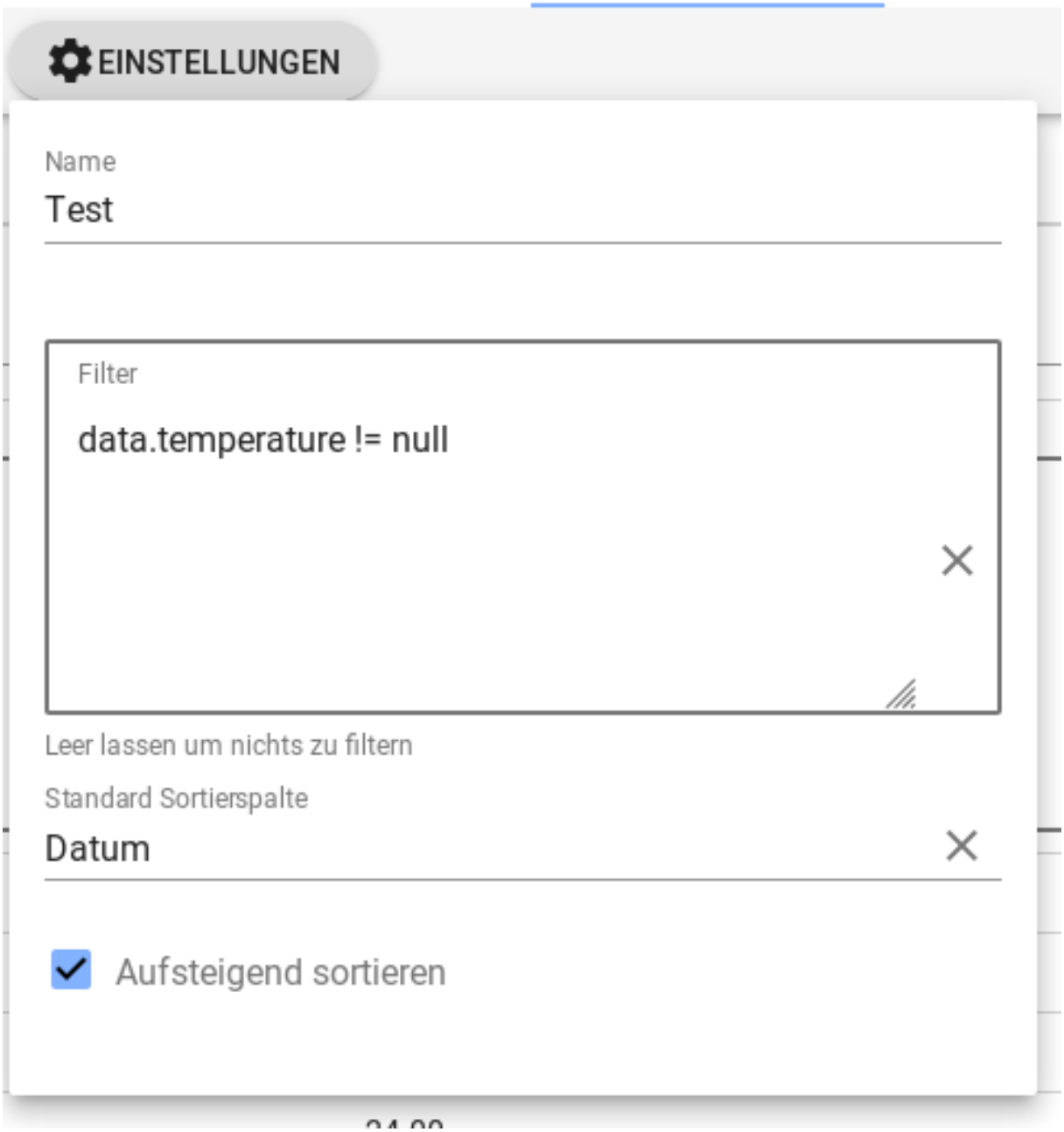


Abbildung 14.4: Sichteneinstellungen

Zunächst kann in ein Feld “Term” ein Abacus-Ausdruck einfügt werden, in dem u.a. auf ein Feld vom Root-Element *Messwert* aus zugegriffen werden kann.

Zum Beispiel kann durch den Ausdruck `float(data.temperature)* 10` in der Spalte die gemessene Temperatur mal 10 angezeigt werden, wenn die Messwerte auch ein Feld mit dem Namen `temperature` haben.

Felder in data müssen, wenn mit ihnen gerechnet werden soll, mittels *Cast*-Funktion in einen passenden Datentyp überführt werden, da diese in der Datenbank schemalos als JSON-gespeichert werden.

Die wichtigsten *Cast*-Funktionen sind:

Funktionsname	Bedeutung
`float`	Der Wert soll als Flieskommazahl behandelt werden
`integer`	Der Wert soll als Ganzzahl behandelt werden
`numeric`	Der Wert soll als Zahl behandelt werden, ob als Ganzzahl oder Fließkommazahl, hängt vom Vorhandensein von Nachkommastellen ab
`text`	Der Wert soll als Text behandelt werden
`boolean`	Der Wert soll als boolscher Wert (`true` oder `false`) behandelt werden

Alle anderen Felder, z.B. `measured_at`, haben schon den passenden Datentyp.

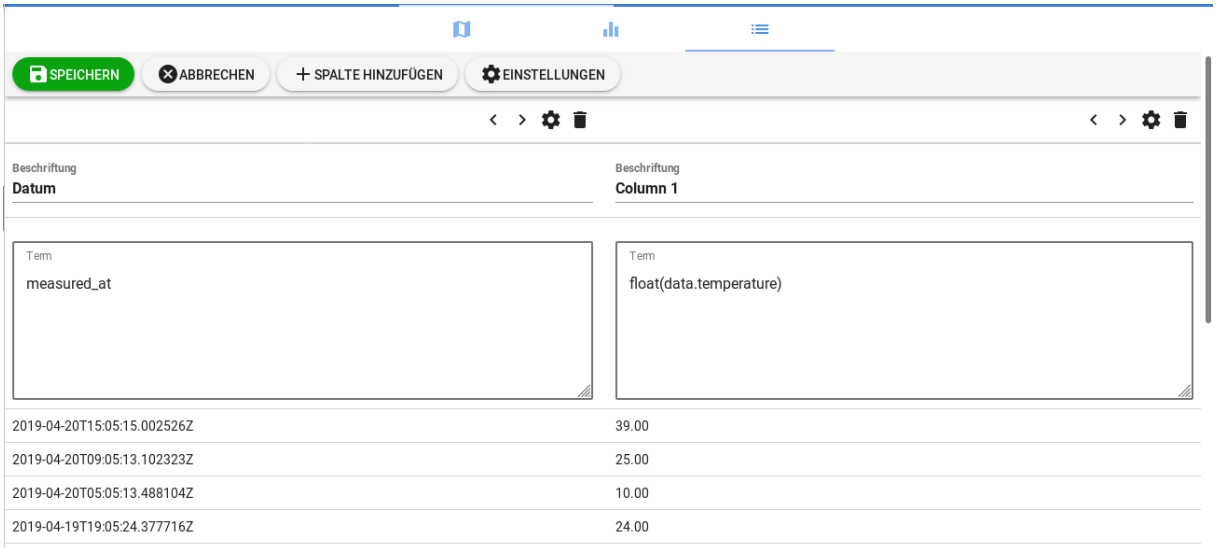


Abbildung 14.5: Abacus Terme zur Definition von Spalteninhalten

Soll nur der Messwert angezeigt werden, ohne damit zu rechnen, muss kein Cast erfolgen, allerdings muss über die Spalteneinstellungen (*Zahnrad*-Symbol) der passende Datentyp ausgewählt werden:

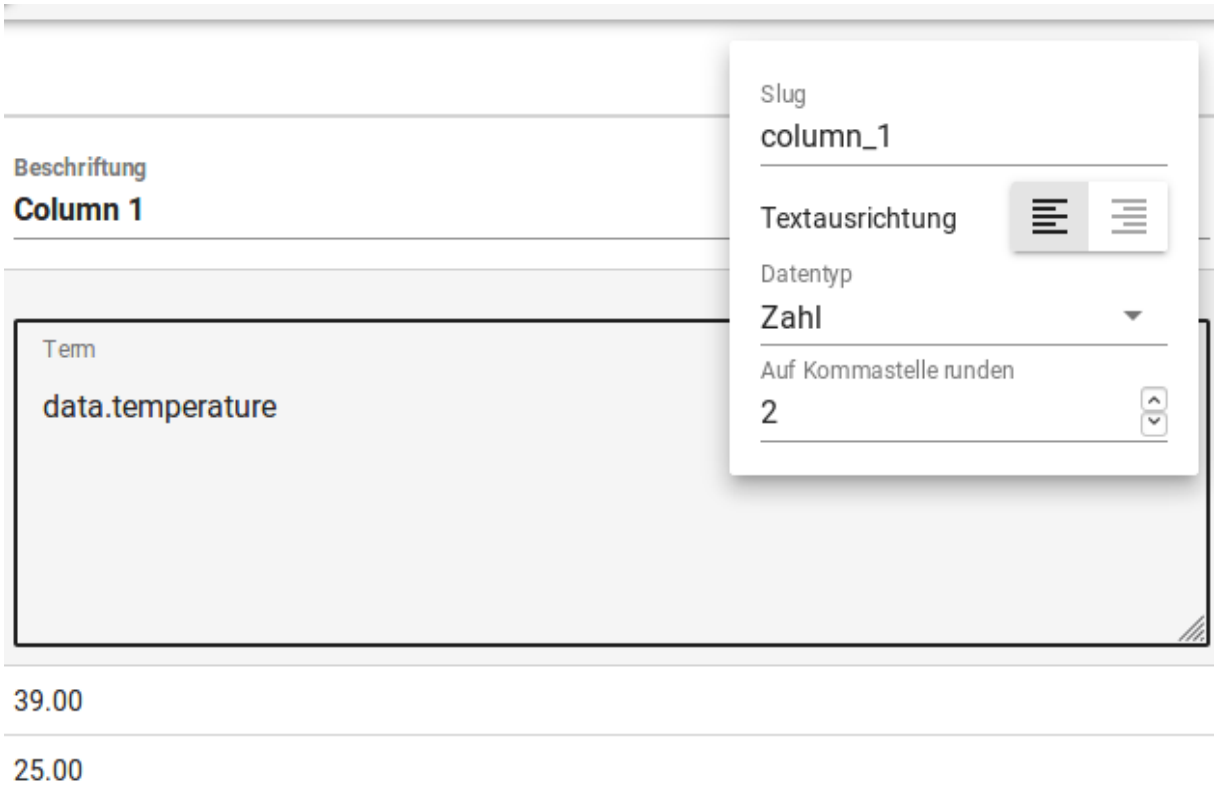


Abbildung 14.6: Spalteneinstellungen, Datentyp einstellen

Es empfiehlt sich, dazu auch einen passenden Filter einzurichten. Die geschieht unter Einstellungen, wenn die Listenansicht bearbeitet wird (*Zahnrad*-Symbol auswählen).

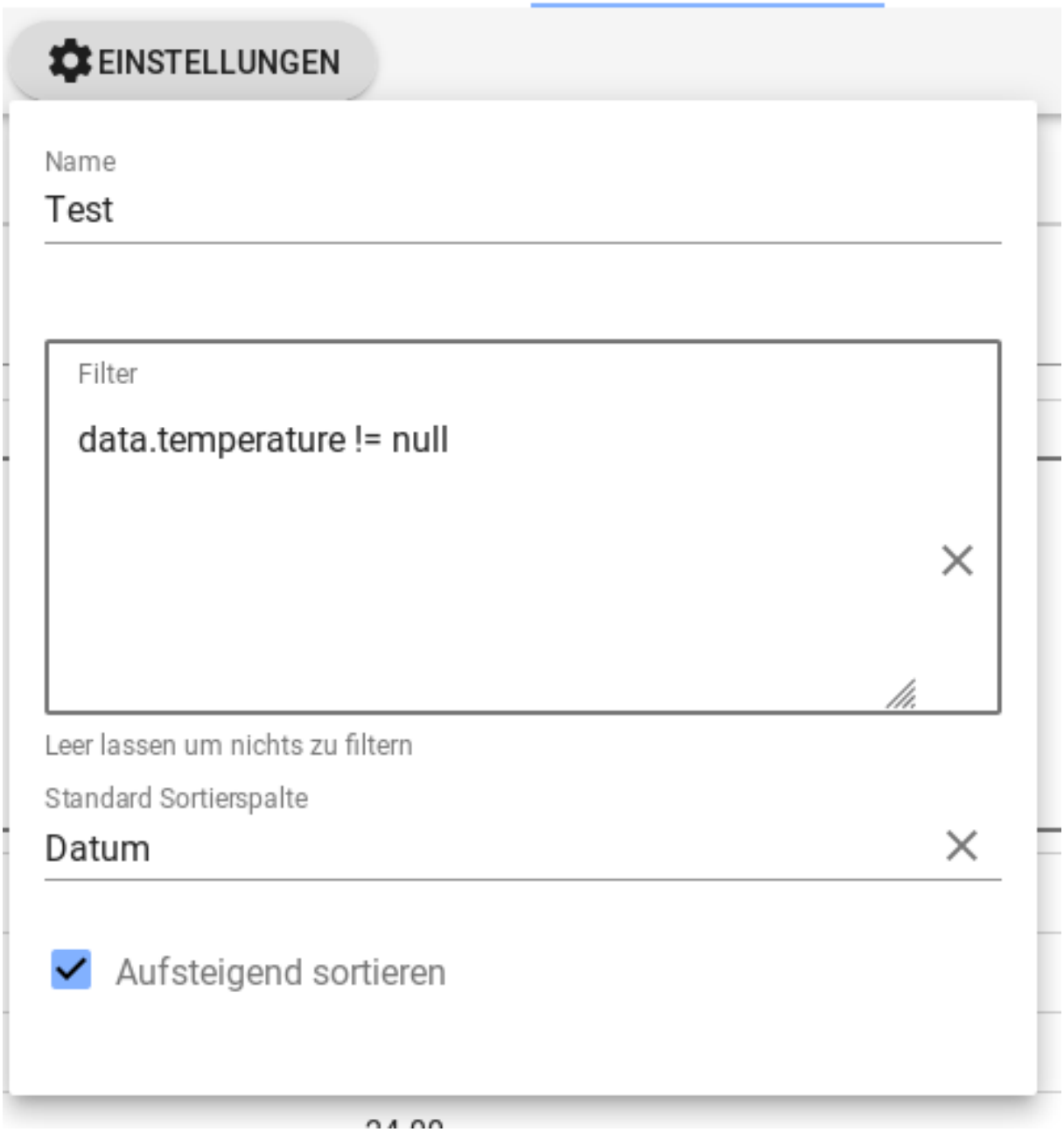


Abbildung 14.7: Filtern von Sichten auf Messwertlisten

In den Einstellungen der Ansicht kann auch eingestellt werden, nach welcher Spalte und in welcher Reihenfolge sortiert werden soll. Außerdem ist die Textausrichtung in der Spalte einstellbar.

Soll ein Datum oder eine Uhrzeit in ein anderes Format gebracht werden, kann über die Spalteneinstellung, nach Festlegung des passenden Datentyps, das Format in der *strftime*-Notation angegeben werden (siehe <hexdocs.pm/timex/Timex.Format.DateTime.Formatters.Strftime.html>).

Beispiele:

- `%d.%m.%Y` für eine deutsche Datumsformatdarstellung, z.B. 15.02.2019
- `%d.%m.%Y %T` für eine deutsche Datumsformatdarstellung mit Uhrzeit, z.B. 15.02.2019 15:04:00

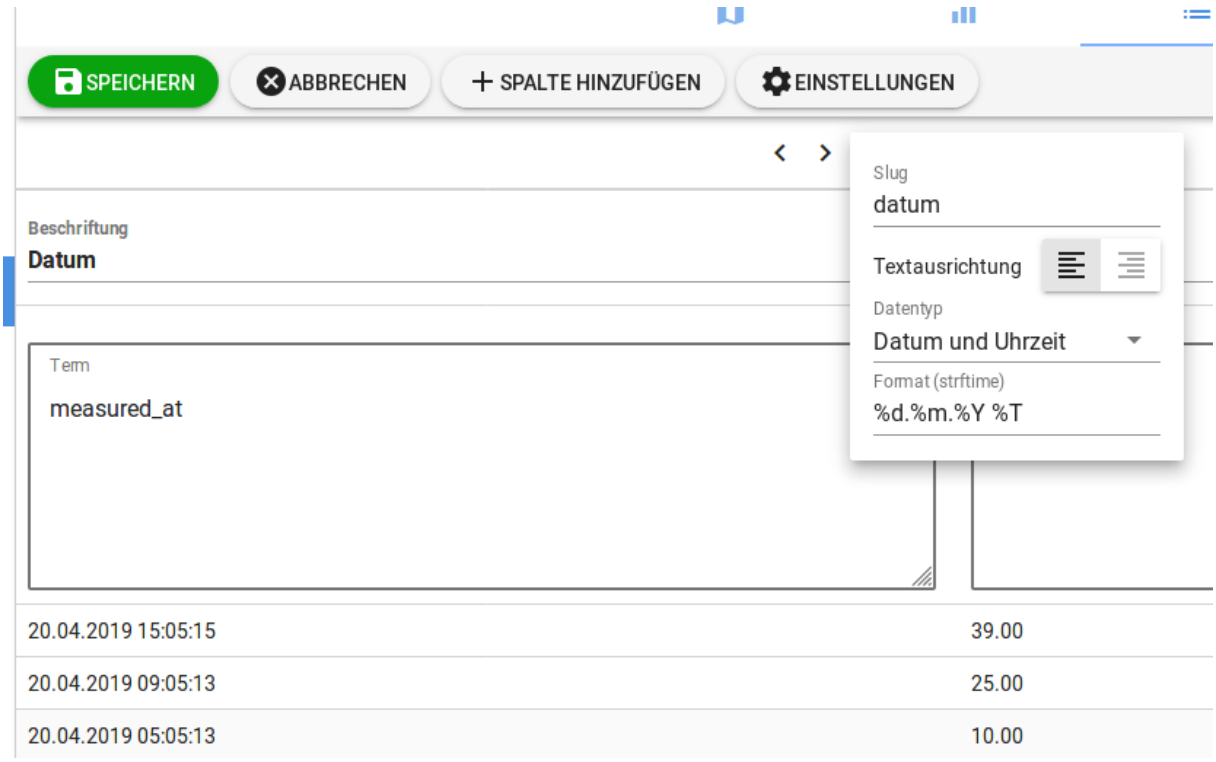


Abbildung 14.8: Datumsformat in Liste

14.5 Graphen und Graphenpresets

Messwerte können in Graphen dargestellt werden, wobei die X-Achse immer den Messzeitpunkt der Messwerte zeigt.

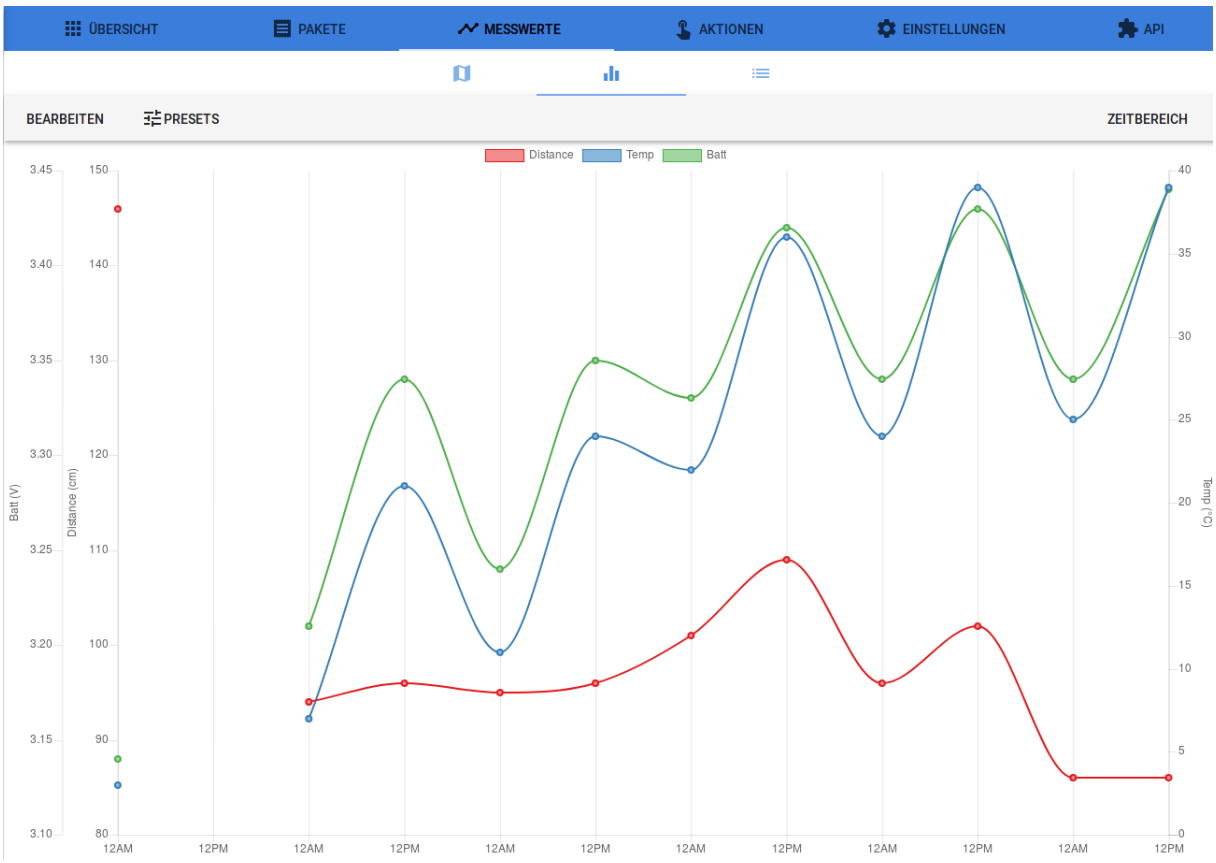


Abbildung 14.9: Graphdarstellung von Messwerten

Standardmäßig werden alle Felder im `data`-Feld der Messwerte und die Anzahl der Messwerte pro Zeiteinheit dargestellt.

Zuerst sollte immer der dargestellte Zeitraum und das Darstellungsintervall gewählt werden:

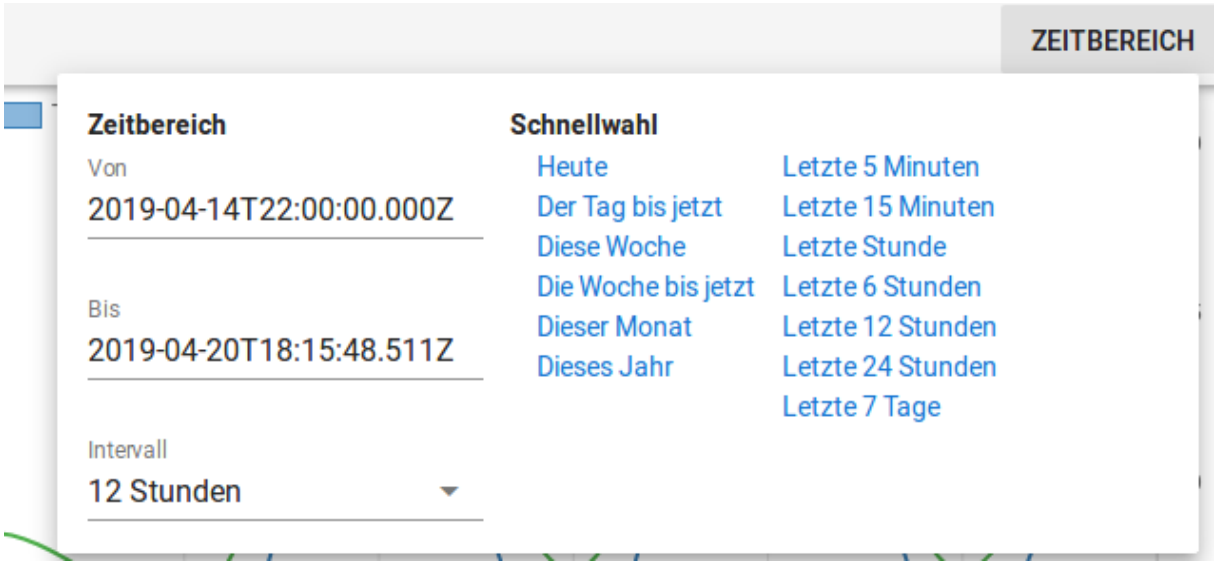


Abbildung 14.10: Graphen Darstellungszeitrum und Interval

Ohne Auswahl des Darstellungszeitraum wird automatisch ein Zeitraum bis zum letzen Messwert und beginnend 7 Tage davor ausgewählt.

Ein Intervall muss ausgewält werden, um eine effiziente Darstellung viele Messwerte über einen längeren Zeitraum zu gewährleisten. Pro Intervall wird immer nur ein Aggregat (zum Beispiel das Maximum oder der Durchschnitt) der Messwerte dargestellt.

Welche Daten auf der Y-Achse abgetragen werden, kann über die Einstellungen (*Bearbeiten*-Knopf) gewählt werden.

Dort lassen sich Datenreihen hinzufügen und entfernen. Es können die Diagrammtypen *Linie*, *Balken*, *Linie mit Fläche darunter* und *Punkt* ausgewählt werden.

Ist die Option Delta gewählt, wird die Änderung zum jeweils vorherigen Intervall dargestellt (z.B. für die Darstellgung von Strom- oder Wasserverbrauch).

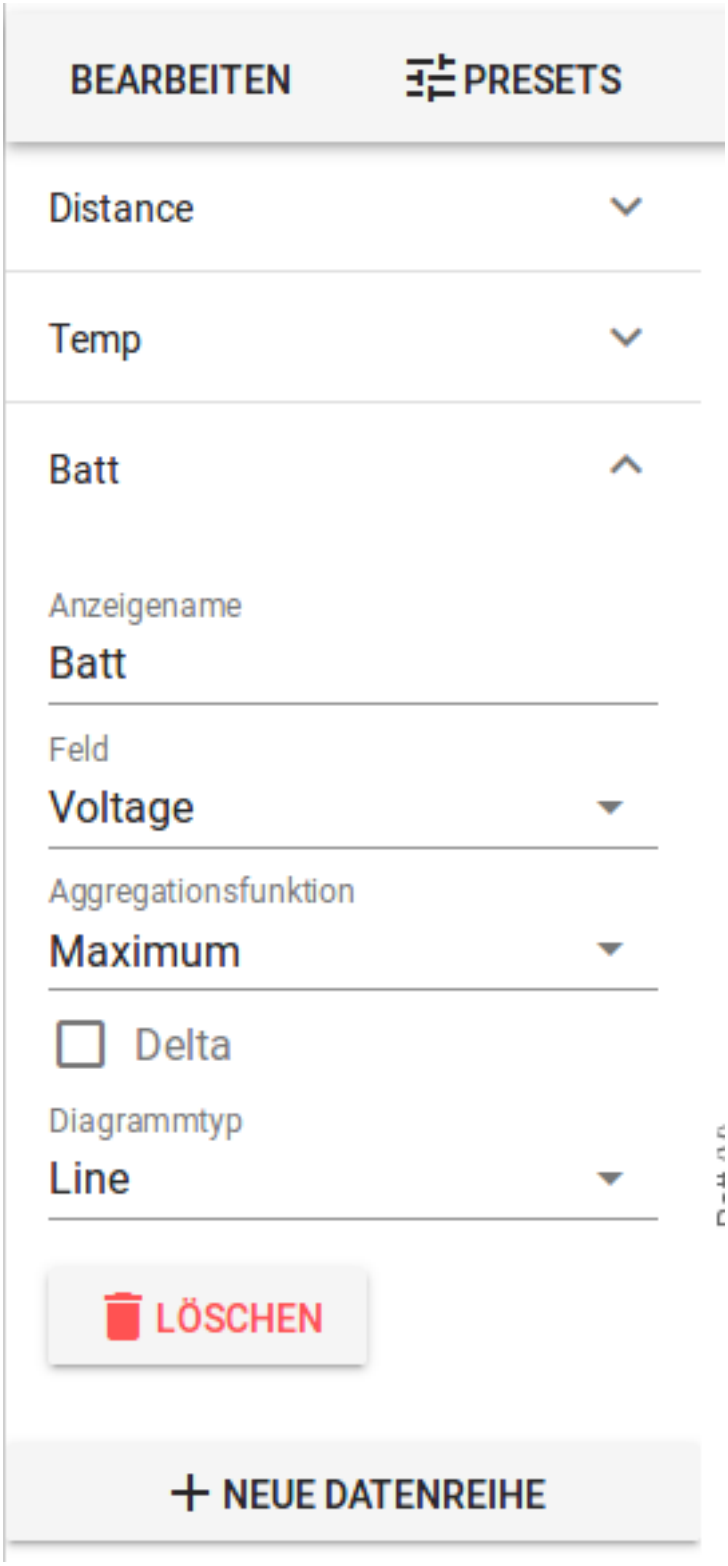


Abbildung 14.11: Graphen bearbeiten

Pro Datenreihe kann ein Feld aus dem Feldern des `data`-Felds der Messwerte ausgewählt werden (Wenn ein Parser geändert wurde und neue Felder unter `data` bereitstehen, dauert es 10 Minuten, bis diese in der Auswahlliste für Datenreihen auftauchen).

