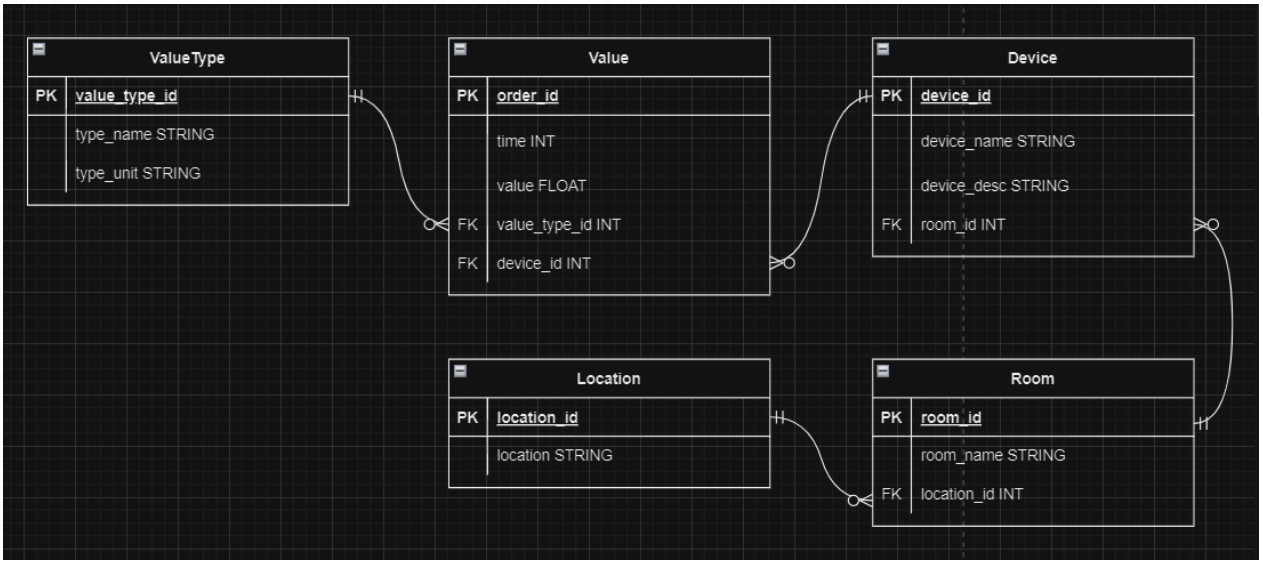
Larissa Lazau

Cheat-sheet – API

New Model – model.py



- Base: Basisklasse für alle Modelle
- ValueType:
  - Tabelle: "value_type"
  - Spalten: id (PK, INT, autoincrement), type_name (STRING), type_unit (STRING)
  - Beziehungen: Werte (One-to-Many mit "Value")
- Value:
  - Tabelle: "value"
  - Spalten: id (PK, INT, autoincrement), time (INT, nicht nullable), value (FLOAT), value_type_id (INT, FK), device_id (INT, FK)
  - Einzigartigkeit: Kombination aus Zeit, value_type_id und device_id muss einzigartig sein
- Device:
  - Tabelle: "device"
  - Spalten: id (PK, INT, autoincrement), device_name (STRING), device_desc (STRING), room_id (INT, FK)
  - Beziehungen: Werte (One-to-Many mit "Value"), Raum (Many-to-One mit "Room")
- Room:
  - Tabelle: "room"
  - Spalten: id (PK, INT, autoincrement), room_name (STRING), location_id (INT, FK)
  - Beziehungen: Geräte (One-to-Many mit "Device")
- Location:
  - Tabelle: "location"
  - Spalten: id (PK, INT, autoincrement), location (STRING)
  - Beziehungen: Räume (One-to-Many mit "Room")

Larissa Lazau

API types – api_types.py

Sie werden in „main.py“ und in „crud.py“ verwendet und dienen zur Deklaration eines bestimmten Datentyps. In der „main.py“ entsprechen die Rückgabewerte von „@app.get/post/put“ den in api_type deklarierten Klassen.

Crud.py

Definiert Methoden zum Erstellen, Lesen, Aktualisieren und Löschen (CRUD) von Datensätzen in der Datenbank.

Für jede neue add_or_update / get_ Methode, kann die vorhandene übernommen werden und dann entsprechend angepasst werden.

- **add_or_update_value_type**: Fügt einen Wertetyp hinzu oder aktualisiert ihn.
- **add_or_update_device**: Fügt ein Gerät hinzu oder aktualisiert es.
- **add_or_update_room**: Fügt einen Raum hinzu oder aktualisiert ihn.
- **add_or_update_location**: Fügt einen Ort hinzu oder aktualisiert ihn.
- **add_value**: Fügt einen Wertedatensatz hinzu.
- **get_room, get_rooms**: Ruft einen Raum oder alle Räume ab.
- **get_location, get_locations**: Ruft einen Ort oder alle Orte ab.
- **get_device, get_devices**: Ruft ein Gerät oder alle Geräte ab.
- **get_value_types**: Ruft alle Wertetypen ab.
- **get_value_by_device_id**: Ruft alle Werte für eine bestimmte Geräte-ID ab.
- **get_value_type**: Ruft eine bestimmte Wertart ab.
- **get_values**: Ruft Werte mit optionalen Filtern ab.

Logging:

Verwendet das Protokollierungsmodul, um Fehler wie Integritätsverletzungen zu protokollieren.

Client – CSV-Import

Dieses Python-Skript verwendet Flask zur Erstellung eines einfachen Webservers, Pandas zur Verarbeitung von CSV-Daten und Requests zum Senden von HTTP-Anfragen.

- **@app.route('/')**: Die Index-Route, die die Vorlage "index.html" wiedergibt.
- **@app.route("/", methods=['POST'])**: Die Route, die Datei-Uploads verarbeitet. Sie speichert die hochgeladene Datei und ruft dann die Funktion send_data auf, um die Daten zu verarbeiten und zu senden.
- **GET http://localhost:8080/api/type/**: Ruft eine Liste von Datentypen ab.
- **POST http://localhost:8080/api/value/**: Sendet die verarbeiteten Daten, um sie zu speichern oder weiter zu verarbeiten.

Die Anwendung erwartet, dass die CSV-Datei mit bestimmten Spalten strukturiert ist, wobei:

- Die erste Spalte enthält Datums- und Zeitangaben.
- Die zweite Spalte enthält die Device-ID.
- Die dritte Spalte enthält die Wertdaten (z. B. Temperaturmesswerte).

```
time,station,T
2024-01-04T14:00+00:00,11001,23.0
2024-01-04T15:00+00:00,11001,33.4
```