

Duale Hochschule Baden-Württemberg

Mannheim

**Portfolio**

**Studiengang Wirtschaftsinformatik**

Studienrichtung Software Engineering

Verfasser: Nele Ecker, Lars Langhammer, Mireille Puschmann, Jan Vögeli, Philip Wagner

Gruppe: Ecker

Kurs: WWI14 SE A

Studiengangsleiter: Prof. Dr.-Ing. Jörg Baumgart

Modul: Projekt

Lehrveranstaltung: Projekt I

Dozent: Tarek Auel

Abgabedatum: 20. November 2016

**Inhaltsverzeichnis**

[**Abbildungsverzeichnis** II](#_Toc467424783)

[**1 Git Workflow** 1](#_Toc467424784)

[**2 Pipeline** 2](#_Toc467424785)

[**2.1 Docker** 2](#_Toc467424786)

[**2.2 Taktstraße** 2](#_Toc467424787)

[**2.3 Apache Kafka** 2](#_Toc467424788)

[**2.4 Apache ActiveMQ** 3](#_Toc467424789)

[**2.5 Consumer** 3](#_Toc467424790)

[**2.6 Apache Spark** 3](#_Toc467424791)

[**2.7 Datenbank** 4](#_Toc467424792)

[**2.8 Webserver** 6](#_Toc467424793)

[**2.9 UI** 6](#_Toc467424794)

[**3 Änderungen zwischen Präsentations- und dem Abgabetermin** 7](#_Toc467424795)

[**4 Anleitung für die Pipeline** 8](#_Toc467424796)

[**Literaturverzeichnis** 9](#_Toc467424797)

# **Abbildungsverzeichnis**

[Abbildung 1: Pipeline 2](#_Toc467425738)

[Abbildung 2: Funktionalität von Spark Streaming 4](#_Toc467425739)

[Abbildung 3: Entity Relationship Diagramm 5](#_Toc467425740)

# **1 Git Workflow**

Es wurde sich entschieden GitHub zu verwenden. Das Branching Modell für GitHub ist das Feature Branching. Für das Feature Branching sprach, dass durch dieses Modell eine Trennung der einzelnen Feature vorgenommen wird. Zudem bietet dieses Modell eine bessere Übersichtlichkeit. Fertige Komponenten sollen in den Master Branch commitet werden.

Ein weiteres Modell wäre die Aufteilung der Branches nach Gruppenmitgliedern gewesen. Hierbei kommt das Problem auf, dass Gruppenmitgliedern an den gleichen Features wie zum Beispiel den Docker oder der Datenbank gearbeitet haben.

Link zum Git-Repository: https://github.com/philipwagner1994/ProjectDHBW

# **2 Pipeline**

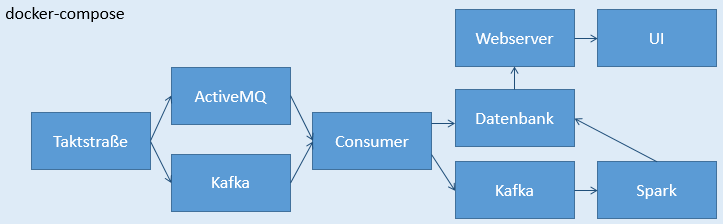


Abbildung 1: Pipeline

## **2.1 Docker**

Für die Pipeline gibt es Docker für die Taktstraße, für ActiveMQ, für die Datenbank, für das UI, für Kafka, für den Consumer, für Spark und für den WebServer. Durch das Docker-Compose werden alle Docker in einem Docker gestartet.

## **2.2 Taktstraße**

De Taktstraße ist eine Applikation, welche die Produktionsdaten für Kafka liefert. Diese Applikation wurde von Professor Reichwald zur Verfügung gestellt.

## **2.3 Apache Kafka**

Kafka ist ein Message Broker, der asynchrone Nachrichte sendet. Diese Nachrichten stellen die Daten der Taktstraße dar. Kafka nimmt die Nachrichten auf und wartet dann bis der Consumer diese Nachrichten konsumiert. Zusätzlich überträgt Kafka Datenströme und sendet diese weiter an Spark.

## **2.4 Apache ActiveMQ**

Ein zusätzlicher Message Broker stellt ActiveMQ dar. Dieser wandelt eine synchrone Kommunikation in eine asynchrone Kommunikation um. Dadurch übernimmt ActiveMQ die Rolle eines Servers. Dieser liefert ERP-Daten. Somit werden Daten aus zwei Quellen simuliert.

## **2.5 Consumer**

Der Consumer verbindet und vereinheitlich die Daten aus verschiedenen Quellen, da diese in verschiedenen Formaten vorliegen. Die Quellen hierbei sind Kafka, ActiveMQ und eine Datei auf dem Dateisystem. Zusätzlich sorgt der Consumer für eine Zuteilung der unterschiedlichen Datensätze zu den korrekten Produkten über eine State Maschine. Die State Maschine filtert die Daten, welche für ein Produkt wichtig sind heraus. Nach der Zuordnung schreibt der Consumer die Daten in die Datenbank und gibt die Daten zusätzlich noch an einen Server weiter.

