

Dokumentation MidEng 7.3 Message Oriented Middleware (Grundlagen plus Vertiefung)

Projektdokumentation: Implementierung und Verifizierung der MOM-Architektur

1. Architektur und Zielsetzung

Die Aufgabe bestand in der Implementierung einer Message Oriented Middleware (MOM) basierend auf **Apache Kafka** und **Spring Boot**, um die Kommunikation zwischen dezentralen Lagerstandorten (Producer) und einer Zentrale (Consumer/REST-API) zu gewährleisten.

Komponente	Rolle	Technologie
Broker/Controller	Vermittler der Nachrichten.	Apache Kafka (KRaft-Modus)
Producer	Sendet Lagerdaten über den Endpunkt <code>/send</code> .	Spring Boot / KafkaTemplate
Consumer	Zentrale, empfängt Daten vom Topic <code>quickstart-events</code> .	Spring Boot / @KafkaListener
DataController	Stellt gespeicherte Daten über eine REST-API bereit.	Spring Boot / @RestController (<code>/warehouse/status</code>)

2. Behebung von Infrastrukturfehlern (Kritischer Abschnitt)

Vor dem erfolgreichen Start traten zwei kritische Infrastrukturfehler auf, die behoben werden mussten.

a) Fehler 1: KRaft-Formatierung und Listeners-Konfiguration

Der initiale Fehler lag in der falschen Konfiguration der KRaft-Cluster-ID und der Listener, was dazu führte, dass der Broker nicht erreichbar war.

- **Ursache:** Falsche oder fehlende Ausführung der KRaft-Formatierung und unvollständige Konfiguration der `listeners` in `server.properties`.
- **Lösung 1 (Konfiguration):** Korrektur der `server.properties` zur Verwendung von `localhost:9092` (für Producer/Consumer) und `localhost:9093` (für Controller):
Properties

```
listeners=PLAINTEXT://:9092,CONTROLLER://:9093  
advertised.listeners=PLAINTEXT://localhost:9092,CONTROLLER://localhost:  
9093
```

- **Lösung 2 (Topic-Erstellung):** Nach Neustart des Brokers wurde das benötigte Topic erfolgreich erstellt: Bash

```
/usr/share/kafka/bin/kafka-topics.sh --create --topic quickstart-events  
--bootstrap-server localhost:9092 --partitions 1 --replication-factor 1  
# Ausgabe: Created topic quickstart-events.
```

b) Fehler 2: Berechtigungen (AccessDeniedException)

Nach der Formatierung stürzte der Broker im KRaft-Modus unmittelbar nach dem Start ab, da der Dienst nicht auf die Metadaten-Dateien zugreifen konnte.

- **Fehlermeldung (Auszug aus journalctl -u kafka):**

```
java.nio.file.AccessDeniedException: /var/lib/kafka/_cluster_metadata-  
0/leader-epoch-checkpoint
```

- **Ursache:** Die vorherige kafka-storage.sh format Ausführung wurde als sudo (root) durchgeführt, wodurch der kafka -Systembenutzer keine Schreibrechte auf sein eigenes Datenverzeichnis (/var/lib/kafka) hatte.
- **Lösung:** Korrektur der Besitzrechte des Datenverzeichnisses: Bash

```
sudo chown -R kafka:kafka /var/lib/kafka
```

- **Ergebnis:** Der Kafka-Dienst startete stabil (**Active: active (running)**).

3. Funktionstest und Verifizierung (Nachweis der Grundlagen)

Die Funktionsfähigkeit wurde durch drei Schritte nachgewiesen: **Senden (Producer)**, **Empfangen (Consumer)** und **Abrufen (REST)**.

a) Nachweis der MOM-Grundlagen (Producer & Consumer)

Die gesendeten Nachrichten über den /send -Endpunkt wurden erfolgreich auf dem Topic platziert (Producer-Bestätigung) und unmittelbar vom Consumer verarbeitet (Consumer-Bestätigung).

Aktion: Aufruf der URLs:

- <http://localhost:8080/send?id=Wien&count=450>
- <http://localhost:8080/send?id=Linz&count=210>
- <http://localhost:8080/send?id=Graz&count=99>

Beweis (Auszug aus dem Log-Terminal der Spring Boot Anwendung):

Code-Snippet

```
# Beweis 1: Producer sendet die Nachricht
... com.example.demo.MessageProducer : Message sent to Kafka topic
'quickstart-events': {"warehouseId":"Wien","date":"2025-12-
16","stockCount":450}

# Beweis 2 & 3: Consumer empfängt und verarbeitet ALLE Nachrichten
2025-12-16T15:43:03.829... com.example.demo.MessageConsumer : Successfully
processed and stored data: WarehouseData{warehouseId='Wien', date='2025-
12-16', stockCount=450}
2025-12-16T15:43:12.057... com.example.demo.MessageConsumer : Successfully
processed and stored data: WarehouseData{warehouseId='Linz', date='2025-
12-16', stockCount=210}
2025-12-16T15:43:19.887... com.example.demo.MessageConsumer : Successfully
processed and stored data: WarehouseData{warehouseId='Graz', date='2025-
12-16', stockCount=99}
```

b) Nachweis der Erweiterten Grundlagen (REST-API)

Der Consumer speichert die empfangenen Daten. Die Verfügbarkeit dieser Daten wurde über den implementierten REST-Endpunkt geprüft.

Aktion: Aufruf der URL: <http://localhost:8080/warehouse/status>

Beweis (JSON-Antwort im Browser):

JSON

```
[{"warehouseId": "Wien", "date": "2025-12-16", "stockCount": 450}, {"warehouseId": "Linz", "date": "2025-12-16", "stockCount": 210}, {"warehouseId": "Graz", "date": "2025-12-16", "stockCount": 99}]
```

