



FOM Hochschule für Oekonomie & Management

Hochschulzentrum Siegen

Seminararbeit

im Rahmen der Lehrveranstaltung

IT-Infrastruktur

über das Thema

Dialogbasierter Zugriff auf eine KI über eine Django REST API

von

Henning Beier

Betreuer : Daniel Bitzer

Matrikelnummer : 517370

Abgabedatum : 27. August 2022

Inhaltsverzeichnis

Abbildungsverzeichnis	III
Tabellenverzeichnis	IV
Abkürzungsverzeichnis	V
1 Einleitung	1
1.1 Praktische Relevanz und Problemstellung	1
1.2 Zielsetzung	2
1.3 Methodik	2
2 Theoretischer Teil	3
2.1 Begrifflichkeiten und Grundlagen	3
2.1.1 API	3
2.1.2 REST	3
2.1.3 KI	3
2.1.4 Django	3
2.1.5 Dialogarten Direkt-, Dienst- und Jobverarbeitung	3
2.1.6 Synchrone- und Asynchrone Verarbeitung	3
2.2 Stand der Technik	3
3 Praktischer Teil	3
3.1 Einrichtung Django-Basissystem	3
3.2 Erweiterungen für API	3
3.3 Einbindung einer KI-Anwendung	3
3.4 Python Module und Pakete	3
3.5 Aufruf als Subprozess	3
3.6 Zugriff	3
3.6.1 Konsolenbasiert mit Zeitmessung	3
3.6.2 Webbasiert über Vue.js mit Axios	3
4 Fazit und Reflexion	3
Anhang	4

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
KI	Künstliche Intelligenz
REST	Representational State Transfer

1 Einleitung

In dieser Seminararbeit soll eine Künstliche Intelligenz (KI)-Anwendung über eine REST¹ API² aufrufbar gemacht werden. Eine KI-Anwendung muss dazu in einer Art eingebunden werden, die es erlaubt auf diese selbst in geeigneter Weise über die Endpunkte einer API zuzugreifen. Das bedeutet, dass die KI-Anwendung als Webservice zur Verfügung gestellt wird und damit auch einen diabloasierten Zugriff auf die KI-Anwendung und dessen Funktionen selbst ermöglicht.

Verwendet werden dazu soll Django, ein quelloffenes Web-Framework das weit verbreitet im produktiven Einsatz ist und ebenso wie viele KI-Anwendungen in der Programmiersprache Python geschrieben wurde. Dieses zusammenführen und bereitstellen als eine Lösung im Softwarebereich, bzw. hier als Dienst, stellt dabei einen typischen Arbeitsbereich der IT-Infrastruktur dar.

1.1 Praktische Relevanz und Problemstellung

Aktuell werden in vielen Bereichen neue Projekte und Lösungen mit KI entworfen, entwickelt und getestet. Dazu werden KI-Frameworks wie TensorFlow, PyTorch oder Keras benutzt. Für die vorgenannten, aber ebenso wie auch für andere KI-Frameworks, hat sich Python als Programmiersprache durchgesetzt. Dies, da sich mit Python große Datenmengen vergleichsweise einfach aufnehmen, verarbeiten und ausgeben lassen. Große Datenmengen sind für die KI-Entwicklung, häufig z.B. als Trainingsdatensätze, unverzichtbar.

Entwürfe und Testentwicklungen von KI-Anwendungen erfolgen insbesondere in frühen Phasen meist lokal in einer Entwicklungsumgebung. Ergebnisse zu präsentieren oder ein Austausch mit Dritten darüber ist damit nicht immer einfach möglich. Zumeist müssten Dritte erst einen Python-Interpreter sowie umfangreiche weitere Abhängigkeiten, wie z.B. das verwendete KI-Framework selbst, installieren um die KI-Anwendung überhaupt erst ausführen zu können. Das fällt nicht jedem interessieren leicht. Auch gibt es, durch die dynamische Entwicklung, meist verschiedenste Versionen der Software und Komponenten die zu unbeabsichtigten und unerwarteten Problemen und Hürden führen können.

¹ Representational State Transfer, API-Architekturansatz, konzipiert und entwickelt für Webservices

² Application Programming Interface, Programmierschnittstelle

1.2 Zielsetzung

In dieser Seminararbeit soll eine KI-Anwendung durch die Bereitstellung für den dialogbasierten Zugriff über eine REST API leichter für Dritte zugänglich gemacht werden. Durch die Bereitstellung als Webservice in schon sehr frühen Entwicklungs- und Testphasen, wird ein sehr einfacher Zugriff auf die KI-Anwendung selbst ermöglicht.

