Dokumentation Projekt Industrie 4.0

NICO BOLLMANN, ALEXANDER DECKERT, TIM GAUDICH, TORSTEN SCHMITT, MALGORZATA WDZIECZNY

Gruppe Schmitt

Inhaltsverzeichnis

Inhalt

Aufgabenstellung

Durch die Modernisierung vieler Industrieanlagen auf den aktuellen Industrie 4.0 Standard, war es die Aufgabe des Projektes eine Verarbeitungspipeline für eine simulierte Produktionsstraße zu erstellen.

Eine wichtige Anforderung war hierbei das Verwenden von Docker, einem Virtualisierungstool, um die verwendeten Komponenten autark in einer virtuellen Umgebung starten zu können. Die einzelnen Komponenten sollen in einem Docker-Compose zusammengefasst werden.

Durch die gegebene Taktstraße sollen die beiden Message-Broker-Systeme *Apache ActiveMQ* und *Apache Kafka* genutzt werden, um Meldungen der Produktionsstraße zu empfangen. Hinzu kommt, dass ein Teil der Informationen auf das Dateisystem geschrieben wird.

Außerdem soll ein Cluster Computing Framework, wie zum Beispiel *Apache Spark*, eingesetzt werden, um Daten der Taktstraße mittels „big data processing“ zu analysieren.

