Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Fachbereich Informatik
Department of Computer Science

Bachelorarbeit

im Bachelor-Studiengang Wirtschaftsinformatik

Entwicklung einer Schnittstelle für die Anbindung von austauschbaren Datenquellen an KI-Algorithmen

von

Laurenz Anton Dilba

Erstprüfer:	Prof. Dr. Matthias Bertram
Zweitprüfer:	Prof. Dr. Wolfgang Heiden
Unternehmen:	CONET Solutions GmbH
Eingereicht am:	4. Dezember 2022

Erklärung

Hiermit erkläre ich wahrheitsgemäß, dass ich den vorliegenden Bericht selbst angefertigt habe. Der Bericht gibt die tatsächlich durchgeführten Arbeiten wieder. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher keiner Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht. Vertrauliche Informationen sind nicht enthalten.

Datum	Unterschrift Studierender	Unterschrift Betreuer

Inhaltsverzeichnis

Abkürzungsverzeichnis

AJAX Asynchronous JavaScript and XML
API Application Programming Interface
HTML Hypertext Markup Language
KI Künstliche Intelligenz
UI User Interface
BL Business Logic
RAM Read-Access Memory
RDBMS relationalen Datenbankmanagementsystemen
HTTP Hypertext Transfer Protocol
JSON JavaScript Object Notation
AMQP Advanced Message Queuing Protocol

1 Einleitung

text

1.1 Motivation und Hintergrund

text

1.2 Problemstellung

text

1.3 Aufbau

text

2 Grundlagen

2.1 Python API mit Flask

Python ist eine um 1991 von Guido van Rossum entwickelte Programmiersprache. Bei der Entwicklung von Python wurde ein besonderer Fokus auf die Lesbarkeit von Code gesetzt. Dank der simplifizierten Syntax im Vergleich zu anderen höheren Programmiersprachen wie Java oder C#, ist Python auch in Bereichen, wie in der Mathematik oder der Wissenschaft ein häufig genutztes Werkzeug. Python bietet ebenfalls die Möglichkeit, von anderen Entwicklern bereitgestellte Bibliotheken in das eigene Projekt zu integrieren.[1]

Flask ist eine der verfügbaren Bibliotheken, die ein Framework für die Implementierung eines Webbasierten Application Programming Interface (API) bereitstellt. Eine API dient dazu, Funktionen und Routen zu definieren, um die Kommunikation zwischen dem Frontend und dem Backend herzustellen. Das Flask Framework ist im Gegensatz zu anderen Frameworks sehr klein. Dies ermöglicht ein schnelles aufsetzen und entwickeln. Da Flask nur die nötigsten Grundlagen für eine API mitliefert, ist der Code besser lesbar und damit für andere Entwickler besser wartbar.[2]

Die Flask API wird für die Anbindung des Frontends an die Datenbank, sowie die Anbindung an die Kommunikationsschnittstelle von RabbitMQ verwendet. Sie nimmt die Daten oder die Eingaben des Nutzers entgegen und vermittelt sie an den richtigen Dienst, damit sie von einer KI-Schnittstelle ausgewertet werden können. Anschließend kann die API angefragt werden, ob es bereits Antworten von einer Künstliche Intelligenz (KI) zu der vorher geschickten Anfrage gab. Falls die API die Auswertung der KI erhalten hat, wird diese ans Frontend geschickt, um sie dort anzeigen zu können.

2.2 Angular Frontend

Eine grundlegende Website wird klassisch mit Hypertext Markup Language (HTML) und JavaScript erstellt. Um eine moderne Website zu entwickeln, die ihren Inhalt nicht beim ersten Aufrufen lädt, sondern erst dann, wenn er benötigt wird, müssen Konzepte wie Asynchronous JavaScript and XML (AJAX) verwendet werden. Angular ist ein von Google gebaut und gepflegtes Open Source Framework, welches das Entwickeln von komplexen webbasierten Anwendungen vereinfachen soll. Angular bietet im Gegensatz zu anderen Webframeworks wie React und Vue.js eine vollumfängliche Bibliothek, mit der nahezu alle Aspekte in der Web Entwicklung abgedeckt werden können.[3]

In Angular wird die Programmiersprache TypeScript verwendet. Diese ist eine Erweiterung der Programmiersprache JavaScript und implementiert Konzepte wie feste Typisierung von Variablen. Weitere Konzepte wie Dependency Injection oder die Trennung von Business Logic (BL) und User Interface (UI) ermöglichen eine schnelle Entwicklung von komplexen Systemen.

