Dokumentation zur Entwicklung einer Lösung für die Speicherung von Verwaltungsschalen

Mario Letsche

September 28, 2023

1 Informationen

Der Stand an Informationen ist problematisch. Die meisten Informationsquellen sind sehr abstrakt. Die Open Industry 4.0 Alliance wollten für ihren GitHub-Account eine Plattform bieten, wo sich die Entwickler versammeln können, um an den Konzepten der Industrie 4.0 zu arbeiten. Https://github.com/OI4 führt zum Account, der aber aktuell keine Repositories hat.

Ein weiterer Ansatzpunkt ist das Projekt BaSyx von Eclipse. Auch hier mangelt es an Dokumentation. Schon bei der Installation gibt es Probleme, auch wenn man dem Video oder dem Wiki folgt. Ich weiß meine Recherche ist bisschen mager, ich hatte eben schnell die Idee das selbst zu machen, um alles besser kennenzulernen.

Am vielversprechendsten sind die Schnittstellen von Faaast. Auch, wenn die Dokumentation hier leidet, sind alle Komponenten zumindest benutzbar. Zwar weiß ich nicht, wie die Komponenten miteinander interagieren sollen, aber der Abruf der Test AAS's war zumindest möglich. Die Weboberfläche, also Faaast Portal, war in der kurzen übrigen Zeit nicht mehr zu entschlüsseln.

Ich möchte jetzt die Gelegenheit nutzen ein paar mögliche Tools zu erwähnen, die hier zum Einsatz kommen könnten und langfristig (also wenn das Projekt richtig genutzt werden soll) sollten. Einzelne Tools kann ich dabei auch nur erwähnen, da mir selbst noch die Expertise für diese Sachen fehlt. Wichtig wäre schon mal REST oder etwas in der Art. Die Kommunikation zwischen der Datenbankschnittstelle und einem potentiellen Webbrowser (oder Webapp) mit MQTT ist nicht wirklich realistisch. Nicht nur sind die Packages, die dafür benötigt werden veraltet, es entspricht auch nicht dem aktuellen Standard der Technologien. Eine der Möglichkeiten, wäre MQTT (für die Kommunikation zwischen App-Komponenten) mit Kafka zu ersetzen. Kafka ist ähnlich wie MQTT aber abgestimmt für Java. Um zusätzliche Bauchschmerzen zu verhindern könnte man den Part mit der JavaScript Web-App auch ersetzen mit Vaadin. Ich habe damit keine Erfahrung, aber es ermöglicht wohl die Gestaltung von Webkomponenten mit Java Code. Alle genannten Tools sind unten in den Quellen zu finden.

Eine der besprochenen Probleme war die gemeinsame Code-Plattform für die HWL. Die Möglichkeit das umzusetzen, wäre ein gemeinsamer GitHub Account für die Firma, wo Mitarbeiter dann registriert werden können. So können alle, die an einem Projekt arbeiten diesem Repository zugeordnet werden.

Meine Empfehlung für einen Server ist in erster Linie ein virtueller Server. Solange das Projekt im kleinen Umfang ist, sollte für die ersten Tests ein virtueller Server genutzt werden, z. B. Hostinger. Da kann man nichts kaputt machen. Da lassen sich so Dinge wie RAM und genutzter Speicher auch individuell anpassen, was für den Anfang ja wichtig ist, solange ihr nicht wisst, wie viele Ressourcen benötigt werden.

2 Implementierung

Eine mögliche Lösung ist der native Ansatz. Dafür habe ich ein Projekt über Maven eröffnet, also eine Java Schnittstelle erstellt, die über MQTT mit einer MongoDB Datenbank kommuniziert.

Für den Umgang mit der Software braucht man folgendes:

1. Maven (optional): Ich habe einen Wrapper für Maven eingebaut, so dass man die Maven Funktionen ohne eigene Maven Installation in diesem Projekt nutzen kann.
2. MongoDB und (für einfache Bedienung und Überwachung) MongoDB Compass.
3. NodeRed (o. Ä.) zum Erhalten und Versenden von Nachrichten.

Um das Projekt aufzubauen gibt man im Terminal (im Projektordner) den Befehl: "mvn clean install" ein. Danach kann die Anwendung über die gewählte IDE gestartet werden und zwar über die Main-Klasse. (Später kann auch die .exe gemacht werden, bzw. das passiert mit mvn clean install schon, aber zum entwickeln ist es so halt einfacher).
Zum Verständnis ist ein kurzer Überblick wichtig, welche Komponenten wir statisch gesehen brauchen.

1. MongoDB Anschluss. Über den Datenbank Driver, den wir über die Maven Dependencies in der pom.xml Datei bekommen haben, wollen wir nicht nur auf die Datenbank zugreifen, sondern auch die üblichen Funktionen ausüben können. Die CRUD Operationen, also Create, Read, Update, Delete, sind für diese Funktionen da. Die Implementation soll also diese Operationen auf eine möglichst einfache und sichere Methode beinhalten. Bestenfalls soll diese Anschluss austauschbar sein, für jegliche Datenbank.
2. MQTT Anschluss. Über den MQTT Driver, den wir über die Maven Dependencies in der pom.xml Datei bekommen haben, können wir Daten mit dem MQTT-Format an einen gewählten Broker schicken oder von dem

Broker erhalten. Dieser Anschluss soll im besten Fall austauschbar sein für andere Datenübertragungsmittel oder auch andere Formate (bisher JSON) oder auch mit einem Logger.

3. Eine Schnittstelle für die beiden Anschlüsse. Der einfache Ansatz, also die beiden Klassen zu implementieren und die Daten auszutauschen, ist nicht nur fehleranfällig, da man auf beiden Seiten die jeweils andere berücksichtigen muss, es ist auch sehr statisch und nur schwer austauschbar. Das ist jetzt beim bisherigen Code vielleicht nicht so schlimm, wenn das ganze aber größer wird, kommt man da schnell in Probleme. Daher bietet sich eine Schnittstelle an, die unabhängig von den beiden Anschlüssen agiert. Sie weiß nur, was sie schicken und bekommen soll und verteilt die Arbeit an die jeweilige Implementation der Anschlüsse. Da die Anschlüsse nicht nur Daten bekommen sondern auch an die anderen Anschlüsse verteilen sollen, wurde ein Interface eingebaut, dass bestimmte Callback-Methoden Implementiert, damit die beiden Anschlüsse bei Bedarf die Daten wieder an die Schnittstelle zurückschicken können. Ganz wichtig: Holt euch jemanden, der sich mit diesen Sachen auskennt, aber halt auch auf diesem Niveau, dass das eine richtige App wird und eben nicht mich.

Durch diese drei Komponenten verlieren wir jegliche Abhängigkeit von den Einzelteilen untereinander. Wenn wir also die Datenbank durch eine andere ersetzen wollen, dann können wir dafür den Anschluss coden, der Schnittstelle bescheid sagen und dann weitermachen.

Jetzt noch ein paar TODO's:

1. Die Datenbank ist noch sehr offen, also hat noch keine Einschränkungen, was da überhaupt rein soll.
2. Es gibt noch wenige Sicherheitsmechaniken. Ich weiß nicht wie anfällig das System ist, da müsste vielleicht ein IT-Sec Experte ran.

3 Quellen bzw. wichtige Links für mich

https://developers.redhat.com/articles/2022/04/05/developers-guide-using-kafka-java-part-1

https://www.javaguides.net/p/rest-api-tutorial.html

https://vaadin.com/

https://www.hostinger.de/

https://github.com/FracturedRodey/DataManager