

3749915, 2927803, 2274162, 6473957

Anmeldedaten Grafana:

Mail: kein.hacker@yoursystem.org

PA: Droptable_users

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.

Thema	MQTT v3	MQTT v5
Fehlermeldungen	Wenig aussagekräftig	Reason Codes für detaillierte Fehler
Properties (Metadaten)	Nicht vorhanden	Properties in fast allen Nachrichten
User Properties	Nicht vorhanden	Key-Value Paare zur freien Nutzung
Last Will Optionen	Einfacher Payload	Zusätzliche Properties möglich
Session-Management	Clean Session Flag	Session Expiry für Ablaufzeit
Nachrichten Ablauf	Nicht vorhanden	Message Expiry verfügbar
Subscription Identifiers	Nicht vorhanden	Ermöglicht Identifikation von Abos

```
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;T°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 Siemensdaten

/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):

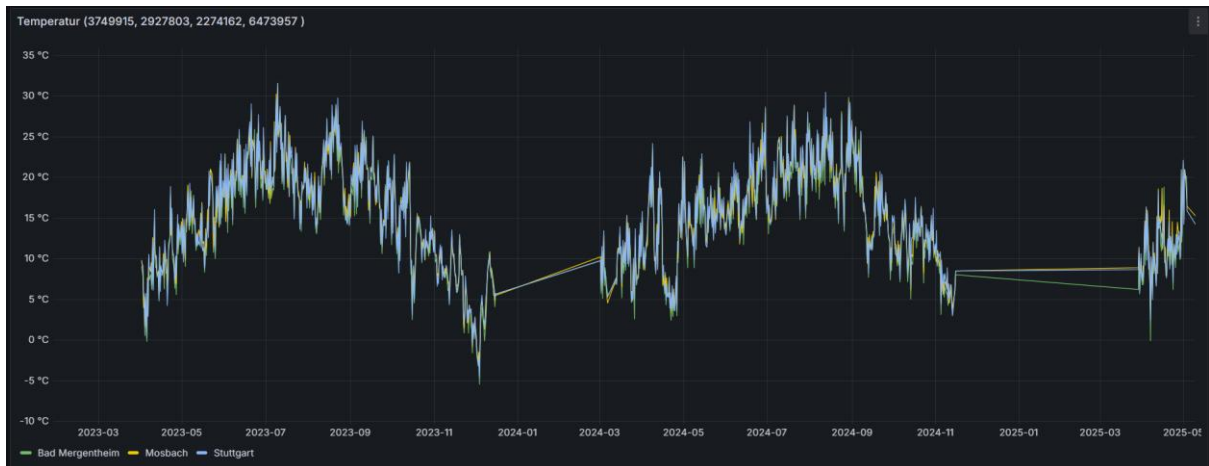
- Kafka: Broker speichert die Nachrichten einfach persistent in Topics und stellt sie zur Verfügung. Consumer verantwortlich für: Verwaltung des Offset, Fehlerhandling,
- MQTT: Broker verwaltet Zustellung & Session-Management, Puffern für Offline Clients. Consumer müssen sich nur verbinden, abonnieren,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

d)

Screenshot von Grafana:



Null-Werte wurden Connected.

Aufgabe 3

1. Wie sind die Daten in den Topics organisiert?

Die Kommunikation erfolgt über JSON-Objekte mit Feldern wie type, gameld, player, column, actions und state.

Das verwendete Topic ist in Partitionen unterteilt, was eine parallele Verarbeitung der Nachrichten ermöglicht.

- Das Topic game-requests enthält die Spielzüge der Spieler.
- Das Topic game-events liefert die entsprechenden Antworten des Game-Masters.