Über die Aggregatsfunktion wird festgelegt, wie mehrere Messwerte in einem Zeitintervall dargestellt werden sollen:

Aggregatsfunktion	Bedeutung
`Average`	Der Durchschnitt aller Messwerte in dem jeweiligen Intervall
`Minimum`	Der Kleinste aller Messwerte in dem jeweiligen Intervall
`Maximum`	Der Größte aller Messwerte in dem jeweiligen Intervall
`Sum`	Die Summe aller Messwerte in dem jeweiligen Intervall
`Count`	Die Anzahl aller Messwerte in dem jeweiligen Intervall
`n-th percentile`	Der Wert, für den n % der Messwerte darunter liegen. 50 % ist damit der Median.

14.6 Hinweise zur Darstellung diskreter Werte mit Graphen

Bei den dargestellten Messwerten handelt es sich um diskrete Werte; es wurde jeweils nur eine Probe eines ggf. kontinuierlichen Signals zu einem bestimmtem Zeitpunkt gemessen. Daher ist eine Darstellung als durchgehende Linie nicht korrekt - es gibt schlicht keine Informationen darüber, was in den Zeitraum zwischen zwei Messwerten passiert ist. Daher sind die folgenden Einschränkungen in ELEMENT zu beachten:

- Auch die Darstellungsform `Linie` zeigt immer die Messpunkte als Punkt an
- Die dargestellte Linie zeigt eine optische ansprechende Kurvenform zwischen zwei Messpunkten. Diese Kurve dient der Optik und ist nicht mehr oder weniger korrekt als eine gerade Linie zwischen zwei Messpunkten.
- Fehlt für ein Intervall ein Messwert (präziser: ist nicht wenigstens ein Messwert in dem Intervall), so wird die Linie auch nicht zum nächsten Messpunkt weitergezogen. Ggf. sollte das Darstellungsintervall größer gewählt werden.

14.7 Karten und Kartenpresets

Sofern Geräten eine Position zugeorndet wurde, entweder fix über die Geräteinstellungen oder als Metadatum zu einem Messwert, kann das Gerät auf einer Karte visualisiert werden.

Wir die Kartenansicht im Bereich Messwerte für ein einzelnes Gerät oder einen Ordner angewählt, wird in der Standardeinstellung das oder die Gerät(e) auf der Karte positioniert angezeigt. In dem Marker auf der Karte ist das eingestellte Gerätesymbol zu sehen:

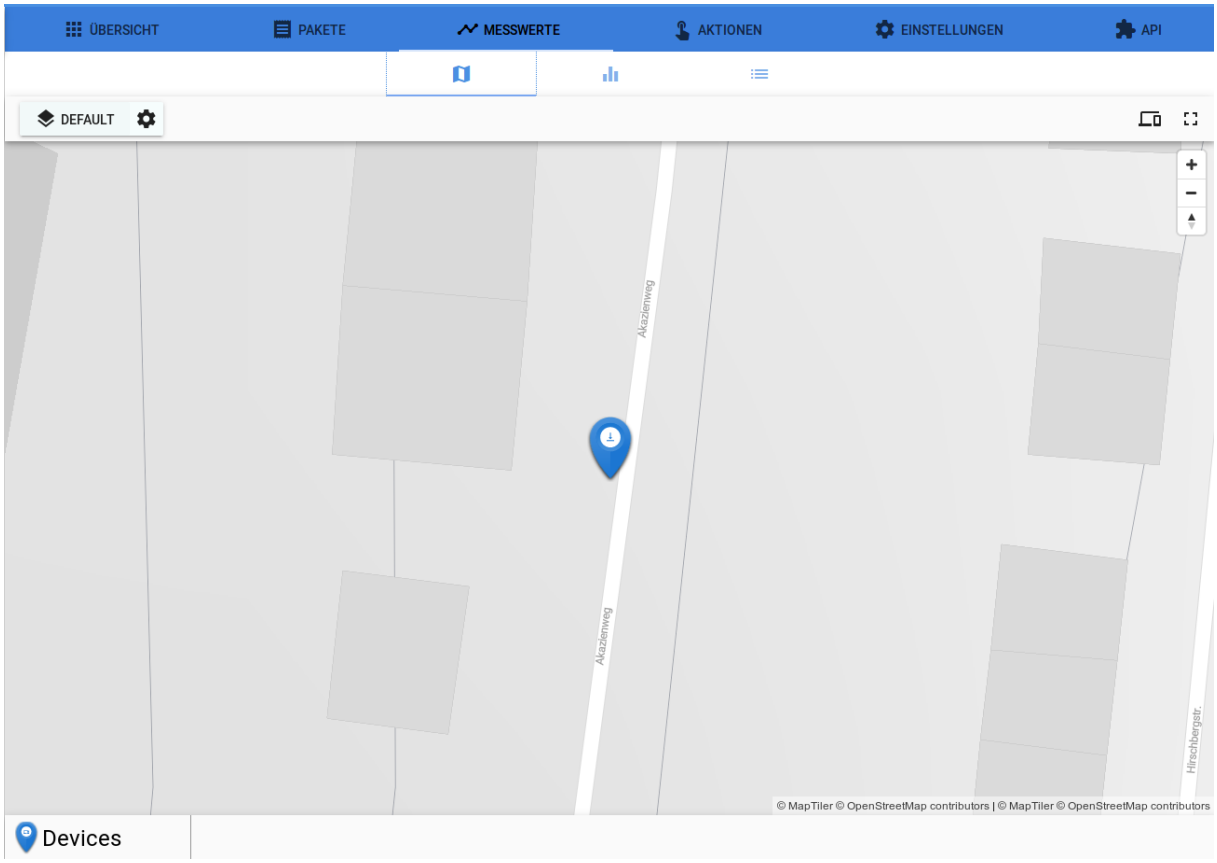


Abbildung 14.12: Standardansicht für Karten

Genau wie bei den Graphen können auch verschiendene Kartenansichten konfiguriert werden.

In den Einstellungsansichten können die folgenden Punkte konfiguriert werden:

- Name der Sicht (Preset)
- Stil der Karte (Dunkel, Hell, Satellit und Straßen)
- Initialer Kurs (Ausrichtung der Karte) und Neigung
- Initialer Kartenausschnitt (Mittelpunk und Zoomfaktor)

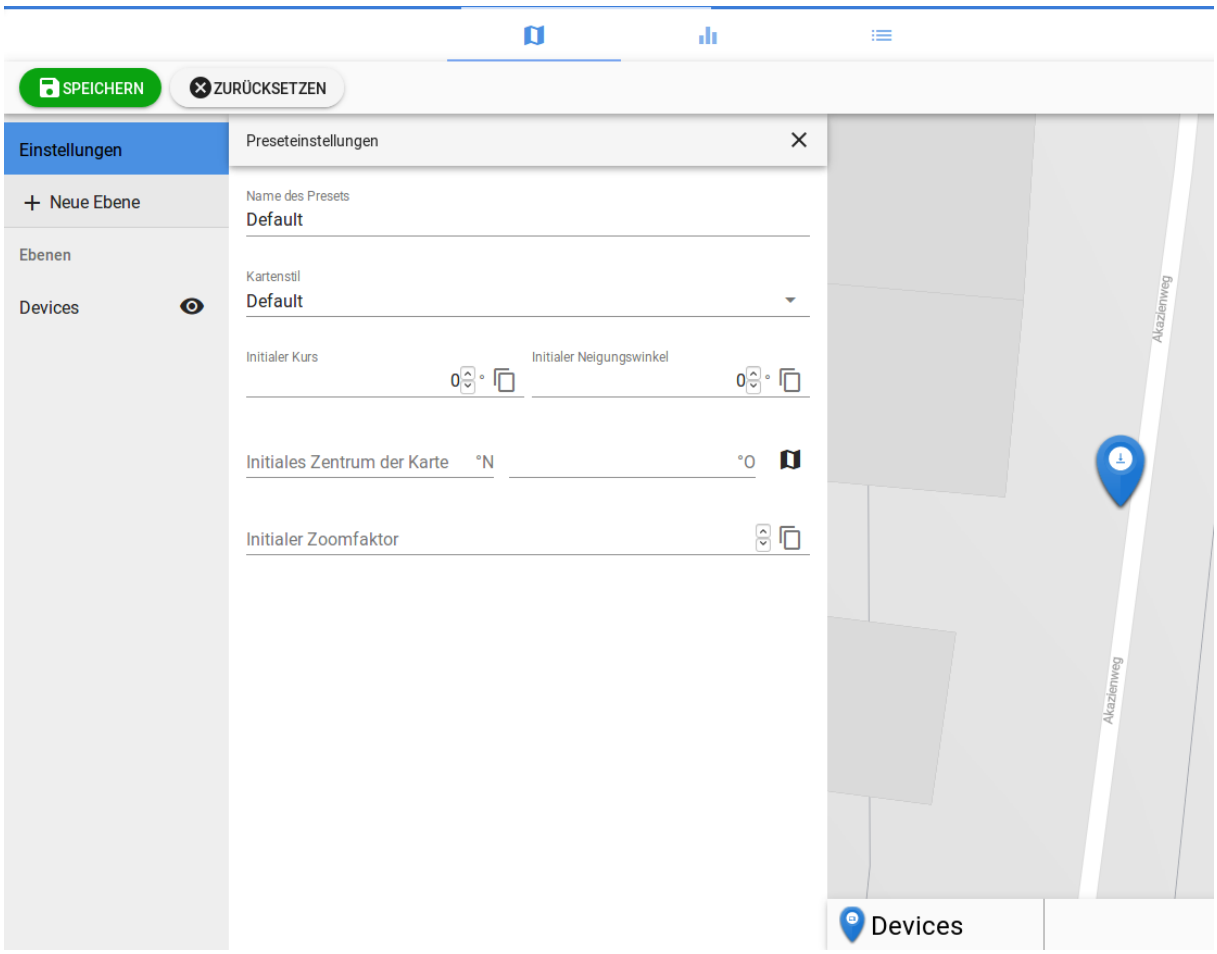


Abbildung 14.13: Karteneinstellungen

Eine Kartenansicht kann mehrere Ebenen haben. Ebenen können dann ein- und ausgeblendet werden.

Es gibt 6 verschiedene Ebenenarten:

Ebenenart	Beschreibung
`Messwerte`	Die Messwertebene stellt pro Gerät den jeweils letzten Messwert dar
`Devices`	Die Geräteebene stellt die Position des Gerätes dar
`Linien`	Die Linienebene stellt die letzten Messwerte eines Geräts an der jeweiligen Messposition dar, verbunden durch Linien (z.B. Weg eines GPS-Trackers)

Ebenenart	Beschreibung
`Heatmap`	Die Heatmapebene stellt nummerische Messwerte wie ein Wärmebild dar
`Gebäude`	Die Gebäudebene stellt Gebäude auf der Karte als 3D-Objekte dar
`Erweitert`	Die Ebenenart “Erweitert” kann für eigene Definitionen in Mapbox GL verwendet werden

Jeder Ebene kann ein Name gegeben werden, der dann unter der Karte als Knopf zum Ein- und Ausblenden der Ebende angezeigt wird.

Abbildung 14.14: Ebenenarten

Messwertebene

Die Messwerteebene dient der Visualisierung des letzten Messwerts eines Geräts, auf das der eingestellt Messwertfilter zutrifft. Standardmäßig ist der Messwertfilter `true`, was bedeutet, dass der letzte Messwert angezeigt wird.

In den Messwertfiltern können Abacus-Ausdrücke mit dem Root-Element `reading` verwendet werden (siehe [Datenstrukturen]).

Um zum Beispiel nur Messwerte mit einer Temperatur über 10 °C anzuzeigen, wäre ein Messwertfilter `data.temperatur > 10` verwendbar.

Für Geräte, welche nicht nur Messwerte mit den zu visualisierenden Daten senden (z.B. weil ein gelegentliches Statuspaket mit dem Batteriestand gesendet wird), sollte ein entsprechender Filter eingestellt werden.

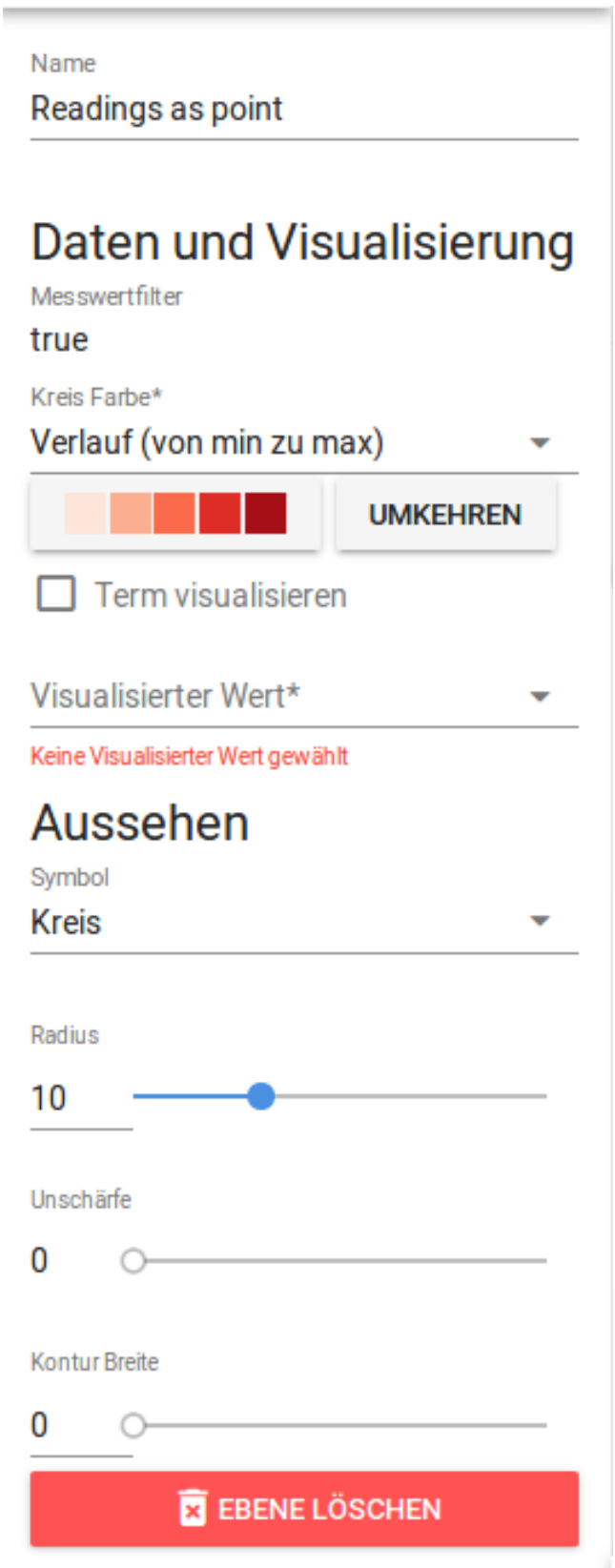


Abbildung 14.15: Messwertebeneneinstellungen

Messwerte werden durch eine Farbskala visualisert. Es stehen drei Möglichkeiten zur Verfügung:

Einstellung	Beschreibung
`Statische Farbe`	Eine feste Farbe, nur sinnvoll in Verbindung mit einem Messwertfilter
`Verlauf (von min zu max)`	Ein automatische berechneter Farbverlauf mit der ausgewählten Farbpalette. Es wird das Minimum und Maximium des dargestellten Werts ermittelt und die Differenz in gleich große Schritte eingeteilt. Die Anzahl der Schritte entspricht der Anzahl der Farben in der Palette.
`Verlauf (Schritte manuell einstellen)`	Ein Farbverlauf, bei dem die Schritte manuell eingestellt werden. Eignet sich z.B. gut um auf/zu oder belegt/unbelegt darzustellen.

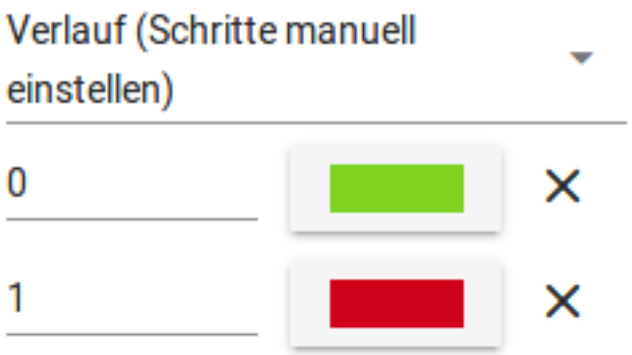


Abbildung 14.16: Belegtzustand mit manuellen Schritten darstellen

Als zu visualisierender Wert kann jedes Feld ausgewählt werden, das mindestens einmal als Feld in `reading.data` auftaucht (neue Felder sind erst nach 10 Minuten nutzbar).

Alternativ kann ein *Abacus-Term* visualisiert werden (siehe Spalteninhalt im Kapitel Listen und Sichten). Das Root-Element ist der Messwert (`reading`).

Abbildung 14.17: Term visualisieren

Sollen in einem Ordner die Messwerte zu mehreren Geräten visualisert werden, deren Parser unterschiedliche benannte Datenfelder erzeugen, kann die `coalesce`-Funktion genutzt werden. Z.B. `coalesce(float(data.temp), float(data.temperatur))`, dann wird das erste in `data` existierende Feld (`temp` oder `temperatur`) dargestellt.

Zur optischen Darstellung können 4 unterschiedliche Symbole (Marker) gewählt werden:

Symbol	Beispiel
Kreis	
Tropfen (wahlweise mit Gerätesymbol oder Messwert im Symbol(Marker))	
Nadel	
Füllstand (3D-Säule mit Füllhöhe entsprechend des Messwerts)	

Devices-Ebene

Mit der *Devices*-Ebene können Geräte auf der Karte an ihrer festgelegten Position mit einer fixen Farbe und einem von drei möglichen Symbolen angezeigt werden.

Sollen nur bestimmte Geräte aus einem Ordner angezeigt werden, können Gerätefilter mit Abacus-Ausdrücken und dem Root-Element `device` eingetragen werden.

Z.B. der Filter `parser.name == "Temperature"` kann verwendet werden, um nur Geräte mit einem Parser, der *Temperature* heißt, anzuzeigen.

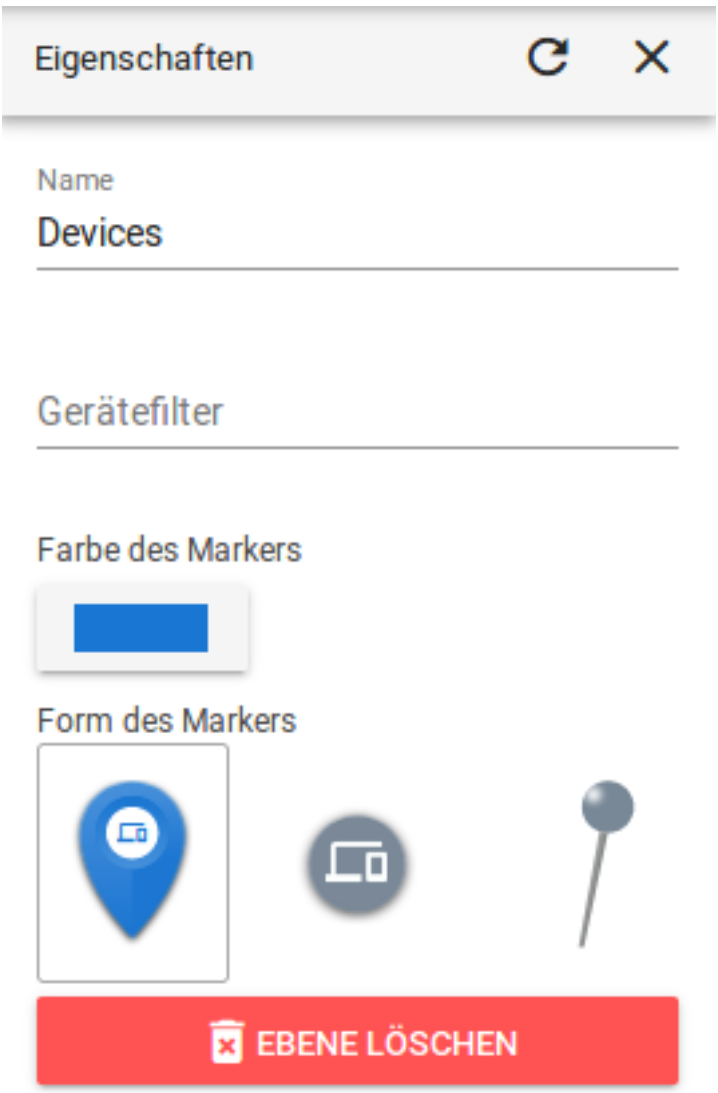


Abbildung 14.18: Geräteebene

Linienebene

Linienebenen verbinden Messwerte, die an ihrer Messposition angezeigt werden, in der zeitlichen Abfolge mit Linien einstellbarer Dicke.

Vorraussetzung ist, dass durch den Parser den Messwerten in den Metadaten eine Position zugerordnet wird.

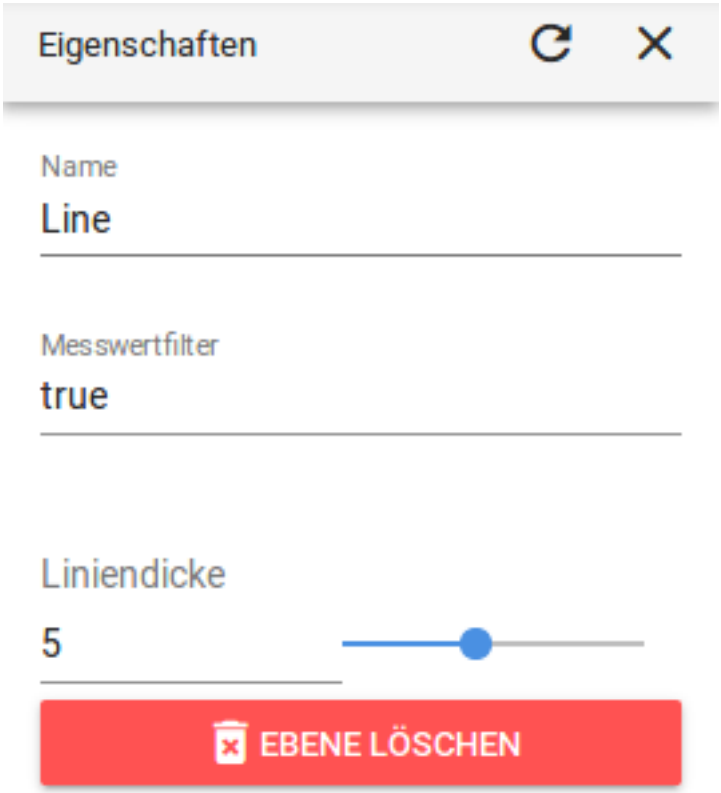


Abbildung 14.19: Linienebene

Heatmapebene

Heatmaps fassen in einem Kartenbereich, dessen Größe abhängig vom Zoomlevel ist, alle Messwerte zusammen. Dazu muss ein Aggregationsfunktion gewählt werden.

Die Berechnung der Karte mit Heatmap kann sehr lange dauern, da alle Messwerte, die dem Filter entsprechen, einbezogen werden. Es empfiehlt sich daher dringend, einen Filter einzutragen.

Z.B. mit dem Filter `measured_at > now()- interval('7 days')` lassen sich die Messwerte der jeweils letzten 7 Tage filtern. (Intervalausdrücke können die Einheiten `millennium`, `century`, `decade`, `year`, `month`, `week`, `day`, `hour`, `minute`, `second`, `millisecond`, `microsecond`, und ihre Pluralformen (`months`, `days`, etc.) nutzen.)

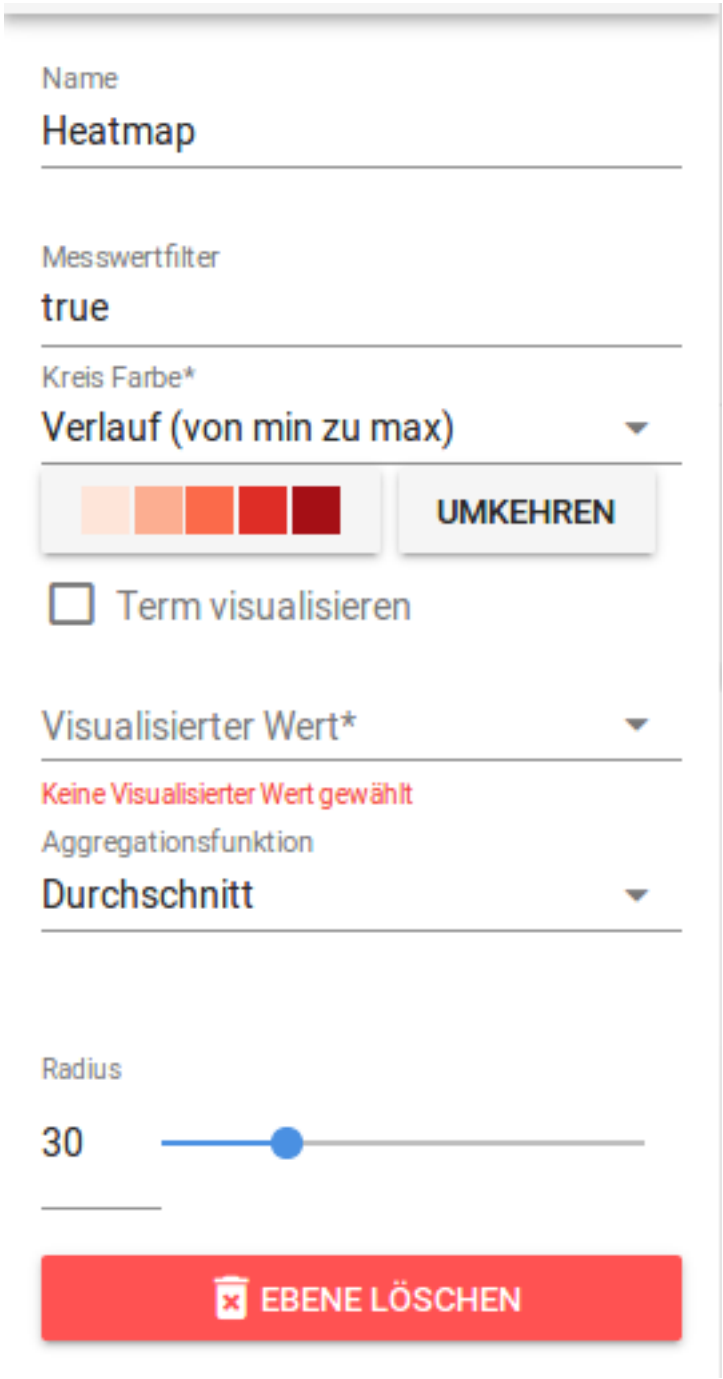


Abbildung 14.20: Heatmapebene

Gebäudeebene

Mit der Gebäudeebene lassen sich die Gebäude als 3D-Objekte in der gewählten Farbe darstellen.