Entwicklung

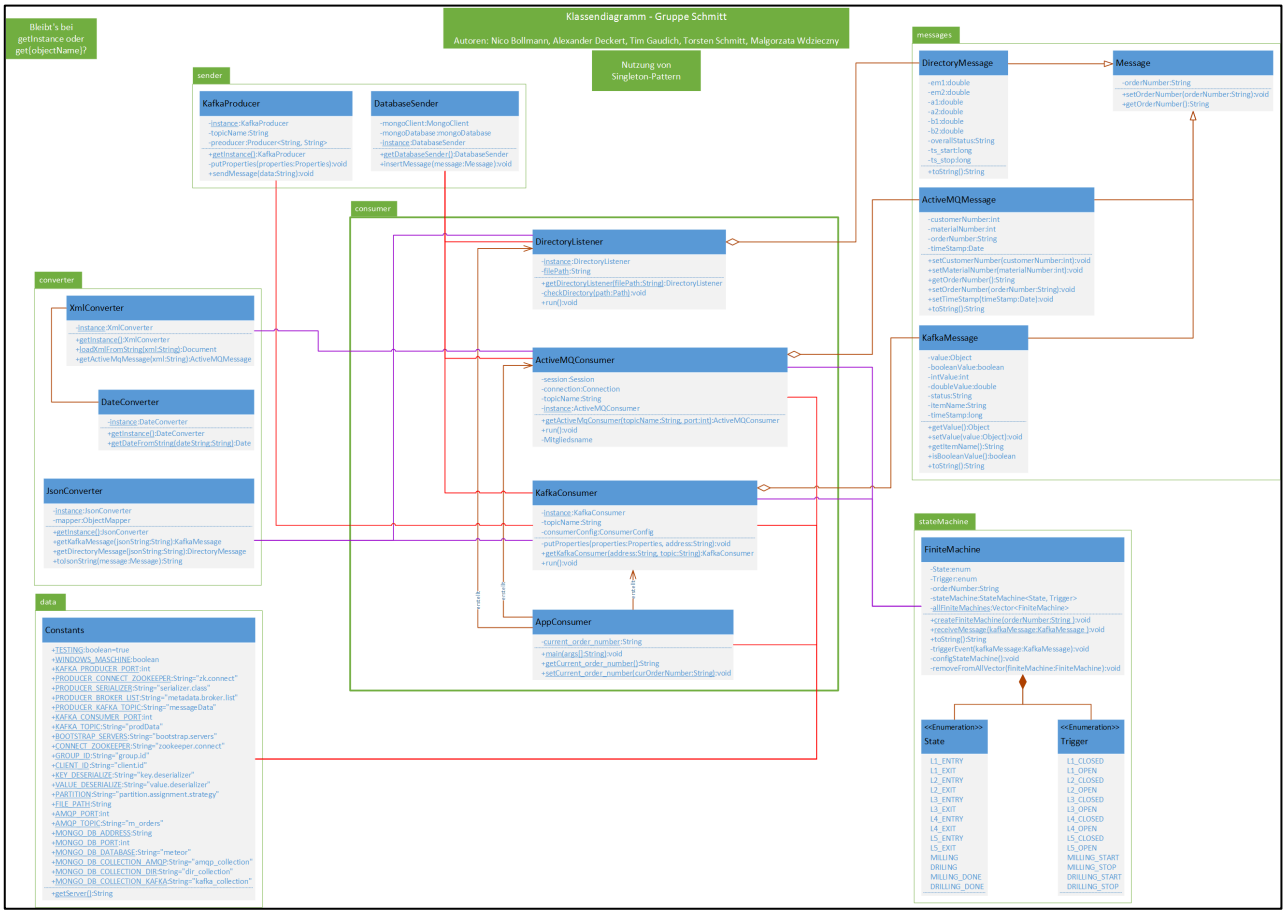
Code-Konventionen

Um die Wart- und Lesbarkeit des entwickelten Codes zu fördern, wurde zu einem frühen Zeitpunkt im Projekt festgelegt, dass geschriebener Code einigen Konventionen folgen und kommentiert werden soll.

Programme

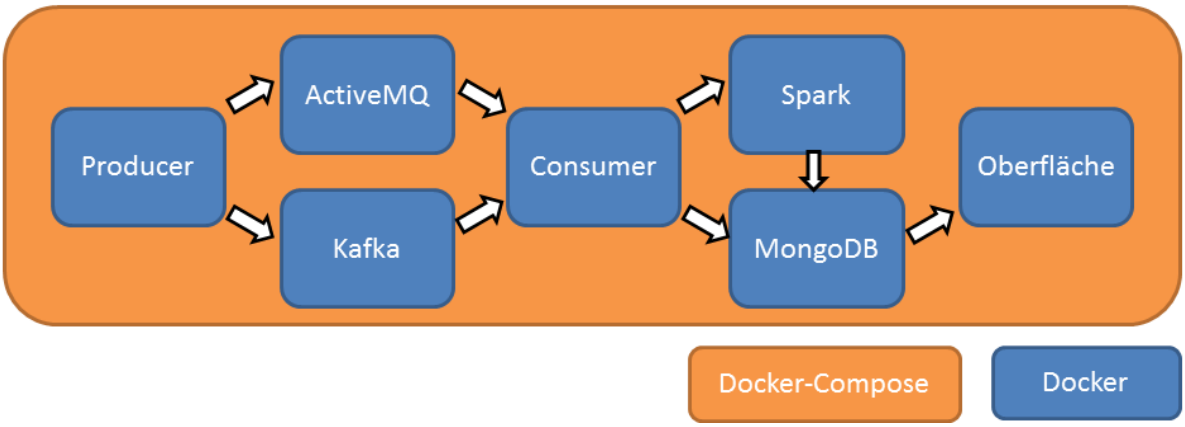
Zur Übermittlung der Meldungen der Produktionsstraße wurde eine Java-Anwendung entwickelt. Diese nutzt, sofern möglich, das **Singleton-Pattern** und sendet empfange Meldungen in ein neues Kafka-Topic, sowie an eine dokumentenorientierte Datenbank (MongoDB). Außerdem wurde ein Zustandsautomat in dieser Anwendung simuliert, wodurch eine mehrteilige Nutzung der Produktionsstraße ebenfalls kein Problem darstellen sollte (wurde nicht mehr getestet, war aber auch keine gestellte Anforderung; kann in weiterem Vorgehen fest integriert werden).

Im Folgenden ist das Klassendiagramm der Backend-Applikation zu sehen:



Um alle Meldungen der Taktstraße zu empfangen, werden verschiedene Threads gestartet, welche die empfangenen Nachrichten in die jeweiligen Message-Objekte übertragen (mit Hilfe verschiedener Converter). Diese werden nach kleineren Anpassungen in Objekte konvertiert, welche an die dokumentenorientierte Datenbank geschickt werden können.

