

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 Publisher 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 Beschleunigungssensor, 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 entfernen.
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 genannte 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 gefahrene 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.
7. 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.