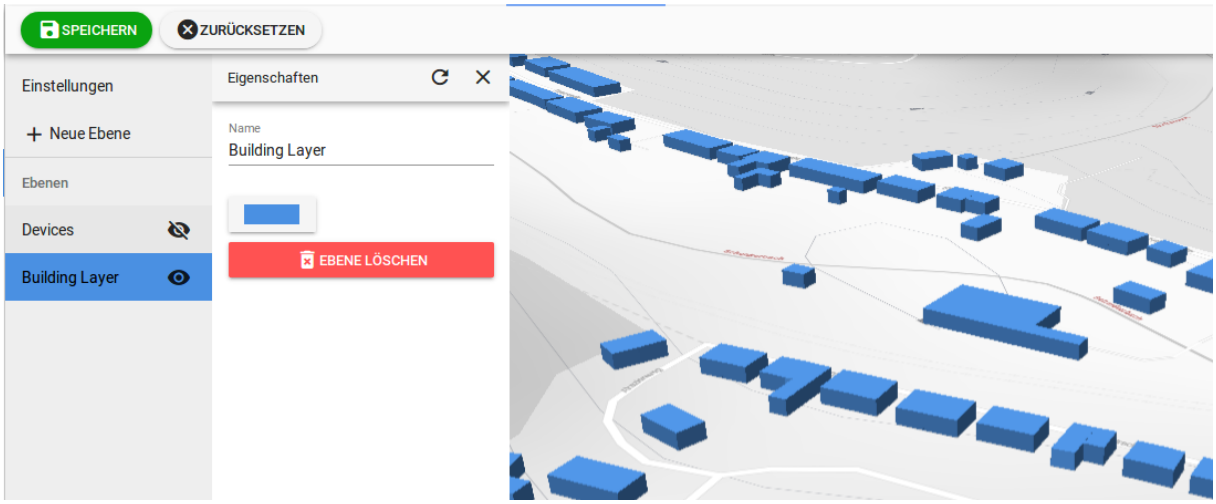


Abbildung 14.21: Gebäudeebene

15 Auditfunktionen

15.1 Einleitung Auditfunktionen

Die ELEMENT IoT-Plattform bietet bei vielen Aktionen die Möglichkeit eines Protokolls, über welches Sie Änderungen nachvollziehen und auch rückgängig machen können. Die einzelnen Funktionen, welche über dieses Feature verfügen, lernen Sie in diesem How To kennen.

15.2 Ordner

Bei Änderungen an Ordnern können Sie diese wie folgt nachvollziehen. Klicken Sie auf den entsprechenden Ordner und anschließend auf das  Symbol.

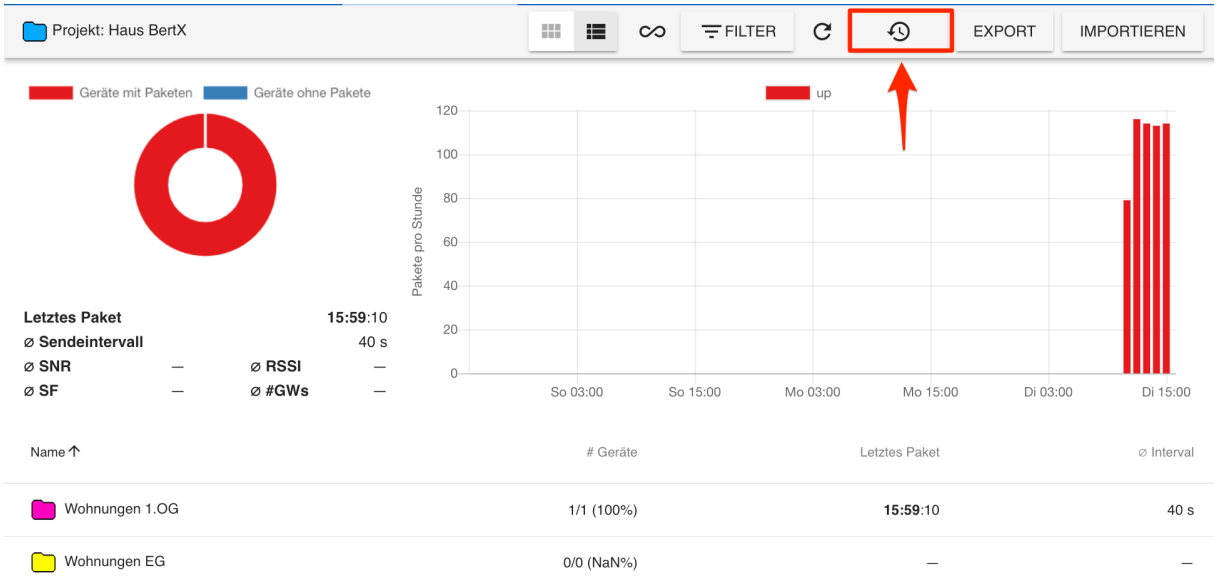


Abbildung 15.1: Screenshot

In der nachfolgenden Ansicht sehen Sie die durchgeführten Änderungen, in diesem Beispiel wurde an

den Namen des Ordners ein “X” angehängt. Wenn Sie jetzt auf den Button **RÜCKGÄNGIG MACHEN** klicken, wird der Name entsprechend wieder auf den Ausgangswert geändert.

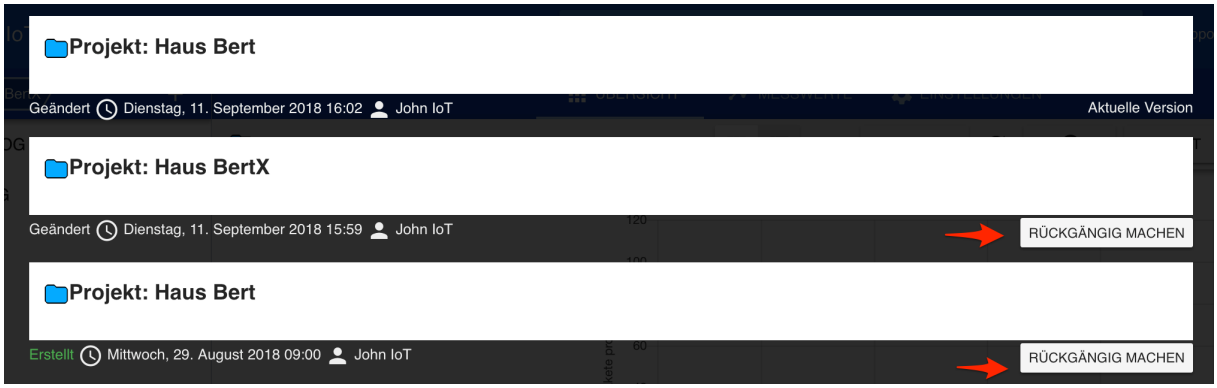


Abbildung 15.2: Screenshot

15.3 Geräte

Einstellungen an Geräten können Sie auf der Übersichtsseite des Gerätes rückgängig machen. In diesem Beispiel wurde der Standort des Gerätes geändert und kann entsprechend wieder auf den Ausgangswert zurückgesetzt werden.

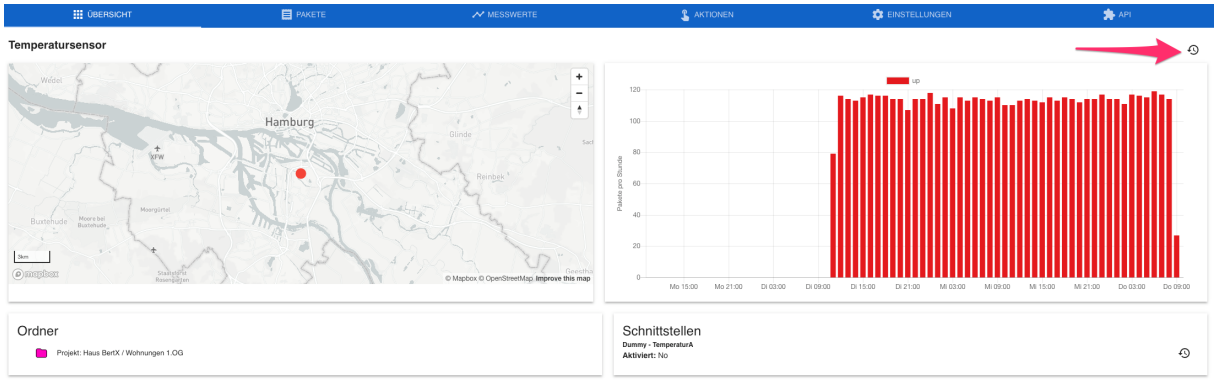


Abbildung 15.3: Screenshot

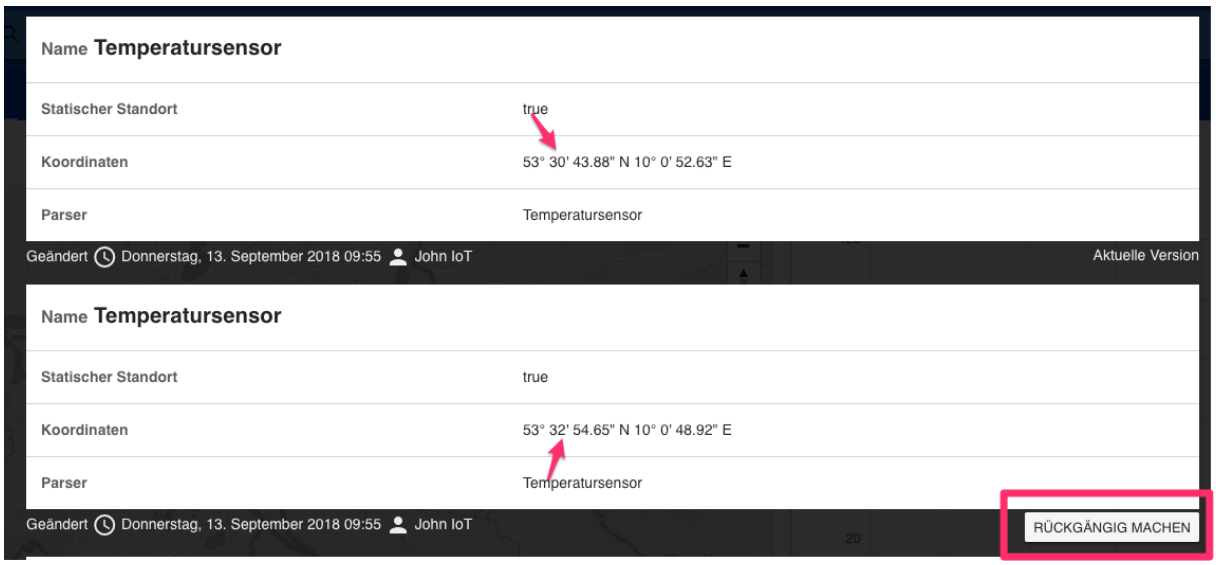


Abbildung 15.4: Screenshot

15.4 Schnittstellen eines Gerätes

Einstellungen, die für die Schnittstelle (Treiber) eines Gerätes geändert werden, können innerhalb der Übersichtsseite eines Gerätes zurückgesetzt werden. Öffnen Sie hierfür in der Navigation den Bereich **Geräte** und wählen Sie das entsprechende Gerät aus.

Um Änderungen an der Schnittstelle rückgängig zu machen, klicken Sie auf das  Symbol und wählen Sie den gewünschten Stand aus. In diesem Beispiel wurde die Schnittstelle deaktiviert und kann nun über das Rückgängigmachen wieder aktiviert werden.

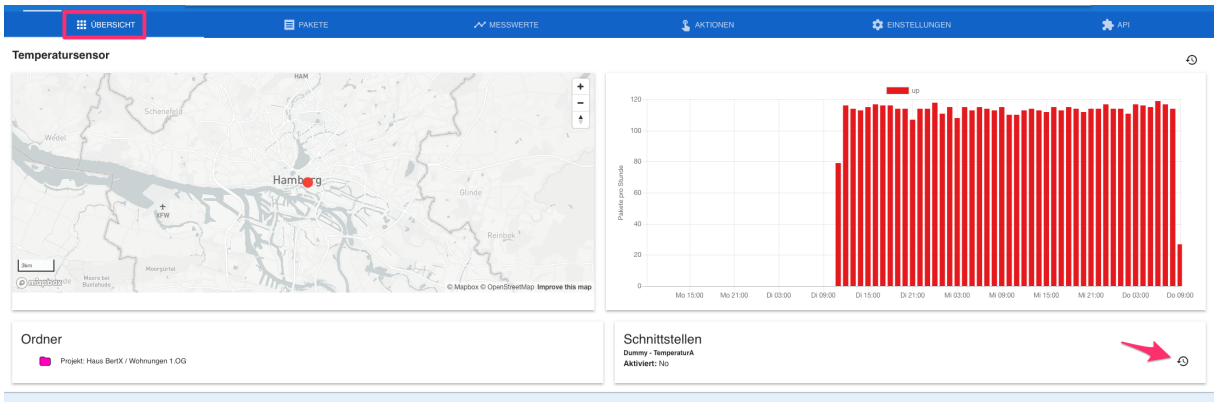


Abbildung 15.5: Screenshot

Abbildung 15.6: Screenshot

15.5 Benutzer

Änderungen, die für Benutzeraccounts durchgeführt werden, können Sie im Bereich **Einstellungen** - **Benutzer** rückgängig machen. Klicken Sie hierfür auf das  Symbol neben dem Benutzer.

In diesem Beispiel wird die Änderung der E-Mail-Adresse widerrufen.

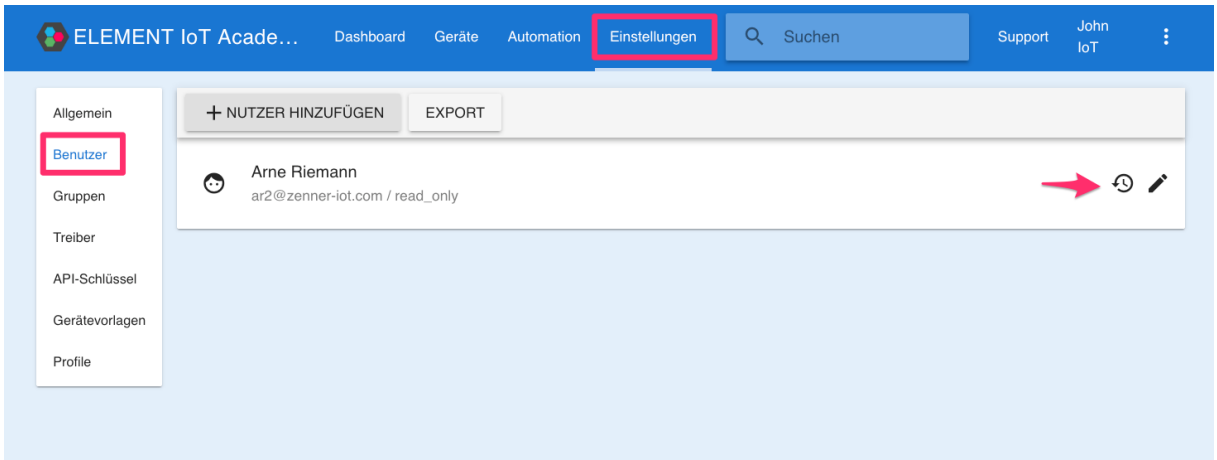


Abbildung 15.7: Screenshot

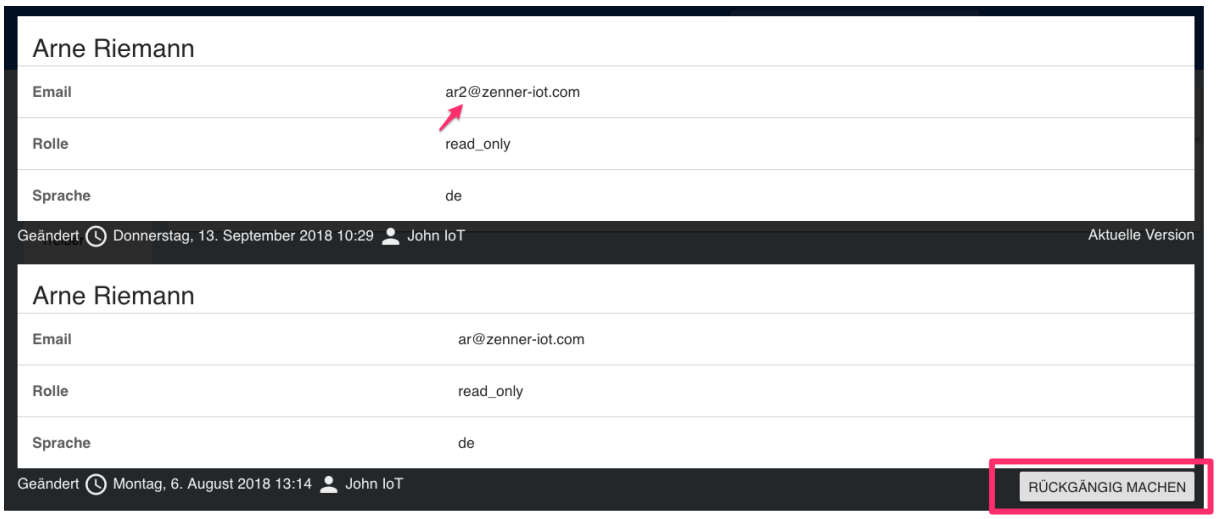


Abbildung 15.8: Screenshot

16 Datenexport

16.1 Exportieren von Paketen

Sie können Pakete direkt aus der ELEMENT IoT-Plattform in eine CSV-Datei exportieren und diese anschießend weiterverarbeiten. In diesem How To werden wir die letzten 50 Pakete eines Gerätes exportieren und diese anschließend in Microsoft Excel weiterverarbeiten.

16.2 Export der Pakete

Melden Sie sich an der ELEMENT IoT-Plattform an und öffnen Sie den Bereich **GERÄTE**.

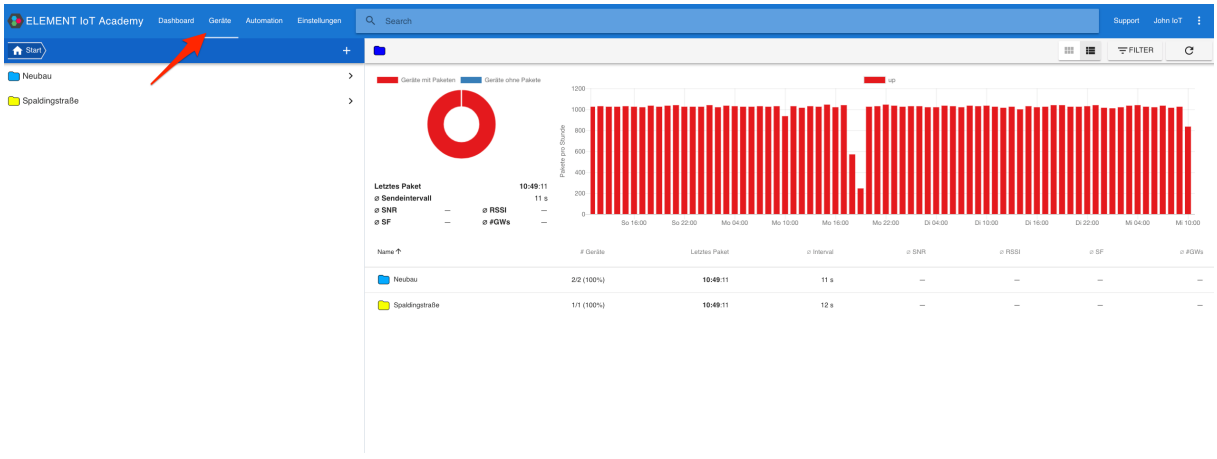


Abbildung 16.1: Öffnen Bereich Geräte

Wählen Sie das gewünschte Gerät aus und öffnen Sie die Ansicht **PAKETE**.

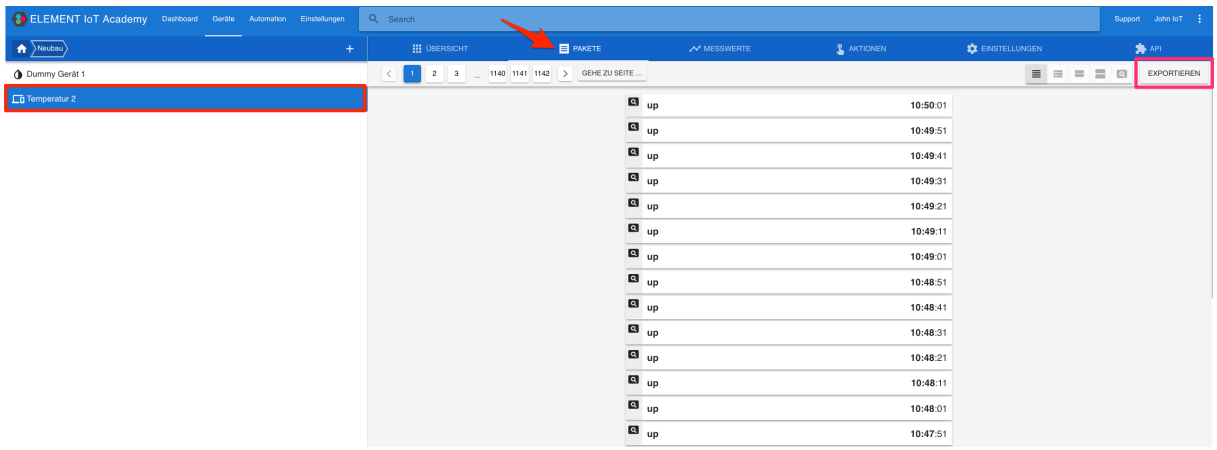


Abbildung 16.2: Ansicht Pakete

Über den Button **EXPORTIEREN** haben Sie jetzt die Möglichkeit, entweder eine beliebige Anzahl von Paketen für den Export auszuwählen, oder alternativ einen Zeitbereich zu definieren. Wählen Sie die letzten 50 Pakete aus und klicken Sie auf den Button **EXPORT**.

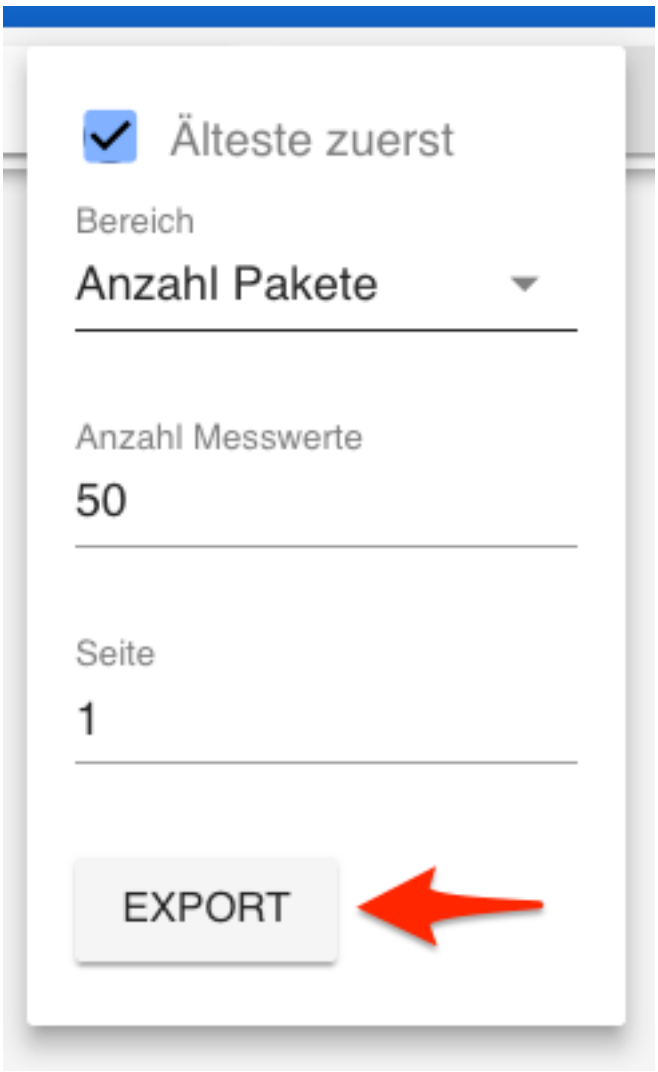


Abbildung 16.3: Ansicht Pakete

Ihr Browser bietet Ihnen nun eine CSV-Datei zum Download an. Speichern Sie diese an einem beliebigen Ort auf Ihrem Computer.

16.3 Exportieren von Messwerten

Sie können Messwerte direkt aus der ELEMENT IoT-Plattform in eine CSV-Datei exportieren und diese anschießend weiterverarbeiten. In diesem How To werden wir die letzten 50 Messwerte eines Gerätes exportieren und diese anschließend in Microsoft Excel weiterverarbeiten. Bitte beachten Sie, dass in dem Export nur Daten enthalten sind, welche durch einen entsprechenden Parser definiert wurden.

16.4 Export der Messwerte

Melden Sie sich an der ELEMENT IoT-Plattform an und öffnen Sie den Bereich **GERÄTE**. Hier können Sie das gewünschte Gerät auswählen und gelangen über die Schaltfläche **MESSWERTE** zu der Exportfunktion.

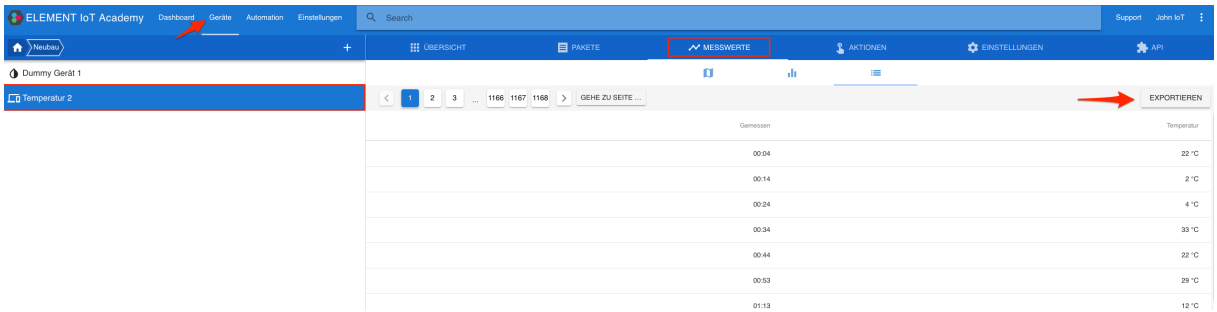


Abbildung 16.4: Öffnen Bereich Geräte

Über den Button **EXPORTIEREN** haben Sie jetzt die Möglichkeit, entweder eine beliebige Anzahl von Messwerten für den Export auszuwählen, oder alternativ einen Zeitbereich zu definieren. Wählen Sie die letzten 50 Messwerte aus und klicken Sie auf den Button **EXPORT**.

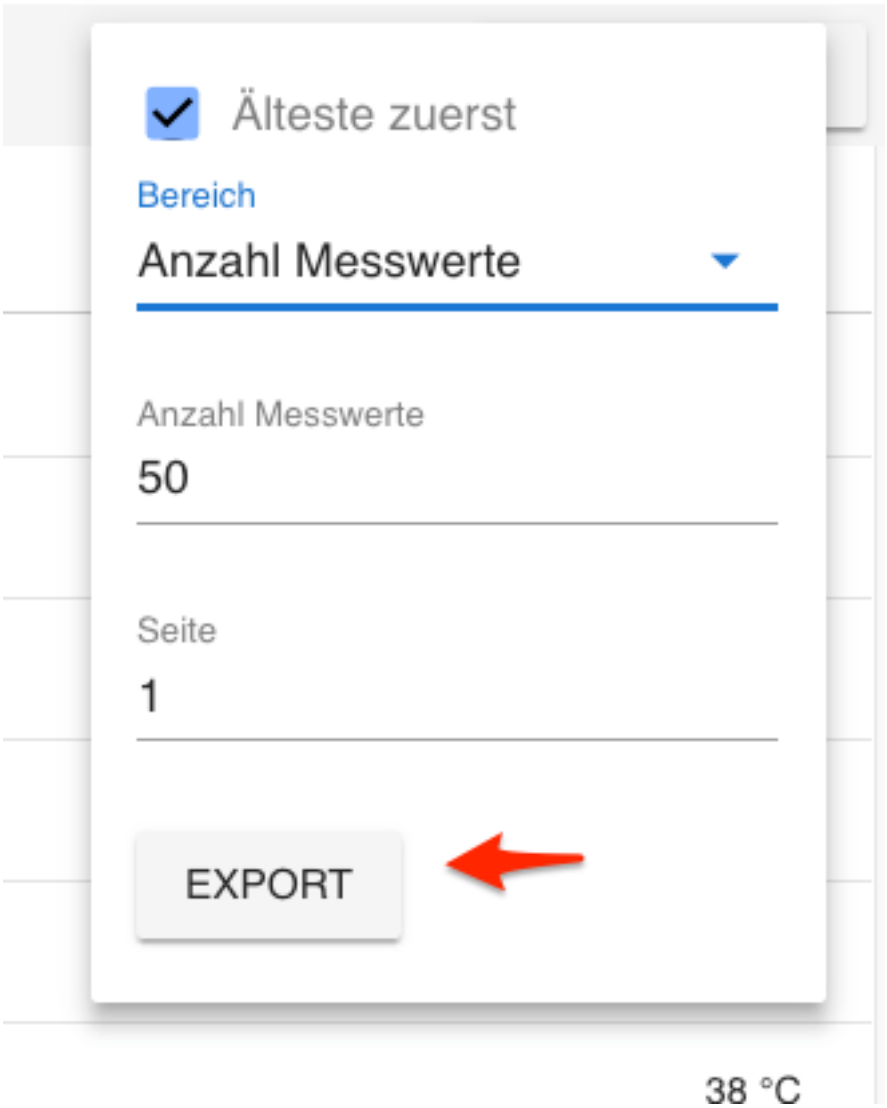


Abbildung 16.5: Anzahl Messwerte

Ihr Browser bietet Ihnen nun eine CSV-Datei zum Download an. Speichern Sie diese an einem beliebigen Ort auf Ihrem Computer.