Aufbau der Pipeline



Die Verarbeitungspipeline beginnt mit der Erzeugung der Daten aus dem Producer (Simulation der Taktstraße). Dieser verteilt die Produktionsdaten unter anderem über *Apache ActiveMQ* und *Apache Kafka*. Diese Daten werden von einem Consumer gesammelt, verpackt und an *Apache Spark* und die Datenbank (MongoDB) weitergeleitet. Aus dieser liest die Oberfläche die nötigen Daten aus und zeigt diese an.

Im abgegebenen Docker-Compose funktioniert die Verarbeitung vom Producer bis zur MongoDB. Auch Spark ist eingebunden, hat jedoch keine Logik implementiert, um in einer gewissen Weise Daten zu analysieren. Die Oberfläche ist ebenfalls im Docker-Compose eingebunden und startet ohne Fehlermeldung. Jedoch wird unter der festgelegten URL diese nicht angezeigt. Unter **docker-compose ps** werden alle Container ohne Fehlermeldung angezeigt.

Visualisierung

Das Hauptaugenmerk des Projektes war die Visualisierung der Daten. Diese sollen für einen Endnutzer ansprechend dargestellt werden.

Die Umsetzung der Visualisierung wurde mit *AngularJS* der Firma Google realisiert. Durch das „2-Way-Databinding“ ist sichergestellt, dass die Oberfläche dynamisch mit Daten versorgt werden kann. Außerdem wird Meteor eingesetzt, welches durch eine integrierte MongoDB die Oberfläche leicht mit Daten versorgen kann. Zusammen mit Meteor und AngularJs spricht man auch vom „3-Way-Databinding“: Die Datenbank teilt dem Frontend automatisch mit, wenn sich der Datenbestand geändert hat und pusht diese Änderungen auch. Durch das Databinding von Angular werden diese Daten danach auch gleich entsprechend aktuell aufbereitet und angepasst. Durch Buttons in der Oberfläche können die Abfragen für die MongoDB auch jederzeit manipuliert werden.

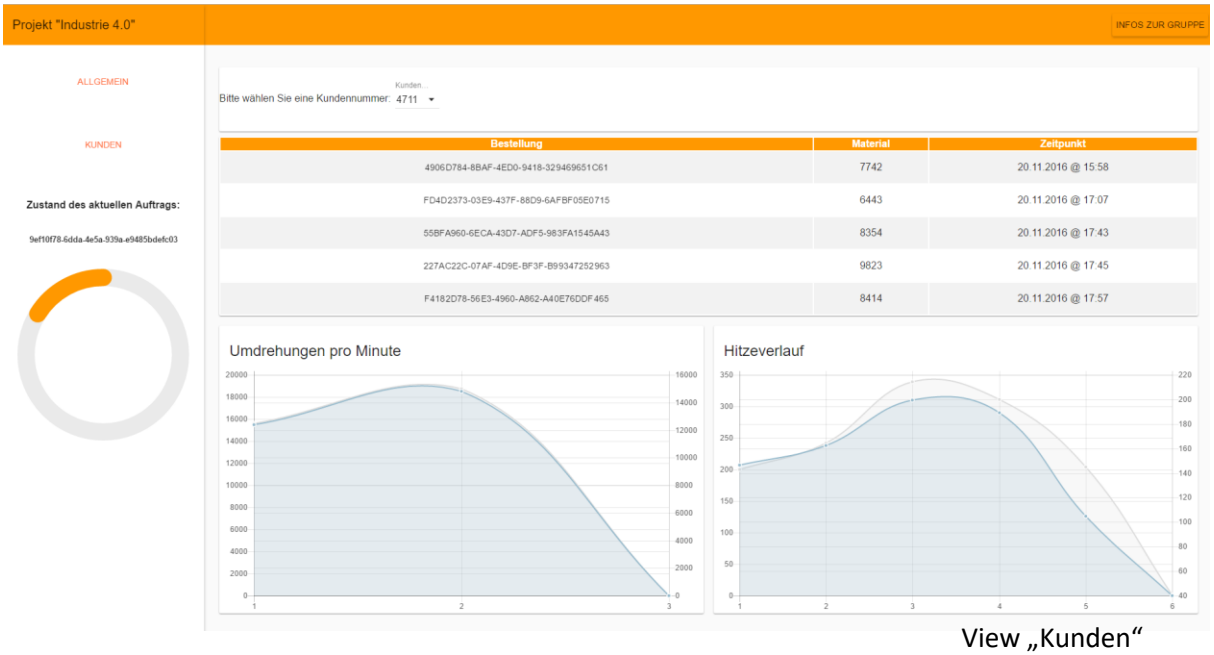
Der Aufbau der Oberfläche teilt sich in verschiedene „Views“ ein, welche über das Menü am linken Rand zu erreichen sind. Zu diesen Views gehören die beiden Bereiche „Allgemein“ und „Kunden“. Mit einem Klick auf die entsprechende View, wird der Inhalt dynamisch in einen vorher definierten Platzhalter geladen. Die Navbar, sowie die Toolbar sind über Templates statisch eingebunden. In der Navigation wird neben den Buttons für die Views, der aktuelle Auftrag sowie dessen Fortschritt angezeigt. Die Toolbar enthält zur Erinnerung einen Button, welcher Informationen über die Gruppe und deren Mitglieder offenbart.

