Diplomarbeit

Höhere Technische Bundeslehranstalt Leonding
Abteilung für Informatik

Leo-Chat

Eingereicht von: **Karsten Köhne, 5AHIF**
Marcel Langoth, 5AHIF
Denys Sheludchenko, 5AHIF
Datum: **April 9, 2019**
Betreuer: **Prof. Mag. Dr. Thomas Stütz**
Projektpartner: **AV Prof. DI Richard Kainerstorfer**

Declaration of Academic Honesty

Hereby, I declare that I have composed the presented paper independently on my own and without any other resources than the ones indicated. All thoughts taken directly or indirectly from external sources are properly denoted as such.

This paper has neither been previously submitted to another authority nor has it been published yet.

Leonding, April 9, 2019

Karsten Köhne, Marcel Langoth, Denys Sheludchenko

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorgelegte Diplomarbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Gedanken, die aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet.

Die Arbeit wurde bisher in gleicher oder ähnlicher Weise keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Leonding, am 9. April 2019

Karsten Köhne, Marcel Langoth, Denys Sheludchenko

Abstract

Leonie is a language assistant who is supposed to support the students in their everyday school life. If you need the current public transport arrival-times, Leonie is on hand. The current information Leonie gets over various interfaces from the Internet, so the timetables and test dates can be queried by the students.

Many already know Alexa or the language assistant of Google. But Leonie was created without their existing hardware platform. Because in the HTL-Leonding you also want to have personal conversations with your avatars and be able to address Leonie with her real name without having to say "Alexa" or "Hey Google". For this a hotword detection was trained which allows the users to initiate the conversation with Leonie with "Hey Leonie". If one thinks of already existing language assistants and their use, speech recognition also plays an important role. Because if users are not properly understood, you can not provide a correct answer. Thus, Leonie was also equipped with a German voice recognition that users from all age groups can understand well.

But Leonie is not just a box you can put in the living room like others. Leonie is more, it's a hologram that allows users to build a personal relationship. Because when you talk to Leonie you do not just hear a voice, you also see an animated avatar in a glass pyramid that can gesticulate while speaking.

Zusammenfassung

Leonie ist eine Sprachassistentin die Schülerinnen im Schulalltag unterstützen soll. Wenn man die aktuellen Öffi-Zeiten benötigt, steht Leonie zur Stelle. Die aktuellen Informationen bekommt Leonie über verschieden Schnittstellen aus dem Internet, so können auch die Stundenpläne und Testtermine von den Schülerinnen abgefragt werden.
Viele kennen schon Alexa oder auch den Sprachassistenten von Google. Doch Leonie wurde ohne deren bestehender Hardware Platform erschaffen. Denn in der HTL-Leonding möchte man auch mit seinen Avataren persönliche Gespräche führen und dabei Leonie mit ihren richtigen Namen ansprechen können ohne "Alexa" oder "Hey Google" sagen zu müssen. Hierfür wurde eine Hotword Detection trainiert die es den Benutzerinnen ermöglicht das Gespräch mit Leonie mit "Hey Leonie" zu initiieren. Wenn man an schon bestehende Sprachassistenten denkt und deren Verwendung, spielt gerade auch die Spracherkennung eine große Rolle. Denn wenn Benutzerinnen nicht richtig verstanden werden, kann man auch keine richtige Antwort liefern. So wurde Leonie auch mit einer deutschen Spracherkennung ausgestattet die Benutzerinnen aus allen Altersgruppen gut verstehen kann.
Doch Leonie ist nicht nur eine Box die man sich ins Wohnzimmer stellen kann wie andere. Leonie ist mehr, sie ist ein Hologramm die es den Benutzerinnen ermöglicht eine persönliche Beziehung aufzubauen. Denn wenn man mit Leonie spricht hört man nicht nur eine Stimme man sieht auch einen animierten Avatar in einer Glaspyramide der beim sprechen gestikulieren kann.

