

Number Guesser

Projektdokumentation

des Studienganges Informatik

an der Dualen Hochschule Baden-Württemberg Stuttgart

von

Markus Limbacher

28. Juni 2020

Matrikelnummer, Kurs 9123121, STG-TINF18C

Fach Microservices mit Node.js

Dozent Dr. Ingolf Buttig

# Inhaltsverzeichnis

[Inhaltsverzeichnis II](#_Toc44168785)

[Abkürzungen III](#_Toc44168786)

[Abbildungsverzeichnis III](#_Toc44168787)

[1. Einleitung 1](#_Toc44168788)

[1.1. Problemstellung 1](#_Toc44168789)

[1.2. Verknüpfung zu Vorlesungsinhalten 1](#_Toc44168790)

[1.2.1. Express.js 1](#_Toc44168791)

[1.2.2. Tensorflow 1](#_Toc44168792)

[2. Umsetzung 2](#_Toc44168793)

[2.1. Lösungsansatz 2](#_Toc44168794)

[2.1.1. verwendete Technologien 2](#_Toc44168795)

[GitHub: 2](#_Toc44168796)

[Swagger: 2](#_Toc44168797)

[Express: 2](#_Toc44168798)

[Tensorflow: 3](#_Toc44168799)

[HTML, CSS, ECMA-Script: 3](#_Toc44168800)

[2.1.2. Architektur 3](#_Toc44168801)

[2.1.3. Projektstruktur 4](#_Toc44168802)

[2.2. Herausforderungen 4](#_Toc44168803)

[3. Zusammenfassung 5](#_Toc44168804)

[3.1. Fazit 5](#_Toc44168805)

[3.2. Ausblick 5](#_Toc44168806)

[3.3. Weitere Dokumentation 5](#_Toc44168807)

# Abkürzungen

**API** Application Programming Interface

**CSS** Cascading Style Sheets

**GPU** Graphics Processing Unit

**HTML** Hypertext Markup Language

**JS** JavaScript

**NPM** Node Package Manager

# Abbildungsverzeichnis

# Einleitung

## Problemstellung

Ziel dieses Projekts ist die Entwicklung einer Webapplikation, die ihren Benutzern die Möglichkeit bietet, mithilfe eines Zeichenfensters Bilder von handgeschriebenen Zahlen zu erstellen. Diese Bilder werden dann von der App analysiert mit dem Ziel, die auf dem Bild enthaltene Zahl zu erkennen. Dem Benutzer soll dann das Ergebnis der Analyse präsentiert werden, wobei für jede Zahl ein Prozentsatz angegeben wird, um zu zeigen, wie sicher die Anwendung ist, dass diese Zahl die richtige ist. Die Analyse der Zahlen soll mithilfe von Machine Learning umgesetzt werden.

## Verknüpfung zu Vorlesungsinhalten

Bei der Entwicklung dieser Applikation kamen Technologien zum Einsatz, die in der Vorlesung „Microservices mit Node.js“ vorgestellt und diskutiert wurden.

### Express.js

Express.js (oder kurz „Express“) ist ein Open-Source-Framework für Webanwendungen, das seinen Benutzern die Möglichkeit bietet, Webanwendungen und APIs zu entwickeln. Mit über 10 Millionen wöchentlichen Downloads ist es eines der beliebtesten und am weitesten verbreiteten NPM-Pakete.

### Tensorflow

Tensorflow ist eine Open-Source-Bibliothek, die vom Google Brain Team entwickelt wurde. Tensorflow bietet viele mathematische Funktionen, die zur Lösung zahlreicher Probleme und Aufgaben verwendet werden können. Die Hauptanwendung ist die Entwicklung von Modellen, die maschinelles Lernen nutzen, wie zum Beispiel neuronale Netze.

# Umsetzung

## Lösungsansatz

### verwendete Technologien

#### GitHub:

GitHub wurde für die Versionsverwaltung verwendet. GitHub ist ein Tochterunternehmen von Microsoft das, seinen Benutzern ermöglicht, ihre Projekte online mithilfe des Versionsverwaltungssystems Git zu verwalten. Dadurch ist es beispielsweise möglich, dass mehrere Nutzer gleichzeitig an verschiedenen Versionen desselben Projekts arbeiten. Obwohl es bei dieser praktischen Arbeit keine Zusammenarbeit zwischen mehreren Entwicklern gab, erwies sich Git trotzdem als ein sehr wichtiges und nützliches Werkzeug zur bessern Verfolgung von Änderungen.

Das GitHub Repository, das für dieses Projekt angelegt wurde, ist unter folgendem Link zu erreichen: <https://github.com/wodyy666/Microservices>.

#### Swagger:

Swagger wurde eingesetzt, um die für dieses Projekt erforderliche API zu entwerfen und zu dokumentieren. Swagger ist ein Open-Source Werkzeug, das zur Beschreibung von APIs gemäß dem OpenAPI-Standard verwendet wird.