View „Allgemein“

In der View „Allgemein“ sind die wichtigsten Daten auf einen Blick gesammelt. Es wird die Bestellnummer des aktuell bearbeiteten Auftrags sowie die aktuellen Maschinenwerte für Milling- und Drilling- Speed/Heat angezeigt. Mit einem Klick auf die dafür vorgesehenen Buttons kann zwischen den Aktuellen- und den Maximalwerten umgeschaltet werden. Durch einen Klick auf die einzelne Card wird auf das Diagramm für die Card umgeschaltet, auf den die letzten 20 Werte im Verlauf visualisiert wurden. Im unteren Bereich ist eine Tabelle abgebildet, in der nach

Kundennummern sortiert, die Anzahl der Bestellung und die Durchführung der letzten Bestellung sichtbar ist.

In der View „Kunden“ kann im oberen Bereich eine Kundennummer ausgewählt werden. Nach der Auswahl dieser erscheint im mittleren Bereich eine Tabelle mit Einträgen über alle getätigten Bestellungen. Wenn ein Klick auf eine Bestellung erfolgt, wird der Verlauf der Umdrehungsgeschwindigkeiten sowie der Hitzeverläufe des gewählten Auftrags grafisch dargestellt.



View „Kunden“

In Bereich „Kunde“ bekommt der Nutzer die Möglichkeit eine Kundennummer sowie eine Bestellnummer auszuwählen. Hat er die Auswahl abgeschlossen, werden ihm die angeforderten Daten zu der Bestellung gezeigt. Außerdem kann zu jeder Bestellung der Verlauf der Maschinenwerte ermittelt werden.

Analyse der Daten

Um zunächst einen groben Überblick über die erzeugten Produktionsdaten zu bekommen, wurden die Maschinendaten in eine Exceldatei exportiert, welche im Anschluss interpretiert werden konnte.

Als Ergebnis konnte herausgefunden werden, dass lediglich eine Kategorisierung und keine Identifizierung der Materialien möglich ist, dabei hat sich gezeigt, dass sowohl die Drilling- als auch die Milling-Geschwindigkeit in Abhängigkeit zum verwendeten Material stehen.

Kategorie	Materialien	MS (1)	MS (2)	MH (1) + MH(2)	MH (3) + MH (4)	DS (1)	DS (2)	DH (1) + DH (2)	DH (3) + DH (4)
1	7423; 4248; 5653; 6443; 7134; 7432	9850	11500	[100;135]	[175;140]	11280	14980	[100;140]	[175;135]
2	8354; 9823; 7742; 8932; 8414; 8235	12480	15000	[140;185]	[245;195]	15500	18500	[190;260]	[330;275]

DH = Drilling-Heat; DS = Drilling-Speed; MH = Millling-Heat MS = Milling-Speed

Des Weiteren konnte herausgefunden werden welche Materialien bestimmte Geschwindigkeiten und Wärmeentwicklungen erzeugen. Die Zusammenhänge sind in der obenstehenden Tabelle zu erkennen. Bei MH (1) und MH (2) erhitzt sich die Fräse, bei MH (3) und MH (4) kühlt diese ab. Bei DH (1) und DH (2) erhitzt sich der Bohrer, bei DH (3) und DH (4) kühlt dieser wieder ab.

