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 ...

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.

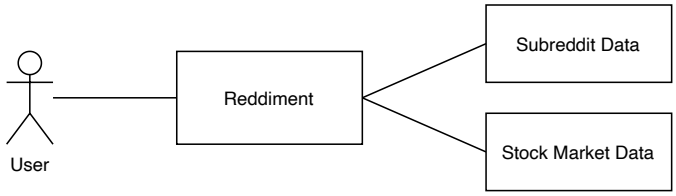

Abbildung 1. Reddiment Gesamtsystem

B. Backend

Dieser Abschnitt beschreibt die server-seitige Backend-Architektur.

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

C. Datenbank

Die Speicherung der Daten erfolgt mit Elasticsearch.

D. Dienste

Dieser Abschnitt beschreibt die Dienste (engl. Services).

1) Sentiment: In diesem Dienst ...

2) Reddit Crawler: Der Reddit Crawler

E. Frontend

Dieser Abschnitt beschreibt die client-seitige Frontend-Architektur. Das Frontend wird unter Zuhilfenahme des Frontend-Frameworks SvelteKit realisiert. Es ist selbst in zwei Unter-Bausteine zerlegt:

1) 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. Zentrale Bestandteile der Verteilungsstruktur sind ...

Tailwind CSS

IV. Entwicklungswerkzeuge

Im folgenden Abschnitt wird auf die verwendeten Entwicklungswerkzeuge eingegangen.

A. Paketverwaltung

Die Verwaltung der Abhängigkeiten erfolgt mit „npm“.

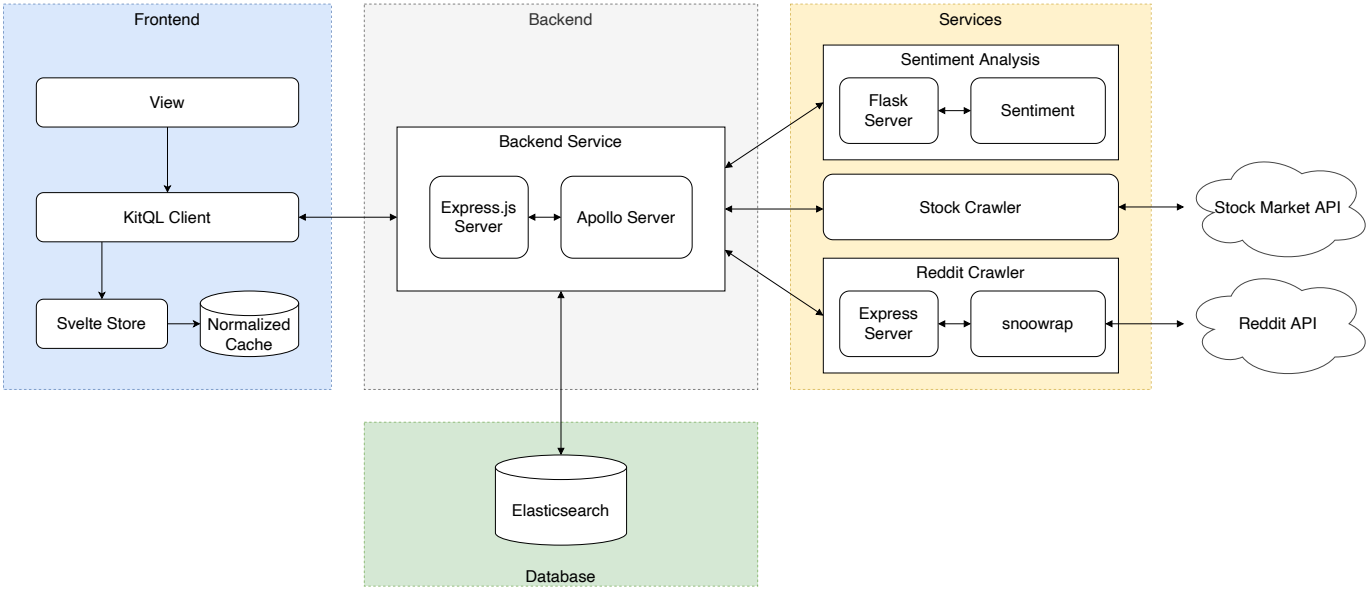

Abbildung 2. Überblick über die Architektur von Reddiment. Die Architektur besteht aus vier Teilen: Das Frontend bietet dem Nutzer eine graphisches Dashboard zur Visualisierung der Metriken (Abschnitt), das Backend umfasst (Abschnitt), die Datenbank (Abschnitt) ist , und den Diensten (engl. Services) zur Sentiment Analyse als auch zum crawlen der Daten von externen APIs.

B. Linting

In beiden Unterprojekten (Frontend und Backend) wird jeweils „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“ [1] durchgeführt. Außerdem wird „nyc“ [2] für die Erzeugung der Test Abdeckung verwendet.

Im Frontend wird „Vitest“ [3] verwendet. Für die Frontend-Komponenten wird zusätzlich die „svelte-testing-library“ [4] 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] Mochajs. (2022). “Mocha - JavaScript test framework running on Node.js and in the browser.” [Online], Adresse: https://mochajs.org/.

[2] Istanbul. (2022). “Istanbul - JavaScript test coverage made simple.” [Online], Adresse: https://istanbul.js.org/.

[3] Vitest. (2022). “Vitest - A Vite-native unit test framework.” [Online], Adresse: https://vitest.dev/.

[4] Testing Library. (2022). “Svelte Testing Library.” [Online], Adresse: https://github.com/testing-library/svelte-testing-library.