Dokumentation der Softwarearchitektur des Projekts Twitter-Dash

Hahn, Bastian
b.hahn@oth-aw.de

Kleber, Martin
m.kleber2@oth-aw.de

Klier, Andreas
a.klier@oth-aw.de

Kreussel, Lukas
l.kreussel@oth-aw.de

Paris, Felix
f.paris1@oth-aw.de

Ziegler, Andreas
a.ziegler1@oth-aw.de

Abstract—In diesem technischem Report wird die Softwarearchitektur des Projekts Twitter-Dash vorgestellt. Das Projekt wurde im Rahmen der Vorlesung Big Data and Cloud Computing (bdcc) implementiert. Ziel der Implementierung ist es, ein Daschboard zu erstellen, dass aktuelle und historische Informationen zu Twitter Trends anzeigen und analysieren kann.

Index Terms—Twitter, Big Data, Database, Web UI, Data Analysis

I. Introduction

Bei der Applikation Twitter-Dash werden von der Twitter API alle 15 Min die aktuellen Trends abgegriffen und in einer lokalen Datenbank gespeichert. Die Kommunikation zwischen den einzelnen Services ist über eine grpc Schnittstelle realisiert. Die Interaktion des Anwenders mit der Applikation erfolgt dabei über eine Web-UI basierend auf dem React Webframework.

II. Architektur Allgemein

Als Architektur wird ein Ansatz verfolgt, der sich stark an Microservices orientiert. Dadurch wird eine abgeschlossene Logik von Front- und Backendkomponenten erreicht, die jeweils von einem Subteam entwickelt wird. Diese gekapselten Einheiten können jeweils als isolierte Applikation der Containervisualisierung Docker betrieben werden. Durch eine zentrale Konfigurationsdatei des Containers können diese einfach zwischen den Teammitgliedern ausgetauscht und ausgeführt werden. Damit wird eine hohe Portabilität gewährleistet. Die Kommunikation der Teilsysteme wird durch eine sprachunabhängige gRPC-Schnittstelle angeboten. Dadurch wird es möglich einzelne Komponenten des Gesamtsystems auszutauschen, ohne größere Veränderungen an den anderen Microservices vornehmen zu müssen. [1]

III. Datenakquise

IV. Datenverarbeitung

V. Datenbank

Alle Datenbank Anfragen werden durch einen database service abstrahiert, der die Funktionen als gRPC Schnittstelle anbietet. Dabei laufen die Datenbank und der database service in jeweils einem eigenen Container. Durch die gRPC Schnittstelle können die Daten typsicher und Platform sowie Programmiersprache unabhängig an den database service übergeben werden und von diesem zurückgeliefert werden.
Vom database service werden folgende Funktionen angeboten:
Zum speichern in die Datenbank:
Trends speichern (StoreTrends)

- Beschreibung: Speichert Trends in die Datenbank
- Parameter:
  - Zeitstempel
  - Liste von Trends mit folgenden Attributen:
    - trendtype
    - name
    - country
    - placement
    - tweetVolume24
- Rückgabe: Keine

Sentiment speichern (StoreSentiment)

- Beschreibung: Speichert Sentiment in die Datenbank
- Parameter:
  - Liste von Sentiments mit folgenden Attributen:
    - tweet
    - Sentiment
    - topic
- Rückgabe: Keine

zum Laden aus der Datenbank:
aktuelle Trends abrufen (GetCurrentTrends)

- Beschreibung: Lädt aktuelle Trends aus der Datenbank
- Parameter:
  - country
  - limit
- Rückgabe:
  - Zeitstempel
  - Liste von Trends mit folgenden Attributen:
    - trendtype
    - name
    - country
    - placement
    - tweetVolume24

