## **Aufgabe 1**

a.)

**Welches Pattern aus der Softwareentwicklung fällt Ihnen zu den Grundprinzipien von MQTT ein?**

Publisher → sendet Nachrichten an bestimmte Topics.

Subscriber → abonniert Topics und empfängt Nachrichten dazu.

Broker → vermittelt zwischen Publisher und Subscriber.

**Wie kann ich beispielsweise alle Topics unter /weather abonnieren?**

MQTT nutzt Wildcards, um mehrere Topics gleichzeitig zu abonnieren:

Wildcard: + / Bedeutung: Platzhalter für genau einen Level

Wildcard: # / Bedeutung: Platzhalter für beliebig viele Level

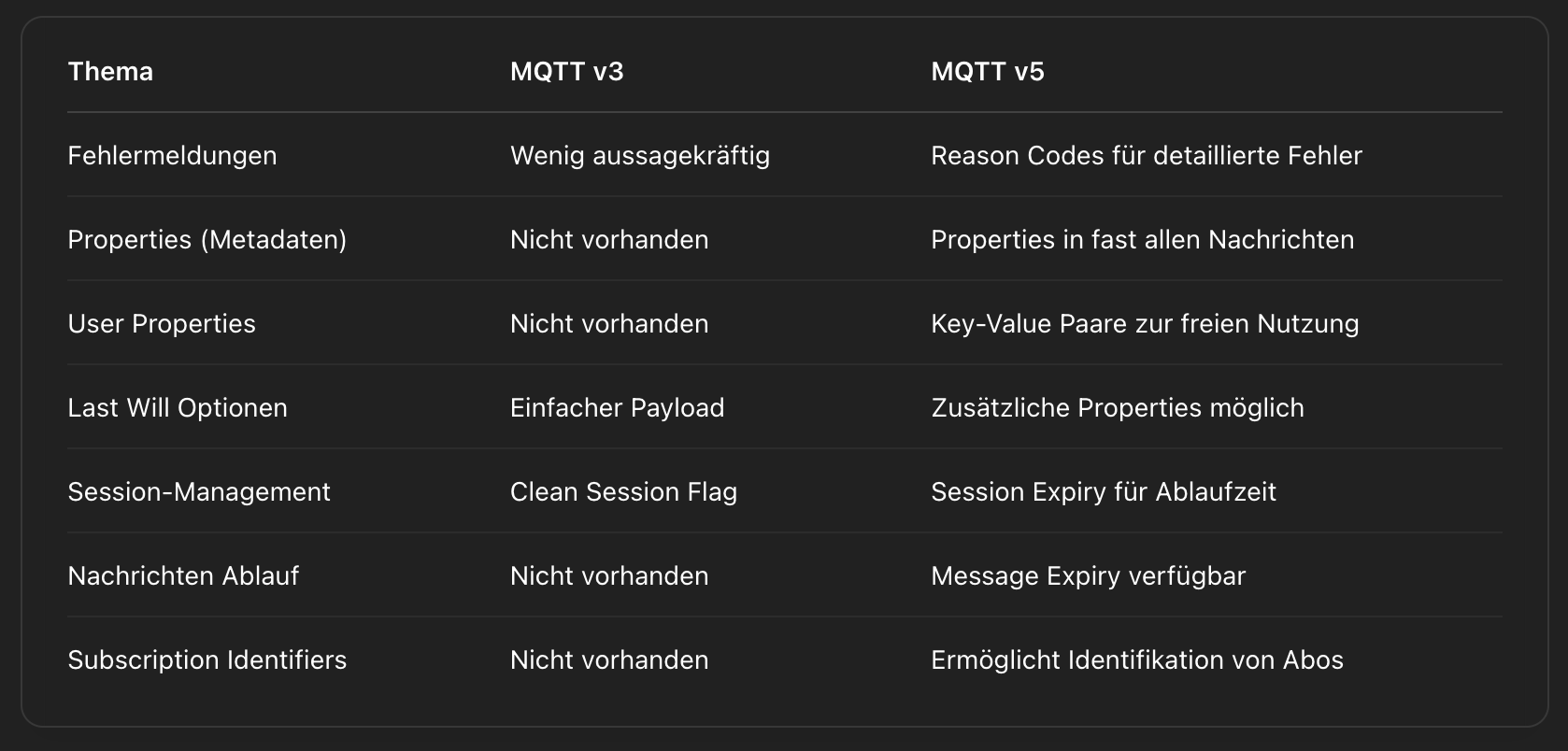
**Was ist der “Last Will”?**

"Last Will and Testament" (LWT) = spezielle Nachricht, die der Broker sendet, wenn ein Client unerwartet offline geht (ohne korrektes DISCONNECT).

Wird beim CONNECT angegeben und vom Broker bei Verbindungsabbruch veröffentlicht.

