

Seminararbeit für das Wahlfach Microservices mit Node.js, Docker und Kubernetes

Wintersemester 2022 - Prüfungszeit Dezember 2022

Aufgabenstellung veröffentlicht am 28.11.2022

**Abgabe der Lösungen in Moodle als PDF (für den
theoretischen Teil) und ZIP Datei (für den praktischen Teil) bis
zum 28.12.2022 20:00 Uhr.**

**Die Seminararbeit ist alleine zu bearbeiten. Gruppenarbeiten
sind nicht zulässig.**

Dozent: Hartmut Seitter

1 Themenstellung: Messdatenerfassungssystem

Es soll ein Messdatenerfassungssystem erstellt werden in dem verschiedene Umweltdaten erfasst, gesammelt und ausgewertet werden sollen.

Es sollen von verschiedenen Räumen oder allgemeiner ausgedrückt, es soll von verschiedenen Lokationen unterschiedliche Messdaten an das zentrale Messdatenerfassungssystem gesendet werden.

Im ersten Schritt sollen folgende Messwerte an das zu schaffende Messdatenerfassungssystem übermittelt werden:

- ID der Messstation
- Die Lokationsdaten (GPS Daten) (optional)
- CO2 Wert in bestimmten Räumen
- Anzahl der Personen in einem Raum (pax counter)
- Temperatur
- Luftfeuchtigkeit
- Feinstaubpartikel in der Luft

Es kann von jeder Lokation entweder nur ein Messwert oder mehrere Messwerte übermittelt werden.

Bevor die Daten abgespeichert werden, sollen die übertragenen Wert auf Gültigkeit überprüft werden und es sollen nur Daten abgespeichert werden, die im Gültigkeitsbereich des Sensors liegen.

Das Messdatenerfassungssystem soll auch beim Erreichen bestimmter Schwellwerte einzelner Messpunkte eine Benachrichtigung an ein oder mehrerer Systeme generieren können. Desweiteren soll die Möglichkeit geschaffen werden, dass eine Benachrichtigung erzeugt wird, wenn eine Kombination von verschiedenen Sensoren ein definiertes Limit über- bzw. unterschreitet

Die Messdaten sind auf eine bestimmte Lokation bezogen und für jede Lokation soll eine kurze Beschreibung hinterlegt werden können und auch die allgemeinen Eigenschaften der verwendeten Sensoren (Sensortyp, Messwertebereich, ...) sollen abgespeichert werden können.

Diese Daten sollen von registrierten Benutzern eingegeben und verändert werden können.

Wenn ein Sensor über eine zu definierende Zeit keinen Messwert liefert, dann soll eine Benachrichtigung generiert werden und Systeme, die sich dafür interessieren, sollen die Möglichkeit bekommen, diese Benachrichtigung zu erhalten.

Es ist zu Beginn noch nicht definiert, wie viele Sensoren und Lokationen erfasst werden und das Datenaufkommen ist auch noch unbekannt. Deshalb soll eine skalierbare Lösung implementiert werden. Das System soll später mit Hilfe von Container in einer Cloudumgebung betrieben werden können.

Ein erstes MVP (minimal viable product) für die Messdatenerfassung soll in etwa so funktionieren:

Ein Messdatensensor oder ein Gerät, das auch mehrere Sensoren beinhaltet kann, bekommt eine eindeutige ID und eine Zieladresse mit dem sich der Messdatensensor verbinden kann. Im Messdatenerfassungssystem wird diese ID hinterlegt und so kann sichergestellt werden, dass nur von registrierten Sensoren Daten an das System gesendet werden.

Sendet dann der Sensor Messwerte, so sollen diese im Messdatenerfassungssystem abgespeichert werden.

Registrierte Benutzer sollen dann Abfragen an das Messdatenerfassungssystem senden können und dann Auswertungen/Daten pro Sensor, pro Lokation, welche Sensoren liefern derzeit keine Daten mehr, usw.

Später soll die Anwendung auch ein grafisches Interface erhalten, aber im ersten Schritt sollen nur tabellarische Daten zurück geliefert werden.

Als Kommunikationsmechanismus zwischen Sensoren und Messdatenerfassungssystem wird MQTT vorgegeben.

2 Aufgabenstellung

Die Seminararbeit teilt sich in einen theoretischen und einen praktischen Teil. Diese werden unabhängig voneinander bearbeitet und bewertet. Geben sie bis zum oben genannten Stichtag ihre Ausarbeitung im DHBW Moodle ab. Den zugehörigen Eintrag finden sie auf der Seite dieses Kurses unter “Seminararbeit”.