16.5 Bearbeiten in Microsoft Excel

Möchten Sie die exportierten Daten zum Beispiel in Microsoft Excel bearbeiten, öffnen Sie Excel und erstellen Sie eine leere Arbeitsmappe. Im Menü **Daten** können Sie über die Schaltfläche **Aus Text** die

im vorigen Schritt erzeugte Datei importieren.

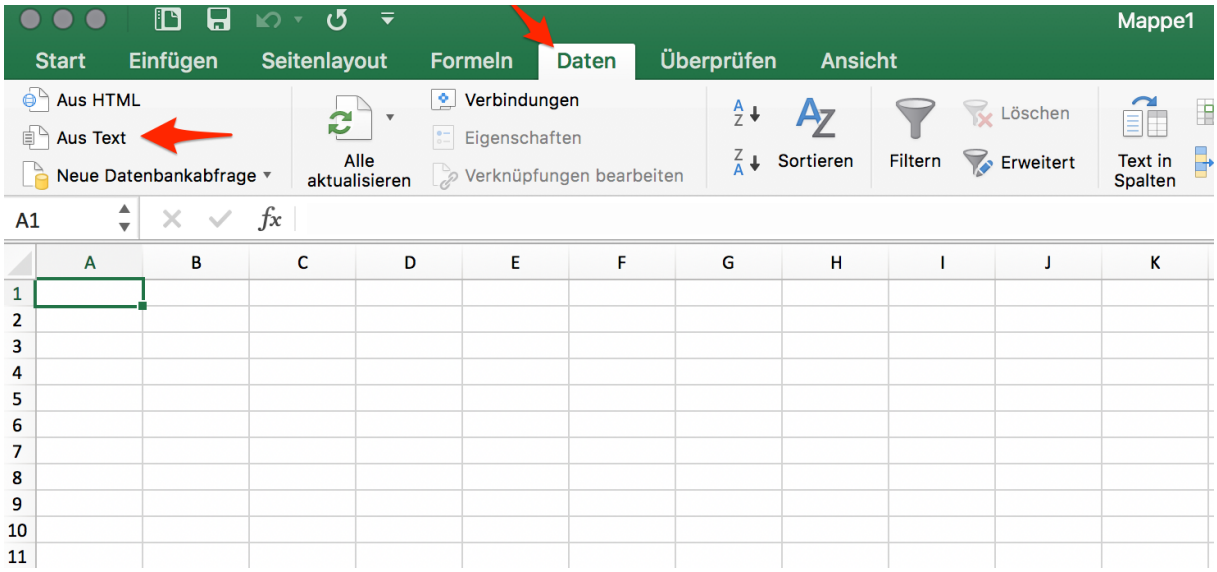


Abbildung 16.6: Ansicht Pakete

Wählen Sie die entsprechende Datei aus, welche von der ELEMENT IoT-Plattform erzeugt wurde.

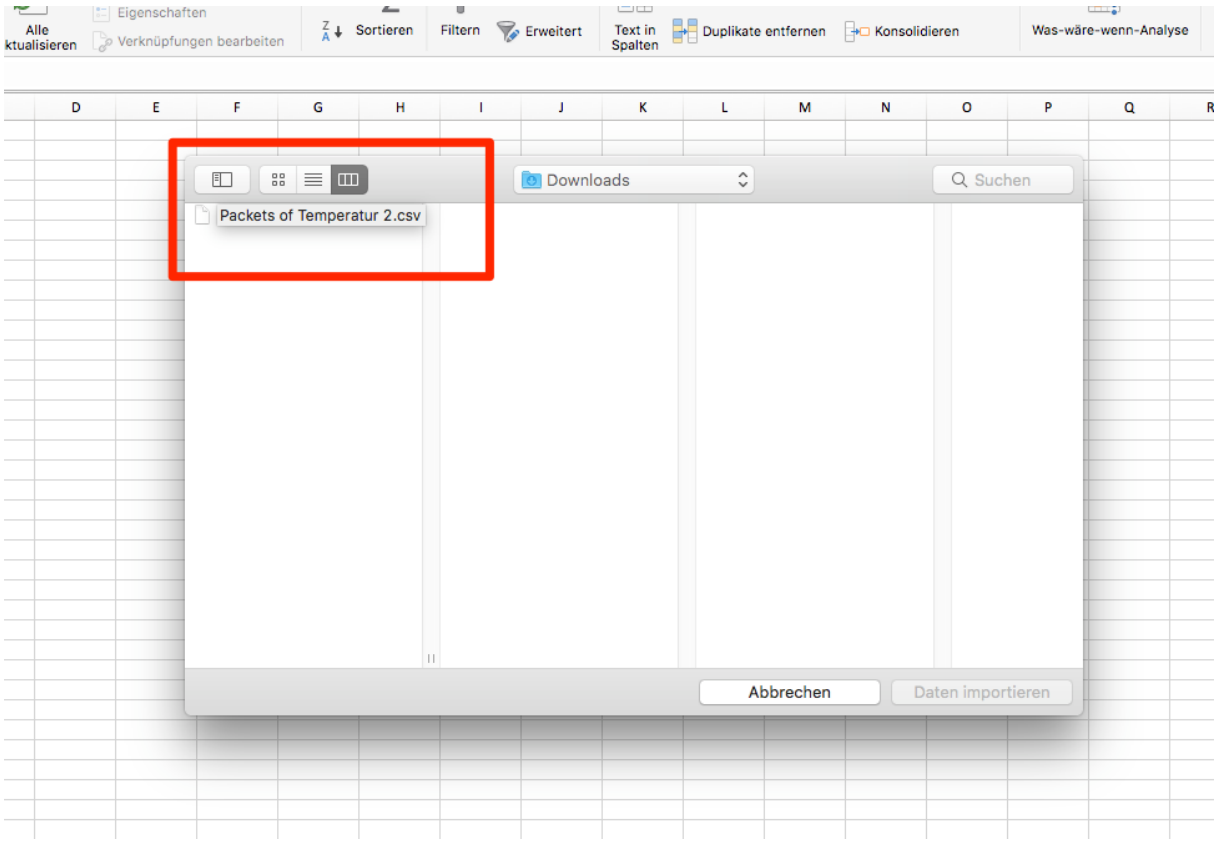


Abbildung 16.7: Ansicht Pakete

Im folgenden Dialog wählen Sie die Optionen

- Mit Trennzeichen versehen
- Dateiursprung: Unicode (UTF-8)

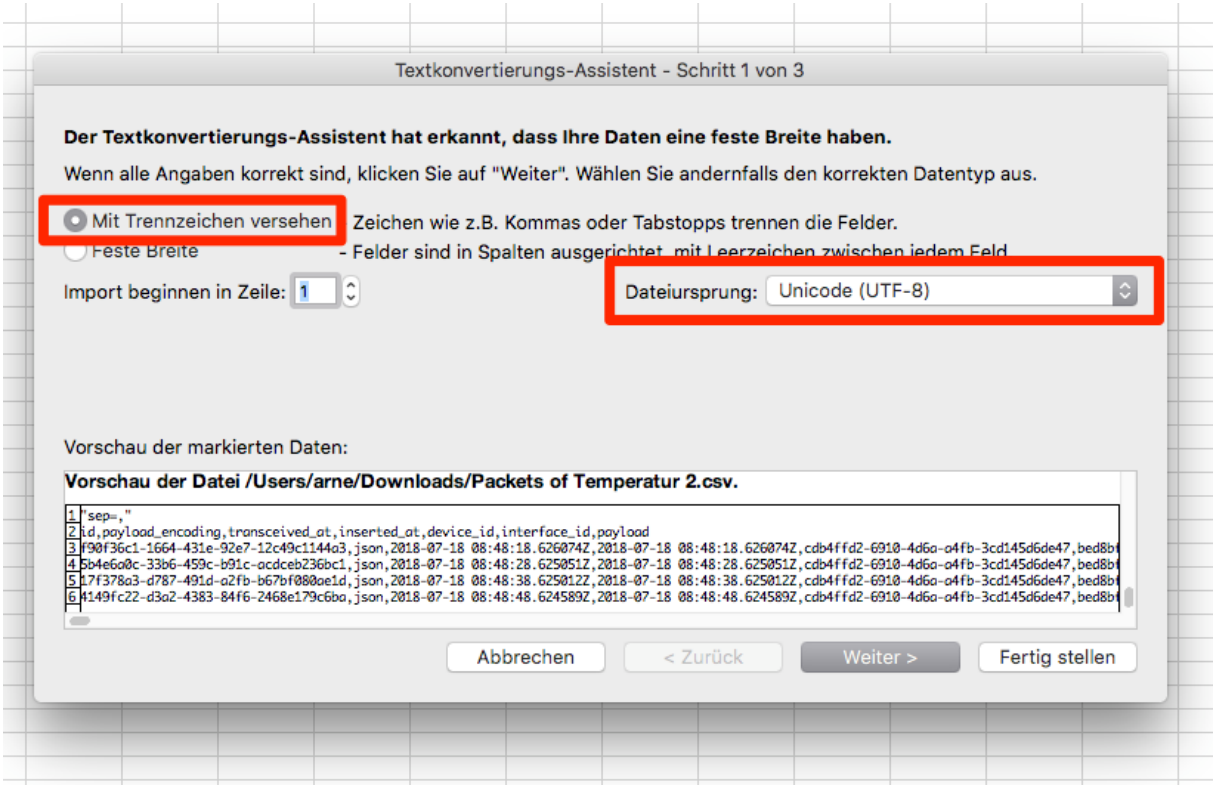


Abbildung 16.8: Import 1

Nach dem Klicken auf den Button **WEITER >** wählen Sie als Trennzeichen **Komma**.

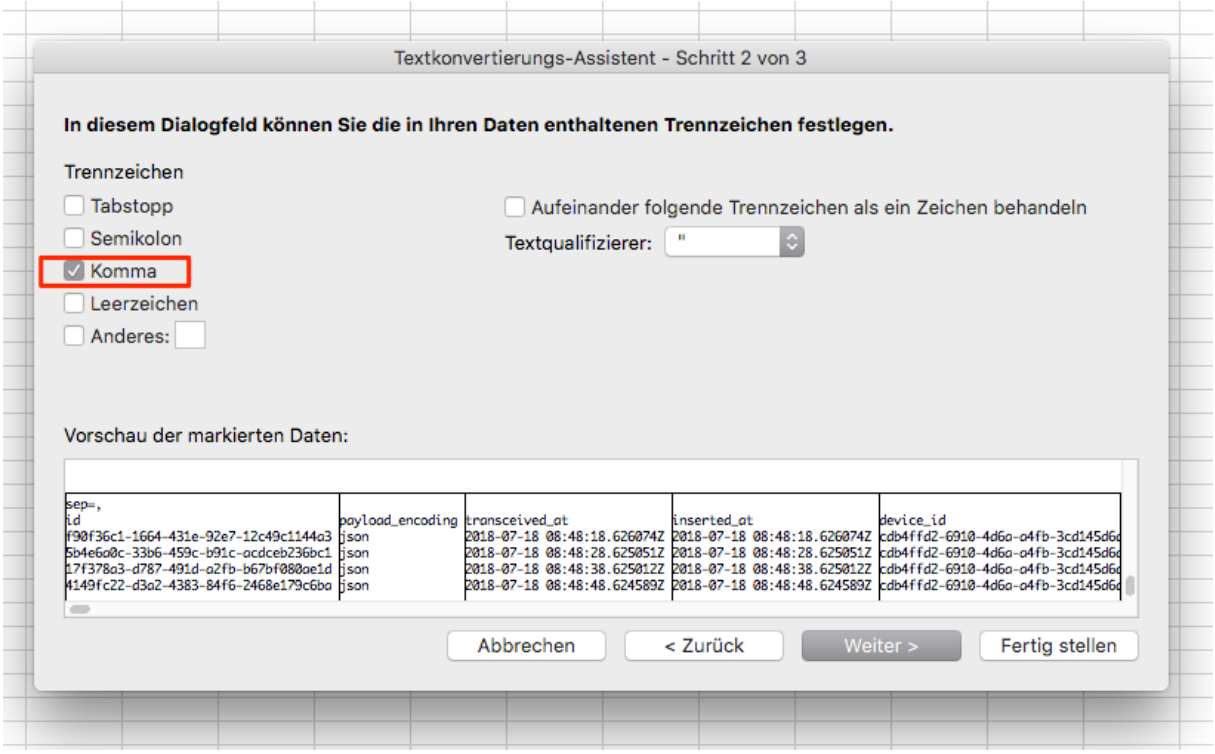
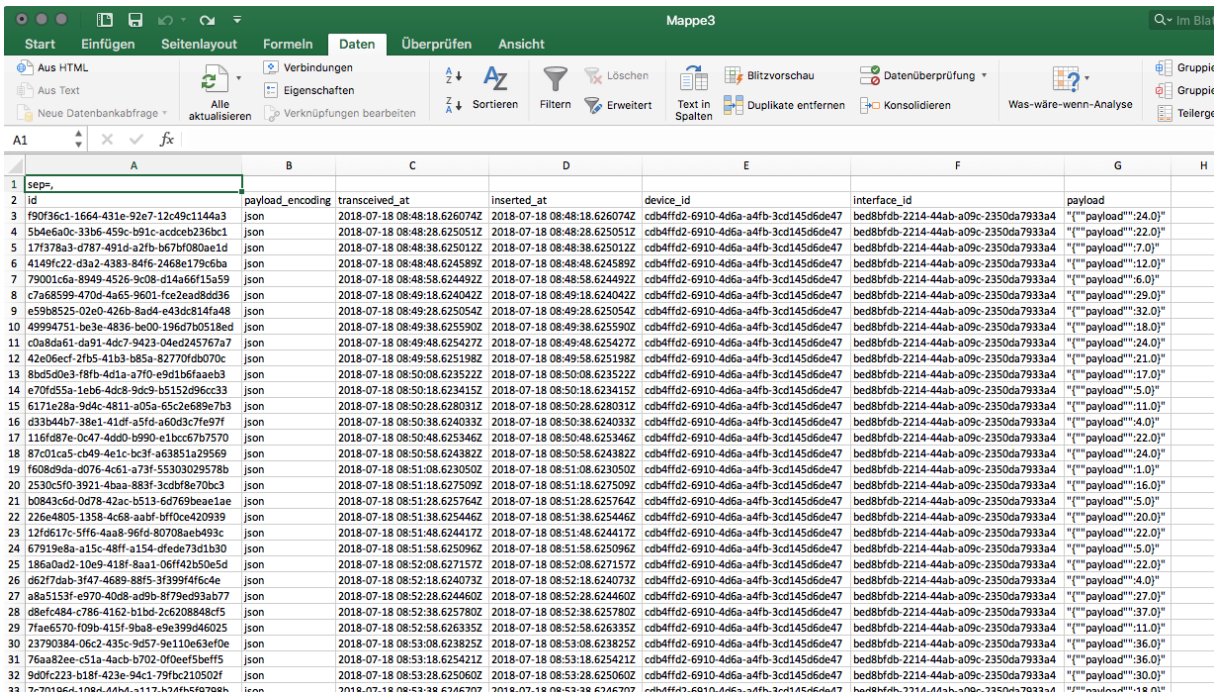


Abbildung 16.9: Import 2

Beenden Sie den Import über **Fertig stellen**.



	A	B	C	D	E	F	G
1	sep=,						
2	id	payload_encoding	transceived_at	inserted_at	device_id	interface_id	payload
3	f90f36c1-1664-431e-92e7-12c49c1144a3	json	2018-07-18 08:48:18.626074Z	2018-07-18 08:48:18.626074Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":24.0}"
4	5b4e6a0c-33b6-459c-b91c-acdceb236bc1	json	2018-07-18 08:48:28.625051Z	2018-07-18 08:48:28.625051Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":22.0}"
5	17f378a3-d787-491d-a2fb-b67bf080ae1d	json	2018-07-18 08:48:38.625012Z	2018-07-18 08:48:38.625012Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":7.0}"
6	4149fc22-d3a2-4383-84f6-2468e179c6ba	json	2018-07-18 08:48:48.624589Z	2018-07-18 08:48:48.624589Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":12.0}"
7	79001c6a-8949-4526-9c08-d14a66f15a59	json	2018-07-18 08:48:58.624492Z	2018-07-18 08:48:58.624492Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":6.0}"
8	c7a68599-470d-4a65-9601-fce2ead8dd36	json	2018-07-18 08:49:18.624042Z	2018-07-18 08:49:18.624042Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":29.0}"
9	e59b8525-02e0-426b-8ad4-e43dc814fa48	json	2018-07-18 08:49:28.625054Z	2018-07-18 08:49:28.625054Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":32.0}"
10	49994751-be3e-4836-be00-196d7b0518ed	json	2018-07-18 08:49:38.625590Z	2018-07-18 08:49:38.625590Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":18.0}"
11	c0a8da61-da91-4dc7-9423-04ed245767a7	json	2018-07-18 08:49:48.625427Z	2018-07-18 08:49:48.625427Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":24.0}"
12	42e06ecf-2fb5-41b3-b85a-82770fdb070c	json	2018-07-18 08:49:58.625198Z	2018-07-18 08:49:58.625198Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":21.0}"
13	8bd5d0e3-f8fb-4d1a-a7f0-e9d1b6faaeb3	json	2018-07-18 08:50:08.623522Z	2018-07-18 08:50:08.623522Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":17.0}"
14	e70fd55a-1eb6-4dc8-9dc9-b5152d96cc33	json	2018-07-18 08:50:18.623415Z	2018-07-18 08:50:18.623415Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":5.0}"
15	6171e28a-9d4c-4811-a05a-65c2e689e7b3	json	2018-07-18 08:50:28.628031Z	2018-07-18 08:50:28.628031Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":11.0}"
16	d33b44b7-38e1-41df-a5fd-a60d3c7fe97f	json	2018-07-18 08:50:38.624033Z	2018-07-18 08:50:38.624033Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":4.0}"
17	116fd87e-0c47-4dd0-b990-e1bcc67b7570	json	2018-07-18 08:50:48.625346Z	2018-07-18 08:50:48.625346Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":22.0}"
18	87c01ca5-cb49-4e1c-bc3f-a63851a29569	json	2018-07-18 08:50:58.624382Z	2018-07-18 08:50:58.624382Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":24.0}"
19	f608d9da-d076-4c61-a73f-55303029578b	json	2018-07-18 08:51:08.623050Z	2018-07-18 08:51:08.623050Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":1.0}"
20	2530c5f0-3921-4baa-883f-3cdbf8e70bc3	json	2018-07-18 08:51:18.627509Z	2018-07-18 08:51:18.627509Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":16.0}"
21	b0843c6d-0d78-42ac-b513-6d769beae1ae	json	2018-07-18 08:51:28.625764Z	2018-07-18 08:51:28.625764Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":5.0}"
22	226e4805-1358-4c68-aabf-bff0ce420939	json	2018-07-18 08:51:38.625446Z	2018-07-18 08:51:38.625446Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":20.0}"
23	12fd617c-5ff6-4aa8-96fd-80708aeb493c	json	2018-07-18 08:51:48.624417Z	2018-07-18 08:51:48.624417Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":22.0}"
24	67919e8a-a15c-48ff-a154-dfede73d1b30	json	2018-07-18 08:51:58.625096Z	2018-07-18 08:51:58.625096Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":5.0}"
25	186a0ad2-10e9-418f-8aa1-06ff42b50e5d	json	2018-07-18 08:52:08.627157Z	2018-07-18 08:52:08.627157Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":22.0}"
26	d62f7dab-3f47-4689-88f5-3f399f4f6c4e	json	2018-07-18 08:52:18.624073Z	2018-07-18 08:52:18.624073Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":4.0}"
27	a8a5153f-e970-40d8-ad9b-8f79ed93ab77	json	2018-07-18 08:52:28.624460Z	2018-07-18 08:52:28.624460Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":27.0}"
28	d8efc484-c786-4162-b1bd-2c6208848cf5	json	2018-07-18 08:52:38.625780Z	2018-07-18 08:52:38.625780Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":37.0}"
29	7fae6570-f09b-415f-9ba8-e9e399d46025	json	2018-07-18 08:52:58.626335Z	2018-07-18 08:52:58.626335Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":11.0}"
30	23790384-06c2-435c-9d57-9e110e63ef0e	json	2018-07-18 08:53:08.623825Z	2018-07-18 08:53:08.623825Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":36.0}"
31	76aa82ee-c51a-4acb-b702-0f0eef5beff5	json	2018-07-18 08:53:18.625421Z	2018-07-18 08:53:18.625421Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":36.0}"
32	9d0fc223-b18f-423e-94c1-79fbc210502f	json	2018-07-18 08:53:28.625060Z	2018-07-18 08:53:28.625060Z	cdb4ffd2-6910-4d6a-a4fb-3cd145d6de47	bed8bfdb-2214-44ab-a09c-2350da7933a4	"{""payload"":30.0}"

Abbildung 16.10: Import 2

17 Ausgangstreiber

Ausgangstreiber können genutzt werden, um Daten aus ELEMENT IoT in andere Systeme zu exportieren. Aktuell stehen 2 Ausgangstreiber zur Verfügung:

- Zu MQTT Broker publizieren
- Regelmässiger Messwertexport

Weiterhin gibt es die Möglichkeit, über die HTTP Push Aktion in einer Regel bestimmte Ereignisse sofort an HTTP Server zu senden.

17.1 Grundwissen

Ausgangstreiber nutzen mehrere fortgeschrittene Systeme in ELEMENT, mit denen Sie sich im Vorfeld vertraut machen sollten:

- **Streams anlegen**

  Jeder Ausgangstreiber wird einen Stream abonnieren, aus dem die Daten bezogen werden, die an das externe System weitergeleitet werden.

- **AbacusSql Ausdrücke schreiben**

  Die Daten, die von den Ausgangstreibern verarbeitet werden, können gefiltert werden. Darüber hinaus gibt oft es die Möglichkeit, über Ausdrücke dynamisch die übertragenen Nutzdaten zu berechnen.

- **Messwertansichten erstellen**

  Der regelmässige Messwertexport macht sich die konfigurierbaren Messwertansichten zunutze, um viele denkbare Tabellenformate zu unterstützen. Mit Ansichten kann definiert werden, in welche Spalten welche Werte eingetragen werden sollen. Zudem kann auch mit den Werten gerechnet werden, um z.B. Einheiten zu konvertieren oder mit Eich- oder Sollwerten aus Profildaten zu vergleichen.

- **Externe Systeme anlegen**

In externen Systemen werden die Zugangsdaten zu den für die Ausgangstreiber benötigten Diensten verwaltet. Ein passendes externes System ist immer notwendig, um einen Ausgangstreiber zu betreiben, allerdings muss ein externes System pro Mandant nur einmal angelegt werden und kann dann wieder verwendet werden.

17.2 Externe Systeme

In externen Systemen werden die Zugangsdaten zu den für die Ausgangstreiber benötigten Dienste verwaltet.

Externe Systeme sind für mehrere Ausgangstreiber wiederverwendbar.

Mittels Gruppenregeln lässt sich ein Berechtigungskonzept einrichten, in dem IT Admins die externen Systeme anlegen, und Integratoren diese dann in Ausgangstreibern nutzen, ohne geheime Zugangsdaten kennen zu müssen.

Zum Anlegen eines externen Systems navigieren Sie in Ihrem Mandanten zu `Einstellungen` -> `Externe Systeme` und klicken dann auf `Externes System hinzufügen`.

In dem sich öffnenden Formular stehen zunächst nur 2 Eingabefelder zur Verfügung:

- `Name`

  Freitext, um das System zu beschreiben, z.B. “MQTT Intern Test”

- `Typ des externen Systems`

  Hier können Sie entscheiden, über welches Protokoll mit dem externen System kommuniziert wird. Nach der Auswahl erscheinen darunter weitere Felder.

Mit dem Button `Einstellungen testen` wird je nach System ein Verbindungs- und Funktionstest durchgeführt und über etwaige Verbindungsprobleme informiert.

17.2.1 MQTT Broker

Dieser Typ steht für Ausgangstreiber, die mit MQTT arbeiten, zur Verfügung.

Zum Testen empfehlen wir den öffentlichen MQTT Broker von HiveMQ: www.hivemq.com/public-mqtt-broker/

Bitte beachten Sie hierbei, dass jeder die Daten, die darüber gesendet werden, lesen kann. Benutzen Sie diesen Broker also wirklich nur zum Testen.

Nach der Auswahl stehen folgende Felder zur Verfügung:

- `Hostname`

  Domainname oder IP des MQTT Brokers, z.B. “broker.hivemq.com”

- `Port`

  TCP Port an dem der MQTT Broker auf Verbindungen wartet; Standard ist `1883`

- `SSL`

  Wenn der Broker die Verbindung über SSL erlaubt oder erfordert, muss hier dafür die Option aktiviert werden

- `Benutzername`

- `Passwort`

- `Client-ID`

  Falls eine bestimmte Client-ID erfordert wird, kann diese hier eingetragen werden.

  Beachten Sie, dass ELEMENT IoT mehrere gleichzeitige MQTT Verbindungen aufbaut und dies bei manchen MQTT Brokern zu Problemen führt, sollte die gleiche Client-ID mehrmals verwendet werden.

  Wenn das Feld leer gelassen wird, erzeugt ELEMENT automatisch eine Client-ID nach dem Schema `element_{{ node id }}_{{ pid }}`.

17.2.2 SFTP (SSH)

Das SFTP-System erlaubt es Ausgangstreibern, auf das Dateisystem an einem SSH-Server zuzugreifen.

Nach der Auswahl stehen folgende Felder zur Verfügung:

- `Wurzelpfad`

  Der Ordner, in dem die benutzenden Ausgangstreiber Dateien lesen und schreiben dürfen.

- `Hostname`

  Domainname des SSH Servers

- `Port`

  Standard: 22

- `Nutzername`

- `Passwort`

17.3 Einen Ausgangstreiber anlegen

Wenn der Stream, das externe System sowie weitere benötigte Dinge (z.B. Ansichten für den Messwertexport) bereit stehen, kann ein Ausgangstreiber angelegt werden.

Dazu navigieren Sie in Ihrem Mandanten zu `Einstellungen` -> `Ausgangstreiber` und klicken dann auf `Neuen Ausgangstreiber hinzufügen`.

In dem Formular zum Hinzufügen / Bearbeiten gibt es folgende Felder:

- `Name`

  Freitext, um den Ausgangstreiber zu beschreiben. zB “Gaszähler Export”

- `Stream`

  Hier kann der im Vorfeld eingerichtete Stream gewählt werden, über den die Ereignisse nach Geräten und Ordnern gefiltert werden.

- `Empfangene Daten`

  Entweder `Pakete` oder `Messwerte`. Manche Treiber unterstützen nicht alle möglichen Daten.

- `Filter auf Ereignisse`

  Um die im Ausgangstreiber verwendeten Ereignisse weiter einzugrenzen, kann hier optional ein Filter definiert werden. Dieser Filter ist immer von dem gewählten Ereignistyp aus `Empfangene Daten` ausgehend.

- `Externes System`

  Das im Vorfeld eingerichtete externe System, an welches die Daten gesendet werden sollen

- `Typ des Ausgangstreibers`

  Sobald hier ein Typ ausgewählt wurde, erscheinen unter diesem Eingabefeld weitere Optionen, die unten näher beschrieben werden.

17.3.1 Zu MQTT Broker publizieren

Mit dem MQTT Ausgangstreiber können Pakete und Messwerte sofort, wenn sie eintreffen, an einen MQTT Broker weitergeleitet werden.

Jedes Ereignis führt in diesem Ausgangstreiber zu mindestens einer Nachricht, die publiziert wird.

Jede MQTT Nachricht hat 3 Parameter, die beim Senden entscheidend sind:

- `Topic`

  Eine mit `/` getrennte Pfadangabe, in welche Kategorie diese Nachricht einzuordnen ist. Beim Topic sind der Kreativität keine Grenzen gesetzt.

  Zum Beispiel könnte man den Namen des Geräts, aus dem das Ereignis stammt im Topic mitteilen, damit weitere Systeme genau ein Gerät abonnieren können.

