Reddiment: Reddit Sentiment-Analyse
Technical Report

Tobias Bauer
t.bauer@oth-aw.de

Fabian Beer
f.beer1@oth-aw.de

Daniel Holl
d.holl1@oth-aw.de

Ardian Imeraj
a.imeraj@oth-aw.de

Konrad Schweiger
k.schweiger@oth-aw.de

Philipp Stangl
p.stangl1@oth-aw.de

Wolfgang Weigl
w.weigl@oth-aw.de

***Zusammenfassung*—Dieser Technical Report beschreibt die Architektur von Reddiment – ein webbasiertes Dashboard zur Sentiment-Analyse von Subreddits.**

I. Einführung und Ziele

`r/wallstreetbets`, auch bekannt als WallStreetBets oder WSB, ist ein Subreddit, in dem über Aktien- und Optionshandel spekuliert wird. Der Subreddit ist bekannt für seine profane Art und die Vorwürfe, dass Nutzer/innen Wertpapiere manipulieren und volatile Kursbewegungen auslösen. Anhand von Sentiment-Analyse sollen nun Subreddits in Bezug auf Aktienkursverläufe analysiert werden. Dazu soll ein webbasiertes Dashboard entwickelt werden welches das Sentiment als auch die Erwähnungen bestimmter Schlüsselwörter in ausgewählten Subreddits über die Zeit und den Aktienverlauf gegenüberstellt.

In den weiteren Abschnitten des Technical Reports wird zuerst die Bausteinsicht des Gesamtsystems in Abschnitt II eingegangen. Im nächsten Abschnitt III wird die Verteilungssicht der Anwendung beschrieben. In Abschnitt IV werden die angewandten Werkzeuge zur Entwicklung der Anwendung vorgestellt. Abschließend wird kurz auf die angewandten Sicherheitskonzepte in Abschnitt V eingegangen und ein Fazit in Abschnitt VI gegeben, gefolgt vom Literaturverzeichnis am Ende.

II. Bausteinsicht

Diese Sicht zeigt die statische Zerlegung des Systems in Bausteine sowie deren Beziehungen.

A. Gesamtsystem

Reddiment bezieht Daten aus mehreren Quellen und stellt diese dem Benutzer aggregiert bereit. Die folgende Abbildung 1 zeigt die Interaktionen des Systems mit Fremdsystemen und dem Benutzer. Die beiden Datenquellen von Reddiment sind

- Reddit API für Subreddit Daten
- Yahoo Finance für Aktienmarkt Daten

B. Backend

Dieser Abschnitt beschreibt die server-seitige Backend-Architektur.

1) Laufzeitumgebung: JavaScript-basierte Plattform Node.js mit dem serverseitigen Webframework ExpressJS.

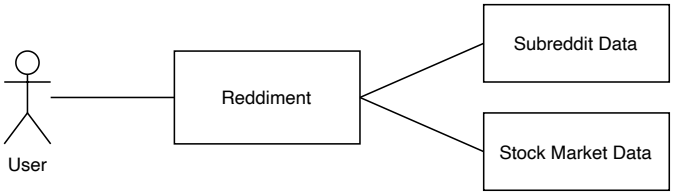

Abbildung 1. Kontextabgrenzung des Reddiment Gesamtsystems

2) GraphQL: Apollo Server wird als Erweiterung der bestehenden Node.js Middleware mit ExpressJS verwendet, um eine GraphQL API zur Verfügung zu stellen.

C. Datenbank

Dieser Abschnitt beschreibt die Persistenzschicht von Reddiment. Die Speicherung der Daten erfolgt mit „Elasticsearch“ [1].

D. Dienste

Dieser Abschnitt beschreibt die Dienste (engl. Services). Reddiment hat zwei Dienste:

1) Sentiment: In diesem Dienst wird für einen Text das Sentiment ermittelt.

2) Reddit Crawler: Der Reddit Crawler verwendet „snoowrap“ [2], das eine Schnittstelle für den Zugriff auf jeden Reddit-API-Endpunkt bereitstellt.