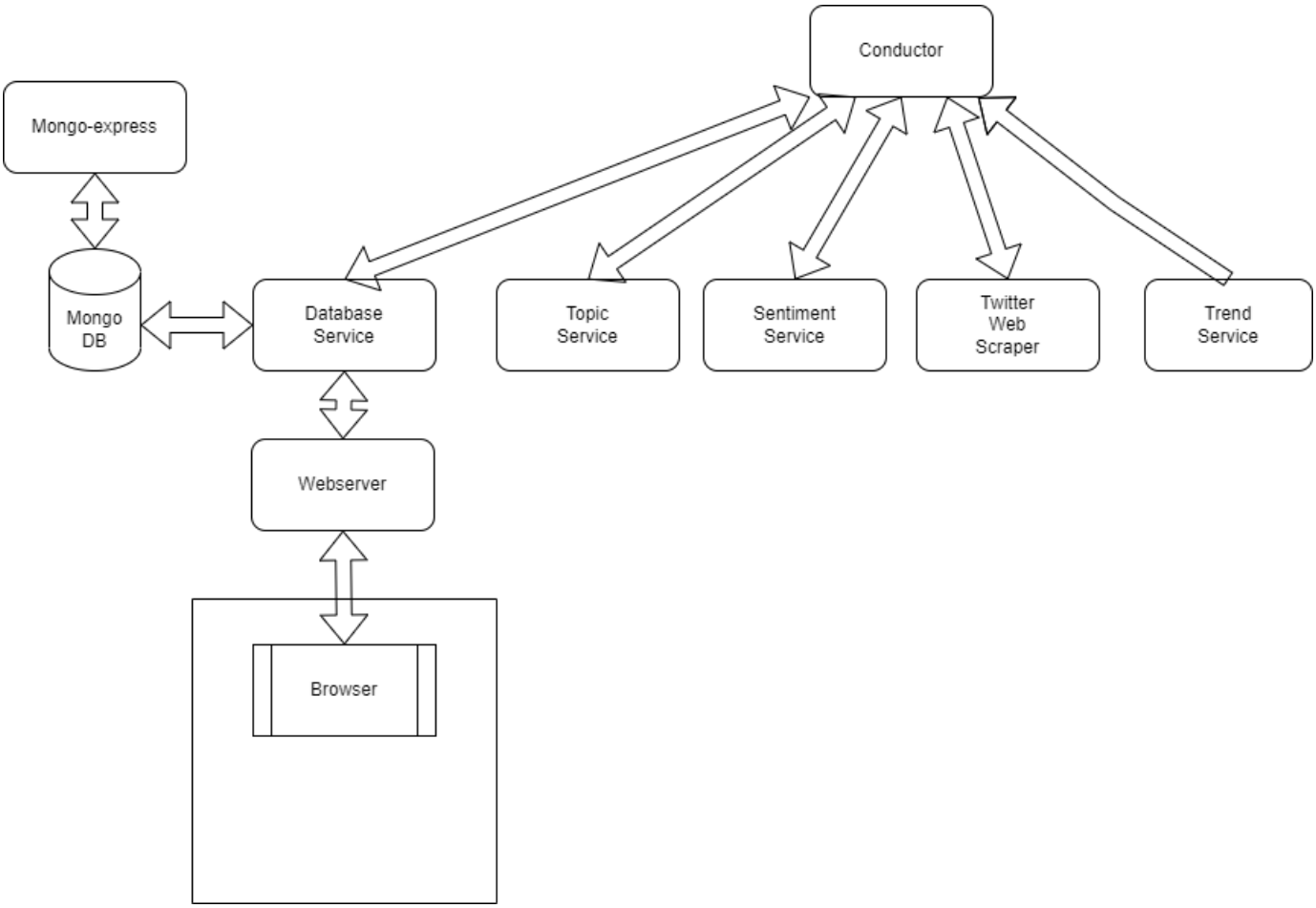

Fig. 1. Übersicht Gesamtsystem

smallskip vergangene Trends abrufen (GetRecentTrends)

- Beschreibung: Lädt vergangene Trends aus der Datenbank
- Parameter:
  - hastag
  - country
  - startdate (optional)
  - enddate (optional)
- Rückgabe:
  - Liste an Trends mit folgenden Attributen:
    - Zeitstempel
    - Liste von Trends mit folgenden Attributen:
      - trendtype
      - name
      - country
      - placement
      - tweetVolume24

Vorhandene Länder abrufen (GetAvailableCountries)

- Beschreibung: Lädt vorhandene Länder aus der Datenbank
- Parameter: Keine
- Rückgabe: Liste mit allen Ländern, die in der Datenbak vorhanden sind

Trends speichern (GetAvailableSentimentTrends)

- Beschreibung: Lädt vorhandene Trends aus der Datenbank
- Parameter:
  - query
  - limit
  - country
- Rückgabe: Liste an vorhandenen Sentiments

aktuelle Sentiment abrufen (GetCurrentSentiment)

- Beschreibung: Lädt aktuellen Sentiment aus der Datenbank
- Parameter:
  - Trendname
  - Country
- Rückgabe: Sentiment

vergangene Sentiment abrufen (GetRecentSentiment)

- Beschreibung: Lädt vergangene Sentiment aus der Datenbank
- Parameter:
  - Trendname
  - Country
  - startdate (optional)
  - enddate (optional)
- Rückgabe: Liste an Sentiments

DB auf Tweet prüfen (GetUniqueTweets)

- Beschreibung: Gibt zurück, welche Tweets noch nicht in der Datenbank sind
- Parameter: Liste an Tweet IDs
- Rückgabe: Liste an Tweet IDs

Vom Backenend erzeugte Daten werden in einer MongoDB Datenbank gespeichert. Dabei werden Trends zu bestimmten Zeitpunkten gespeichert. Diese enthalten Trendtype, Trendname, Placement, Country, TweetVolume24. Für Sentiments werden die Tweets, die zu einem Trend gehören, und das jeweilige Topic gespeichert.

Um die Funktionalität zu gewährleisten wurden Unit Tests entworfen. Dadurch kann die interne Integrität, sowie die Anbindung an externe Systeme, sichergestellt werden. Die Tests wurden dabei mit dem Arrange-Act-Assert-Pattern entworfen. Dieses soll eine übersichtliche Struktur und Einheitlichkeit garantieren.

VI. Frontend

References

[1] Microservice-Frontend-Architekturen [Online] https://www.sigs-datacom.de/uploads/tx_dmjournals/attermeyer_OTS_Microservices_Docker_16.pdf (visited on Jun. 21, 2022)