Hochschule Rosenheim, Fakultät für Informatik

Vorlesung: Verteilte Verarbeitung SS 2018 Prof. Dr. Gerd Beneken
Nachholprüfung WS 18/19 Lehrbeauftragter Thomas Mildner

Übungsblatt 3 von 3 – Praktikum als Grundlage für ein Kolloquium

Ziel dieses Praktikums

Ziel der Aufgabe: Messaging, Systemintegration und Internet Of Things

Wenn Sie diese Aufgabe erfolgreich abgeschlossen haben, können Sie

- Selbstständig eine Nachrichten-basierte Architektur aufbauen.
- Sie haben das Publischer Subscriber Pattern verstanden

Die Aufgabe ist ein Versuch eine mini-IoT (Internet Of Things, Telematik) Lösung zu entwerfen. Wir versuchen folgendes nachzubauen: Es gibt ein kleines Gerät, das Sie z.B. in einen LKW verbauen können. Die Abbildung zeigt ein Gerät der Firma Telic aus Unterhaching (München).

Abbildung 1: http://www.telic.de/de/sbc-avl

Das Gerät meldet bestimmte Fahrdaten über eine Nachricht (z.B. SMS) an einen Server im Internet. Die Nachrichten schickt das Gerät regelmäßig, beispielsweise alle 5 Minuten. Zu den Fahrdaten gehören die gefahrene Strecke, die GPS Koordinaten. Das Gerät hat einen Beschleunidungssensor, sodass es bei zu starker Beschleunigung (= Unfall) einen Alarm schicken kann.

Aufgabe:

1. Schreiben Sie eine Klasse TelematikEinheit mit einer eigenen main() Methode. Die Klasse stellt das oben gezeigte Gerät dar. Jedes Objekt der Klasse TelematikEinheit hat eine eindeutige ID. TelematikEinheit Objekte sollen in einem einstellbaren Zeitintervall, z.B. einmal pro Sekunde, eine Nachricht (hierzu ist auch eine Klasse zu implementieren) an eine Queue „fahrdaten“ schicken. Die Nachricht enthält die ID für die TelematikEinheit, zufällige möglichst plausible GPS Koordinaten (für die Fahrzeugposition) sowie eine Zufallszahl für die gefahrene Strecke und die Uhrzeit zum Zeitpunkt der Erzeugung der Nachricht. Es muss möglich sein, viele „TelematikEinheiten“ gleichzeitig zu starten. So könnten wir eine Fahrzeugflotte simulieren.
2. Schreiben Sie eine zweite Klasse EingangsFilter mit einer eigenen main() Methode. Der Filter entnimmt die Nachrichten aus Queue „fahrdaten“ und schreibt die Nachrichten in ein Topic „verteiler“. Später können wir den Filter dazu verwenden nicht plausible oder verstümmelte Nachrichten zu entferen.
3. Schreiben Sie eine dritte Klasse „Fahrtenbuch“, diese ist Subscriber beim Topic „verteiler“. Sie liest die Nachrichten zu den Fahrdaten aus und führt zu jeder TelematikEinheit eine eigene Liste mit allen Nachrichten zu dieser TelematikEinheit. Die TelematikEinheit wird über die oben genantne ID identifiziert.

Optional: Sie speichern die Liste der Nachrichten auf der Festplatte.

4. Schreiben Sie eine Methode, welche für eine TelematikEinheit aus den Nachrichten die gesamte gefahrende Strecke berechnen kann.

5. Optional: Schauen Sie sich bitte an, was Event Sourcing ist und vergleichen das gerade implementierte Fahrtenbuch damit.

6. Erweitern sie die Klasse TelematikEinheit so, dass diese nach einer zufälligen Zeit eine Alarm-Nachricht in die Queue „fahrdaten“ schickt. Die Alarm-Nachricht ist eine Erweiterung der oben genannten Klasse „Nachricht“. Erweitern Sie den EingangsFilter so, dass dieser die Alarm-Nachricht in eine gesonderte „alarme“-Queue schreibt.

   Schreiben Sie eine Klasse „DataWarehouse“ als Subscriber des Topics „verteiler“. DataWarehouse speichert nicht die Nachrichten selbst, sondern hält eine Datenstruktur, die für jedes Mobile Gerät weiß, wie lang die Fahrstrecke in einer bestimmten Stunde des Tages war. D.h. die Datenstruktur hat 24 Felder. Jedes Feld enthält die in der genannten Stunde gefahrene Strecke. DataWarehouse gibt die Daten in regelmäßigen Abständen auf der Konsole aus.

Software:

Installation auf dem Rechner:

Für dieses Aufgabenblatt ist (leider) eine eigene Middleware erforderlich, die JMS implementiert. Wir verwenden ActiveMQ 5.15.x von der Apache Foundation. Sie können ActiveMQ unter http://activemq.apache.org/ downloaden. Das Installationspaket für Windows oder Linux umfasst jeweils ca. 60 MB. Die folgenden Schritte führen Sie bitte zur Installation von Active MQ durch (unter Windows, Linux kann im Detail von den Schritten abweichen). Die Installation kann etwas dauern … Beschaffen Sie das Installationspaket von ActiveMQ 5.15.x für Ihr Betriebssystem Entzippen Sie das Paket (irgendwo) auf der Festplatte ihres Arbeitsrechners, in dem Verzeichnis müssen Sie schreibrechte haben. Dieses Verzeichnis wird im Folgenden als <ActiveMQ Home> bezeichnet. Starten Sie ActiveMQ mit <ActiveMQ Home>/bin/activemq.bat in einer DOS-Konsole. ActiveMQ sollte ohne Fehlermeldung starten.

Youtube Anleitung:
https://www.youtube.com/watch?v=oaegBVoVvlQ&list=PL73qvSDlAVVixd1FFxiCCQX0fT5ixhJB

Docker Container:

Alternativ kann mit einem Docker Container gearbeitet werden.

```
docker run --name='activemq' -it --rm \
    -e 'ACTIVEMQ_CONFIG_MINMEMORY=512' \
    -e 'ACTIVEMQ_CONFIG_MAXMEMORY=2048'\
        -P
    webcenter/activemq:latest
```

https://hub.docker.com/r/webcenter/activemq/

Tipps:

- Erstellen Sie einen neuen Ordner im Repository mit dem Namen „03_Messaging“
- Erstellen Sie ein neues Projekt mit einer passenden .gitignore Datei
- Verwenden Sie Tools wie Maven / Gradle für die Abhängigkeiten
- Gelb markierte Aufgaben sind optional