Autoren der Diplomarbeit

Karsten Köhne

Chatbot, Sprachverarbeitung & Visualisierung des Avatars

Persönliche Daten

Name:	Karsten Köhne
Geburtsdatum:	26. Oktober 1997
E-Mail:	karsten@koehne.at
Bildungsweg:	
seit 2014	HTBLA Leonding, Informatikabteilung
2012 bis 2014	BORG Hoenauerstraße
2008 bis 2012	Hauptschule St.Florian
2004 bis 2008	Martin Boos Schule
Berufliche Erfahrung:	
Juli 2016	Rosenbauer International AG
Juli 2017	Rosenbauer International AG
Sprachliche Kenntnisse:	
Deutsch	Muttersprache
Englisch	Fließend

Table 1: Persönliche Daten Karsten Köhne

Marcel Langoth

Gesichtserkennung & Prototypenbau

Persönliche Daten

Name:	Marcel Langoth
Geburtsdatum:	27. April 2000
E-Mail:	marcel@langoth.eu
Akademische Laufbahn:	
seit 2014	HTBLA Leonding, Informatikabteilung
2010 bis 2014	BRG Landwiedstraße
2006 bis 2010	Volksschule Weichstetten
Berufliche Laufbahn:	
Juli 2015	RÜBIG GmbH & Co KG
Juli 2016	RÜBIG GmbH & Co KG
Juli 2017	RÜBIG GmbH & Co KG
August 2018	MIC Datenverarbeitung GmbH
Sprachliche Kenntnisse:	
Deutsch	Muttersprache
Englisch	Fließend

Table 2: Persönliche Daten Marcel Langoth

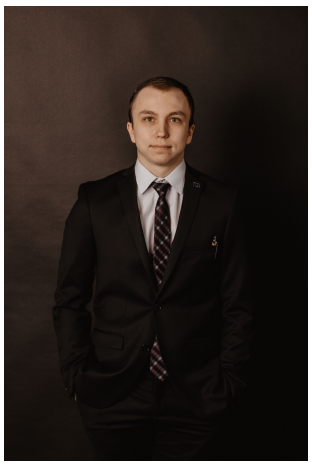
Denys Sheludchenko

Backend

Persönliche Daten

Name:	Denys Sheludchenko
Geburtsdatum:	02. August 1998
E-Mail:	denys.sheludchenko@gmail.com
Bildungsweg:	
2004 bis 2010	Schule in Ukraine
2011 bis 2012	Hauptschule Mauthausen
2012 bis 2013	Hauptschule 11 Linz
seit 2013	HTBLA Leonding, Informatikabteilung
Berufliche Erfahrung:	
Juli 2015	Gebietskrankenkasse OÖ
August 2017	Sprecher Automation GmbH
Sprachliche Kenntnisse:	
Ukrainisch	Muttersprache
Deutsch	Fließend
Russisch	Fließend
Englisch	Fließend

Table 3: Persönliche Daten Denys Sheludchenko



Danksagungen

An dieser Stelle möchten wir uns bei allen bedanken, welche uns durch ihre fachliche und persönliche Unterstützung in dieser Diplomarbeit geholfen haben. Ganz besonders Herrn Prof. Mag. Dr. Thomas Stütz für die Unterstützung durch sein Fachwissen in diesem Bereich, welches uns die Arbeit an der Diplomarbeit wesentlich erleichtert hat. Des weiteren danke wir Prof. DI Richard Kainerstorfer für sein Fachwissen mit dem er uns bei dieser Diplomarbeit unterstützt hat. Auch danken wir unserer Schule, der HTL-Leonding, für die finazielle Unterstützung bei der Arbeit. Ebenso gilt der Dank unseren Eltern, Mitschülern und Freunden für das Korrekturlesen und dem Beistand während der Diplomarbeit.

Inhaltsverzeichnis

Kapitel 1

Pflichtenheft

1.1 Ausgangslage

Die HTL Leonding, eine moderne mittelständiche Schule im Zentralraum von Oberösterreich mit ca. 1000 Schüler, bietet viele Ausbildungsmöglichkeiten mit dem Schwerpunkt Informatik und Elektronik an. Bei diesen Ausbildungen werden neue Technologien näher gebracht. Die HTL Leonding möchte ihren Besuchern diese Werte anschauliche präsentieren.

1.2 Istzustand

Derzeit besuchen schulfremde Personen die HTL Leonding. Diese haben meist Fragen zur Schule. Derzeit kann der Avatar "Leonie" über einen Alexa-Skill Fragen beantworten. Das funktioniert nicht immer perferkt bei höherer Lautstärke.