  Einige Beispiele für Topics sind:

  - `devices/Gaszähler 837401/readings`
  - `parkplatz`

- `Payload`

  Der Inhalt der Nachricht kann in praktisch jedem Format gesendet werden. Nativ unterstützt der MQTT Treiber in ELEMENT das JSON Format, weitere Formate können mit AbacusSql Ausdrücken zusammengebaut werden.

- `QoS` (Quality of Service - Dienstgüte)

  Wie wichtig es ist, dass jede einzelne Nachricht mindestens bzw. maximal einmal beim Empfänger ankommt, kann mit diesem Parameter ausgedrückt werden.

  Faustregel: Je höher die Zahl, desto besser die Dienstgüte. Je kleiner die Zahl, desto weniger Bandbreite wird benötigt.

Es gibt folgende Optionen:

- `Topic`

  Sobald der gewünschte `Rendermodus` gewählt ist, erscheint darunter ein Eingabefeld.

  - Rendermodus `Statisches Topic`

    Im Textfeld `Topic` kann ein statisches Topic gewählt werden.

  - Rendermodus `AbacusSql Ausdruck`

    Im Feld `Topic Ausdruck` kann ein AbacusSql Ausdruck beschrieben werden, der eine Zeichenkette zurückgibt.

- `Nutzdaten`

  - Rendermodus `Ereignis als JSON kodieren`

    Das Ereignis wird als JSON versendet. Die Felder sind ähnlich denen, die auch in der API bei den entsprechenden Routen `/packets` bzw. `/readings` vorkommen.

  - Rendermodus `AbacusSql Ausdruck`

    Im Feld `Nutzdaten (Ausdruck)` kann ein Ausdruck hinterlegt werden, mit dem für jedes Ereignis die Nutzdaten berechnet werden.

    Der Rückgabewert muss auch hier eine Zeichenkette sein.

- `QoS`

  Hier kann die Dienstgüte bestimmt werden, mit der die Daten an den MQTT-Broker gesendet werden. Wenn Sie unsicher sind, wählen Sie `2 - Genau einmal`.

17.3.2 Messwert-Export nach Zeitplan

Der regelmässige Messwertexport verbindet sich mit einem Dateisystem und schreibt dort die Messwerte aus dem gewählten Stream in Dateien.

Dazu muss im Vorfeld eine Messwertansicht erstellt werden, denn diese zeigt dem Treiber, welche Spalten es gibt, und wie sie berechnet werden.

Es gibt folgende Optionen:

- `Ansicht die exportiert wird`

  Hier kann die bereits angelegte Ansicht ausgwählt werden, die bestimmt, welche Spalten in den exportierten Dateien vorkommen.

- `Export-Zeitplan`

  Dieses Feld bestimmt, zu welchen Zeiten der Export stattfinden soll.

  Der Ausdruck ist im Crontab-Format. Dieses besteht aus 5 Parameter, die der Reihenfolge nach für Minute, Stunde, Tag im Monat, Monat und den Wochentag stehen.

  Mit dem Platzhalter `*` kann angegeben werden, dass in diesem Parameter jeder Wert erlaubt sein soll. Zusätzlich lässt sich der Platzhalter durch eine Zahl teilen, zB `*/5` im Minutenparameter steht für alle 5 Minuten.

  Mit `-` können Zahlenbereiche definiert werden. `19-5` steht im Stundenparameter für alle Stunden zwischen 19 und 5 Uhr.

  Beispielsweise steht der Crontab-Ausdruck `30 19 * * 1-5` dafür, an jedem Wochentag um 19:30 etwas auszuführen.

  Weitere Beispiele enthält der Wikipedia-Artikel zu Crontab `de.wikipedia.org/wiki/Cron#Crontab`.

- `Zeitzone des Zeitplans`

  Stellen Sie hier Ihre lokale Zeitzone an, in Deutschland ist das `Europe/Berlin`.

- `Dateiformat`

  - `JSON`

    exportiert ein JSON Objekt pro Zeile

  - `CSV`

    exportiert die Ansicht im CSV-Format

  - `CSV mit Excel-Eigenheiten`

    Exportiert die Ansicht im CSV-Format mit einigen speziellen Steuerzeichen, damit Microsoft Excel diese Dateien besser interpretiert. Sollen die CSV- Dateien später von anderen Programmen als Excel gelesen werden, sollten Sie dieses Format nicht verwenden.

  - `Excel (xlsx)`

    Exportiert die Ansicht im Office Open XML Format.

- `Eine Datei pro Gerät`

  Wenn ausgewählt, erzeugt der Export pro Gerät, das in dem gewählten Stream vorkommt, eine Datei und schreibt dort nur die Messwerte, die in dem Gerät vorkommen.

  Diese Option beeinflusst, welche Möglichkeiten in `Bestimmung des Dateinamens` zur Auswahl stehen.

- `Bestimmung des Dateinamens`

  - `Statischer Dateiname`

    Es wird alles in eine Datei geschrieben. Der Dateiname kann im Feld darunter angegeben werden

  - `Aktuelles Datum voranstellen`

    Dem Dateinamen wird das aktuelle Datum im Format `YYYY-MM-DD-` vorangestellt.

  - `Gerätenamen voranstellen`

    Dem Dateinamen wird der Name des Geräts vorangestellt.

  - `Gerätenamen und aktuelles Datum voranstellen`

    Sowohl Gerätename als auch Datum (im oben angegebenen Format) wird dem Dateinamen vorangestellt.

  - `Mit AbacusSql Ausdruck berechnen`

    Der Dateiname wird pro Gerät mit einem AbacusSql Ausdruck berechnet. Der Rückgabewert des Ausdrucks muss eine Zeichenkette sein.

- `Dateiname` oder `Dateiname (Ausdruck)`

  Je nach Auswahl in der `Bestimmung des Dateinamens` kann hier der Suffix oder direkt ein Ausdruck geschrieben werden, mit dem der Dateiname berechnet wird.

- `Datei überschreiben, wenn sie schon existiert`

  Wenn ausgewählt, werden bei späteren Exports alle bereits vorhandenen Daten überschrieben, sofern die Dateien den selben Namen haben.

  Wenn nicht ausgewählt, sollte im `Filter` ein Zeitfilter definiert werden, damit es nicht zu Duplikaten kommt.

  Das XLSX-Format erlaubt es nicht effizient, weitere Zeilen anzuhängen, daher wird in diesem Format immer die Datei überschrieben.

- `CSV Zelltrennzeichen` (Nur bei CSV, nicht bei CSV für Excel)

  Welches Zeichen genutzt werden soll, um Spalten voneinander zu trennen. z.B. `;`

- `CSV Dezimaltrennzeichen` (Nur bei CSV)

  Welches Dezimaltrennzeichen genutzt werden soll, z.B. `,` oder `.`

18 ELEMENT CSV Export Tool

18.1 Datenexportmöglichkeiten

Auf Anfrage ist ein CSV-Export-Tool für die Linux-Kommandozeile verfügbar, welche Daten über die ELEMENT API abruft und in frei definierbare CSV-Dateien schreibt und per E-Mail versendet oder im Dateisystem ablegt.

18.2 CSV-Export Tool

Die Konfigurationsdatei des CSV-Exporttools enthält die Dokumentation:

```
# URL der Element-Installation die anbegefragt werden soll
# default https://element-iot.com
base_url = "https://element-iot.com"

# API Key für die Abfrage, muss im Mandaten angelegt sein
# kein default# ELEMENT Schnittstellen
# kann Gerätespezifisch überschrieben werden
api_key = "e8d410e52b276c6pf056410c452341f4"

# Ausgabepfad
# default .
out_path = "."

# CSV Trennzeichen
# default ,
separator = ";"

# Dezinaltrennzeichen
# Default .
decimal_separator = ","

# Spaltennamen für den Export
# default ["Name", "Datum", "Wert"]
# kann Gerätespezifisch überschrieben werden
header = []
```

```
# Definition der Spaltenwerte pro Messwertergbniszeile
#
# Jeder Eintrag besteht aus drei Teilen,
# durch ':' getrennt: <Zugriffsart>:<Typ>:<Wert oder Pfad zum Zugriff>
#
# Zugriffsarten bestimmen, wie der Teil <Wert oder Pfad zum Zugriff>
    genutzt
# werden kann
#
# device = Auf ein Feld aus dem Gerät zugreifen,
# siehe https://docs.element-iot.com/resources/devices/
# Es werden Pfadausdrücke unterstützt z.B. fields.example
# Es kann nicht in Arrays navigiert werden
#
# profile = Es kann auf Daten aus einen Geräteprofil zugegriffen werden.
# Z.B. wenn es ein Profil zaehler mit dem Feld seriennummer gibt,
# kann profie:string:zaehler.seriennummer verwendet werden
#
# readings = Es kann auf Felder in einem Reading zugegriffen werden
# (keine Pfadausdrücke)
# Z.b. Wenn es in den Messwerten ein Feld value gibt,
# kann readings:string:value verwendet werden
#
# meta = Es kann auf Metadaten der Readings zugegriffen werden
# (keine Pfadausdrücke)
# typischerweise wäre dies meta:datetime:measured_at
#
# fixed = Es wird nicht zugegriffen sondern der Wert direkt genutzt
#
# Typ-Feld
# Es werden vier Typen unterstützt: string, integer, float und datetime
#
# default
# ["device:string:name", "meta:datetime:measured_at", "reading:string:
    value"]
columns = [
        "meta:datetime:measured_at",
        "device:string:name",
        "reading:string:usage"
]

# Teile aus denen der Dateiname zusammengesetzt werden soll.
# Mögliche Werte wie bei columns ohne readings und meta
# Zusätzlich ist special:datetime:now für aktuelles Datum und Zeit
file_name = ["device:string:name", "fixed:string:_", "special:datetime:now
    "]

# Format für Zeitangaben, wird genutzt zum Formatieren der Zeitangaben und
    wenn
# in columns fixed:datetime verwendet wird
# Mögliche Formate:
```

```
# https://hexdocs.pm/timex/Timex.Format.DateTime.Formatters.Default.html#
    content
#
# default {ISO:Extended:Z} (Isso 8602 mit Zeitzone)
dateformat = "{0D}.{0M}.{YYYY} {0h24}:{0m}"

# Zeitzone die die Zeitangaben haben soll
# default UTC
timezone = "Europe/Berlin"

# Senden der Dateien (einzeln) an folgende Email
# Wenn der leer ("") dann wird nichts gesendet
send_email = "op@zenner-iot.com"

# Löschen nach dem Senden. Nur wenn eine Email gesetzt wurde.
delete_after_send = false

# Sortierrichtung nach Datum
# default descending = absteigend
# ascending = aussteigend
sort_direction = "ascending"

filter = "data.mark == \"E\""

[zaehler]

[zaehler.UZ1hybridhalle]
# ID oder slug des Gerätes, welches abgefragt werden soll
device_id = "4c6ba668-8272-4c3b-bb8c-4da253e603c1"

# Maximales alter in Tagen vom Ausführungsdatum des Skripts
# default 1
max_age = 4

# Wieviele Werte sollen abgeholt werden
# default 1
limit = 384

columns = [
        "meta:datetime:measured_at",
        "fixed:string:UZ1hybridhalle",
        "reading:float:usage"
]

file_name_parts = ["fixed:string:UZ1hybridhalle_", "special:datetime:now"]

[zaehler.UZ2reparaturhalle]
# ID oder slug des Gerätes, welches abgefragt werden soll
device_id = "5bd567db-d7f9-48d6-8e4e-7276e8c91953"

# Maximales alter in Tagen vom Ausführungsdatum des Skripts
```

```
# default 1
max_age = 4

# Wieviele Werte sollen abgeholt werden
# default 1
limit = 384

columns = [
        "meta:datetime:measured_at",
        "fixed:string:UZ2reparaturhalle",
        "reading:float:usage"
]

file_name_parts = ["fixed:string:UZ2reparaturhalle_", "special:datetime:
    now"]
```

19 ELEMENT REST-API

Die ELEMENT IoT-Plattform bietet eine umfängliche Schnittstelle (API) zum programmatischen Zugriff auf alle Entitäten. Nahezu alle wichtigen Funktionen, die über das Nutzerfrontend verfügbar sind, werden ebenfalls über die API bereitgestellt.

> Standardisierte Programmierschnittstellen (APIs) über unterschiedliche Betriebssysteme sorgen für Quelltextkompatibilität, d. h. Quelltext kann ohne Anpassungen für die jeweiligen Systeme erfolgreich kompiliert werden. Eine Programmierschnittstelle, genauer Schnittstelle zur Anwendungsprogrammierung, häufig nur kurz API genannt (englisch application programming interface, wörtlich ‚Anwendungsprogrammierschnittstelle'), ist ein Programmteil, der von einem Softwaresystem anderen Programmen zur Anbindung an das System zur Verfügung gestellt wird. Im Gegensatz zu einer Binärschnittstelle (ABI) definiert eine Programmierschnittstelle nur die Programmanbindung auf Quelltext-Ebene. Zur Bereitstellung solch einer Schnittstelle gehört meist die detaillierte Dokumentation der Schnittstellen-Funktionen mit ihren Parametern auf Papier oder als elektronisches Dokument.

Quelle: Wikipedia

In Folgenden wird der praktische Nutzen der Schnittstelle mit Beispielen beschrieben, eine detaillierte Dokumentation der API findet sich in der Online API Dokumentation.

19.1 API-Endpunkt

Anhängig von der ELEMENT-Instanz, welche genutzt werden soll, wird eine Endpunkt-URL benötigt. Alle API-Routen sind relativ zu dieser Endpunkt-URL.

Die deutsche Clound-Instanz von ELEMENT IoT findet sich unter <element-iot.com>, daher lautet der API-Endpunkt https://element-iot.com/api/v1

19.2 Authentifizierung

Zum Ausweis dafür, dass ein API-Aufruf berechtigt ist, wird ein API-Schlüssel mittels der Query-Parameters `auth` angegeben. Der API-Schlüssel für ein Gerät ist in den Geräteeinstellungen unter dem Punkt *API* auffindbar, sofern es einen API-Schlüssel mit Berechtigungen auf dieses Geräte gibt (siehe API-Schlüssel).

Ein Aufruf der API mit `auth`-Parameter würde entsprechend so aussehen: `https://element-iot.com/api/v1?auth=46cb688e2c0b468e26e914235d4b73ea`.

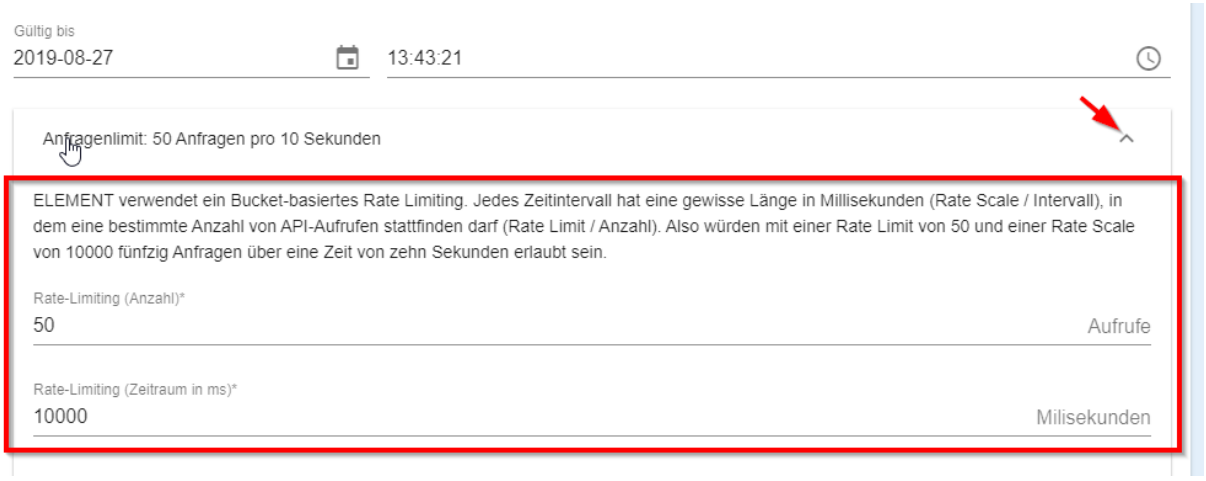
Wird eine API-Route, für welche eine Authentifizierung nötig ist, ohne `auth`-Parameter aufgerufen, so wird der HTTP-Response-Code `401 Unautherized` zurückgegeben.

Wird eine API-Route mit unzureichender Berechtigung aufgerufen, so wird der HTTP-Response-Code `403 Forbidden` zurückgegeben.

19.3 Rate-Limit

Um eine übermäßige Nutzung der API zu Lasten anderere Nutzer zu verhindern, ist die Anzahl der Anfragen pro Zeitraum für jeden API-Schlüssel begrenzt (Rate-Limit).

Die Anzahl der Anfragen und die Länge der Zeiträume kann in der API-Schlüsseleinstellung festgelegt werden. Standardmäßig sind 50 Anfragen in einem Zeitraum von 10 Sekunden erlaubt.



Wird die Anzahl der Anfragen pro Zeitraum überschritten, wird ein HTTP-Response-Code `429 Too Many Requests` gesendet.

Alle API-Antworten enthalten die folgenden HTTP-Kopfzeilen mit Bezug zum Rate-Limit

Kopfzeile	Beispiel	Bedeutung
`x-ratelimit-limit`	`50/10000ms`	das eingestellte Rate-Limit für den angegebenen API-Schlüssel
`x-ratelimit-reset`	`245`	Zeit in Millisekunden bis zum Ende des aktuellen Zeitraums
`x-ratelimit-remaining`	`10`	verbleibende Abfragen im aktuellen Zeitraum

19.4 Pagination

Jeder API-Aufruf, der eine Liste von Entitäten zurückgibt, wird *paginiert*. *Pagination* bedeutet, dass die Listenelemente seitenweise mit einem Offset zurückgegeben werden. Dies betrifft die folgenden Listen:

- Geräteliste
- Geräte-Interface-Liste
- Paketliste
- Messwertliste
- Ordnerliste
- Ordner-Geräte-Liste
- API-Schlüssel-Liste
- Parser-Liste

Die folgenden Query-Parameter können im Zusammenhang mit der *Pagination* angegeben werden:

Parameter	Beispiel	Beschreibung
`limit`	`10`	Elemente pro Seite, Standard ist `10`, das Maximum ist `100`
`sort`	`name`	Feld der Entitäten, nach dem sortiert werden soll; Standard ist `inserted_at`

Parameter	Beispiel	Beschreibung
`sort_direction`	`ascending`	Sortierrichtung: `ascending` (aufsteigend) oder `descending` (absteigend); Standard ist `descending`
`retrieve_after`	`881af0c9-72c2-4758-af09-4901bef0c15c`	ID des Letzten Elements, welches schon bekannt in der vorherigen Seite zurückgegeben wurde, abhängig von `sort` und `sort_direction`
`before`	`2017-09-05T13:46:47.956173Z`	Alter der Listenelemente muss vor dem Datum liegen
`after`	`2017-09-04T03:12:02.882313Z`	Alter der Listenelemente muss nach dem Datum liegen

Datumangaben für `before` und `after` können wie folgt angegeben werden:

- **Unix Epoch Timestamp**
- **ISO 8601 Basic**: `<date>T<time><offset>`
- **ISO 8601 Basic Z**: `<date>T<time>Z`
- **ISO 8601 Extended**: `<date>T<time><offset>` (z.B. `2007-08-13T16:48:01 +03:00`)
- **ISO 8601 Extended Z**: `<date>T<time>Z` (z.B. `2007-08-13T13:48:01Z`)
- **ISO 8601 Date**: `YYYY-MM-DD` (e.g. `2007-08-13`)

Jeder API-Aufruf mit einer Liste von Enitäten im `body` Feld, gibt außerdem ein `retrieve_after_id`-Feld zurück. Diese ID kann mit dem `retrieve_after`-Parameter in dem folgenden API-Aufruf angegeben werden, um die folgenden Entitäten gemäß Sortierung zu erhalten:

```
{
  "status": 200,
  "retrieve_after_id": "0ff0c75a-b2ac-4051-a2df-d2c0fa97655c",
  "ok": true,
  "body": [ /* Items */ ]
}
```

19.5 Geräte-API

Über die API-Route `devices` können Geräte abgerufen, angelegt, geändert und gelöscht werden.

Die [Datenstruktur] von Geräten wird als JSON codiert, z.B. (Auszug):

```
{
  "updated_at": "2017-08-09T09:27:17.641848",
  "tags": [
    // ...
  ],
  "static_location": true,
  "slug": "device-1",
  "profile_data":[
    // ...
  ],
  "parser_id": "cee40cb2-f58c-4ca8-ae91-4ae76cba5119",
  "name": "Device #1",
  "meta": {},
  "mandate_id": "50e032fb-964f-440d-9b9a-47870667373c",
  "location": {
    "type": "Point", "coordinates": [9.959696531296402,53.5501216243068]
  },
  "interfaces": [
    // ..
  ],
  "inserted_at": "2017-08-09T09:24:42.866589",
  "id": "a70a234e-d186-4ca8-836b-74816738939e",
  "fields": {
    // ..
  }
}
```

Wobei die Interfaceliste für ein LoRaWAN-Gerät z.B. so aussieht:

```
  [
    {
      "opts": {
        "network_session_key": "",
        "managed": false,
        "enabled": true,
        "duplicate_framecounter_allowed": true,
        "device_type": "otaa",
        "device_key": "",
        "device_eui": "",
        "device_address": "",
        "app_session_key": ""
      },
      "id": "0f7c53ae-4366-42dd-9ffc-158e1ccae7aa",
      "driver_instance_id": "9cf80c6a-1669-4cbf-96d0-9eefb707d294",
      "device_id": "a70a234e-d186-4ca8-836b-74816738939e"
    }
  ]
```

Beispielhaft werden im Folgenden die API-Aufrufe für Geräte beschrieben. Die Logik kann auf alle ande-

ren Enitäten übertragen werden und ist ausführlich in der Online API Dokumentation beschrieben.

19.5.1 Liste aller Geräte

Methode: `GET`, Endpunkt: `/devices`

Beispielanfrage

```
GET https://element-iot.com/api/v1/ \
devices?auth=46cb688e2c0b468e26e914235d4b73ea
```

Beispielantwort

```
{
  "status": 200,
  "retrieve_after_id": "a70a234e-d186-4ca8-836b-74816738939e",
  "ok": true,
  "body": [
    // Contains an array of devices
  ]
}
```

19.5.2 Einzelnes Gerät

Methode: `GET`, Endpunkt: `/devices/:device_id`

Beispielanfrage

```
GET https://element-iot.com/api/v1/ \
devices/test-1?auth=46cb688e2c0b468e26e914235d4b73ea
```

Beispielantwort

```
{
  "status": 200,
  "ok": true,
  "body": {
    // Contains the device
  }
}
```

19.5.3 Gerät anlegen

Methode: `POST`, Endpunkt: `/devices`

Beispielanfrage

```
POST https://element-iot.com/api/v1/ \
devices?auth=46cb688e2c0b468e26e914235d4b73ea
{
  "device": {
    "name": "New Name",
    "type": "sensor",
    "tags": [
      {
        "id": "9e5121d8-dbb7-42d9-ba46-1bc95af3a350"
      },
      {
        "action": "create",
        "name": "New Tag",
        "color_hue": 100
      }
      ]
  }
}
```

Beispielantwort

```
{
  "status": 200,
  "ok": true,
  "body": {
    // Contains the new device
  }
}
```

Um ein Gerät mit Ordnern (tags) zu speichern, gibt es zwei Möglichkeiten, diese anzugeben. Existierende Tags werden mit ihrer ID referenziert, z.B. `"tags": [{"id": "9e5121d8-dbb7-42d9-ba46-1bc95af3a350"}]`. Neue tags können mit einem Aktionsfeld (`action: create`) angegeben werden, z.B.: `"tags": [{"action": "create", "name": "That new tag", "color_hue": 100}]`.

19.5.4 Gerät ändern

Methode: `PUT` oder `PATCH`, Endpunkt: `/devices/:device_id`

Beispielanfrage

```
PUT https://element-iot.com/api/v1/ \
devices/test-1?auth=46cb688e2c0b468e26e914235d4b73ea
{
  "device": {
    "name": "New Name",
    "type": "sensor"
```

```
  }
}
```

Beispielantwort

```
{
  "status": 200,
  "ok": true,
  "body": {
    // Contains the changed device
  }
}
```

Der Request-Body muss dabei nur die zu änderenden Felder enthalten.

Die ID kann das `id`-Feld oder der `slug` des Gerätes sein.

19.5.5 Gerät löschen

Methode: `DELETE`, Endpunkt: `/devices/:device_id`

Beispielanfrage

```
DELETE https://element-iot.com/api/v1/ \
devices/test-1?auth=46cb688e2c0b468e26e914235d4b73ea
```

Beispielantwort

```
{
  "status": 200,
  "ok": true,
  "body": {
    // Contains the deleted device
  }
}
```

Die ID kann das `id`-Feld oder der `slug` des Gerätes sein.

19.5.6 Interfaces eines Geräts anzeigen

Methode: `GET`, Endpunkt: `/devices/:device_id/interfaces`

Beispielanfrage

```
GET https://element-iot.com/api/v1/ \
devices/test-1/interfaces?auth=46cb688e2c0b468e26e914235d4b73ea
```

Beispielantwort

```
{
  "status": 200,
  "ok": true,
  "body": {
    // Contains a list of the interfaces attached to the device
  }
}
```

19.5.7 Pakete eines Geräts anzeigen

Methode: `GET`, Endpunkt: `/devices/:device_id/packets`

Beispielanfrage

```
GET https://element-iot.com/api/v1/ \
devices/test-1/packets?auth=46cb688e2c0b468e26e914235d4b73ea
```

Beispielantwort

```
{
  "status": 200,
  "ok": true,
  "body": {
    // Contains a list of the packets attached to the device
  }
}
```

Der optionale Parameter `packet_type` kann genutzt werden um nur bestimmte Pakettypen abzurufen, z.B. `up`.

Die Sortierung ist fest eingestellt auf `transceived_at`.

19.5.8 Messwerte eines Geräts anzeigen

Methode: `GET`, Endpunkt: `/devices/:device_id/readings`

Beispielanfrage

```
GET https://element-iot.com/api/v1/ \
devices/test-1/readings?auth=46cb688e2c0b468e26e914235d4b73ea
```

Beispielantwort

```
{
  "status": 200,
  "ok": true,
```

```
  "body": {
    // Contains a list of the readings attached to the device
  }
}
```

Die Sortierung ist fest eingestellt auf `measured_at`.

19.6 Konten-API

Mit dieser API können Benutzer und API-Schlüssel verwaltet werden.

Konten (accounts) sind eine Sammlung aus Benutzern (users), die jeweils in verschiedenen Mandanten mit verschiedenen Rollen und Gruppen sein können.

19.6.1 Felder eines Kontos

Feld	Beschreibung	Bearbeiten?
`name`	Der volle Name des Kontos	ja
`auth_method`	Wie sich das Konto bei ELEMENT authentifiziert. (email_password, public_api_key)	beim Erstellen
`public_api_key`	Der API Schlüssel, der in den `auth` query-Parameter eingefügt werden kann (nur bei `auth_method=public_api_key`)	nein
`email`	Die Emailadresse des Kontos (nur bei `auth_method=email_password`)	ja
`flags`	Bestimmte Nutzerspezifische Einstellungen	ja
`language`	UI-Sprache des Kontos (de/en/fr)	ja
`password_changed_at`	Wann hat der Nutzer sein Passwort das letzte Mal geändert	nein
`confirmed_at`	Wann hat der Nutzer den Link in der Einladungsemail angeklickt und das Formular ausgefüllt	nein
`inserted_at`	Wann wurde das Konto erstellt	nein
`updated_at`	Wann wurde das Konto zuletzt bearbeitet	nein

Feld	Beschreibung	Bearbeiten?
`users`	Liste der Benutzer in dem Konto - dies entscheidet, welche Informationen angesehen und bearbeitet werden dürfen	ja

19.6.2 Felder eines Nutzers

Jede der unten beschriebenen API-Endpunkte arbeitet jeweils mit den gleichen Feldern für jeden Nutzer.

Feld	Beschreibung	Bearbeiten?
`id`	Automatisch erzeugte, eindeutige UUID	nein
`groups`	Liste der Gruppen, in denen der Benutzer ist (Berechtigungen aus Gruppen gelten nur für Nutzer, bei denen `role=user` ist.)	ja
`role`	Rolle des Nutzers (read_only/user/admin)	ja
`mandate_id`	ID des Mandanten, in dem der Nutzer arbeiten darf, oder **`null`** falls der Nutzer global ist	ja
`permissions`	Welche Rechte hat der Nutzer aus seiner Rolle und seinen Gruppen	nein
`inserted_at`	Wann wurde der Nutzer erstellt	nein
`updated_at`	Wann wurde der Nutzer das letzte mal bearbeitet	nein

Weitere Felder, die bei `GET` Routen geliefert werden entfallen mit dem nächsten Release.

