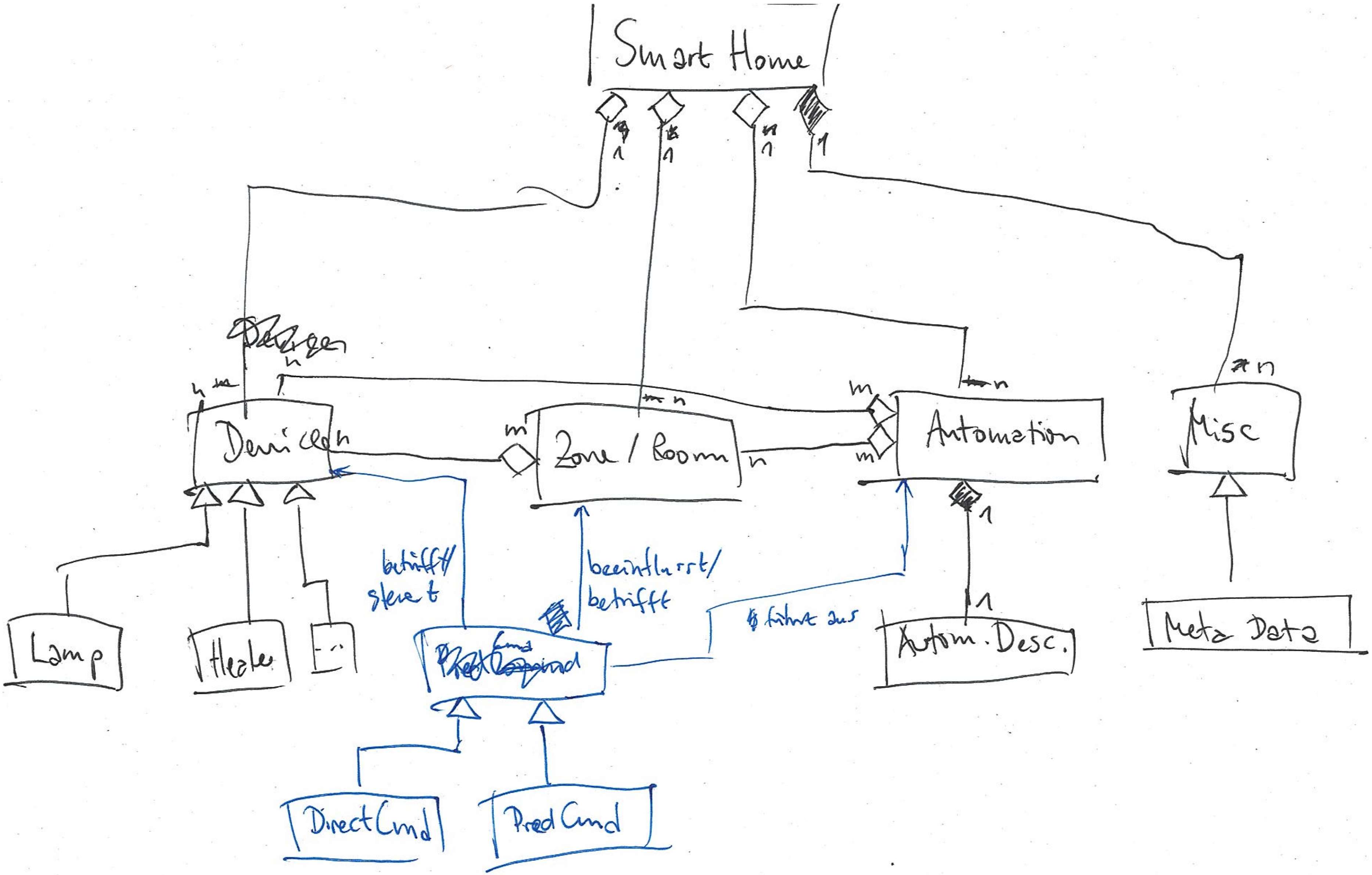
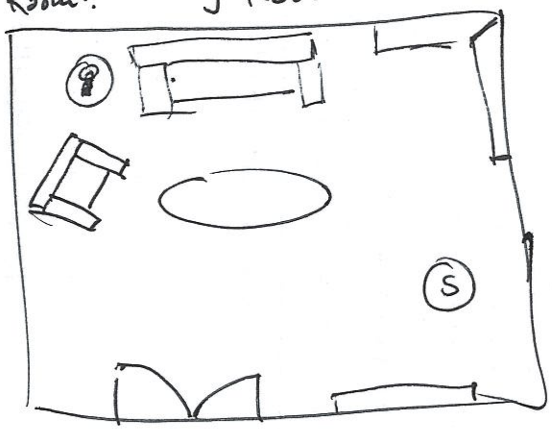


(macht einen "reinventing the wheel" - Eindruck...
→ kann man das auch eleganter lösen?)