1.3 Problemstellung

Besucher der HTL Leonding müssen zuerst das Sekretäriat aufsuchen um herauszufinden wo Klassen, Schüler, Lehrer und etc. zu finden sind.

1.4 Aufgabenstellung

Es soll ein Avatar erstellt werden, der in der Lage sein soll, ein möglichst natürlichsprachliches Gespräch führen zu können. Damit sollen den Besuchern der HTL Leonding Auskünfte erteilt werden.

1.5 Ziele

Der Avatar "Leonie" soll für alle Altersgruppen die die HTL Leonding besuchen zugänglich sein. Im Eigangsbereich soll "Leonie" die Besucher automatisch begrüßen und die meisten Fragen der Besucher/Besucherinnen beantworten können. Die automatische Begrüßung soll über eine Gesichtserkennung erfolgen und den Benutzer beim annähern oder vorbeigehen erkennen. Die Fragen über die Schule, den Stundenplan, das Wetter und die Verkehrsanbindung sollen problemlos beantwortet werden können. Unteranderem soll der Avatar "Leonie" dabei mit dem Benutzer gestikulieren können, um ein möglichst natürliches Gespräch darzustellen. Die Darstellung des Avatars soll als Hologramm von allen drei Seiten sichtbar sein. Dabei ist es wichtig, dass "Leonie" auch von einer weiten Entfernung noch sichtbar ist. Schülern und Lehrern soll der Avatar in Alltagsituationen helfen. Auch am Elternabend soll "Leonie" im Einsatz sein, um Eltern für weiter Informationen zur Seite stehen zu können. Gerade auch am Tag der offenen Tür oder Schnuppertagen ist Leonies Einsatz gefragt, um neuen Besuchern und Interessenten den Besuch einfacher zu machen. Es ist wichtig, dass "Leonie" einfach und schnell zu bedienen ist, das heißt ohne großes Vorwissen soll "Leonie" hilfe stellen können.

1.6 Funktionale Anforderungen

Besucher kommen in den Eingangsbereich der HTL Leonding und werden dort von dem Avatar "Leonie" begrüßt. Dies funktioniert durch eine Gesichtserkennung. Der Besucher kann sich den Avatar dann in einer Pyramide von 3 verschiedenen Seiten ansehen und ihr Fragen über die Schule, das Wetter oder die Verkehrsanbindung stellen. "Leonie" antwortet mittels Chatbot und bewegt sich entsprechend dazu. Auf Wunsch des Besuchers kann sie kann auch tanzen.