Laden Sie dazu in ihrer Abgabe zwei Dateien hoch:

1. Die Ausarbeitung des theoretischen Teils als PDF-Datei.
2. Die Ausarbeitung des praktischen Teils als ZIP-Archiv.

Dieses Archiv soll den Quellcode sowie alle weiteren Artefakte (Dockerfile, Kubernetes-manifeste, etc...) beinhalten. Es soll je Microservice ein Unterverzeichnis mit dem Namen des Microservice haben, in dem der jeweilige Quellcode und die weiteren Artefakte liegen.

Vorschlag – verwenden Sie ihre Matrikelnummer als filename der Zip Datei und der PDF Datei.

In der Summe umfasst die Seminararbeit 8 Aufgaben (4 im theoretischen Teil und 4 im praktischen Teil).

Maximal können 56 Punkte vergeben werden.

Ich wünsche Ihnen viel Erfolg bei der Ausarbeitung.

2.1 Theoretischer Teil: Aufgabe 1 – 8 Punkte

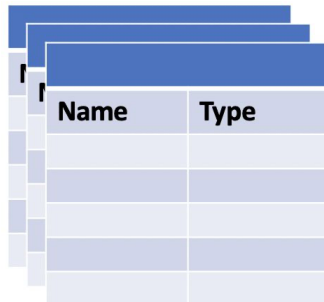
Identifizieren Sie mit Hilfe der Domain Driven Design Methode die Microservices die für das zentrale Messdatenerfassungssystem notwendig sind.

Stellen Sie die von Ihnen identifizierten Microservices in dieser Form dar.



2.2 Theoretischer Teil: Aufgabe 2 – 8 Punkte

Beschreiben Sie die Daten für jeden Microservice mit Name und Type in folgender Form.
(Hinweis: Achten sie auch auf die Referenzierung der Daten)



| Name | Type |
|------|------|
| | |
| | |
| | |
| | |
| | |

2.3 Theoretischer Teil: Aufgabe 3 – 6 Punkte

Erstellen sie eine Eventliste für Events, die in dem Messdatenerfassungssystem für bestimmte Ereignisse notwendig sind

| entity type | Event type | Event topic | Event data | Event data status | Comments |
|-------------|------------|-------------|------------|-------------------|----------|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

2.4 Theoretischer Teil: Aufgabe 4 – 6 Punkte

Erstellen Sie ein Deployment Modell der Microservices aus denen ihre Anwendung bestehen soll

3 Aufgabenstellung praktischer Teil

Eine erste Version der Microservices Application steht bis jetzt nur zwei Service als Prototypen zur Verfügung:

[rectemp \(receive temperatur data\)](#) -> Microservice, um Temperaturdaten von einem Sensor zu empfangen abzuspeichern.

Des weiteren steht ein Prototyp eines Simulationsprogramms zur Verfügung, das dazu dient, Sensordaten zu erzeugen und an die Anwendung zu senden.

[Sim \(simulate data\)](#) -> node.js Prototyp- Programm um Daten von einem Temperatursensor zu erzeugen und an einen MQTT broker zu senden.

Diese Prototypen soll von Ihnen entsprechend angepasst werden, so dass es mit den Microservices der Anwendung zusammenarbeitet und die Funktionen getestet werden können.

Laden sie den ersten Prototypen der Anwendung von Moodle herunter ([node.js Skeletons für die Seminararbeit](#)[Datei](#) – praxis-aufgabe.zip).

Führen sie die entsprechende Programmänderungen / Programmerweiterungen durch, die in den folgenden Aufgaben beschrieben werden.

Testen Sie ihre Programmänderungen

Und laden sie eine ZIP Datei (Archiv) als ihre Lösung wieder in Moodle hoch und das Archiv soll den Quellcode sowie alle weiteren Artefakte (Dockerfile, Kubernetes-manifeste, etc...) beinhalten.

Laden Sie auch die PDF Datei für den theoretischen Teil hoch.

3.1 Praktischer Teil: Aufgabe 5 - Node.js Programmierung – sim Programm - 6 Punkte

Das sim prototype Programm soll von Ihnen so erweitert und verändert werden, dass

- die Daten an einen „mqtt eclipse mosquito broker“, der als Dockercontainer local auf ihrem PC läuft, gesendet werden.
- neben den Temperaturdaten auch die Daten
 - CO2 Werte
 - Anzahl der Personen
 - Feinstaubwerte und
 - Luftfeuchtheitswerte an den mqtt broker gesendet werden können.

(Hinweis, das Programm kann auch mehrfach zur Ausführung kommen und jeweils einen anderen Messwert senden.)

Erstellen sie eine Docker Compose file damit der eclipse mosquito event broker vom dockerhub geladen wird und als lokaler container gestartet wird.