E. Frontend

Dieser Abschnitt beschreibt die client-seitige Frontend-Architektur. Das Frontend wird unter Zuhilfenahme des Frontend-Frameworks SvelteKit [3] realisiert. Das Frontend besteht aus zwei Bausteinen: 1) den Routen zur Navigation, und 2) der *Library* für Komponenten und weitere Module.

Die Routen zur Navigation sind selbst in zwei Unter-Bausteine zerlegt und befinden sich im Ordner `routes`

1) Layout: Es gibt Elemente, die auf jeder Seite sichtbar sind, z. B. die Navigationsleiste oder eine Fußzeile. Anstatt sie auf jeder Seite zu wiederholen, wird eine Layout-Komponente namens `src/routes/__layout.svelte` verwendet.

2) Route: Die *Library* ist eine Sammlung von Frontend-Komponenten, ...

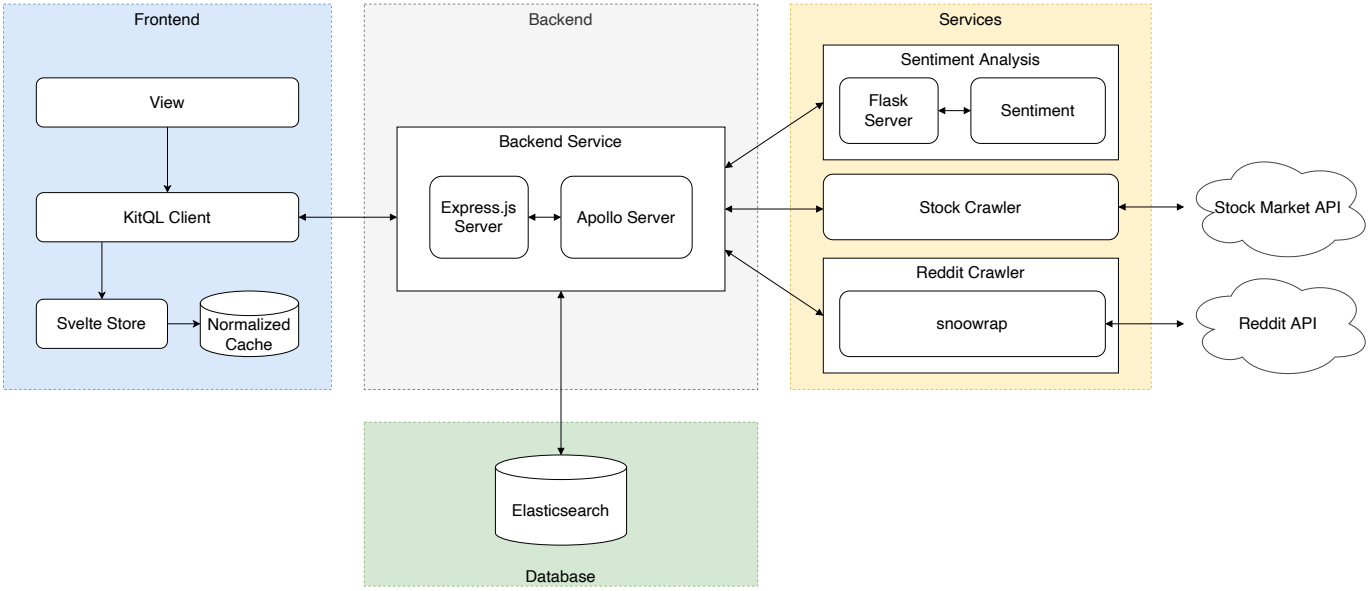

Abbildung 2. Überblick über die Architektur von Reddiment. Die Architektur besteht aus vier Teilen: Das Frontend bietet dem Nutzer ein graphisches Dashboard zur Visualisierung der Metriken (Abschnitt II-E), das Backend stellt (Abschnitt II-B) die GraphQL API bereit, die Datenbank (Abschnitt II-C) speichert persistent Reddit und Aktienmarkt Daten, und den Diensten (Abschnitt II-D) zur Sentiment Analyse als auch zum crawlen der Daten von APIs.