1.6.1 UseCase Diagramm

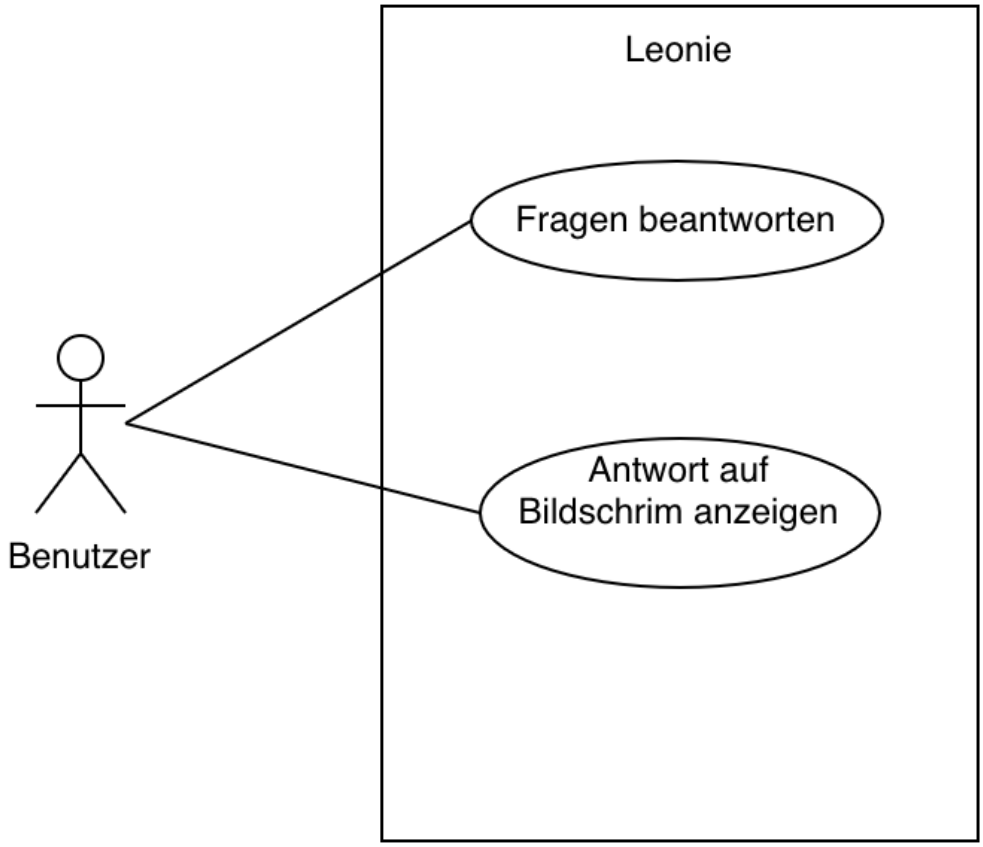


Abbildung 1.1: UseCase Diagramm

1.7 Nichtfunktionale Anforderungen

Unser Team hat sich dazu entschieden für die nichtfunktionalen Anforderungen die ISO9126 Zertifizierung als Vorlage zu nehmen. Bei dieser Arbeit wurden einige Punkte besonders hervorgehoben, diese waren im Konkreten Stabilität, Zeitverhalten, Resourcenverbrauch, Sicherheit, Robustheit und der Punkt Benutzbarkeit, Gebrauchstauglichkeit. Kurz zusammengefasst war es wichtig ein komplettes System zu haben das ohne Hilfe von eingeschultem Personal zu bedienen ist. Außerdem sollte es jederzeit die Möglichkeit geben das System von außen zu warten und es sollte selbst-lernend sein. Zusätlich war es wichtig, dass das System innerhalb kurzer Zeit, für jeden verständlich, antwortete und alle Daten sicher vor Missbrach schützte.

Abbildung 1.2: ISO9126 - Nicht funktionale Anforderungen [53]

1.7.1 Stabilität

Das System muss über einen Zeitraum von einem Monat ohne eingreifen von außen bedienbar sein. Dafür wird ein Internetzugang von mindestens 2 Mbit/s vorrausgesetzt.

1.7.2 Zeitverhalten

Das System muss dem Benutzer innerhalb von 3 Sekunden eine gültige Antwort liefern können.

1.7.3 Resourcenverbrauch

Das System muss auf einem handelsüblichen Computer mit einem Kaufpreis von ca. 500 Euro ohne Probleme funktionieren können.

1.7.4 Sicherheit

Das System muss alle Anfragen über SSL-Verschlüsselung absichern, damit keine Zugriffe auf die Fragen an dritte gelangen können. Dafür muss ein Reverse Proxy für die einzelnen Server-Container mit gültigem und unterzeichnetem SSL-Zertifikat aufgesetzt werden.

1.7.5 Robustheit

Das System muss trotz falscher Frage des Benutzer weiter funktionstüchtig sein und darf nicht abstürzen. Dafür muss bei einer nicht verstandenen Frage des Benutzers, das System dies erkennen und erneut fragen, was der Benutzer möchte.

1.7.6 Benutzbarkeit, Gebrauchstauglichkeit

Das System muss für einen nicht eingeschulten Benutzer durch einfaches starten benutzbar sein. Dafür muss mit Gesichtserkennung, das System gestartet werden. Das Layout auf dem nebenstehenden Bilschirm muss übersichtlich und leicht verständlich sein. Die Sprache und Ausdrucksweise des Avatars muss gut verständlich sein.