19.6.3 `GET /accounts`

Listet alle Konten auf, die der benutzte API-Schlüssel sehen darf.

Diese Route benutzt die gleiche Paginierung wie alle anderen Auflistungsrouten auch.

Beispiel `GET /api/v1/accounts?auth=xyz&limit=2`


```
{
  "body": [
    {
      "avatar_url": "https://www.gravatar.com/avatar/
         b3bb7f515176fa0e46be2ddf6b893a3c?d=retro&s=40",
      "updated_at": "2021-09-28T12:29:17.614511Z",
```

```
      "inserted_at": "2021-09-28T12:29:17.614511Z",
      "allowed_origins": null,
      "public_api_key": null,
      "password_changed_at": null,
      "confirmed_at": null,
      "email": "user_a@example.com.com",
      "flags": {},
      "auth_method": "email_password",
      "language": null,
      "name": "Account A",
      "id": "9ff43afb-fb61-4809-b65e-3453885f56ef"
    },
    {
      "avatar_url": "https://www.gravatar.com/avatar/015521
         e509baf0df76403212081ed530?d=retro&s=40",
      "updated_at": "2021-09-01T13:49:18.317004Z",
      "inserted_at": "2021-09-01T12:24:00.541793Z",
      "allowed_origins": null,
      "public_api_key": "AABBCCDDEEFF...",
      "password_changed_at": null,
      "confirmed_at": null,
      "email": null,
      "flags": {},
      "auth_method": "public_api_key",
      "language": null,
      "name": "API Key B",
      "id": "a040f7bc-5bc5-4ab8-8ee4-9a058d2f5a2a"
    }
  ],
  "ok": true,
  "retrieve_after_id": "a040f7bc-5bc5-4ab8-8ee4-9a058d2f5a2a",
  "status": 200
}
```


19.6.4 POST /account

Erstellt ein neues Konto und schickt eine Einladungsemail an die angegebene Emailadresse, falls `auth_method` auf `email_password` gesetzt ist.

Beispiel

```
POST /api/v1/accounts?auth=xyz
Content-Type: application/json

{
  "account": {
    "name": "C",
    "email": "c@example.com",
    "auth_method": "email_password",
```

```
    "language": "de",
    "users": [
      {
        "mandate_id": "ac7327a3-5d6f-4ce1-aace-d830769cddba",
        "role": "user",
        "groups": [{"id": "4f63c93d-e168-4a4e-9f1c-c0a38f996eb9"}]
      }
    ]
  }
}
```

Rückgabe: Siehe `GET /accounts/:id`

Die `mandate_id` in jedem Objekt der `users` Liste muss immer gesetzt sein und kann über die `/mandates` API ermittelt werden.

Gruppen können über ihre ID bestimmt werden, die in einem Objekt wie im Beispiel übermittelt werden. Die IDs der Gruppen können über die `/groups/` API ermittelt werden.

Beim Erstellen eines Kontos muss mindestens ein Benutzer in der `users` Liste enthalten sein.

Je nach angegebener `auth_method` ergeben sich folgende Pflichtfelder:

Feld	`auth_method`
`name`	immer
`auth_method`	immer
`users`	immer
`email`	`email_password`

19.6.5 PUT /accounts/:id

Alternative Routen:

- `PUT /accounts/by-email/:email`
- `PUT /accounts/by-api-key/:api_key`

Ändert bestimmte Felder des Kontos mit der gegebenen ID.

Beispiel

```
PUT /api/v1/accounts/cb25d9ae-4e4d-4353-ac9e-20a88d7ad149?auth=xyz
Content-Type: application/json

{
```

```
  "account": {
    "name": "Peter",
    "language": "fr"
  }
}
```

Rückgabe: Siehe `GET /users/:id`

Felder, die nicht angegeben werden, werden auch nicht geändert.

Sollen einzelne Nutzer geändert werden, müssen alle anderen Nutzer zumindest mit ihrer ID aufgeführt werden, weil sie ansonsten gelöscht werden. Das gleiche gilt auch, wenn dem Konto ein neuer Benutzer hinzugefügt werden soll.

```
PUT /api/v1/accounts/cb25d9ae-4e4d-4353-ac9e-20a88d7ad149?auth=xyz
Content-Type: application/json

{
  "account": {
    "users": [
      {"id": "9ff43afb-fb61-4809-b65e-3453885f56ef"},
      {"id": "6049ca4c-b4f3-438c-9ae2-a5be5e951ea0", "role": "admin"},
      {"role": "read_only", "mandate_id": "409402fc-9108-44e4-8260-070
          d8e814870"}
    ]
  }
}
```

Angenommen, die beiden user 9ff43afb… und 6049ca4c… würden bereits im Konto existieren, so würde bei dem 2. (6049ca4c…) die Rolle geändert, die Gruppen und der Mandant jedoch unangetastet bleiben.

Das dritte Objekt in der users Liste würde dafür sorgen, dass ein neuer Nutzer im Mandant 409402fc… angelegt werden würde.

19.6.6 GET /accounts/:id

Alternative Routen:

- `GET /accounts/by-email/:email`
- `GET /accounts/by-api-key/:api_key`

Gibt ein einzelnes Konto anhand seiner ID zurück.

Beispiel `GET /api/v1/accounts/cb25d9ae-4e4d-4353-ac9e-20a88d7ad149?auth=xyz`

Rückgabe:

```
{
  "body": {
    "users": [
      {
        "permissions": {
          "account": {
            "create": true,
            "delete": true,
            "read": true,
            "update": true
          },
          "action_button": {
            "create": true,
            "delete": true,
            "read": true,
            "update": true
          },
          //...
        },
        "groups": [],
        "global_access": true,
        "updated_at": "2021-10-20T14:29:16.343414Z",
        "inserted_at": "2021-10-20T14:29:16.343414Z",
        "uses_limit": true,
        "account_id": "6d0deab5-42b8-44cc-a179-a1c7b752ac89",
        "mandate_id": null,
        "role": "admin",
        "id": "fddbc77d-5a87-4311-8e16-4b07d42ce9ad"
      }
    ],
    "avatar_url": "https://www.gravatar.com/avatar/
       f110a5b8feacea25275521f4efd0d7f2?d=retro&s=40",
    "updated_at": "2021-10-20T14:29:16.339000Z",
    "inserted_at": "2021-08-30T11:28:08.271262Z",
    "allowed_origins": null,
    "public_api_key": null,
    "password_changed_at": "2021-10-13T14:08:48.089430Z",
    "confirmed_at": null,
    "email": "a@b.com",
    "flags": {
      "actions_show_create": true,
      "custom_expr_tag": "inserted_at",
      "sidebar_width": 299
    },
    "auth_method": "email_password",
    "language": "en",
    "name": "Test Nutzer",
    "id": "6d0deab5-42b8-44cc-a179-a1c7b752ac89"
  },
  "ok": true,
```

```
  "status": 200
}
```

Die Routen zum Erstellen und Ändern geben den Nutzer im selben Schema zurück.

19.6.7 DELETE /accounts/:id

Alternative Routen:

- `DELETE /accounts/by-email/:email`
- `DELETE /accounts/by-api-key/:api_key`

Löscht das Konto mit der gegebenen ID.

Beispiel: `DELETE /api/v1/accounts/cb25d9ae-4e4d-4353-ac9e-20a88d7ad149?auth=xyz`

Rückgabe:

```
{
  "ok": true,
  "status": 200
}
```

19.7 Mandanten-API

Mit der Mandanten-API können für den API-Schlüssel sichtbare Mandanten gelistet, bearbeitet erstellt und gelöscht werden.

19.7.1 Felder eines Mandanten

Feld	Beschreibung	Bearbeiten?
`id`	Automatisch erzeugte, eindeutige ID für den Mandanten.	nein
`name`	Voller Name des Mandanten, wie er in der UI angezeigt wird	ja
`slug`	Kürzel des Mandanten, wie er zB in der URL der Webapp gezeigt wird	ja
`default_language`	Standardsprache, die neuen Nutzern, die in diesem Mandanten hinzugefügt als `selected_language` zugewiesen wird	ja

Feld	Beschreibung	Bearbeiten?
`support_email`	Emailadresse, die in der UI im Hilfemenü unter “Support” hinterlegt ist	ja
`theme`	Farbschema in das die UI gehüllt wird, wenn man sich in dem Mandanten befindet	ja
`email_whitelist`	Liste von Emailadressen, an die Regeln Emails senden dürfen	ja
`limit_notifications`	Liste von Limits betreffenden Ereignissen, die eine Warnemail auslösen. Gültige Ereignisse sind: `rate_limit_exceeded` und `quota_exceeded`.	ja
`limit_notification_email`	Emailadresse, an die solche Limitüberschreitungsbenachrichtigungen gesendet werden	ja
`default_position`	Beschreibt die Startposition von Karten in dem Mandanten, sofern noch keine anderen Daten vorliegen (GeoJSON Point)	ja
`updated_at`	Wann wurde der Mandant das letzte Mal bearbeitet	nein
`inserted_at`	Wann wurde der Mandant erzeugt	nein

19.7.2 GET /mandates

Listet alle für den API-Schlüssel sichtbaren Mandanten auf. Bei direkt an einen Mandanten gebundenen Schüsseln ist das immer nur einer, ansonsten alle Mandanten auf der ELEMENT IoT Instanz.

Beispiel `GET /api/v1/mandates?auth=xyz`


```
{
    "body": [
        {
            "id": "b5e858f2-9d47-4298-97c4-db55a8034434",
            "inserted_at": "2021-08-23T13:07:02.136824Z",
            "name": "Moritz Spielplatz",
            "slug": "moritz",
            "updated_at": "2021-08-23T13:07:02.136824Z"
        },
        {
            "id": "7daa3bc4-7ed3-4f7e-b3b3-a3d898ffee23",
            "inserted_at": "2021-05-25T08:51:42.192205Z",
            "name": "birdz",
            "slug": "birdz",
```

```
            "updated_at": "2021-05-25T08:51:42.192205Z"
        }
    ],
    "ok": true,
    "retrieve_after_id": "7daa3bc4-7ed3-4f7e-b3b3-a3d898ffee23",
    "status": 200
}
```

19.7.3 POST /mandates

Erstellt einen neuen Mandanten.

Beispiel:

```
POST https://stage.element-iot.com/api/v1/mandates?auth=83
   f1c2c9869030688133f7fc4530
Content-Type: application/json

{
  "mandate": {
    "name": "Mandate Name",
    "slug": "mandate-slug",
    "default_language": "en"
  }
}
```

Antwort:

```
{
  "body": {
    "id": "88bb3162-2a0e-44ed-9c60-0d0635e4b154",
    "inserted_at": "2021-08-24T09:30:05.388235Z",
    "name": "Mandate Name",
    "slug": "mandate-slug",
    "updated_at": "2021-08-24T09:30:05.388235Z"
  },
  "ok": true,
  "status": 200
}
```

19.7.4 PUT /mandates/:id

Ändert einen bestehenden Mandanten.

Beispiel:

```
PUT /api/v1/mandates/88bb3162-2a0e-44ed-9c60-0d0635e4b154?auth=xyz
```

```
Content-Type: application/json
```

Antwort:


```
{
  "body": {
    "id": "88bb3162-2a0e-44ed-9c60-0d0635e4b154",
    "inserted_at": "2021-08-24T09:30:05.388235Z",
    "name": "Mandate Name",
    "slug": "blub",
    "updated_at": "2021-08-24T12:29:58.339427Z"
  },
  "ok": true,
  "status": 200
}
```


19.8 Limits-API

Viele Features können Mandantenscharf in ihrer Anzahl oder Verfügbarkeit limitiert werden. Mit der entsprechenden API kann das automatisch geschehen.

Limits funktionieren immer additiv, wenn ein Limit nicht gesetzt ist, bedeutet das immer, dass die Funktion nicht zur Verfügung steht.

19.8.1 Felder eines Limits

Feld	Beschreibung
`id`	Automatisch erzeugte, eindeutige ID für das Limit
`category`	Dieses Feld bestimmt die Funktion, die beschränkt werden soll
`category_info`	Ein Objekt das die Kategorie beschreibt
-> `display`	Anzeigename für die Kategorie
-> `type`	Typ der Kategorie
-> `hint`	Hilfetext mit weiteren Infos zu der Kategorie
`mandate_id`	ID des Mandanten für den dieses Limit gilt
`enabled`	Wenn die Kategorie ein `feature` ist, entscheidet dieses Feld, ob das Feature zur Verfügung steht

Feld	Beschreibung
`quota`	Wenn die Kategorie eine `quota` ist, können von diesem Typ nur so viele Elemente angelegt werden, wie in diesem Feld stehen. `unlimited` Kann auch gesetzt werden, um diese Kategorie nicht zu limitieren
`rate_limit`	Wenn die Kategorie ein `rate_limit` ist, kann hier die Grösse des normalen Buckets eingestellt werden
`burst_limit`	Wie oben, nur ein zweites, kurzfristigeres Ratenlimit, um peaks zu vermeiden.
`last_notification_sent`	Datum, an dem das letzte Mal eine Benachrichtigung über das Überschreiten dieses Limits gesendet wurde.
`updated_at`	Datum, an dem das Limit zuletzt bearbeitet wurde
`inserted_at`	Datum an dem das Limit erstellt wurde

19.8.2 Verfügbare Kategorien

Action buttons (*quota*)

```
["Elixir.Platform.ActionButton"]
```

Emails from rules (*rate_limit*)

```
["Elixir.Platform.Rule.Action.SendNotification","send_email"]
```

Generic UDP (*quota*)

```
["Elixir.Platform.Driver.Instance","Elixir.Platform.Drivers.UDP"]
```

Sigfox (*quota*)

```
["Elixir.Platform.Driver.Instance","Elixir.Platform.Drivers.SigfoxHttp"]
```

Bridged devices (*quota*)

```
["Elixir.Platform.Driver.Instance","Elixir.Platform.Drivers.Bridged"]
```

Loriot (*quota*)

```
["Elixir.Platform.Driver.Instance","Elixir.Platform.Drivers.LoriotHttp"]
```

Generic HTTP Endpoint (*quota*)

```
["Elixir.Platform.Driver.Instance","Elixir.Platform.Drivers.HttpRest"]
```

Manual (*quota*)

```
["Elixir.Platform.Driver.Instance","Elixir.Platform.Drivers.Manual"]
```

Detailed history of actions (*feature*) Allows showing the full list of events on actions.

```
["Elixir.Platform.Driver.Action","events_ui"]
```

ThingPark by Actility (*quota*)

```
["Elixir.Platform.Driver.Instance","Elixir.Platform.Drivers.
   ActilityThingPark"]
```

Cancel actions after duration (*feature*) Enables an additional field 'cancel_after', which will automatically cancel that action if it is not in a terminal state when the duration expires.

```
["Elixir.Platform.Driver.Action","cancel_after"]
```

Mandates (*quota*)

```
["Elixir.Platform.Mandate"]
```

Dummy Generator (*quota*)

```
["Elixir.Platform.Driver.Instance","Elixir.Platform.Drivers.DummyGenerator
   "]
```

Parking Pilot by Smart City System GmbH (*quota*)

```
["Elixir.Platform.Driver.Instance","Elixir.Platform.Drivers.ParkingPilot"]
```

MQTT Broker (*quota*)

```
["Elixir.Platform.ExternalSystem.Schema","Elixir.Platform.ExternalSystem.
   MQTTBroker"]
```

ELEMENT IoT (*quota*)

```
["Elixir.Platform.Driver.Instance","Elixir.Platform.Drivers.ElementProxy"]
```

Gateway Management by ZIS (*quota*)

```
["Elixir.Platform.Driver.Instance","Elixir.Platform.Drivers.
   GatewayManagement"]
```

Users (*quota*)

```
["Elixir.Platform.User"]
```

Parsers (*quota*)

```
["Elixir.Platform.Parsing.Parser"]
```

Chirpstack by Orne Brocaar (*quota*)

```
["Elixir.Platform.Driver.Instance","Elixir.Platform.Drivers.Chirpstack"]
```

LNS Multicast groups (*quota*)

```
["Elixir.Platform.Drivers.ElementLns","multicast_groups"]
```

Total API requests (*rate_limit*)

```
["Elixir.Platform.Web.API","requests"]
```

Override actions (*feature*) Enables additional checkbox 'override', which will cancel all other actions that are not in a terminal state, if set.

```
["Elixir.Platform.Driver.Action","override"]
```

Rules (*quota*)

```
["Elixir.Platform.Rule"]
```

API Keys (*quota*)

```
["Elixir.Platform.APIKey"]
```

ELEMENT LNS by ZIS (*quota*)

```
["Elixir.Platform.Driver.Instance","Elixir.Platform.Drivers.ElementLns"]
```

Wireless-M-Bus (*quota*)

```
["Elixir.Platform.Driver.Instance","Elixir.Platform.Drivers.WmbusBridge"]
```

Views on readings (*quota*)

```
["Elixir.Platform.View","Elixir.Platform.Parsing.Reading"]
```

Profiles (*quota*)

```
["Elixir.Platform.Profile"]
```

Send files in Email (*quota*)

```
["Elixir.Platform.ExternalSystem.Schema","Elixir.Platform.ExternalSystems.
    EmailFileSystem"]
```

Scheduled actions (*quota*)

```
["Elixir.Platform.Driver.Action.Schedule"]
```

TrackCentral (discontinued) by TrackNet (*quota*)

```
["Elixir.Platform.Driver.Instance","Elixir.Platform.Drivers.
    TracknetWebsocket"]
```

Reading map presets (*quota*)

```
["Elixir.Platform.MapLayers"]
```

Device templates (*quota*)

```
["Elixir.Platform.Device.Template"]
```

WebSocket connections (*rate_limit*) Opening a WebSocket API connection is more expensive than API requests so these are rate limited separately

```
["Elixir.Platform.Web.API.WebSocket.GenericSocket","open"]
```

Publish to MQTT (*quota*)

```
["Elixir.Platform.OutputDriver.Schema","Elixir.Platform.OutputDriver.
    PublishMQTT"]
```

Gateway Management by Hessware (*quota*)

```
["Elixir.Platform.Driver.Instance","Elixir.Platform.Drivers.HesswareGms"]
```

Generic MQTT Client (*quota*)

```
["Elixir.Platform.Driver.Instance","Elixir.Platform.Drivers.MQTTClient"]
```

Reading graph presets (*quota*)

```
["Elixir.Platform.GraphPreset"]
```

Scheduled reading export (*quota*)

```
["Elixir.Platform.OutputDriver.Schema","Elixir.Platform.OutputDriver.
    ExportView"]
```

Concurrent WebSocket API connections (*quota*) Limits how many WebSocket API connections can exist at the same time

```
["Elixir.Platform.Web.API.WebSocket.GenericSocket","concurrent_connections
    "]
```

Streams (*quota*)

```
["Elixir.Platform.Rabbit.Stream"]
```

Packet API write operations (*feature*) Enables the create, update and delete endpoints for the packets API.

```
["api_packets_writeable"]
```

Devices (*quota*)

```
["Elixir.Platform.Device"]
```

SFTP (SSH) (*quota*)

```
["Elixir.Platform.ExternalSystem.Schema","Elixir.Platform.ExternalSystem.
    SFTP"]
```

19.8.3 GET /mandates/:mandate_id/limits

Fragt die aktuell auf den Mandanten mit der gegebenen ID gesetzten Limits ab

Beispiel `GET /api/v1/mandates/dfddd4de-9ffd-4c60-8110-16cc9c5464f8/limits?auth=xyz`

Antwort


```
[
  {
    "category_info": {
      "display": "Packet API write operations",
      "hint": "Enables the create, update and delete endpoints for the
         packets API.",
      "type": "feature"
    },
    "updated_at": "2020-05-26T14:36:27.957816Z",
    "inserted_at": "2020-05-26T14:36:27.957816Z",
    "mandate_id": "dfddd4de-9ffd-4c60-8110-16cc9c5464f8",
    "last_notification_sent": null,
    "enabled": true,
    "burst_limit": {
      "bucket_duration": 3600000,
      "bucket_size": 0
    },
    "rate_limit": {
```

```
      "bucket_duration": 3600000,
      "bucket_size": 0
    },
    "quota": 0,
    "category": [
      "api_packets_writeable"
    ],
    "id": "4a1a2cbd-6c99-447e-806c-29016ae2219b"
  },
  {
    "category_info": {
      "display": "Action buttons",
      "type": "quota"
    },
    "updated_at": "2020-05-20T08:59:24.097166Z",
    "inserted_at": "2020-04-30T11:24:04.501149Z",
    "mandate_id": "dfddd4de-9ffd-4c60-8110-16cc9c5464f8",
    "last_notification_sent": null,
    "enabled": true,
    "burst_limit": {
      "bucket_duration": 10000,
      "bucket_size": 500
    },
    "rate_limit": {
      "bucket_duration": 60000,
      "bucket_size": 10000
    },
    "quota": "unlimited",
    "category": [
      "Elixir.Platform.ActionButton"
    ],
    "id": "83269d8f-b67f-412b-81ad-60d696208db5"
  },
  {
    "category_info": {
      "display": "API Keys",
      "type": "quota"
    },
    "updated_at": "2020-05-20T08:59:24.096199Z",
    "inserted_at": "2018-05-25T08:40:24.403470Z",
    "mandate_id": "dfddd4de-9ffd-4c60-8110-16cc9c5464f8",
    "last_notification_sent": null,
    "enabled": true,
    "burst_limit": {
      "bucket_duration": 10000,
      "bucket_size": 500
    },
    "rate_limit": {
      "bucket_duration": 60000,
      "bucket_size": 10000
    },
```

```
    "quota": "unlimited",
    "category": [
      "Elixir.Platform.APIKey"
    ],
    "id": "9be37888-1ca7-45c4-9543-4ed2040261f0"
  }
]
```

19.8.4 Limits schreiben

Es gibt 2 Möglichkeiten, Limits in einen Mandanten zu setzen:

- Vorige Limits löschen und die gegebene Liste schreiben (mit `POST /mandates/:id/limits`)
- Vorige Liste an Limits erhalten, aber Elemente mit der gleichen Kategorie ersetzen (mit `PUT /mandates/:id/limits`)

Bis auf die HTTP-Methode unterscheiden sich die Anfragen und die Antworten dieser API nicht.

Beispiel

```
POST /api/v1/mandates/b5e858f2-9d47-4298-97c4-db55a8034434/limits?auth=xyz

Content-Type: application/json

{
  "limits": [{
    "category": ["Elixir.Platform.Device"],
    "quota": 5
  }]
}
```

Antwort

```
[
  {
    "category_info": {
      "display": "Devices",
      "type": "quota"
    },
    "updated_at": "2021-08-24T13:09:49.216943Z",
    "inserted_at": "2021-08-24T13:09:49.216943Z",
    "mandate_id": "b5e858f2-9d47-4298-97c4-db55a8034434",
    "last_notification_sent": null,
    "enabled": null,
    "burst_limit": null,
    "rate_limit": null,
    "quota": 5,
    "category": [
```

```
      "Elixir.Platform.Device"
    ],
    "id": "afd6a0d6-1423-4971-b71f-89bf9db082c4"
  }
]
```

2. Anfrage

```
PUT /api/v1/mandates/b5e858f2-9d47-4298-97c4-db55a8034434/limits?auth=83
    f1c2c9869030688133f7fc4530
Content-Type: application/json

{
  "limits": [{
    "category": ["Elixir.Platform.Device"],
    "quota": 3
  },
  {
    "category": ["Elixir.Platform.Profile"],
    "quota": "unlimited"
  }]
}
```

Antwort

```
[
  {
    "category_info": {
      "display": "Devices",
      "type": "quota"
    },
    "updated_at": "2021-08-24T13:20:31.738678Z",
    "inserted_at": "2021-08-24T13:09:49.216943Z",
    "mandate_id": "b5e858f2-9d47-4298-97c4-db55a8034434",
    "last_notification_sent": null,
    "enabled": null,
    "burst_limit": null,
    "rate_limit": null,
    "quota": 3,
    "category": [
      "Elixir.Platform.Device"
    ],
    "id": "afd6a0d6-1423-4971-b71f-89bf9db082c4"
  },
  {
    "category_info": {
      "display": "Profiles",
      "type": "quota"
    },
    "updated_at": "2021-08-24T13:20:31.739915Z",
    "inserted_at": "2021-08-24T13:20:31.739915Z",
```

```
    "mandate_id": "b5e858f2-9d47-4298-97c4-db55a8034434",
    "last_notification_sent": null,
    "enabled": null,
    "burst_limit": null,
    "rate_limit": null,
    "quota": "unlimited",
    "category": [
      "Elixir.Platform.Profile"
    ],
    "id": "b8c7f97e-af08-4e27-9e41-bfede8f8c74e"
  }
]
```


19.9 Hilfsprogramme für API-Tests

Um die Beispiele im Folgenden nachvollziehen zu können, empfehlen wir die Nutzung eines entsprechenden Tools. In den folgenden Screenshots wird dafür Postman verwendet. Postman können Sie kostenlos downloaden: https://www.getpostman.com/

19.10 Beispiele für Geräte

Sie benötigen für die nächsten Beispiel einen API-Schlüssel und jeweils den “Slug” des Gerätes. Den Slug finden Sie zum Beispiel in der URL des Gerätes:

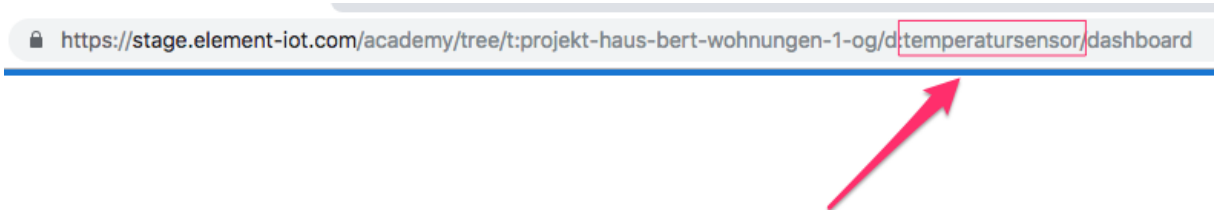


Abbildung 19.1: Screenshot

In diesem Beispiel wäre der Slug für das Gerät: **temperatursensor** , anstelle des Slug können Sie auch die ID des Gerätes verwenden.

Alle Beispiele sind identisch aufgebaut, es wird jeweils die entsprechende URL (Endpunkt), eine Beispiel-Abfrage anhand des Temperatursensors und ein Ergebnis gezeigt. Für diese API-Abfragen wird das Tool Postman genutzt.

Um Ihnen den Einstieg in die Nutzung von Postman zu erleichtern, hier ein Beispiel zur Abfrage einer

API mit Testdaten. Öffnen Sie Postman und wählen Sie die Option Request (die Registrierung nach dem Öffnen können Sie überspringen, diese ist nicht notwendig):

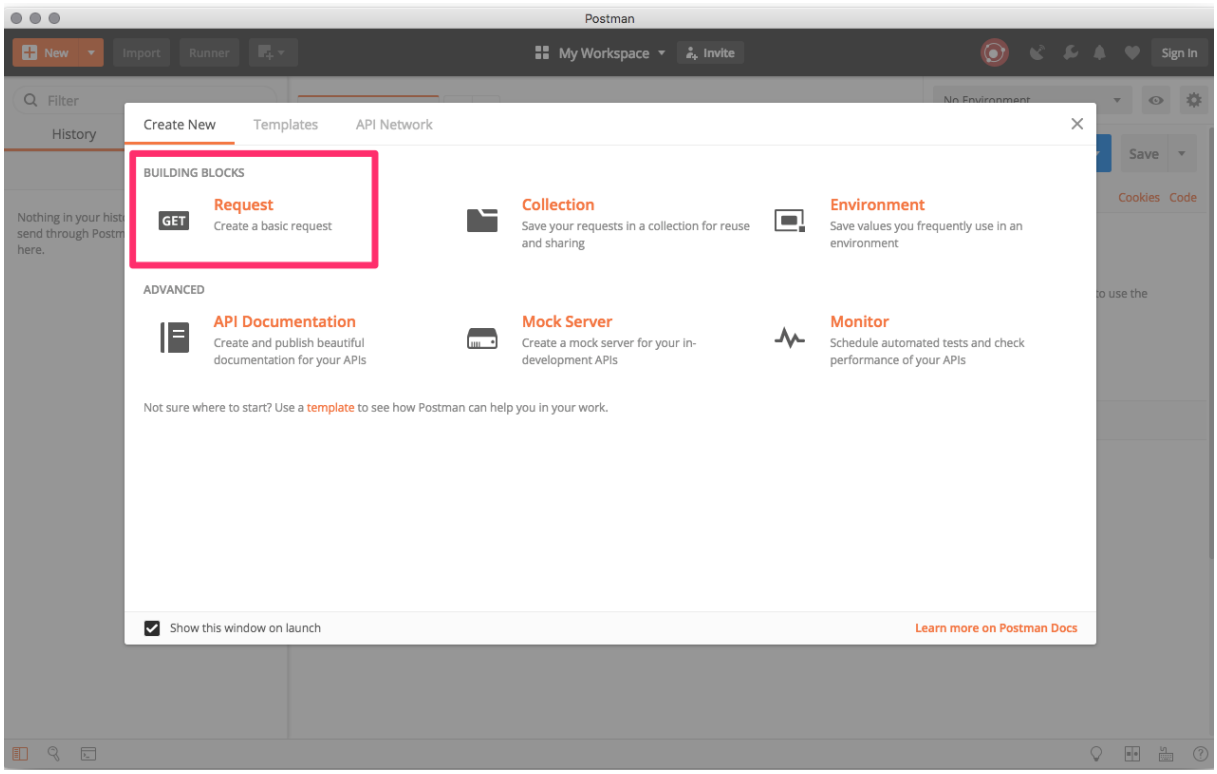


Abbildung 19.2: Screenshot

Vergeben Sie im nachfolgenden Dialog einen Namen (dieser bezeichnet eine Sammlung an Requests) und legen Sie einen entsprechenden Ordner an.