3) Components: In diesem Modul sind die Svelte-Komponenten gesammelt, mit denen die eigentliche Anzeige im Webbrowser realisiert wird. Die Komponenten behandeln alle Eingaben und kommunizieren bei Bedarf mit dem Backend über die GraphQL API Schnittstelle.

III. Verteilungssicht

Das Verteilungssicht beschreibt die Verteilung des Gesamtsystems, wichtige Begründungen für diese Verteilungsstruktur und die Zuordnung von Softwareartefakten zu Bestandteilen der Infrastruktur. Zentraler Bestandteile der Verteilungsstruktur sind „Docker“ [4] Container.

IV. Entwicklungswerkzeuge

Im folgenden Abschnitt wird auf die verwendeten Entwicklungswerkzeuge eingegangen.

A. Paketverwaltung

Die Verwaltung der Abhängigkeiten erfolgt mit „npm“ [5].

B. Linting

Im Frontend wird „eslint“ in Verbindung mit „prettier“ verwendet,um die Einhaltung der Codierrichtlinien zu gewährleisten. Die Konfigurationen sind jeweils in den Dateien `eslintrc.js` und `prettierrc.js` hinterlegt.

C. Build-Tools

1) Backend: Im Backend wird der Typescript Compiler „tsc“ verwendet, um die Dateien in ein Format zu überführen, welches mit `node` ausgeführt werden kann.

Die Konfiguration findet sich dabei in der Datei `tsconfig.json`.

2) Frontend: Im Frontend ist Vite dafür zuständig, die Anwendung aus dem Quellcode zu erstellen. Dabei gibt es zwei Varianten: Für Entwicklungszwecke wird ein Vite-Dev-Server (mit Reload-Funktionalität) zum Bereitstellen der Anwendung verwendet, während für den Produktiveinsatz nur die benötigten Zieldateien unter Verwendung des `Static adapter` erstellt werden, die dann mit einer beliebigen Server-Software ausgeliefert werden können.

Ferner sind Alias-Namen für häufig genutzte Verzeichnisse definiert, um Pfadangaben zu vereinfachen.

D. Unit Tests

Im Backend werden Unit-Tests anhand von „mocha“ [6] durchgeführt. Außerdem wird „nyc“ [7] für die Erzeugung der Test Abdeckung verwendet.

Im Frontend wird „Vitest“ [8] verwendet. Für die Frontend-Komponenten wird zusätzlich die „svelte-testing-library“ [9] verwendet. Diese ermöglicht es, die Komponenten zu *rendern* und Details über die verschiedenen Elemente innerhalb der Komponente zu erhalten.

V. Sicherheitskonzepte

Dieser Abschnitt beschreibt die angewendeten Sicherheitskonzepte, die für eine sichere Entwicklungs- und Produktionsumgebung notwendig sind.

A. Secrets Verwaltung

VI. Fazit und Ausblick

Literatur

[1] Elastic. (2022). “Elasticsearch.” [Online], Adresse: https://www.elastic.co/de/elasticsearch/.

[2] T. Katz. (2022). “snoowrap - A fully-featured JavaScript wrapper for the reddit API.” [Online], Adresse: https://github.com/not-an-aardvark/snoowrap.
[3] Svelte. (2022). “Svelte Kit - The fastest way to build Svelte Apps.” [Online], Adresse: https://kit.svelte.dev/.
[4] Docker. (2022). “Docker.” [Online], Adresse: https://www.docker.com/.
[5] npm. (2022). “npm.” [Online], Adresse: https://www.npmjs.com/.
[6] Mochajs. (2022). “Mocha - JavaScript test framework running on Node.js and in the browser.” [Online], Adresse: https://mochajs.org/.
[7] Istanbul. (2022). “Istanbul - JavaScript test coverage made simple.” [Online], Adresse: https://istanbul.js.org/.
[8] Vitest. (2022). “Vitest - A Vite-native unit test framework.” [Online], Adresse: https://vitest.dev/.
[9] Testing Library. (2022). “Svelte Testing Library.” [Online], Adresse: https://github.com/testing-library/svelte-testing-library.