Übersicht

Im Rahmen des Praktikums *Verteilte Systeme* soll eine Anwendung aus dem Bereich Internet-of-Things entwickelt werden. Dazu sollen die Technologien *Sockets*, *RPC* (Apache Thrift), *REST*, sowie eine *Message-Oriented-Middleware* (MQTT) verwendet werden.

Rahmenbedienungen

- Das Bearbeiten der Praktikumsaufgaben findet in Zweierteams statt.
- Die Praktikumsaufgaben 1 - 4 sollten zuhause vorbereitet werden, da die Bearbeitungszeit während des Praktikumstermins u.U. nicht ausreicht um eine vollständige Lösung zu implementieren. Eine nachträgliche Abgabe ist i.A. nicht möglich.
- Die Lösungen sollen in GitLab zur Verfügung gestellt werden. Die Lösungen müssen auf den Macs im Labor laufen!
- **Jede Lösung muss getestet werden**. Schreiben Sie zu jeder ihrer Lösungen mindestens einen Test zum Testen einer funktionalen Anforderung, einer nicht-funktionalen Anforderung, sowie einen Performance-Test. Die Tests sollen weitgehend automatisiert sein.
- Die Aufgaben werden im Rahmen das der Abnahme mündlich mit dem Dozenten durchgesprochen. **Hierbei sollten beide Teammitglieder die Lösung erklären können**.
- **Nach Abschluss jeder Aufgaben soll ein Protokoll angefertigt werden**, welches die wesentlichen Ergebnisse beinhaltet. Weitere Informationen befinden sich in den jeweiligen Aufgabenstellungen. Das Protokoll muss spätestens eine Woche nach dem mündlichen Testat abgegeben werden.
- Sie können zwischen Java und C++ wählen. Das Mischen verschiedener Sprachen ist auch erlaubt. Andere Programmiersprache sind nur nach Absprache mit dem Dozent erlaubt. Das selbe gilt auch für andere Middlewarekomponenten als die oben erwähnten Sockets, REST, Apache Thrift und MQTT.
- Das Moodle stellt eine Forenfunktion bereit. Sollten Sie Probleme bei der Lösung einer Aufgabe oder der Installation der Software haben, nutzen Sie bitte das Forum. Die Dozenten werden im Forum mitlesen und ggf. Tipps geben.

Aufgabe 0

Zunächst soll eine Analyse der Anforderungen erstellt werden, die von der Gruppe in den Aufgaben 1 - 4 implementiert werden. Schauen Sie sich hierfür die Aufgabenstellungen 1 - 4 an und erstellen Sie die Anforderungen für jeden Termin *schriftlich*. Beachten Sie, dass das Bestehen der weiteren Termine davon abhängt, ob die erstellten Anforderungen umgesetzt wurden. Die Anforderungen sollen neben der zu verwendenden Programmiersprache auch Übertragungsformate und ein

Konzept der Test- und Simulationsumgebung für jeden Termin enthalten. Änderungen am Konzept sind in Lauf des Praktikums erlaubt, müssen jedoch dokumentiert werden. Am Ende des Termins sollen dem Dozenten die Anforderungen vorgestellt und ggf. ergänzt werden. Schließlich wird ein Anforderungsdokument als Protokoll abgegeben. Dieses dient als Checkliste für die folgenden Termine.

Aufgabe 1

Im ersten Schritt soll eine Anwendung entwickelt werden, die einen Kühlschrank mit Internet Anbindung simuliert. Der Kühlschrank arbeitet als Server und soll mit Hilfe von Sockets von einem einfachen Client angesprochen werden und für insgesamt 5 Artikel den aktuellen „Füllstand“ angeben. Jeder Kühlschrank hat zu jedem Zeitpunkt die vorhandene Menge aller Artikel. Die Menge jedes Artikels im Kühlschrank soll sich nach einem zu bestimmenden Zeitintervall reduzieren, so dass irgendwann der Kühlschrank leer ist. Das heißt, es müssen Sensoren simuliert werden, die den Füllstand "messen" und eine "Zentrale", die alle Messungen sammelt und mit dem Besitzer des Kühlschranks kommuniziert.

In dieser Aufgabe (und in Aufgabe 2) soll alle Kommunikation mittel Sockets erfolgen. Java oder C++ sind erlaubte Programmiersprachen.

Die Sensoren verwenden UDP, um deren Messungen an die Zentrale zu schicken. Die Zentrale soll sowohl die aktuell vorhandene Menge eines einzelnen Artikels, wie auch die Historie des Füllstandes ausgeben.

Aufgabe 2

Die zweite Aufgabe besteht darin, in den Kühlschrank einen Webserver zu integrieren. Der Webserver soll mit einem beliebigen Browser (Chrome, Firefox, Internet Explorer, Safari, etc.) angesprochen werden können und jeweils eine einfache HTML Seite mit einer Übersicht über die im Kühlschrank befindliche Menge von jeweils 5 Artikeln an den Browser schicken.

Dieser Webserver soll mit TCP Sockets realisiert werden - und nicht etwa mit fertigen Webserver-Klassen aus Bibliotheken.

Die Sensoren aus Aufgabe 1 müssen weiter laufen. Das heißt, dass die "Zentrale" gleichzeitig mit den Sensoren als auch mit Browser (HTTP Klienten) in Kontakt bleiben soll.

Aufgabe 3

In der dritten Aufgabe soll eine RPC Anbindung des Kühlschranks an ein Geschäft mittels Apache Thrift implementiert werden. Wenn die Menge eines der 5 Artikel im Kühlschrank unter einen zuvor festgelegten Schwellwert sinkt, soll der Kühlschrank über die Thrift Schnittstelle selbständig beim Geschäft den Artikel nachbestellen. Über die Schnittstelle soll es möglich sein, Artikel nachzubestellen und zu beliebigen Zeitpunkten Rechnungen über die bisher getätigten Bestellungen anzufordern. Artikel haben neben einem Namen auch Preise für eine bestimmte Menge, z.B.

€/100g. Der Webserver im Kühlschrank soll dabei so erweitert werden, dass eine Nachbestellung auch manuell über einen Webbrowser erfolgen kann, wenn ausreichend Platz vorhanden ist.

Hinweis: Wie erfahren die Sensoren, dass ein Artikel geliefert wurde? Das soll *simuliert* werden -- genau wie das erfolgen sollte, ist ein "Design-Decision", die Sie treffen sollen.

Für diese Aufgabe muss nur ein Geschäft simuliert werden, aber dieses Geschäft soll mehr als ein Kühlschrank bedienen.

Aufgabe 4

Die vierte Aufgabe besteht darin, dass die Geschäfte (!) über Message-Oriented-Middleware, d.h. MQTT, direkt von Erzeugern (Bauernhof, Metzger Importeur und andere *Lieferanten*) Informationen über Sonderangebote erhalten und dann Artikel nachbestellen kann. Dazu generieren die Lieferanten und Geschäfte periodisch Angebote bzw. Nachfragen für die angebotenen Artikel. Geschäfte konkurrieren miteinander wenn gute Angebote erscheinen („First come, first served“); schlechte Angebote können liegen bleiben bzw. vom Anbieter „verbessert“ werden, bis Käufer gefunden werden. Überlegen Sie ganz genau, wie Ihrer „virtuelle Markt“ mittels „Publish/Subscribe“-Kanälen funktionieren kann und soll (wie viele Kanäle brauchen Sie? Wer fängt an, wer wartet auf wen, wie erfahren Kunden, ob sie bzw. wie viele sie gekauft haben, usw.).