## **2.6 Apache Spark**

Spark ist ein Framework für Cluster-Computing. Spark teilt sich in fünf Komponenten, die teilweise voneinander abhängig sind. Diese sind Spark Core, Spark SQL, Spark Streaming, MLlib (Machine Learning Library) und GraphX. Spark Core bildet die Grundlage des Spark Systems. Die grundlegenden Infrastruktur-Funktionalitäten werden durch Spark Core bereitgestellt. Spark SQL ermöglicht es RDD, verteilte Datensätze in ein Data Frame zu wandeln. Auf Data Frames könne SQL Abfragen getätigt werden. Zusätzlich werden Data Frames als temporäre Tabellen gespeichert. Spark Streaming, welche auf in der Pipeline verwendet wird, ermöglicht die Verarbeitung von Datenströmen. Die Datenströme werden hierbei in einzelne Pakete zerlegt, auf diese dann Transformationen ausgeführt werden können. In dem Projekt nimmt wie in Abbildung 2 gezeigt wird, Spark Streaming den Datenstrom von Kafka auf und schreibt die daraus transformierten Daten in die Datenbank.



Abbildung 2: Funktionalität von Spark Streaming[[1]](#footnote-1)

Eine Funktionsbibliothek stellt MLlib dar. Sie macht Machine Learning Algorithmen für Spark-Systeme verfügbar.

GraphX basiert auf Spark und ist ein verteiltes Framework, welches Graphen berechnet.

## **2.7 Datenbank**

Die Datenbank basiert aus PostgreSQL. Für die Verwendung von PostgreSQL sprach, dass PostgreSQL eine relationale Datenbank ist. Durch die relationale Datenbank ist eine leichte Aggregation der Daten möglich. Zusätzlich werden Datenredundanzen verhindert. Ein weiteres Argument für die relationalen Datenbank war, dass diese bereits in den Vorlesungen besprochen wurden. Analysen werden über SQL Abfragen vorgenommen. Die Datenbank enthält nur historische Daten. Livedaten werden direkt an das UI übermittelt.

In der Datenbank werden zwei Datenbanktabellen erzeugt.

Die Tabelle Data enthält Daten von den ERP-Daten und den ActiveMQ-Daten.

Die Tabelle Properties enthält Daten aus Kafka. Die Attribute der Tabellen sind in Abbildung 3 veranschaulicht.

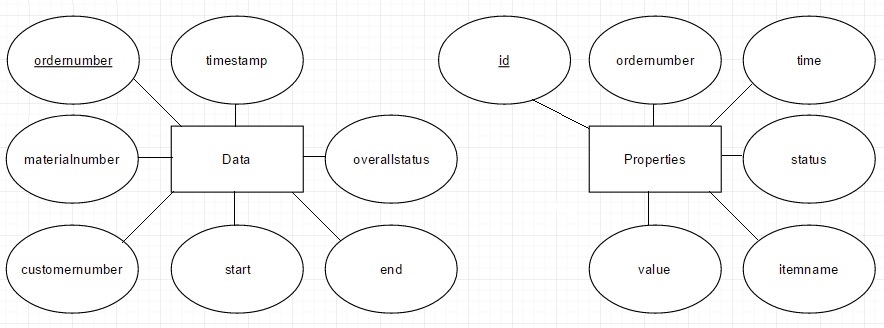


Abbildung 3: Entity Relationship Diagramm

## **2.8 Webserver**

Der Webserver, wie er in der Pipeline aufgezeigt wird teilt sich in zwei Server auf. Zu einem gibt es einen Server, der Daten von dem Consumer enthält. Dieser Server reagiert auf Liveanfragen. Ein weiter Server ist ein Server, der auf Anfragen zu historischen Daten, die sich in der Datenbank befinden, reagiert. Dazu liest er die geforderten Daten aus der Datenbank aus und teilt diese dem UI mit.

## **2.9 UI**

Die grafische Oberfläche basiert auf OpenUI5. Für OpenUI5 sprach, dass es ein einfaches Grundgerüst besitzt. Zudem sind leichte Anpassungen an dem UI möglich. Zusätzlich wurde die c3.js-Bibliothek, die auf d3.js basiert eingebunden. Probleme bei OpenUI5 waren, dass viele Eigenkonfigurationen vorgenommen werden mussten, welche einen hohen Aufwand darstellten. Das UI zeigt zwei Arten von Daten an. Dur h den einen Server werden Livedaten angezeigt. In den Grafiken werden durch einen Server historische Daten an das UI mitgeteilt und angezeigt.

# **3 Änderungen zwischen Präsentations- und dem Abgabetermin**

Zwischen dem Präsentationstermin und dem Abgabetermin gab es noch einige kleine Änderungen.

Schleifen, die in den Graphen angezeigt wurden, werden nun nicht mehr angezeigt.

Der Durchlauf des Werkstückes auf der Taktstraße wird angezeigt.

Die Graphen, welche die Geschwindigkeit und die Temperatur veranschaulichen werden auf einer Seite angezeigt, sodass ein Vergleich zwischen diesen beiden Werten möglich ist.

# **4 Anleitung für die Pipeline**

Dos2unix

# **Literaturverzeichnis**

Spark Streaming Programming Guide. Online verfügbar unter http://spark.apache.org/docs/latest/streaming-programming-guide.html, zuletzt geprüft am 20.11.2016.

1. Spark Streaming Programming Guide [↑](#footnote-ref-1)