Im weiteren Ausblick des Projektes wäre es möglich die manuell ausgeführten Auswertungen und Analysen über *Apache Spark* zu automatisieren. Dabei könnten die Analysen deutlich schneller sowie ausführlicher ausgeführt werden.

Zuständigkeiten

Der Bereich der Dockerisierung sämtlicher Anwendungen und Technologien wurde von **Alexander Deckert** übernommen. Zu diesem gehörten das Erstellen von Dockerfiles sowie die Erstellung eines Docker-Compose.

Die Backendprogrammierung im Rahmen des Projektes wurde durch **Nico Bollmann** durchgeführt. Zu den hierbei durchgeführten Tätigkeiten gehören unter anderem die Entwicklung des Kafka-Consumers, die Entwicklung einer State-Machine, die Ausgabe der Daten in die Datenbank sowie die (zeitweise) Bereitstellung der Daten zur Analyse in Excel-Dokumenten.

Zuständig im Bereich der UI-Entwicklung war **Tim Gaudich**. Er hat eine gemeinsam designte Benutzeroberfläche und mit den Technologien Meteor, Javascript und AngularJS ansprechend sowie technisch sauber umgesetzt. Auf diesen Bereich wurde, wie bereits erwähnt, ein Schwerpunkt in diesem Projekt gelegt.

Das Erstellen des UML-Klassendiagramms, fiel in den Aufgabenbereich von **Malgorzata Wdzieczny**.

Die organisatorischen Arbeiten wurden von **Torsten Schmitt** durchgeführt. Zu diesem Bereich gehörten die Organisation und Leitung von Meetings sowie die Strukturierung des Arbeitsumfeldes. Auch die Präsentation der Gruppenergebnisse gehörte in diesen Aufgabenbereich.

Das Thema rund um die Verwendung von Apache Spark wurde aufgrund der Komplexität von **Nico Bollmann** und **Malgorzata Wdzieczny** bearbeitet. Infolge dessen konnte Spark erfolgreich in die Pipeline integriert werden.

Die manuelle Analyse wurde von **Malgorzata Wdzieczny, Nico Bollmann** und **Alexander Deckert** durchgeführt, um durch verschiedene Sichtweisen möglichst optimale Analyseergebnisse zu erhalten.

Startanleitung der Pipeline

Sobald das Archiv „Dockerfiles“ verfügbar ist, muss mittels einer SHELL in dieses gewechselt werden. In diesem Verzeichnis befinden sich einzelne Unterverzeichnisse für die verschiedenen Technologien, sowie eine Datei mit dem Namen **docker-compose.yml**. Diese kann mit dem Befehl *docker-compose build* gebuildet werden. Sobald dieser Vorgang beendet ist, kann das Compose mit dem Befehl *docker-compose up* gestartet werden.

Installation ohne Docker

Die Java-Dateien können auch manuell gestartet werden:
Die Taktstraße kann mit dem Befehl *java -jar TaktstrasseOpcServer-0.0.1-SNAPSHOT.jar -o \dockerDir -d 1000 -amqp tcp://192.168.99.100:32768 -kafka 192.168.99.100:1000 -topic prodData* gestartet werden. Die normale Consumer-Jar mit dem Befehl *java –jar Consumer.jar*. Gegebenenfalls muss die Consumer-Jar neu erzeugt werden und in der Klasse *Constants* Anpassungen an den Ports gemacht werden.

Aufgrund erwähnter technischer Probleme funktioniert die Anzeige der Daten in der Oberfläche nicht. Um sich die Daten anzeigen zu lassen, muss auf einen manuellen Start ohne Docker Compose zurückgegriffen werden. Dazu sind folgende Schritte durchzuführen:

- Installation von Meteor nach der Anleitung auf der Github-Seite: https://github.com/nbollmann/projekt-dhbw/tree/master/Oberflaeche/industrie_40