Laden sie das node.js Programm und die dazugehörige Docker Compose yaml Datei als ZIP Datei in Model hoch mit dem Hinweis für Aufgabe 5.

Die Eigenschaften des Programms und die console.log Ausgaben müssen erhalten bleiben. #

3.2 Praktischer Teil: Aufgabe 6 - NodeJS Programmierung - rectemp, tempalert Programm - 8 Punkte

- Ändern Sie den rectemp-Service ab, so dass die Temperaturdaten die vom mqtt broker (lokaler docker container) empfangen und in einer MongoDB persistiert werden.
- In der ‚Request‘ Beschreibung wird aufgeführt, das beim über- bzw. unterschreiten von bestimmten Schwellwerten eine Benachrichtigung erzeugt wird. Es soll ein neu zu erstellender node.js Microservice erstellt werden, dem eine mqtt message geschickt wird, wenn der Temperatursensor einen Wert über- bzw. unterschreitet. (Die Schwellwerte werden direkt im rectemp ‚hard coded‘ (zur Reduzierung des Programmieraufwands).

Erstellen Sie ein node.js Programm mit dem Namen tempalert das die mqtt message entgegen nimmt und als console.log Info aus gibt. Fügen Sie den entsprechenden Programmcode in rectemp ein damit die mqtt Nachricht versendet wird.

- Sorgen sie dafür, dass die MongoDB Zugriffsparmeter wie Benutzername, Passwort, MongoDB Hostname, Port oder DBName konfigurierbar sind, z.B. durch Environment-Variablen oder durch Auslesen aus einer Datei. Liegen diese Konfigurationen nicht vor (keine Environment-Variable gesetzt bzw. Config-Datei nicht gefunden) soll mit folgenden Defaults gearbeitet werden:
 - Username: kein Benutzername
 - Passwort: kein Passwort
 - Hostname: localhost
 - Port: 27017
 - DBName: Temperatur

Eigenschaften des zur Verfügung gestellten Quellcodes wie Ausgaben auf das Console Log müssen erhalten bleiben.

3.3 Praktischer Teil: Aufgabe 7 - lokaler Container Bau - 6 Punkte

Erzeugen Sie für rectemp und tempalert ein Dockerfile, mithilfe dessen die NodeJS Anwendung in ein Container Image gepackt wird. Legen sie das Dockerfile direkt im Ordner für den jeweiligen Service ab.

Erzeugen sie eine Docker-compose file, mit der die Dockerimages erzeugt und gestartet werden können. Inkl. der Anwendungen mqtt und mongodb vom dockerhub.

Testen Sie die Anwendung.

Das Simulationsprogramm sendet die entsprechenden Daten

Die Daten werden in der MongoDB abgespeichert und sollen durch das mongoCLI verifiziert werden können.

Wenn Daten ausserhalb der Schwellwerte vom Simulationsprogramm gesendet werden, so werden diese im tempalert Programm auf der console ausgegeben.

Anmerkung: Bei diesen Docker Images müssen die Änderungen aus Aufgabe 6 mit beinhaltet sein.

Achten Sie auch darauf, dass beim Neustart der Dockercontainer die Daten in der Datenbank erhalten bleiben.

3.4 Praktischer Teil: Aufgabe 8 Kubernetes Deployment - 8 Punkte

Erzeugen Sie Kubernetes Deployment-Artefakte und legen diese für jeden Service in einem Unterordner "manifests" unterhalb des Ordners für jeden einzelnen Service ihrer Abgabe ab. Diese müssen kompatibel mit Kubernetes 1.18 oder früher sein.

1. (2 Punkte) Anwendungsdeployments für die NodeJS Anwendung bzw. Services (der rectemp service unter Nutzung der MongoDB und MQTT.
2. (1 Punkte) Stellen sie sicher, dass Lastverteilung auf mehrere Instanzen/Pods des rectemp Services möglich ist.
3. (1 Punkte) Konfigurierbare Verbindungsparameter für den Zugriff auf MongoDB via eigenständigem Kubernetes Manifest (auf das vom Deployment aus verwiesen wird)
4. (1 Punkt) Resilienz gegen Ausfall; wenn die Receive Service Anfragen nicht mehr bedienen kann soll dies erkannt und die Anwendung neu gestartet werden.
5. (1 Punkte) der rectemp Service soll maximal 128 MByte RAM und 0,1 cores konsumieren können.
6. (2 Punkte) Integrationstest: Sind alle Komponenten inkl. MongoDB auf dem Cluster deployed, funktioniert die Anwendung als ganzes und speichert die Daten in der MongoDB. Werden Services zwischenzeitlich auf 0 und später wieder hoch skaliert, können alle gesendeten Daten im rectemp-Service trotzdem angezeigt werden.