Die KI-Anwendung muss dazu in eine geeignete IT-Infrastruktur eingebunden werden. Ein sicherer und belastbarer Webservice, ist das angestrebte Ziel dieser Seminararbeit.

1.3 Methodik

Die Seminararbeit soll das gewünschte Ziel, durch exploratives, vertikalens Prototyping erreichen. Vertikal bedeutet dabei, dass von der einfachen Konsolen KI-Anwendung bis hin zum dialogbasierten Aufruf über den Webserver, alle Ebenen durchdrungen werden sollen. Dabei sollen alle Schritte, jede eingesetzte Software und Lösung gründlich betrachtet und alle Schritte und Entscheidungen nachvollziehbar dokumentiert werden.

Zuerst soll eine einfache KI-Anwendung, die als Konsolenanwendung lokal in einer Entwicklungsumgebung, lauffähig ist erstellt werden. Hier ist ein üblicher Prototyp einer KI-Anwendung, z.B. aus einem einfach Tutorial, aus. Zum Erreichen des Ziels, reicht es dabei aus, wenn nur wenige einfache Funktionen des verwendeten KI-Frameworks zum Einsatz kommen.

Im Anschluss daran, soll ermittelt werden auf welche Arten die Einbindung einer solchen KI-Anwendung in das Django-Framework möglich ist. Mögliche Lösungsansätze sollen ermittelt, betrachtet und bewertet werden.

Die schrittweise Einrichtung einer Django-Installation wird analog durchgeführt und alle gemachten Einstellungen, eingerichteten Erweiterungen und Anpassungen nachvollziehbar dokumentiert. Dies umfasst dabei auch die notwendigen Module für Bereitstellung der API sowie den Einsatz von Django in einem produktiven Umfeld.

Im Schlussschritt werden die Django-Installation und die, für die Einbindung vorbereitete, KI-Anwendung zusammengeführt. Der dann mögliche, dialogbasierte Zugriff auf die KI-Anwendung als Webservice, die Django REST API, soll durch den Einsatz eines dynamischen Frontends, belegt werden.

2 Theoretischer Teil

2.1 Begrifflichkeiten und Grundlagen

2.1.1 API

2.1.2 REST

2.1.3 KI

2.1.4 Django

2.1.5 Dialogarten Direkt-, Dienst- und Jobverarbeitung

2.1.6 Synchrone- und Asynchrone Verarbeitung

2.2 Stand der Technik

3 Praktischer Teil

3.1 Einrichtung Django-Basissystem

3.2 Erweiterungen für API

3.3 Einbindung einer KI-Anwendung

3.4 Python Module und Pakete

3.5 Aufruf als Subprozess

3.6 Zugriff

3.6.1 Konsolenbasiert mit Zeitmessung

3.6.2 Webbasiert über Vue.js mit Axios

4 Fazit und Reflexion

Anhang

Anhang 1: Beispielanhang

Dieser Abschnitt dient nur dazu zu demonstrieren, wie ein Anhang aufgebaut sein kann.

Anhang 1.1: Weitere Gliederungsebene

Auch eine zweite Gliederungsebene ist möglich.

Ehrenwörtliche Erklärung

Hiermit versichere ich, dass die vorliegende Arbeit von mir selbstständig und ohne unerlaubte Hilfe angefertigt worden ist, insbesondere dass ich alle Stellen, die wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen sind, durch Zitate als solche gekennzeichnet habe. Ich versichere auch, dass die von mir eingereichte schriftliche Version mit der digitalen Version übereinstimmt. Weiterhin erkläre ich, dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde/Prüfungsstelle vorgelegen hat. Ich erkläre mich damit nicht einverstanden, dass die Arbeit der Öffentlichkeit zugänglich gemacht wird. Ich erkläre mich damit einverstanden, dass die Digitalversion dieser Arbeit zwecks Plagiatsprüfung auf die Server externer Anbieter hochgeladen werden darf. Die Plagiatsprüfung stellt keine Zurverfügungstellung für die Öffentlichkeit dar.

Wilnsdorf, 27. August 2022

(Ort, Datum)

A handwritten signature in black ink, reading "Henning Beier". The signature is written in a cursive style with a horizontal line underneath it.

(Eigenhändige Unterschrift)