**Was sind die wichtigsten Unterschiede zwischen MQTT v3 und v5?**

MQTT v5 ist deutlich flexibler, moderner und transparenter in der Kommunikation, bleibt aber kompatibel zum einfachen v3-Ansatz.



mosquitto\_sub -h mqtt.fim.uni-passau.de -t "#" -v

b)

**Wie können Sie alle Topics abonnieren, auch ohne diese alle im Voraus zu kennen?**

mosquitto\_sub -h 10.50.12.150 -p 1883 -t "#" -v

oder

mosquitto\_sub -h 10.50.12.150 -p 1883 -t "/weather/#"

**Welche Topics und Werte können Sie sehen?**

Wetterdaten: Mosbach, Stuttgart, Mergentheim,

/AZ-Envy/Lpg 0.01;ppm

/AZ-Envy/Co 0.01;ppm

/AZ-Envy/Smoke 0.04;ppm

/AZ-Envy/Temp 29.32;┬░C

/AZ-Envy/Hum 22.12;%

/siemens/1200CPU/Time S7-1200 Time: +9h +12min +32sec

/siemens/1200CPU/Poti +16860

/siemens/1200CPU/IO +0

/siemens/1200CPU/OPC/iRcv1 +12345

/siemens/1200CPU/OPC/sRcv Hallo

/siemens/1200CPU More Values under /Time /Poti /IO /OPC/iRcv1 /OPC/sRcv

**->Hier sieht man Simensdaten**

/weather/mosbach {"tempCurrent":11.990021,"tempMax":11.990021,"tempMin":10.869995,"comment":"Publ.Id 8528","timeStamp":"2025-04-09T08:12:17.024+00:00","city":"Mosbach","cityId":2869120}

/weather/stuttgart {"tempCurrent":11.059998,"tempMax":12.029999,"tempMin":9.52002,"comment":"Publ.Id 8528","timeStamp":"2025-04-09T08:11:17.032+00:00","city":"Stuttgart","cityId":2825297}

/weather/mergentheim {"tempCurrent":7.6400146,"tempMax":7.6400146,"tempMin":7.6400146,"comment":"Publ.Id 8528","timeStamp":"2025-04-09T08:13:17.037+00:00","city":"Bad Mergentheim","cityId":2953402}

**->Wetter**

Bei nur # kriegen wir alles was auf der Maschine läuft

**Wie können Sie ein bestimmtes Topic abonnieren, z.B. das Wetter für Mosbach?**

Teste: mosquitto\_sub -h 10.50.12.150 -t "/weather/mosbach" -v

**In welchem Datenformat werden die Wetterdaten bereitgestellt?**

Die Wetterdaten werden im JSON-Format bereitgestellt.  
  
Beispiel:  
  
{

"temperature": 15,

"humidity": 70,

"wind": 5

}

c)

**Client Programmierung**

## **Aufgabe 2**

a)

**Kafka vs. MQTT – Gemeinsamkeiten & Unterschiede:**

**Gemeinsamkeit:**

* Beide sind Messaging-Systeme (Publisher-Subscriber-Modell).
* Asynchrone Kommunikation.
* Kafka als auch MQTT arbeiten mit einem zentralen Broker.

**Unterschiede:**

* MQTT ist leichtgewichtig, ideal für IoT/Embedded Systeme, Kafka für Big Data, Streaming und Analytics (für große Datenmengen)
* Kafka speichert Daten langfristig, MQTT oft nur temporär (Retained Messages)
* MQTT hat eine geringe Latenz, ideal für Echtzeit-Kommunikation. Kafka Höher, da auf Persistenz ausgelegt.

**Wann ist Kafka besser geeignet?**

* Große Datenmengen / Big Data
* Event Streaming, Analytics, Logging
* Systeme mit mehreren Consumer-Gruppen (unabhängige Verarbeitung)

**“Dumb Broker / Smart Consumer” (Kafka) vs. “Smart Broker / Dumb Consumer” (MQTT):**

* Kafka: Broker speichert die Nachrichten einfach persistent in Topics und stellt sie zur Verfügung. Consumer verantwortlich für: verwaltung des Offset, Fehlerhandling,
* MQTT: Broker verwaltet Zustellung & Session-Management, Puffern für Offline Clients. Consumer müssen sich nur verbinden, abonnieren, Empfangene Nachrichten verarbeiten.

**Was sind Partitionen (Kafka)?**

* Unterteilung eines Topics → parallele Verarbeitung möglich (Erhöht Durchsatz und Verarbeitungsgeschwindigkeit)
* Neben Load-Balancing auch für:
  + Skalierbarkeit
  + Nachrichtenreihenfolge innerhalb einer Partition
  + Verteilte Verarbeitung nach Schlüssel (z. B. User-ID)