2. Was muss man bei Producer/Consumer mit max. 10 Consumern beachten?

Mindestens 10 Partitionen erstellen, damit jeder Consumer eine eigene Partition verarbeiten kann.

Consumer müssen in einer Consumer Group sein, um die Partitionen automatisch zu verteilen.

Zur Vermeidung mehrfacher Verarbeitung:

- Setze `enable_auto_commit=True` oder
- Committe die Offsets manuell.


Wichtig: Keine mehrfach gestarteten Instanzen mit der gleichen Consumer Group ID verwenden.

3. Für welche Zwecke eignen sich Standalone Consumer (ohne Gruppe)?

- Gezielte Verarbeitung einzelner Partitionen oder Offsets (z. B. Debugging, Replay).
- Ideal bei Einmalverarbeitung, Monitoring oder wenn man volle Kontrolle über den Offset braucht.
- Verwendung mit `assign()` und `seek()`.

Dokumentation von einem Spiel (Spieler 1 / Rot)

Zwei Spieler starten das Spiel. Nach dem Start erscheint folgendes Fenster:



player

client

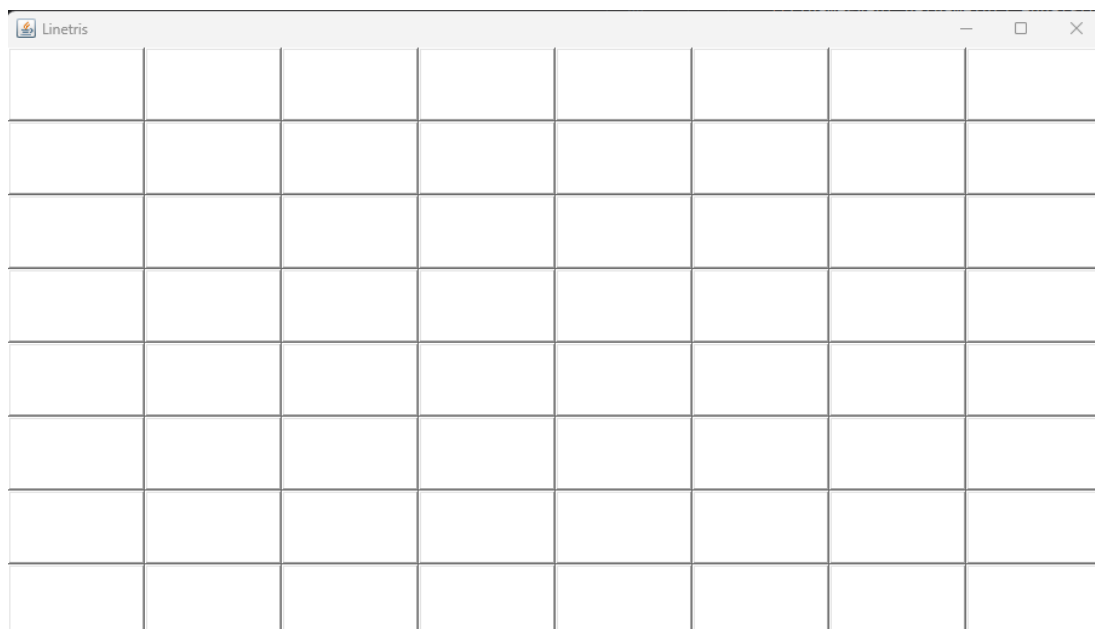
Spiel starten

Auf Spiel warten

In diesem Fenster kann der Spieler seinen Namen anpassen.

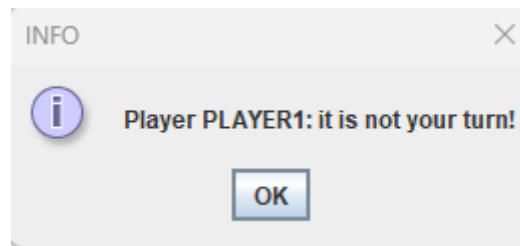
Ein Spieler klickt auf „Auf Spiel warten“, während der andere auf „Spiel starten“ klickt. Diese Spielvermittlung findet über einen anderen Channel („game-mediation“) statt.

Sobald die Verbindung hergestellt ist, erscheint bei beiden Spielern folgendes Fenster:

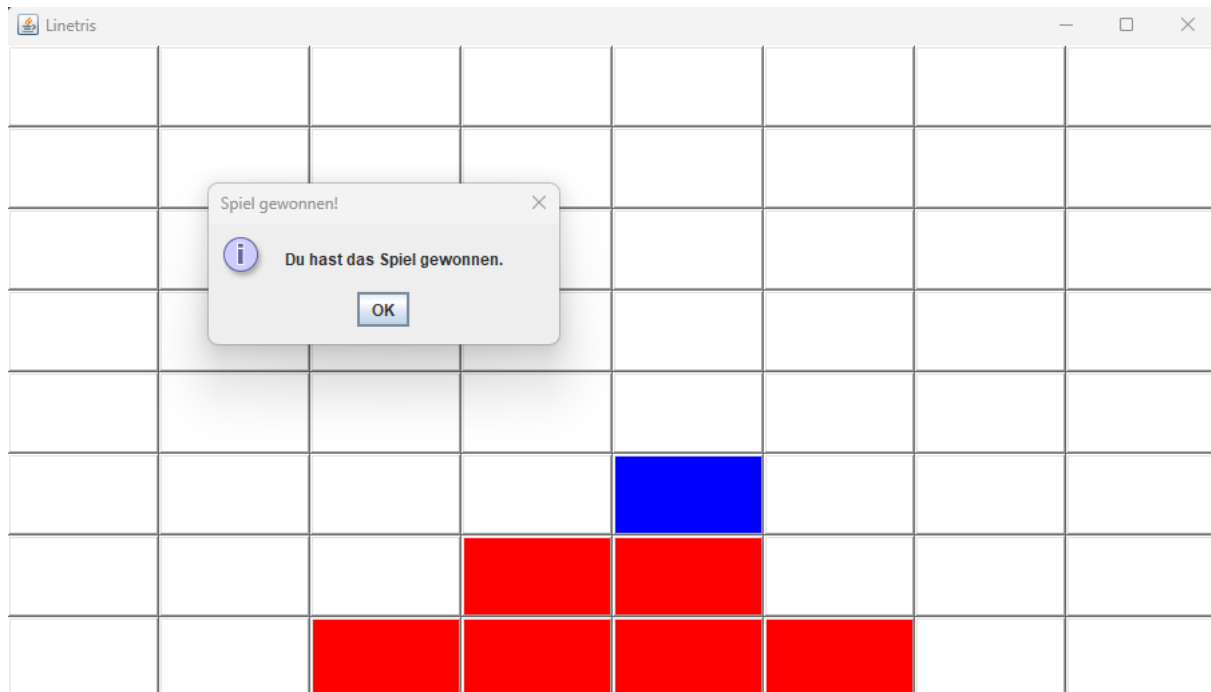


Nun können die Spielblöcke abwechselnd von den Spielern befüllt werden.

Versucht ein Spieler, zwei Züge hintereinander auszuführen, erscheint folgende Meldung:



Sobald ein Spieler gewonnen hat, erscheint folgende Meldung:



Konsolenausgabe von dem Spiellauf:

New game proposed: {"gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f","player1":{"name":"player"},"client1":{"name":"client"}}

Event empfangen:

```
{"timeStamp":1747211318067,"actions":[{"type":"newGame","client1":{"name":"client"},"player1":{"name":"player"},"client2":{"name":"client"},"player2":{"name":"player"},"rows":8,"cols":8}],state:"OK","gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f"}
Move sent: {"type":"move","gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f","player":"PLAYER1","column":2}
```

Event empfangen:

```
{"timeStamp":1747211345629,"actions":[{"type":"move","player":"PLAYER1","column":2}],state:"OK","gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f"}
```

Event empfangen:

```
{"timeStamp":1747211349165,"actions":[{"type":"move","player":"PLAYER2","column":3}],state:"OK","gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f"}
```

Move sent: {"type":"move","gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f","player":"PLAYER1","column":6}

Event empfangen:

```
{"timeStamp":1747211350902,"actions":[{"type":"move","player":"PLAYER1","column":6}],state:"OK","gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f"}
```

Event empfangen:

```
{"timeStamp":1747211355876,"actions":[{"type":"move","player":"PLAYER2","column":4}],state:"OK","gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f"}
```

Move sent: {"type":"move","gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f","player":"PLAYER1","column":4}

Event empfangen:

```
{"timeStamp":1747211362356,"actions":[{"type":"move","player":"PLAYER1","column":4}],state:"OK","gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f"}
```

Event empfangen:

```
{"timeStamp":1747211370557,"actions":[{"type":"move","player":"PLAYER2","column":5}],state:"OK","gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f"}
```

Move sent: {"type":"move","gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f","player":"PLAYER1","column":5}

Event empfangen:

```
{"timeStamp":1747211372807,"actions":[{"type":"move","player":"PLAYER1","column":5}],state:"OK","gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f"}
```

Event empfangen:

```
{"timeStamp":1747211376076,"actions":[{"type":"move","player":"PLAYER2","column":1}],state:"OK","gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f"}
```

Move sent: {"type":"move","gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f","player":"PLAYER1","column":4}

Event empfangen:

```
{"timeStamp":1747211376831,"actions":[{"type":"move","player":"PLAYER1","column":4}],state:"OK","gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f"}
```

Event empfangen:

```
{"timeStamp":1747211379122,"actions":[{"type":"move","player":"PLAYER2","column":7}],state:"OK","gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f"}
```

Move sent: {"type":"move","gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f","player":"PLAYER1","column":5}

Event empfangen:

```
{"timeStamp":1747211379955,"actions":[{"type":"move","player":"PLAYER1","column":5}],state:"OK","gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f"}
```

Event empfangen:

```
{"timeStamp":1747211383212,"actions":[{"type":"move","player":"PLAYER2","column":8},{type":"deleteBottomRow","row":8}],state:"OK","gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f"}
```

Move sent: {"type":"move","gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f","player":"PLAYER1","column":6}

Event empfangen:

```
{"timeStamp":1747211385066,"actions":[{"type":"move","player":"PLAYER1","column":6}],state:"OK","gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f"}
```

Event empfangen:

```
{"timeStamp":1747211386680,"actions":[{"type":"move","player":"PLAYER2","column":5}],state:"OK","gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f"}
```

Move sent: {"type":"move","gameId":"f2e8507c-e58f-455b-ac6a-e835c3efed2f","player":"PLAYER1","column":3}

Event empfangen:

```
{ "timeStamp": 1747211388455, "actions": [{ "type": "move", "player": "PLAYER1", "column": 3 }, { "type": "winAction", "player": "PLAYER1" } ], "state": "OK", "gameId": "f2e8507c-e58f-455b-ac6a-e835c3efed2f" }
```