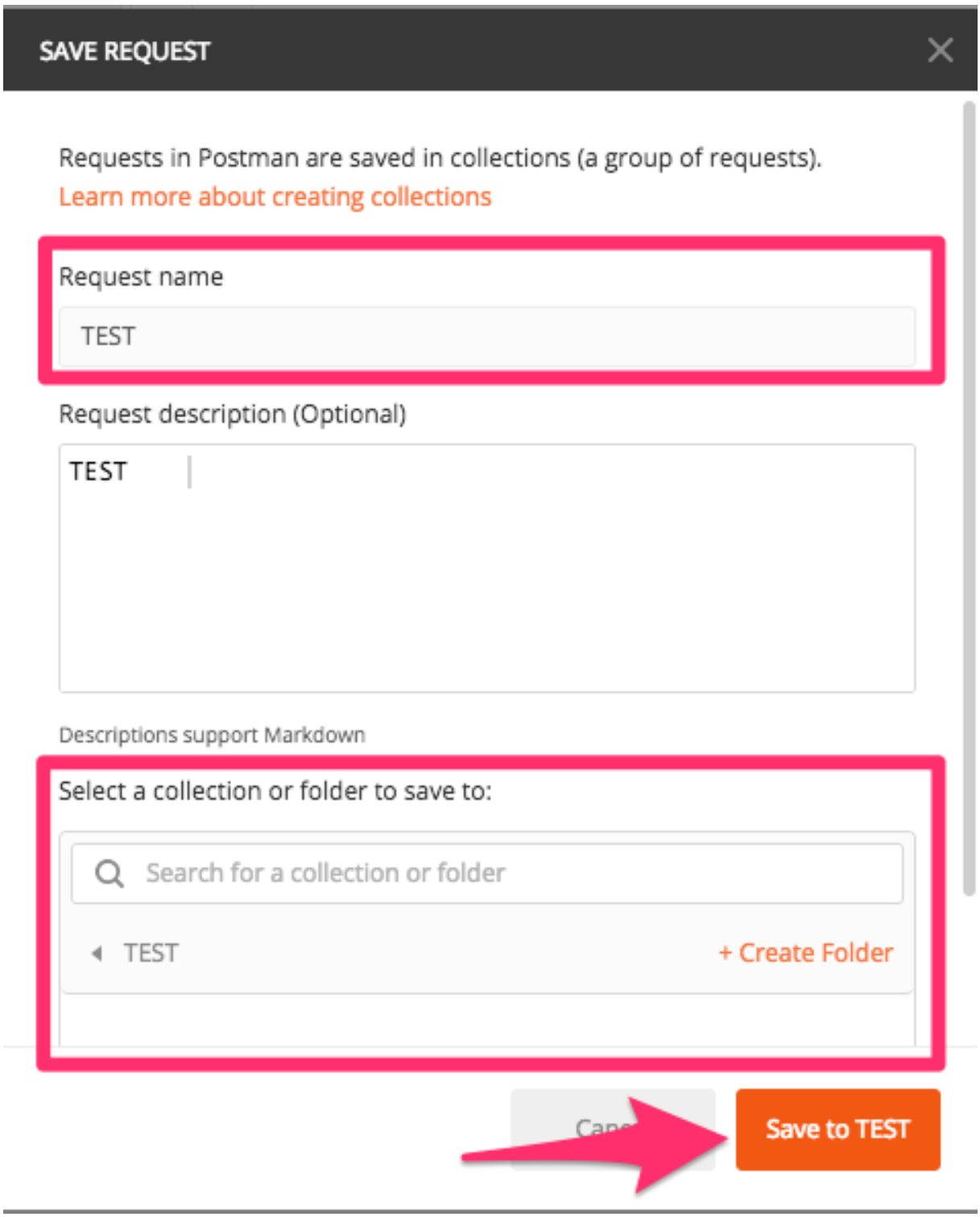


Abbildung 19.3: Screenshot

Nach dem Speichern können Sie ihre erste Abfrage absenden. Geben Sie hierzu in das URL Feld “https://jsonplaceholder.typicode.com/posts” ein und klicken Sie auf **SEND**. Im unteren Bereich sehen Sie jetzt das Ergebnis der Anfrage.

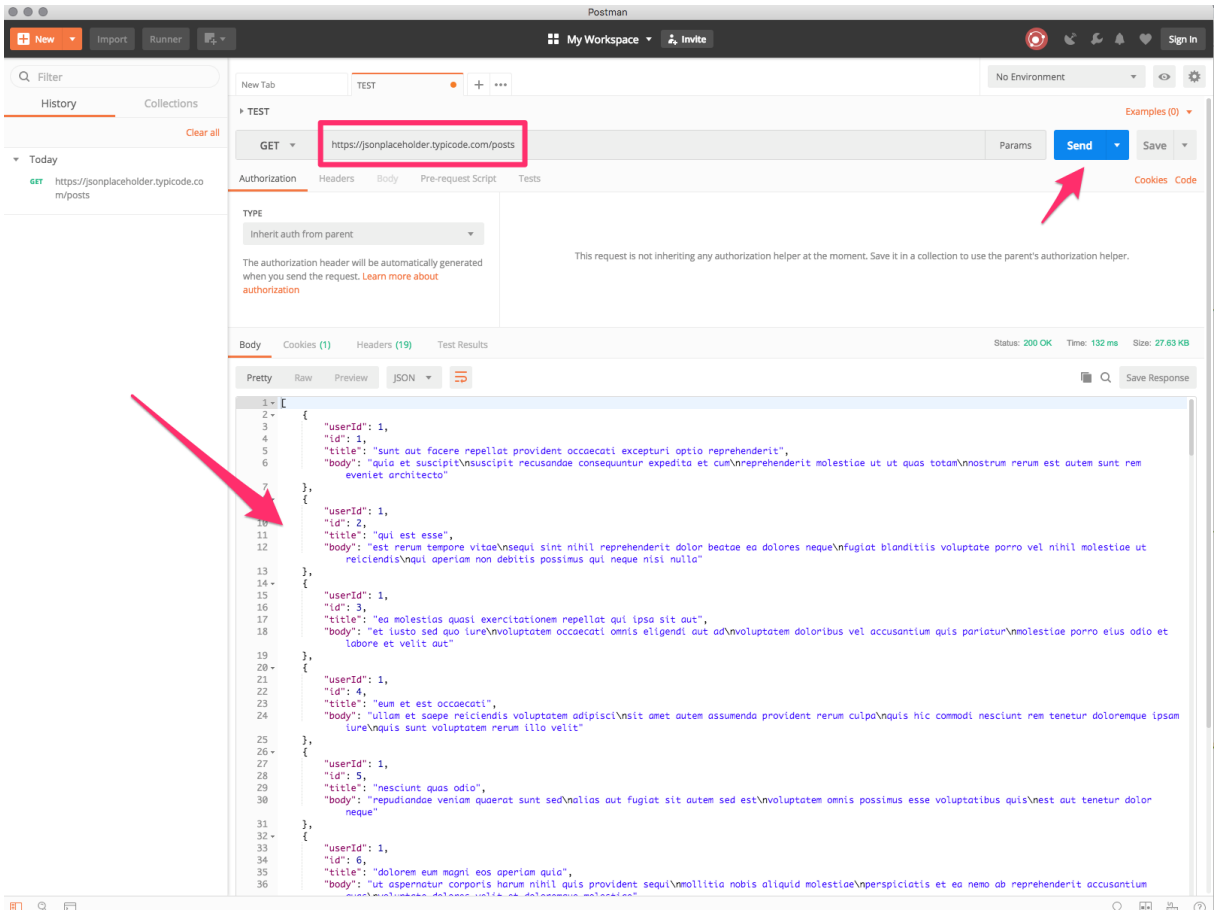


Abbildung 19.4: Screenshot

19.11 Beispiel: Informationen über ein Gerät abfragen

Sie erhalten alle Informationen welche Sie auch in der ELEMENT Oberfläche sehen, und können diese in anderen System oder Anwendungen verwenden.

In Postman geben Sie im URL Feld die folgende Adresse ein:

URL

```
https://element-iot.com/api/v1/devices/temperatursensor? \
auth=8774556701e370e409264025327a5c39
```

Bitte ersetzen Sie “8774556701e370e409264025327a5c39” durch Ihren eignen API-Schlüssel. Klicken Sie nun auf **SEND** um das Ergebnis der Abfrage zu erhalten.

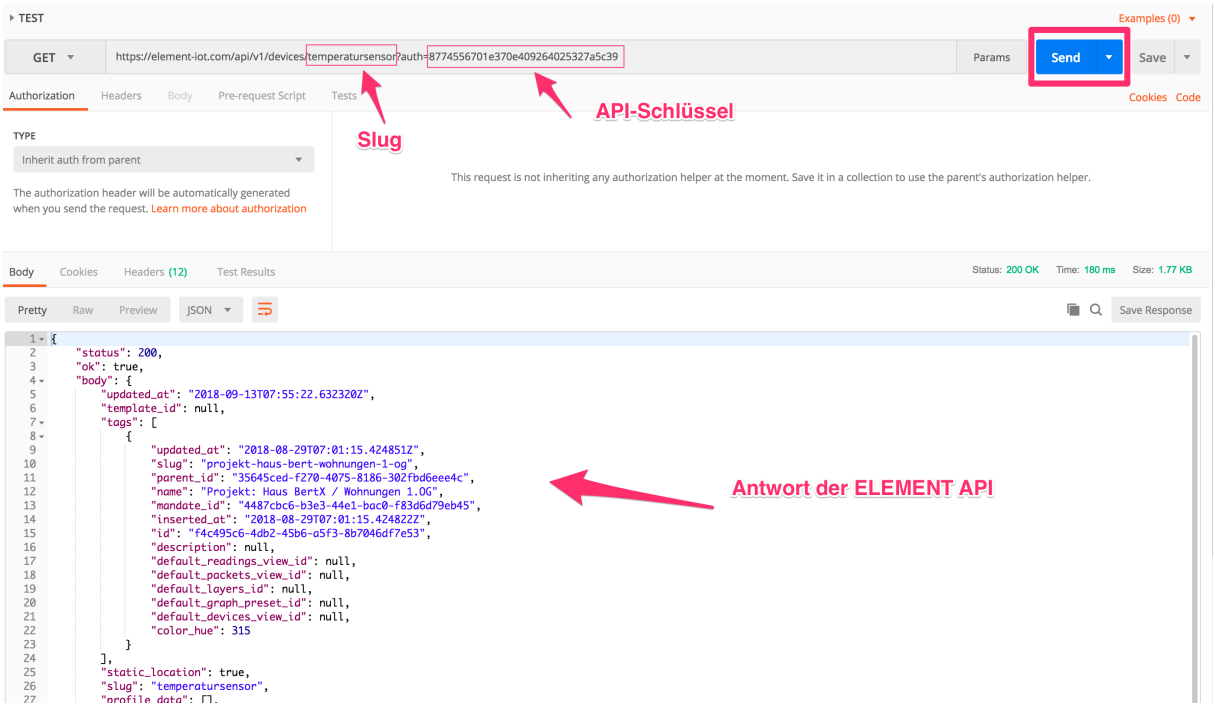


Abbildung 19.5: Screenshot

19.12 Beispiel: Abfrage der 50 letzten Messwerte eines Gerätes

Mit dieser Abfrage erhalten Sie die letzten 50 Messwerte des Gerätes.

URL

```
https://element-iot.com/api/v1/devices/temperatursensor/readings? \
auth=8774556701e370e409264025327a5c39
```

Bitte ersetzen Sie “8774556701e370e409264025327a5c39” durch Ihren eignen API-Schlüssel. Klicken Sie nun auf **SEND**, um das Ergebnis der Abfrage zu erhalten.

Wie Sie in dem Screenshot sehen können, bekommen Sie die selben Ergebnisse wie innerhalb der ELEMENT IoT Oberfläche und können diese Werte beliebig weiterverarbeiten.

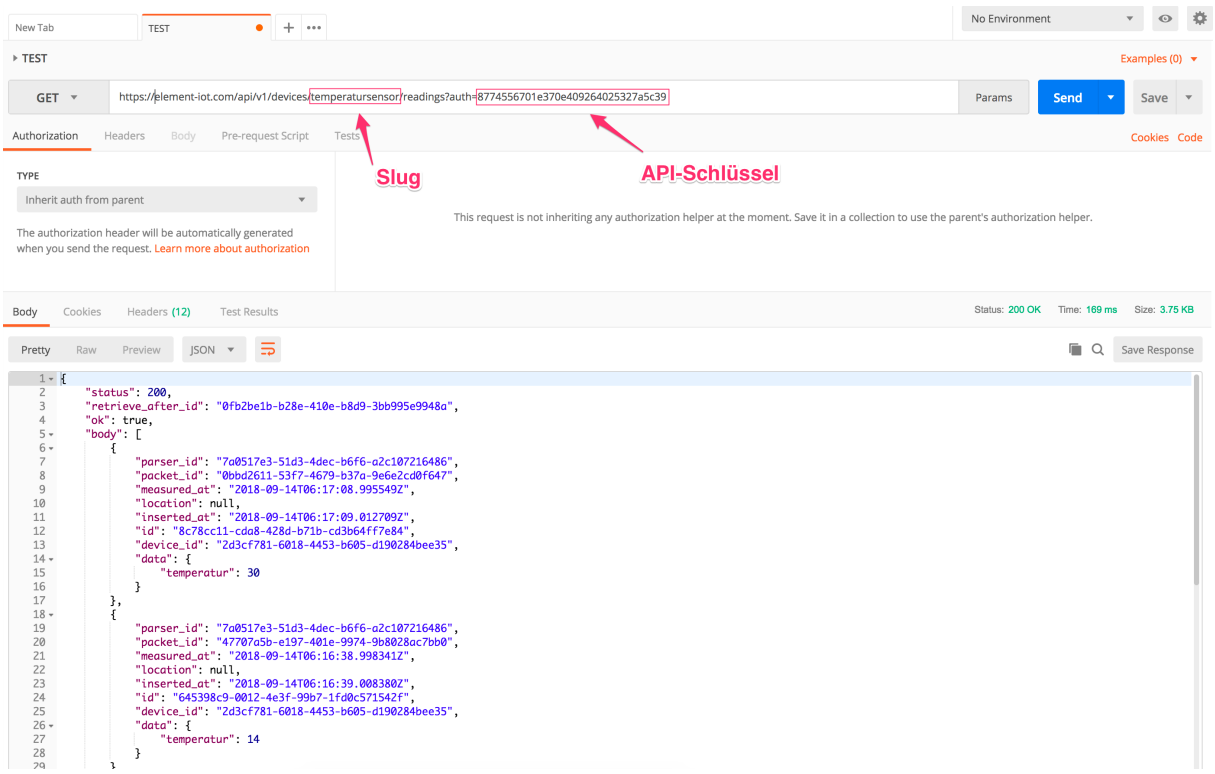


Abbildung 19.6: Screenshot

19.13 Beispiel: Alle Geräte aus einem Ordner abfragen

Sie können alle Informationen über Geräte, die sich innerhalb eines Ordners befinden, über die API Abfragen. Anstelle des Geräte-Slugs benötigen Sie nun den Slug des Ordners:

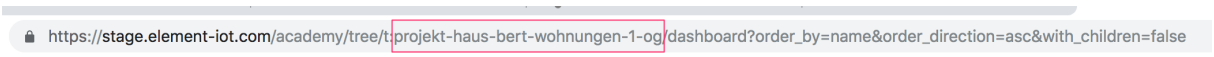


Abbildung 19.7: Screenshot

URL

```
https://element-iot.com/api/v1/tags/ \
projekt-haus-bert-wohnungen-1-og/devices?auth=8774556701
    e370e409264025327a5c39
```

Bitte ersetzen Sie “8774556701e370e409264025327a5c39” durch Ihren eignen API-Schlüssel. Klicken Sie nun auf **SEND**, um das Ergebnis der Abfrage zu erhalten.

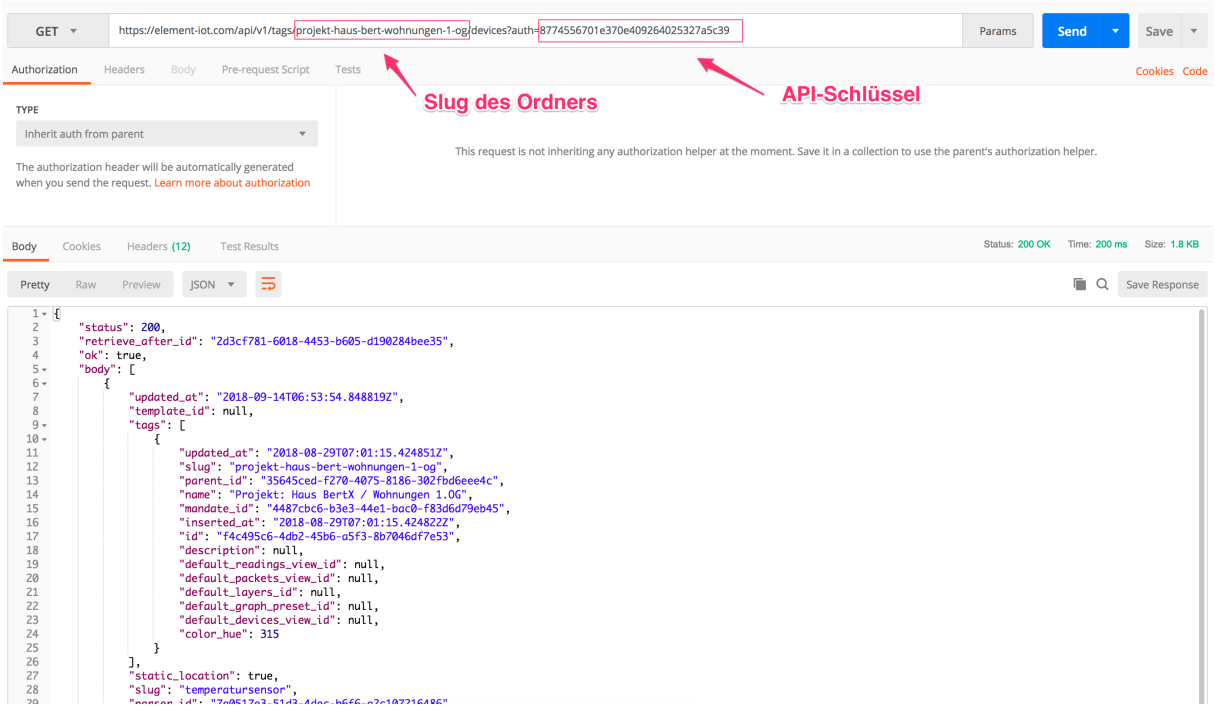


Abbildung 19.8: Screenshot

19.14 Beispiel: Abfragen der 20 letzten Pakete eines Gerätes

URL

```
https://element-iot.com/api/v1/devices/temperatursensor/packets?limit=20&
    \
sort=inserted_at&sort_direction=desc&auth=8774556701e370e409264025327a5c39
```

Bitte ersetzen Sie “8774556701e370e409264025327a5c39” durch Ihren eignen API-Schlüssel. Der Parameter “limit” gibt an, wieviele Pakete sie abfragen möchten. Klicken Sie nun auf **SEND**, um das Ergebnis der Abfrage zu erhalten.

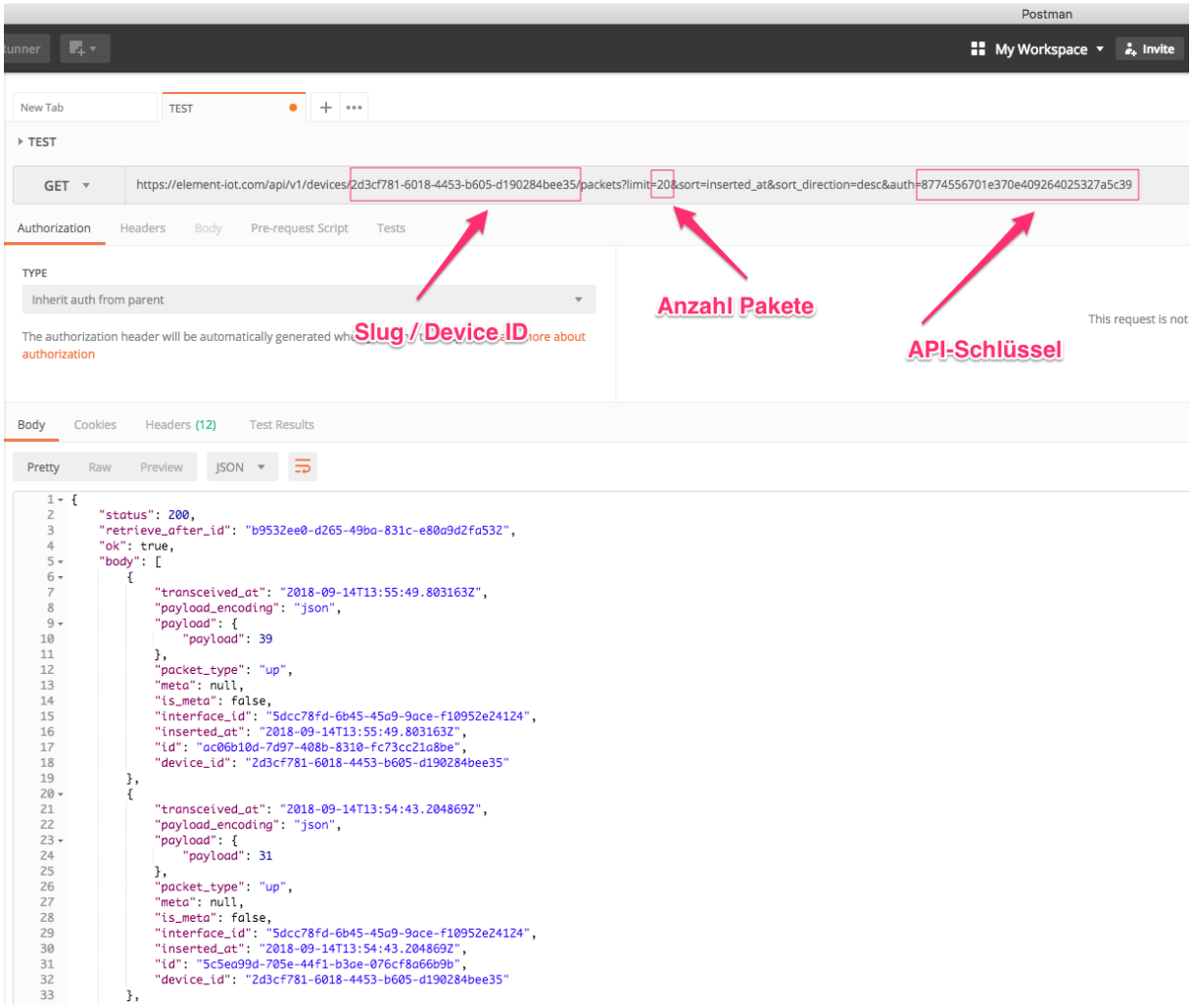


Abbildung 19.9: Screenshot

19.15 Praxisbeispiel

Anhand dieses kleinen PHP/HTML-Skriptes können Sie sehen, wie man die ELEMENT-API für eine einfach Webseite nutzen kann,welche die Temperatur des übergebenen Gerätes ausgibt. Das Beispiel dient nur der Veranschaulichung.

```
<?php
function do_request($deviceId, $apiKey)
{
    // ELEMENT API URL, wir übergeben die ID des Gerätes und den API-Schlü
       ssel
    $apiUrl = "https://stage.element-iot.com/api/v1/devices/$deviceId/
       readings?limit=10&sort=inserted_at&sort_direction=desc&auth=$apiKey
```

```
        ";

    $ch = curl_init();
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_URL, $apiUrl);

    // Als Rückgabeformat erwartet wir JSON
    curl_setopt($ch, CURLOPT_HTTPHEADER, array(
        'Content-Type: application/json',
        'Accept: application/json'
    ));

    $result = curl_exec($ch);
    curl_close($ch);
    return json_decode($result);
}

$apiKey = '8774556701e370e409264025327a5c39';
$deviceId = '2d3cf781-6018-4453-b605-d190284bee35';
$data = do_request($deviceId, $apiKey);
?>

<!doctype html>

<html lang="de">
<head>
  <meta charset="utf-8">
  <title>"ELEMENT IoT-Plattform - Messwerte</title>
  <meta name="description" content="ELEMENT IoT-Plattform - Messwerte">
  <meta name="author" content="ZENNER IoT Solutions GmbH">
</head>
    <body>
        <h1>Geräte ID: <?php echo $deviceId; ?></h1>
        <div>
            <ul>
            <?php
                foreach ($data->body as $v) {
                    echo '<li> Temperatur: ' . $v->data->temperatur . '
                        Grad Celsius </li>';
                }
            ?>
            </ul>
        </div>
    </body>
</html>
```

Abbildung 19.10: Screenshot

19.16 Websocket Verbindungen

Neben der REST API existiert auch noch eine WebSocket API. Diese ermöglicht ihnen Pakete und Messwerte in fast Echtzeit zu erhalten. Zum Testen empfiehlt sich die Nutzung eines entsprechenden WebSocket Client wie zum Beispiel der Google Chrome Erweiterung Simple Websocket Client.

URL

```
wss://element-iot.com/api/v1/devices/temperatursensor \
/packets/socket?auth=8774556701e370e409264025327a5c39
```

Bitte ersetzen Sie “8774556701e370e409264025327a5c39” durch Ihren eignen API-Schlüssel. Weitere Bespiele für das Aufbauen einer Websocket-Verbindung finden Sie im entsprechenden Gerät im Tab “API”.

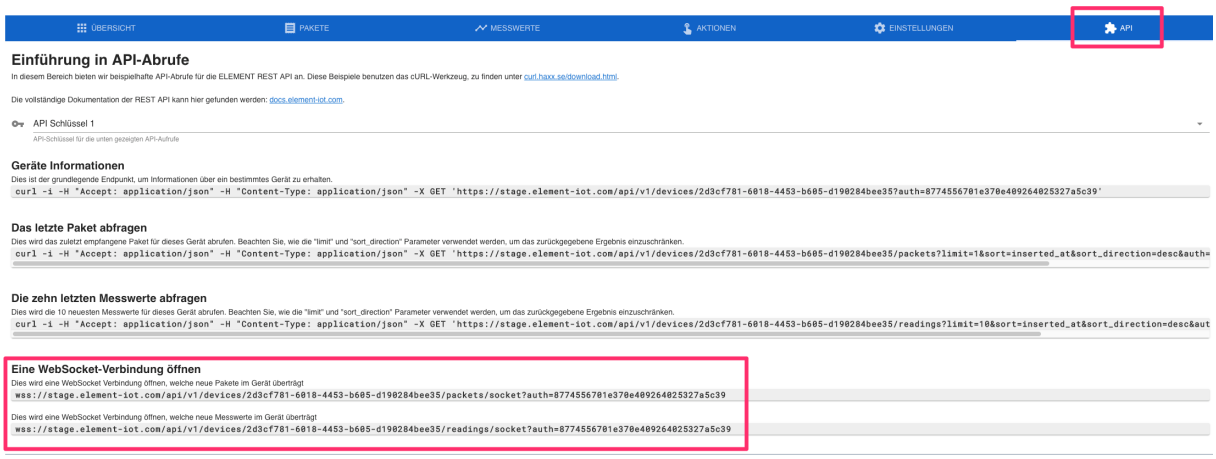


Abbildung 19.11: Screenshot

Geben Sie nun die entsprechende Websocket URL im Client an, wenn jetzt ein neues Paket eintrifft wird dieses umgehend angezeigt.