Das Frontend wird für die Ein- und Ausgabe der Daten verwendet. Der Nutzer kann auf der Webseite seine Suchanfrage in ein Textfeld schreiben und anschließend auf den Server

[1] Josheph, 2021.
[2] Grinberg, 2018.
[3] Moiseev u. a., 2018.

hochladen. Im nächsten Schritt wird die Möglichkeit bereitgestellt, die eingegeben Daten automatisiert zu bearbeiten und zu manipulieren. Im gleichen Zug wird die Eingabe des Nutzers in ein für die KI verständliches Format konvertiert. Im letzten Schritt kann der Nutzer die Anfrage an das Backend schicken, dass mit der Analyse der Eingabe begonnen werden soll. Das Frontend fängt daraufhin an beim Backend in regelmäßigen Abständen nach Antworten der KI zu fragen. Wenn Antworten vorhanden sind, können diese in einer Liste visualisiert werden.

2.3 Redis API Cache

Redis ist eine In-Memory Key-Value Datenbank. Im Gegensatz zu relationalen Datenbankmanagementsystemen (RDBMS) wie MySQL oder PostgreSQL werden in Redis keine festen Tabellenstrukturen hinterlegt. Redis gehört damit zur Kategorie der NoSQL Datenbanken (Not Only SQL). Key-Value Stores sind kein Ersatz für eine relationale Datenbank, bieten aber für bestimmte Bereiche große Vorteile. Durch das Fehlen von komplexen Strukturen innerhalb der Datenbank, kann Redis Anfragen weitaus schneller als andere Datenbanksysteme bearbeiten. Da Redis im Read-Access Memory (RAM) ausgeführt wird, werden die Daten grundsätzlich nicht persistent gespeichert. ACID (Atomicity, Consistency, Durability and Isolation) Konformität wird mit Redis ebenfalls nicht gewährleistet. Für den Einsatzzweck als Cache in einer Cloud Umgebung ist Redis allerdings sehr gut geeignet.[4]

Innerhalb des Redis Key-Value Stores werden alle relevanten Daten gespeichert, die ein Nutzer während seiner Benutzung der Software produziert. Dort werden ebenfalls die Zwischenergebnisse abgespeichert, die die KI während der Analyse erstellt.

2.4 MySQL Datenbank für Services und Logs

MySQL ist ein um 1995 erschienenes Open-Source RDBMS. MySQL ist eines weitverbreitetsten und schnellsten Datenbanksysteme in seiner Kategorie.[5]

In relationalen Datenbanken werden Daten strukturiert in Tabellenform abgespeichert. Einzelne Tabellen können Verlinkungen und Referenzen auf andere Tabellen haben, damit die Zusammengehörigkeit der Daten beschrieben werden kann, ohne Daten redundant speichern zu müssen. In MySQL, wie auch anderen RDBMS, werden Tabellenstrukturen und Daten persistent abgespeichert. In-Memory Datenbanken wie Redis können Daten über Umwege auch persistent speichern, jedoch müssen dafür größere Anpassung an der Konfiguration von Redis vorgenommen werden.

Das RDBMS MySQL wird unter Anderem für die Speicherung der Logs, die der Flask Server während der Verarbeitung von Requests oder Nachrichten an die KI produziert, verwendet. Ein weiterer Einsatzzweck der MySQL Datenbank ist die Speicherung der im System registrieren KI-Services. Ein Dienst kann über die Flask API im System registriert oder deregistriert werden. Das Frontend kann sich im Anschluss eine Auflistung der verfügbaren Services vom Backend ziehen.

[4] Paksula, 2010.
[5] DuBois, 2008.

2.5 Kommunikation mit RabbitMQ

Damit eine Kommunikation zwischen unabhängigen Programmen möglich wird, muss es einen Zwischendienst geben, der die Nachrichten von von Programm zum anderen transportiert. Beider Kommunikation zwischen einer Website und einer API wird das Hypertext Transfer Protocol (HTTP) verwendet. Dieses stellt sicher, dass die Information, ob die Nachrichten am anderen Ende angekommen sind, vorhanden sind. Sollte eine Nachricht nicht angekommen sein, hat der Absender die Möglichkeit die Nachricht erneut zu schicken. Problematisch wird diese Herangehensweise, wenn die Antwortzeit sehr lang wird oder ungewiss ist, ob überhaupt eine Antwort kommen wird.