SH THING

Room: Living Room



💡 Lampe ::Thing

→ Kind: Lamp

→ Status: ON/OFF

→ Value: 💡 <T>

→ Actions: ...

Ⓢ : RoomSensor ::Thing

→ Kind: Sensor

~~→ Status: ON/OFF~~

~~→ Value~~

Items:

→ Humidity Sensor

→ Status: ON/OFF

→ Value: <T>

→ Temp. Sensor

→ Status: ON/OFF

→ Value: <T>

→ Unterscheiden zwischen Read-only Things (Sensoren) und actionable ("steuerbare") Things.

→ Status: ON/OFF

Proof of Concept Scenario

- Lampe in Büro (smarte Lampe oder smarter Stecker + "dumme" Lampe)
- Messstation an Arbeitsplatz

→ Lampe randomisiert für ~~15/20 Minuten In~~ 2 Minuten, alle 15/30 Minuten an/ausschalten.

→ Helligkeit(en) messen

Sensor:

⇒ {time : 2023-06-09 12:00:00,
brightness : X
}

⇒ ~~B~~ Hub : → Lampensteuerung

→ Logged Lampenänderung

sendet an API →

{ {time : ...
brightness : ...,
lamp : ON/OFF }

↓

Trainingsdaten

SH import

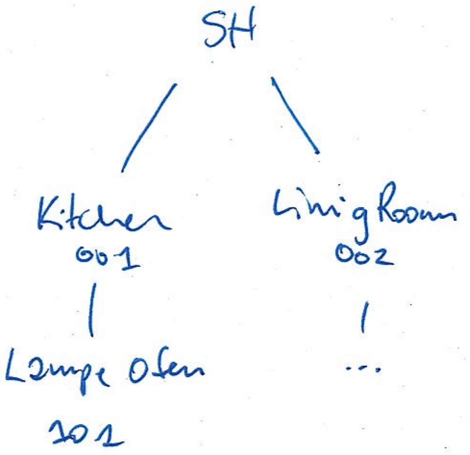
als JSON Hierarchy?

1. Liste mit Räumen + Geräten (inkl. IDs)
Zonen

2. Hierarchy von/zwischen Zonen/Räumen inkl. enthaltener Geräte

z.B.

```
1. {rooms: [
      {
        ~~room~~ ID: 001,
          name: "Kitchen"
      },
      {
         ID: 002,
       name: "LivingRoom"
      }
    ],
  devices: [
    {    ID: 101, name: "Lampe Ofen", type: lamp/RGB}]}
```



```
2.  { root: [
       {
```

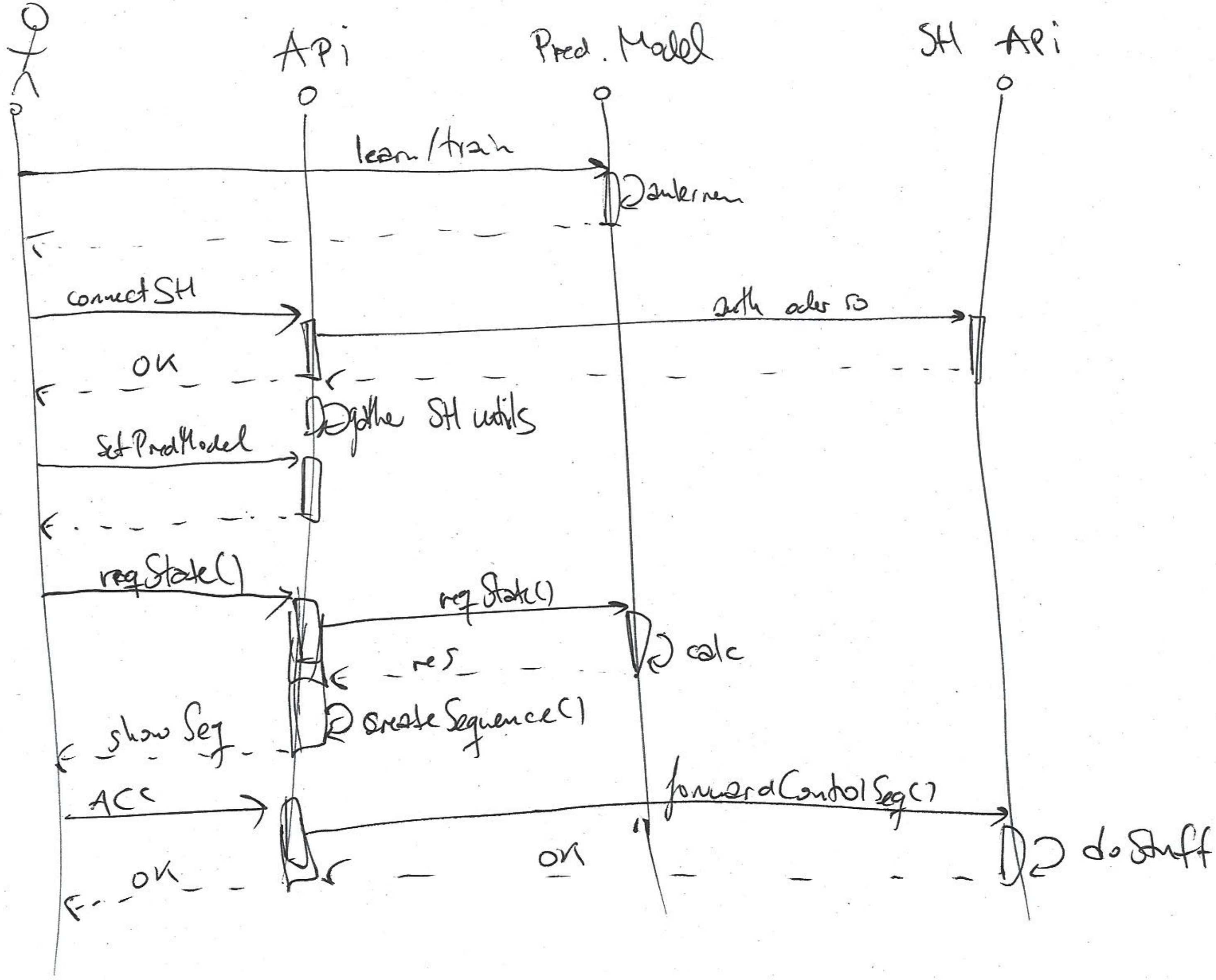
wie sieht Format aus, was man von SH APIs bekommt?