Die Swagger-Dokumentation enthält Informationen zu allen Endpunkten, die von dem Backend bereitgestellt werden. Hier kann man also nachlesen, wie auf die verschiedenen Methoden des Backends zugegriffen werden kann und welche Daten übergeben werden müssen, damit das Backend die Anfrage bearbeiten kann. Zuletzt werden alle Antwortoptionen einschließlich ihres Inhalts festgehalten. Dies würde es einem Entwickler ermöglichen, ein Frontend an ein Backend anzubinden, ohne die Implementierungsdetails des Backends zu kennen. Stattdessen kann dieser Entwickler alle Informationen, die er benötigt, aus der Swagger-Dokumentation für die API entnehmen.

#### Express:

Wie in der Einleitung erwähnt wurde in diesem Projekt das NPM-Paket Express verwendet. Mithilfe von Express wurde ein Webserver entwickelt. Dieser Webserver enthält sowohl das Frontend als auch das Backend dieser Anwendung.

#### Tensorflow:

Zur Analyse der Bilder wurde Tensorflow verwendet. Es wäre auch möglich gewesen, einen der Azure Cognitive Services zu nutzen, die diese Funktionalität bereits bieten, aber das Ziel des Projekts war es ein eigenes Modell zu entwickeln, zu trainieren und dann zu verwenden.

Eine wichtige Entscheidung für dieses Projekt war die Wahl des richtigen Tensorflow-Moduls, denn obwohl es im NPM-Repository ein Modul namens Tensorflow gibt, ist dieses Modul nicht die offizielle Tensorflow-Bibliothek für Node.js und nur mit Linux kompatibel. Stattdessen wurde in diesem Projekt das „@tensorflow/tfjs-node“-Paket verwendet, das mit Linux, Windows und Mac kompatibel ist.

Von diesem Paket gäbe es noch eine GPU-Version, die die Grafikkarte des jeweiligen Rechners verwendet. Diese Version wurde nicht verwendet, da der für die Entwicklung verwendete Laptop nur über eine integrierte Grafikkarte verfügte. In Zukunft wäre es jedoch möglich, das derzeit verwendete Paket durch seine GPU-Version zu ersetzen, um eine bessere Leistung zu erzielen. Die zusätzliche Leistung einer Grafikkarte sorgt jedoch nur dafür, dass das Modell schneller trainiert werden kann und die Analyse eines Bildes schneller abgeschlossen ist. Eine Grafikkarte hätte keinen Einfluss auf die Qualität der Ergebnisse.

#### HTML, CSS, JS:

Das Frontend wurde mit HTML, CSS und JS entwickelt. Es wurden keine Frontend-Frameworks verwendet, da der Umfang der zu entwickelnden Website nicht groß genug war, um den Einsatz eines solchen Frameworks lohnenswert zu machen.

Drei Funktionalitäten wurden in dem Frontend implementiert. Zum einen wird hier das Bild von dem Benutzer erstellt, damit dieses später analysiert werden kann. Zu diesem Zweck wurde ein Canvas-Element in die Website integriert, in dem der Benutzer die Zahlen mithilfe seiner Maus zeichnen kann.

Zum anderen ist das Frontend in der Lage, das vom Benutzer gezeichnete Bild in ein Format zu konvertieren, das vom Backend verarbeitet werden kann. Ein Teil der Bildverarbeitung und -analyse wird also bereits auf der Client-Seite erledigt. Nach dieser Verarbeitung übernimmt das Frontend auch die Kommunikation mit dem Backend über die zuvor erwähnte API, die über http-Aufrufe angesprochen wird.

Zuletzt dient das Frontend auch zur Anzeige der Analyseergebnisse mithilfe mehrerer Balken, die die erkannte Zahl darstellen.s

Das Frontend wurde mit dem Firefox-Browser entwickelt und getestet. Aus diesem Grund kann die Benutzererfahrung bei der Verwendung von Firefox besser sein. Die Anwendung sollte jedoch auf allen Browsern funktionieren.

### Architektur

Allgemeine Architektur

(mit diagramm)

Tensorflow Modell

<https://github.com/tensorflow/tfjs-examples/tree/master/mnist-node>

MNIST

API specs

<https://app.swaggerhub.com/apis/wodyy666/Microservices-NumberGuesser/1.0.0>

### Projektstruktur

<https://www.terlici.com/2014/08/25/best-practices-express-structure.html>

## Herausforderungen

Data ingest

Canvas resize and downsampling

🡪client side

Overfitting:

594922 🡪 33082 (Zeiteinsparung auch noch erwähnen?)

Probeleme mit dataset: sehr klein; zentriert; schreibweise 1 (erkennt ne 7) (einfach beispiel MNIST bild einfügen)

# Zusammenfassung

## Fazit

Problem overfitting

Train endpoint besser nutzen

Draw update speed

## Ausblick

## Weitere Dokumentation

-swagger

.git

Readme

code