RabbitMQ ist ein eine nachrichtenorientierte Middleware, die die Kommunikation zwischen zwei oder mehreren Programmen durch das Advanced Message Queuing Protocol (AMQP) ermöglicht. Im Gegensatz zu einer direkten Kommunikation zwischen Client und Server wie bei HTTP, wird in RabbitMQ eine Queue implementiert, in der alle Anfragen gesammelt werden. Jeder Client kann Nachrichten in die Queue reinschreiben. Diese Nachrichten werden dort so lange gespeichert, bis sie von einem Dienst ausgelesen werden. Durch diese Herangehensweise wird eine asynchrone Kommunikation zwischen Client und Server ermöglicht. Da RabbitMQ frei von den Handshakes des HTTP ist, sind die Schreib- und Lesezeiten deutlich schneller.[6]

Die Middleware RabbitMQ wird für die Kommunikation zwischen der Flask API und den KI-Services genutzt. Der im Frontend vom Nutzer eingegebene Text-Input wird an die Flask API geschickt. Die Flask API modifiziert den Text im Anschluss so, dass es mittels der JavaScript Object Notation (JSON) über den RabbitMQ Service in die Queue geschrieben werden kann. Jeder KI-Service hat eine Queue einprogrammiert, aus der die Nachrichten ausgelesen werden. Diese Nachrichten können dann verarbeitet und im Anschluss in eine Response-Queue geschrieben werden. Das Flask Backend kann diese Response-Queue auslesen und die einzelnen Antworten dann zusammenbauen.

2.6 KI-Service

Die KI-Services sind alleinstehende Programme, die die Aufgabe haben, Nachrichten anzunehmen, sie zu transformieren, zu analysieren und anschließend ein oder mehrere Ergebnisse zurückzugeben.

Um die Nachrichten empfangen und die Ergebnisse zurücksenden zu können, muss in jedem Service eine AMQP Verbindung zu RabbitMQ hergestellt werden.

2.7 Logs visualisieren in Grafana

text

2.8 Deployment mit Docker

text

[6] Ionescu, 2015.

3 Methodik

text

3.1 Design Science Research

text

3.2 Evaluationsmethode

text

4 Projektergebnisse

text

4.1 Software Architektur

text

4.2 REST-API mit Flask

text

4.2.1 Aufbau und Implementierung der REST-API

text

4.2.2 Nutzeridentifizierung mit JWT

text

4.2.3 Caching mit Redis Datenbank

text

4.2.4 Fehlerbehandlung

text

4.2.5 Event Logging

text

4.3 Visualisierung der Logs in Grafana

text

4.4 Webseite mit Angular

text

4.4.1 Aufbau des User Interfaces

text

4.4.2 Funktionen der Komponenten

text

4.4.3 Kommunikation zur API

text

4.5 Kommunikation zwischen API und Services mit RabbitMQ

text

4.5.1 RabbitMQ vs. REST-API

text

4.6 Implementierung des KI-Services

text

4.6.1 Dynamische Registrierung neuer Services

text

4.6.2 Interpretation der Eingabe mit BERT

text

4.6.3 Cosinusähnlichkeitssuche in Elastic Search

text

5 Evaluation

text

5.1 Performanceanalyse

text

5.2 Skalierbarkeit

text

5.3 Ergebnisse des Code-Reviews

text

6 Fazit

6.1 Fazit

6.2 Einschränkungen

6.3 Ausblick

7 Literaturverzeichnis

DUBOIS, P., 2008. *MySQL*. Pearson Education.

GRINBERG, M., 2018. *Flask web development: developing web applications with python*. O'Reilly Media, Inc.

IONESCU, V.M., 2015. The analysis of the performance of RabbitMQ and ActiveMQ. In: *2015 14th RoEduNet International Conference-Networking in Education and Research (RoEduNet NER)*. IEEE, S. 132–137.

JOSHEPH, T., 2021. Python. *Python Releases for Windows*. Jg. 24.

MOISEEV, A.; FAIN, Y., 2018. *Angular Development with TypeScript*. Simon und Schuster.

PAKSULA, M., 2010. Persisting objects in redis key-value database. *University of Helsinki, Department of Computer Science*. Jg. 27.