→ openHAB REST API
Abfrage aller "Things" & "Items"
& etc.

→ ...

~~El Gamal Sign. (mod p-1)~~

8



Api

- Model (ggf. was wurde gelernt / was ist möglich)
  → Inputvektor? → Wie steuern? Trainieren?

.Train() ← data → Result maybe

.Calc() ← input vector → result ↑ format?

- Wie sieht SH aus? Was für Utils sind vorhanden?
  → Lampen
  → Türen
  → etc...
  } in Datenstruktur speichern, da konkrete Steuerung möglich sein soll → simpler Forward an SH Api

  ↳ Unterscheidung zwischen "normalem" Command und pred.-unterstützten Command. → Steuerelemente aus/von SH Api (?)
  → Anknüpfen an SH Api

- User Interaction
  → REST Api (?) → Spring? (Bitte kein Webdev...) :(
  → Ui (?) → Web?
  ↳ Java Ui + REST Api

~~KEINE ANZ~~ ANZEIGE DES AKTUELLEN SH-STATUS? → WANN REFRESHEN?

Prediction Model

Create() / Select() : ~~Status~~ Result

- FeedData(...) : ~~Status~~ Result
- Train (...) : ~~<Status, Update>~~ <Result, Status>
- Predict(...) <ControlSequence(?), ~~Update~~ Status>

→ Wie ~~[illegible]~~ Elemente d. Pred. Models mit SH-Devices / Rooms ← irrelevant?
verknüpfen? → Mapping?

↳ SH-Device : ID

→ Output von Pred. beinhaltet ID(s) / muss auf ID(s) gemappt werden

1. SH importieren (~~noch~~ quasi: Verbindung zu SH aufbauen)

2. Training Data importieren / Trainieren

2.1.

A	B	C	D	E
⋮	⋮	⋮	⋮	⋮

⇨ Spalten den Geräten / Eigenschaften des SH zuordnen und auswählen, was gelernt werden soll.

Ui | REST

- ~~GET~~ der Geräte / Eigenschaften
- POST der Trainingsdaten
- POST des Mappings
- POST → "Trainiere!"

2.2. Prediction Model auswählen / ~~einrichten~~ (während Laufzeit wechselbar?)

2.3. Trainieren! (im neuen Thread)

oder bereits trainiertes Model verwenden → mit bool bei Einrichtung angeben, ob Modell bereits trainiert ...

↳ Training im separatem Thread

⇒ DONE!

3. Normal Command

3.1. Ui Element auswählen → Steuern

Ui | REST

- ~~GET~~ POST Geräte-ID + ~~[illegible]~~ Instruktion

↳ ~~Weitere~~ Forward an SH-API

~~2.1~~

4. Pred. Command

4.1.

↳ Pred. Modell

→ Instr. an SH-API + Res. anzeigen

Ui | REST

- POST Geräte-ID + Outcome

↳ Anschmeißen von Pred. Modell ⇒

→ Instruktion an SH-API + Response