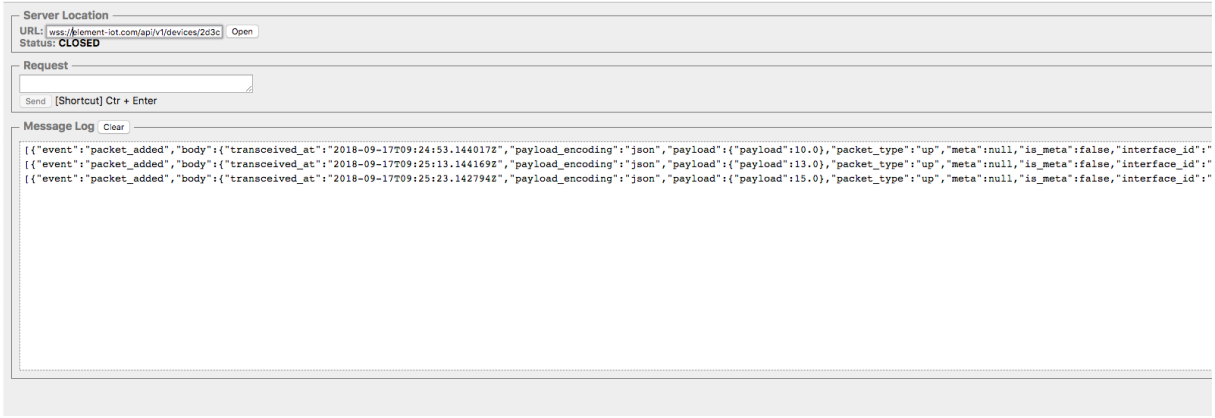


Abbildung 19.12: Screenshot

20 Lizenzen und Limits

Seit Version 2.18 verfügt ELEMENT über ein Limit-System.

Superadmins einer ELEMENT IoT Instanz können pro Mandant Limits vergeben sowie global über einige Limits entscheiden, die Einfluss auf die Performance haben.

Nutzern, die auf einen Mandanten beschränkt sind, werden die Limitierungen in Form von Popups angezeigt, wenn sie zum Beispiel versuchen, gesperrte Features zu benutzen oder zu viel von etwas anzulegen:

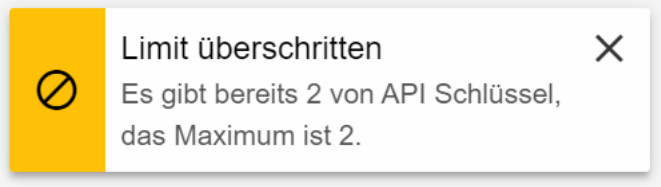


Abbildung 20.1: Popup, wenn Limit überschritten wurde

Darüber hinaus gibt es Lizenzen, die die Obergrenzen für die gesamte ELEMENT IoT Instanz definieren:

Abbildung 20.2: Eine beispielhafte Lizenz

20.1 Glossar

Im Kontext von Limits und Lizenzen werden oft folgende Begriffe genutzt:

- **Kategorie**

  Eine bestimmte Sache, die limitiert wird. Jede Kategorie hat einen Typ (Limit, Ratenlimit oder Funktion) und kann ausserdem unlizensiert sein.

  Einige Beispiele für Kategorien:

  - Geräte (Limit)

    Bestimmt die maximale Anzahl an Geräten, die angelegt werden können.

  - Alle API-Abfragen (Ratenlimit, unlizensiert)

    Bestimmt, wie oft innerhalb eines bestimmten Zeitraums die API genutzt werden kann.

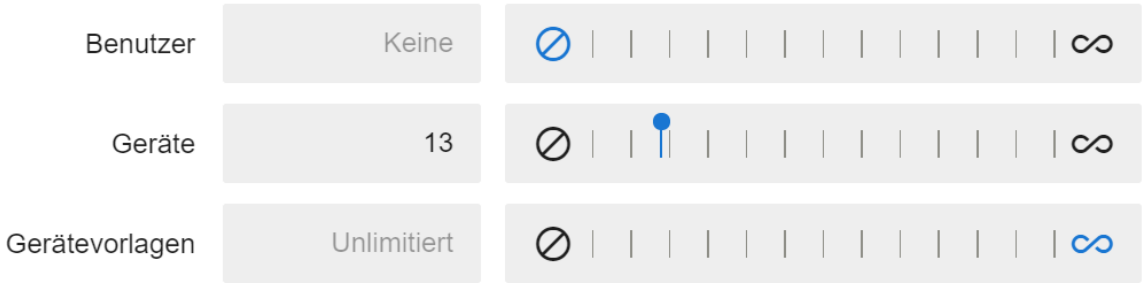
  - Gleichzeitige WebSocket API-Verbindungen (Limit, unlizensiert)

    Bestimmt, wie viele WebSocket API Verbindungen gleichzeitig geöffnet sein dürfen.

- **Limit** (Quota)

  Limitiert die Anzahl der jeweiligen Kategorie.

  Es kann auf der linken Seite eine genaue Zahl eingetippt werden, auf der rechten Seite ist ein Slider mit dem man schnell einige oft genutzte Werte zwischen 0 und unendlich einstellen kann.

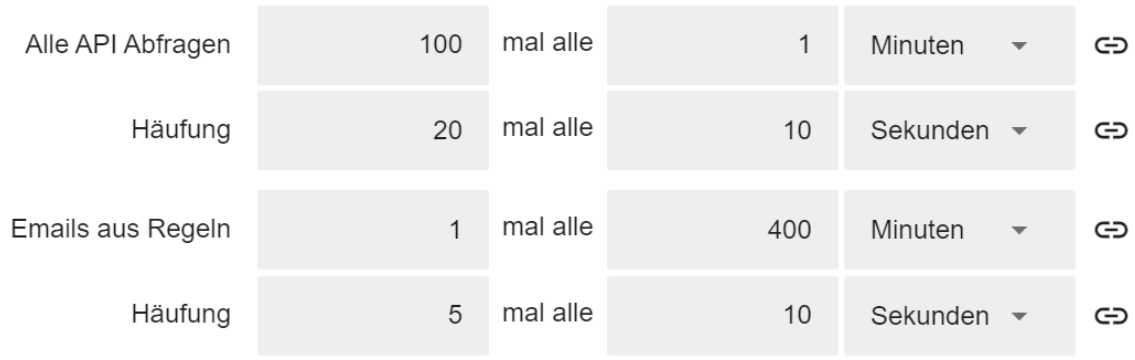


- **Ratenlimit** (Rate Limit)

  Limitiert die Frequenz von bestimmten Aktionen.

  Bei Ratenlimits erscheinen immer 2 Zeilen:

  - In der ersten Zeile soll das generelle Limit über einen längeren Zeitraum eingestellt werden.
  - In der zweiten, mit dem Wort “Burst” bzw. “Häufung” beschrifteten Zeile soll ein kleineres Limit mit einem kürzeren Zeitraum eingestellt werden, um zu verhindern, dass das gesamte Limit innerhalb von kürzester Zeit aufgebraucht werden kann.



Die einzelnen Eingabefelder von links nach rechts bedeuten:

1. **Grösse des Limits**

   Wie viele Aktionen der Kategorie dürfen im angegebenen Zeitraum maximal passieren?

2. **Dauer**

   Wie lange gilt ein Ratenlimit, bis der Verbrauch auf 0 zurückgesetzt wird?

3. **Einheit der Dauer**

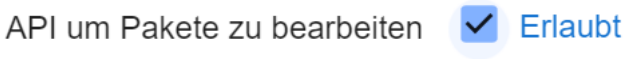
   Hier kann zwischen Sekunden, Minuten, Stunden und Tagen gewählt werden.

4. **Dauer-Wert belassen**

   Mit dem Verknüpfungssymbol kann umgeschaltet werden, ob bei Änderung der Einheit der Wert beibehalten werden soll oder der Wert in die andere Einheit umgerechnet werden soll

- **Funktion** (Feature Flag)

  Funktionen, bei denen es keinen Sinn ergibt, sie in der Rate oder Anzahl zu beschränken, werden mit dem Kategorietyp “Funktion” versehen, die man nur erlauben oder verbieten kann.

- **unlizensiert**

  Grundsätzlich müssen alle Kategorien, die in einer Instanz verwendet werden sollen, zunächst über eine Lizenz freigeschaltet werden. Es gibt aber einige Ausnahmen für Kategorien, bei denen es wichtig ist, dass man sie im laufenden Betrieb reduzieren kann, um z.B. auf Überlastungen bestimmter Systeme reagieren zu können.

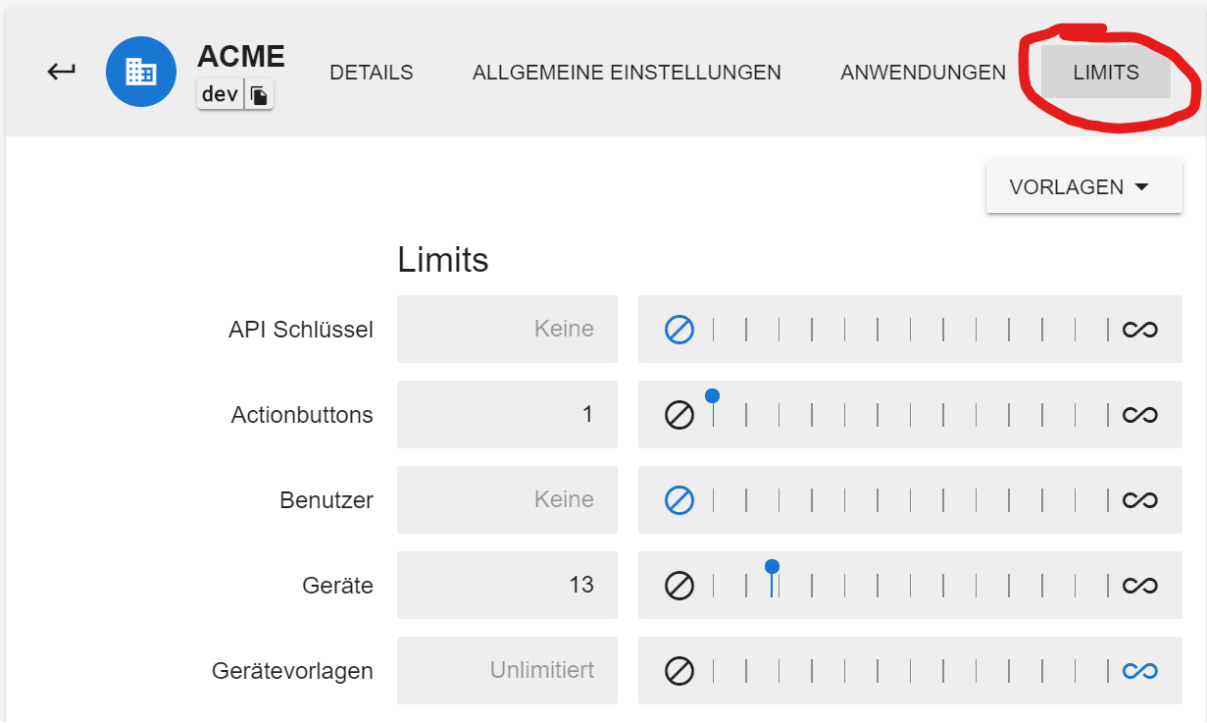
  Unlizensierte Kategorien können von Superadmins im Adminbereich unter `Limits` -> `Globale Limits` eingestellt werden.

20.2 Mandantenlimits

Superadmins einer ELEMENT IoT Instanz können die Limits für einen Mandanten frei einstellen.

Zu Beachten ist hierbei, dass es keinen Effekt hat, wenn in Mandanten bestimmte Kategorien höher freigeschaltet werden als es von den Lizenzen bzw. den globalen Limits her gedeckelt wurde.

Zum manuellen Verstellen der Limits in einem Mandanten navigiert man im Adminbereich zu dem gewünschten Mandanten und klickt dann oben auf den `Limits` Tab.



Über das Menü “Vorlagen” können zuvor definierte Listen von Limits in den Mandanten geladen werden. Wenn eine Vorlage gewählt wird, fragt ELEMENT nach, wie die Vorlage geladen werden soll. Dabei gibt es folgende Möglichkeiten:

- **Abbrechen**

  Die Vorlage doch nicht importieren.

- **Mischen**

  Wenn in diesem Mandanten bereits Kategorien konfiguriert sind, die auch in der Vorlage vorkommen, werden diese nach folgenden Regeln addiert:

  - Bei **Limits** werden die Werte addiert. Wenn mindestens einer unendlich ist, so ist auch das Ergebnis unendlich.

  - Bei **Ratenlimits** gewinnt das höchste Ratenlimit.
  - Bei **Funktionen** ist die Funktion danach dann aktiviert, wenn in der Vorlage oder im Mandanten die Funktion auch schon aktiviert war.

- **Überschreiben**

  Anders als bei **Mischen** werden hier bereits konfigurierte Kategorien mit den neuen Werten überschrieben, anstatt sie zu kombinieren.

- **Ersetzen**

  Setzt alle Kategorien zurück und überschreibt danach die Kategorien aus der Vorlage.

Zum Speichern der verstellten Limits kann danach unten auf **Speichern** geklickt werden.

20.3 Limitvorlagen

Mit Vorlagen können vordefinierte Listen von Limits, Ratenlimits und Funktionen gespeichert werden, um sie später in Mandanten-Limits oder dem Lizenzgenerator zu verwenden.

Zum Erstellen einer Vorlage navigiert man im Adminbereich zu `Limits` -> `Vorlagen` und klickt dann auf `Limitvorlagen`.

Im Editor kann ein **Name** vergeben werden, mit dem die Vorlage später identifiziert werden kann.

Ganz unten befindet sich immer ein Auswahlmenü, in dem nach der gewünschten Kategorie gesucht werden kann. Hier kann auch durch Eintippen einiger Zeichen gesucht sowie mit den hoch/runter Tasten navigiert und mit Enter ausgewählt werden.

Dann kann für diese Kategorie das Limit entsprechend eingestellt werden.

Sobald die Vorlage gespeichert ist, kann sie außerdem automatisch auf alle aktuell bestehenden Mandanten angewendet werden, indem man ganz oben auf **Auf alle Mandanten anwenden** klickt. Dabei gibt es 3 Optionen:

- **Abbrechen**

  Den Vorgang nicht ausführen.

- **Überspringen**

  Wenn für eine Kategorie in einem Mandanten bereits etwas definiert wurde, wird diese Kategorie im Limit übersprungen und nicht angewendet.

  Das heißt nur Kategorien, die noch nicht in dem Mandanten definiert wurden, werden gespeichert.

- **Ersetzen**

  Egal, ob eine Kategorie in einem Mandnaten schon definiert wurde, wird sie immer mit dem Wert aus der Vorlage überschrieben.

20.4 Globale Limits

Kategorien, die nicht lizensiert sind, können von Superadmins limitiert werden.

Dazu gehören folgende Kategorien:

- **Gleichzeitige WebSocket API-Verbindungen**

  Limitiert die maximale Anzahl an WebSocket *API* Verbindungen, die gleichzeitig bestehen dürfen. Da WebSocket API Verbindungen mit viel Durchsatz unter Umständen viel Datenbanklast verursachen können, kann hier angesetzt werden, falls die Datenbanklast zu hoch sein sollte.

- **Alle API Abfragen**

  Limitiert die Frequenz aller API-Abfragen. API Keys limitieren zwar jeder für sich bereits die Zugriffsfrequenz, jedoch kann es trotzdem zu Überlastungen kommen, wenn besonders viele API Keys gleichzeitig benutzt werden.

- **Emails aus Regeln**

  Sollten Sie einen Dienstleister für den E-Mail Versand benutzen, bietet es sich an, hier die Anzahl an E-Mails, die aus der Regel-Aktion “Benachrichtigung senden” heraus verschickt werden, zu begrenzen, so dass man nicht durch Überschreitung einer monatlichen Quota Mehrkosten verursacht.

- **WebSocket API Verbindungen**

  Limitiert die Frequenz mit der neue WebSocket API-Verbindungen geöffnet werden können. Diese Aktion wird gesondert abgerechnet, da beim WebSocket API Verbindungsaufbau kurzzeitig mehr Last verursacht wird, als bei einer einfachen API-Abfrage.

Wenn eine unlizensierte Kategorie nicht global limitiert wurde, ist sie grundsätzlich unlimitiert. Es ist nicht nötig, etwas einzustellen, wenn man nicht limitieren möchte.

20.5 Dashboard

Das Limit-Dashboard gibt einen Überblick über den Verbrauch von globalen Limits, Lizenzen und Mandantenlimits.

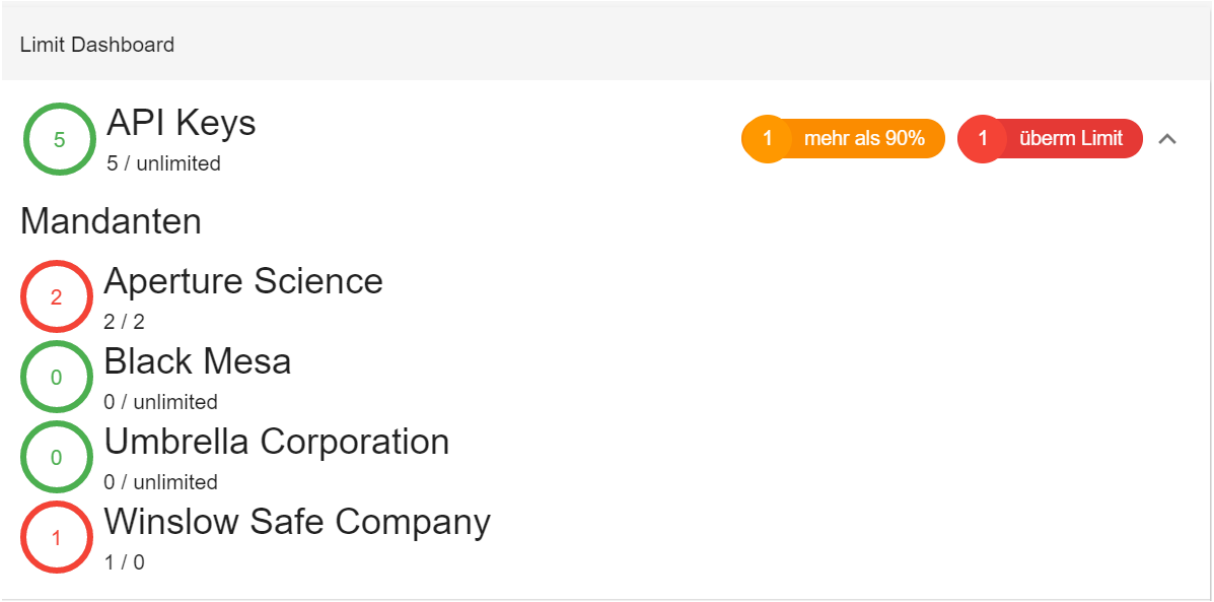


Abbildung 20.3: Das Limitdashboard mit aufgeklappter Kategorie “API Keys”

Zu jedem Limit ist der aktuelle Verbrauch und die Obergrenze sichtbar.

Bei den Ratenlimits sieht man an den beiden Kreisen wie es um das normale und das burst Limit steht.

Um eine einzelne Kategorie mandantenscharf zu betrachten, kann auf die Zeile geklickt werden, um sie auszuklappen. Dort ist dann für jeden Mandanten aufgelistet, wie viel von dessen Limit aufgebraucht ist.

Einen schnellen Überblick gewähren auch die Warnungen auf der rechten Seite jeder Kategorie, die anzeigen wie viele Mandanten bei dieser Kategorie am Limit bzw kurz davor sind.

21 Logging

ELEMENT IoT nutzt Logs um allerlei Informationen über den laufenden Betrieb mitzuteilen. Diese Logs werden in die Standardausgabe geschrieben, von dort werden die Logs je nach Infrastruktur in einen Log-Aggregator wie Graylog übertragen.

21.1 Logausgabe konfigurieren

Im Administrationsbereich findet man unter “Konfiguration” den Punkt Konsolenlog.

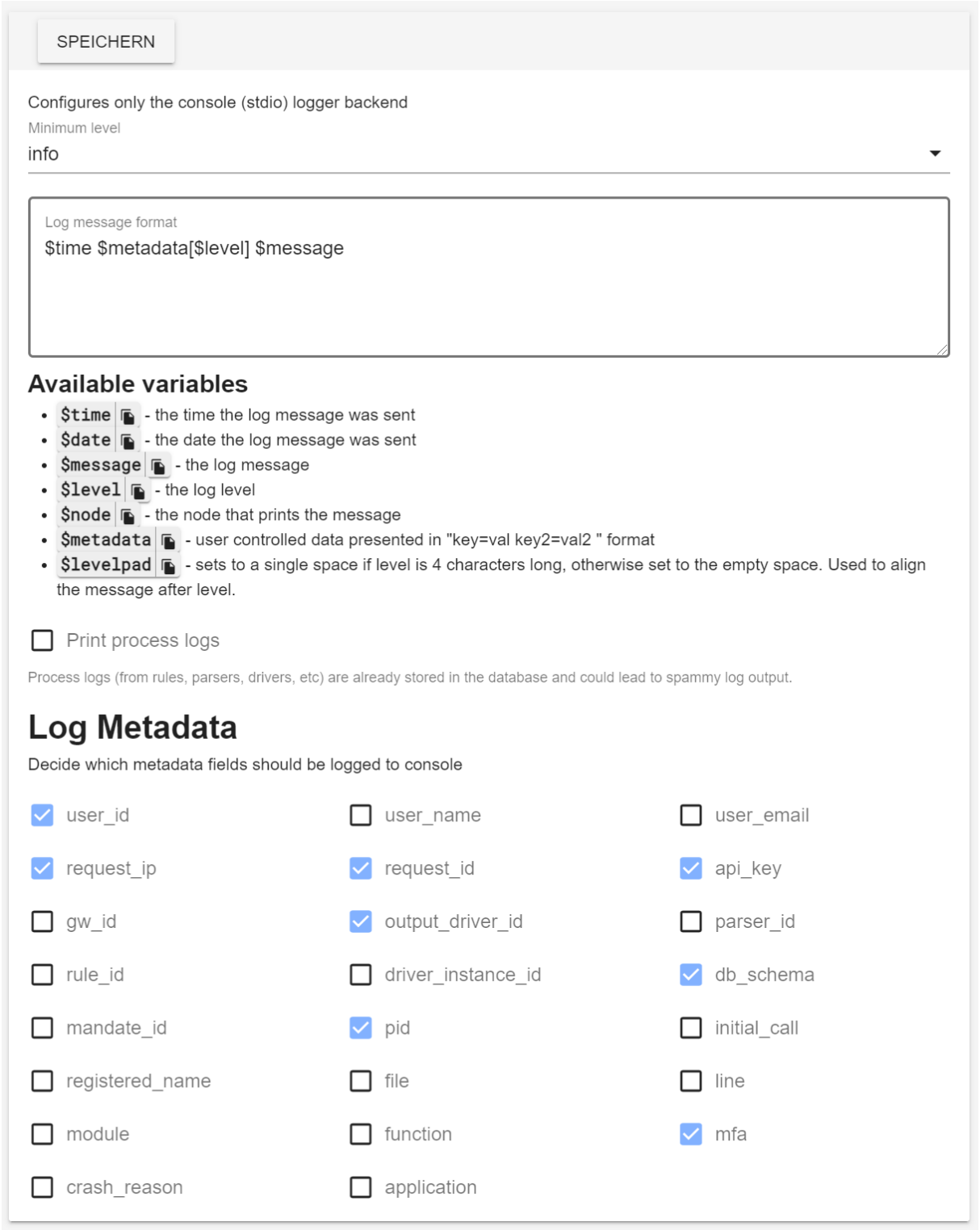


Abbildung 21.1: Log Konfiguration

21.2 Minimum level

Stellt ein, ab welchesm Loglevel eine Lognachricht wirklich auf der Konsole ausgegeben wird.

- `debug` Enthält viele Informationen, die eher für die Entwicklung und zur Fehlersuche praktisch sind. Im Normalbetrieb sollte diese Option nicht verwendet werden
- `info` ist das Stadardlevel

Es gibt weitere Loglevel, die man aber nicht als Minimallevel konfigurieren kann:

- `warn` Lognachrichten, die auf potentielle Fehler hinweisen
- `error` Fehlermeldungen und Abstürze

21.2.1 Log message format

Eine Vorlage, nach der jede Lognachricht zusammen mit ihren Metadaten formatiert wird.

Wichtig ist, dass am Ende der Vorlage mindestens 2 Leerzeilen stehen sollten, damit die Nachrichten korrekt von Graylog getrennt werden können.

Unterhalb des Textfelds befinden sich alle verfügbaren Variablen mitsamt Beschreibung.

21.2.2 Print process logs

Wenn aktiviert, werden Logs aus Benutzerkonfigurierten Unterprozessen auch mit auf die Standardausgabe geschrieben. Da diese Logs eigentlich innerhalb der von ELEMENT IoT einsehbar sind, ist diese Option standardmässig deaktiviert.

21.2.3 Log Metadata

Metadaten sind strukturierte Daten, die an jede Lognachricht gehängt werden und bei Bedarf ausgegeben werden können.

In diesem Bereich kann man auswählen, welche Metadaten über die Variable `$metadata` auf die Standardausgabe geschrieben werden, sofern sie auch in der Lognachricht enthalten sind.

Der Bereich steht nicht zur Verfügung, wenn die `$metadata` Variable nicht in der Vorlage verwendet wird.

21.2.3.1 Metadaten

Nicht alle Metadaten sind in jeder Nachricht vorhanden.

- HTTP - Nur bei Logs zu HTTP Anfragen verfügbar:
  - Benutzerdaten - Nur, wenn ein Mensch die UI benutzt
    * `user_id` Datenbank ID des Nutzers
    * `user_name` Name des Nutzers
    * `user_email` Email des Nutzers
  - API Keys - Nur, wenn ein API Key die API benutzt
    * `api_key` verwendeter API Schlüssel
  - `request_id` Eine Eindeutige ID, die jede HTTP Abfrage identifiziert
  - `request_ip` Die IP, von der die Anfrage ausgeht.

    Hinweis die IP wird durch Proxies und VPNs verschleiert. Sofern Proxies die IP nicht weitergeben (auch Reverse-Proxies), wird hier nicht die wirkliche IP geloggt.
  - `gw_id` HTTP Traffic, der von Gateways ausgelöst wird, enthält auch deren ID
- Prozessabhängig
  - `output_driver_id` Datenbank ID eines Ausgangstreibers
  - `parser_id` Datenbank ID eines Parsers
  - `rule_id` Datenbank ID einer Regel
  - `driver_instance_id` Datenbank ID eines Eingangstreibers
  - `db_schema` Typ des Prozesses (Regel/Treiber/Parser)
  - `mandate_id` Datenbank-ID des Mandanten, um den es geht
- Immer verfügbar

  Die meissten dieser Metadaten sind nur für Programmierer interessant. Evtl. werden Sie gebeten, eine dieser Metadaten zu Debugzwecken anzuschalten.
  - `pid` eindeutige ID des Prozesses, der die Lognachricht geschrieben hat
  - `initial_call` erster Funktionsaufruf des Prozesses, (zB `MyGenServer.init/1`)
  - `registered_name` falls der Prozess einen Namen hat, wird dieser hier gezeigt
  - `file` Pfad der Datei, in der der Logaufruf geschrieben wurde
  - `line` Zeile in der Datei (macht nur zusammen mit `file` Sinn)
  - `module` Modul, aus dem der Logaufruf kommt
  - `function` Funktion, aus der der Logaufruf kommt

- `mfa` kurz für Modul/Funktion/Arität - kürzere Form für `module` und `function`
- `crash_reason` selten ausgegebene Fehlermeldung, die der Grund dafür war, dass in einem GenServer `terminate/2` aufgerufen wurde.
- `application` Anwendung/Bibliothek, aus der der Logaufruf stammt. ELEMENT IoT heisst hier `platform`