1.8 Systemarchitektur

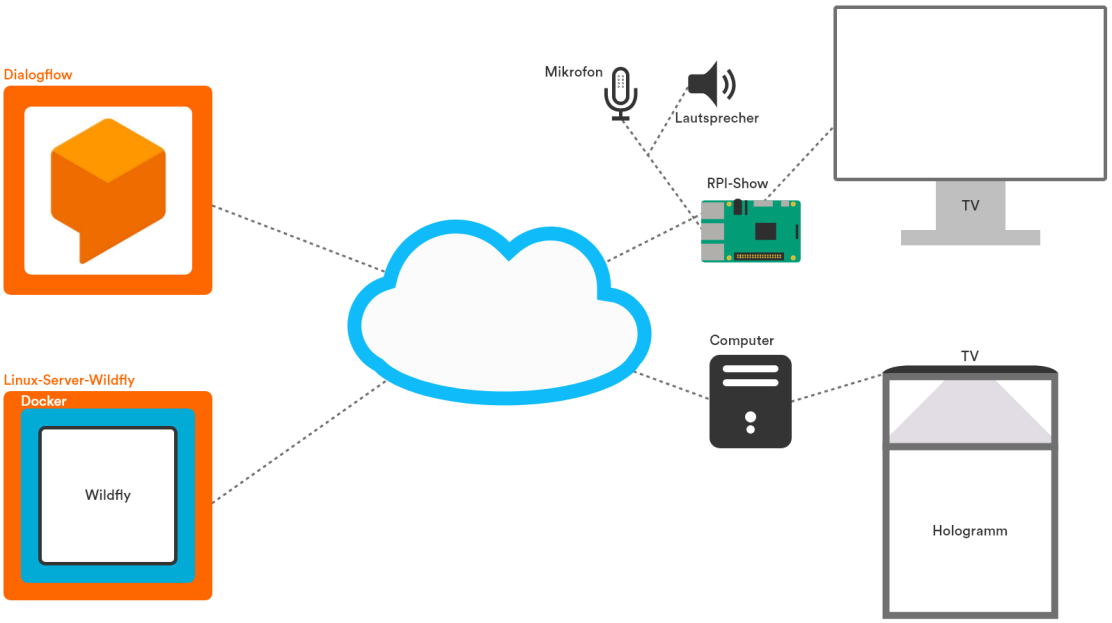


Abbildung 1.3: Systemarchitektur

Kapitel 2

Entwurfsentscheidungen

2.1 Chatbot

"Ein Chatterbot, Chatbot oder kurz Bot ist ein textbasiertes Dialogsystem, welches das Chatten mit einem technischen System erlaubt. Er hat je einen Bereich zur Textein- und -ausgabe, über die sich in natürlicher Sprache mit dem dahinterstehenden System kommunizieren lässt. Chatbots können, müssen aber nicht in Verbindung mit einem Avatar benutzt werden. Technisch sind Bots näher mit einer Volltextsuchmaschine verwandt als mit künstlicher oder gar natürlicher Intelligenz. Mit der steigenden Computerleistung können Chatbot-Systeme allerdings immer schneller auf immer umfangreichere Datenbestände zugreifen und daher auch intelligente Dialoge für den Nutzer bieten. Solche Systeme werden auch als virtuelle persönliche Assistenten bezeichnet". [4]

2.1.1 Chatbot Bestandteile

Intent

Wenn man mit einem Chatbot kommuniziert hat man eine gewisse Intention. Durch die Frage "Wann fährt der nächste Bus bei der Meixnerkreuzung" möchte der Benutzer wissen, wann der nächste Bus bei einer gewissen Haltestelle fährt. Für einen Menschen ist diese Frage klar, jedoch muss ein Chatbot auch erkennen, welche Intention der Benutzer hat und dieses auch meist mit einer Wahrscheinlichkeit definieren (Dialogflow JSON Response), um die Antwort dementsprechend zu wählen. In Abbildung 2.5 ist ein Beispiel Intent bei dem erstellten System für Dialogflow zu sehen. In diesem Fall soll das System ein Begrüßung "Hallo Leonie" als die Intention "Greet" erkennen.

Damit ein System einen Text auch richtig versteht, muss ein sogenannter Text-Klassifizierer trainiert werden. Das System wird mit Trainingsdaten darauf trainiert, die Intention des Eingabe Textes zu erkennen. Hierfür wird eine hohe Anzahl an Trainingsdaten benötigt, die auch richtig gelabelt sind (einfaches Beispiel: ["Hallo Leonie", "Greet"]).

Doch auch die Wahl des richtigen Klassifizierungs Algorithmus spielt eine große Rolle. Bevor man mit der eigentlichen Programmierung des Textklassifizieres beginnen kann, müssen noch die Werkzeuge für die Verarbeitung von Sprach verstanden werden. Damit das Modell so akkurat wie möglich ist und Texte immer auf die selbe Weise klassifiziert werden können, muss ein Folge von Methoden der Sprachverarbeitung verwendet werden.

Sprachverarbeitung

Wenn ein Computersystem die menschliche Sprache verarbeiten soll, spricht man von einem NLP (Natural Language Processing) [23]. Für die Verarbeitung der Texte wurde die Python Library NLTK verwendet. [24]

Tokenization Beim Tokenization-Verfahren werden Zeichenketten in kleine Komponenten für die weitere Verarbeitung gewandelt. [39] Das Ergebnis ist im im Beispiel mit "Ergebnis" gekennzeichnet

```
1 import nltk
2 text = "Wie geht es dir heute?"
3 tokenizer = nltk.tokenize.TreebankWordTokenizer()
4 tokens = tokenizer.tokenize(text)
5 print(tokens)
6
7 ['Wie', 'geht', 'es', 'dir', 'heute', '?'] #Ergebnis
```

Listing 2.1: Tokenization mit NLTKs TreebankWordTokenizer

Stemming Beim Stemming werden die Wörter des Eingabe-Textes nachdem Tokenization-Verfahren in die Stammform gebracht. Der Stemming-Algorithmus ist durch sein statisches Verfahren sehr performant, jedoch ist das Ergebnis nicht immer ein gültiges Wort. Das Ergebnis der beiden Stemming-Algorithmen ist immer im Beispiel mit "Ergebnis" gekennzeichnet

```
1 import nltk
2 text = "Wie geht es dir heute?"
3 tokenizer = nltk.tokenize.TreebankWordTokenizer()
4 tokens = tokenizer.tokenize(text)
5 stemmer = nltk.stem.Cistem()
6 " ".join(stemmer.stem(token) for token in tokens)
7
8 'wie geh es dir heu ?' #Ergebnis
```

Listing 2.2: Stemming mit NLTKs Cistem-Algorithmus

```
1 import nltk
2 text = "Wie geht es dir heute?"
3 tokenizer = nltk.tokenize.TreebankWordTokenizer()
4 tokens = tokenizer.tokenize(text)
5 stemmer = nltk.stem.snowball.GermanStemmer()
6 " ".join(stemmer.stem(token) for token in tokens)
7
8 'wie geht es dir heut ?' #Ergebnis
```

Listing 2.3: Stemming mit NLTKs Snowball-Algorithmus

Lemmatization Bei den zuvor verwendeten Stemming-Algorithmen wird ein gültiges Wort nicht garantiert. Bei der Lemmatization wird der Text nach dem Tokenization-Verfahren mit Hilfe eines Wörterbuches in die Stammform gebracht und somit ist das Ergebnis abhängig von dem verwendeten WordNet [46]. Im folgenden Beispiel wird als Sprache für den Eingabetext Englisch verwendet, da ein Englisches Wörterbuch/WordNet verwendet wird. Das Ergebnis ist mit "Ergebnis" gekennzeichnet

```
import nltk
text = """How are you doing today?"""
tokenizer = nltk.tokenize.TreebankWordTokenizer()
tokens = tokenizer.tokenize(text)
stemmer = nltk.stem.WordNetLemmatizer()
" ".join(stemmer.lemmatize(token, pos='v') for token in tokens)

'How be you do today ?' #Ergebnis
```

Listing 2.4: Lemmatization mit NLTKs WordNet-Lemmatizer

Stopwords Der Trainingsdatensatz und in weiterer Folge auch der Eingabetext des Chatbot-Benutzers, wird viele Füllwörter beinhalten. Wenn man Texte Klassifizieren möchte, führen Füllwörter meist nicht zu einer höheren Genauigkeit der Textklassifizierung. Deshalb kann man die für den Anwendungsfall abhängigen Füllwörter für die Trainingsdaten und für die Verwendung des Klassifizieres beim Chatbot herausfiltern. Das Ergebnis ist mit "Ergebnis" gekennzeichnet

```
import nltk
from nltk.corpus import stopwords
stopWords = set(stopwords.words('german'))
words = nltk.word_tokenize("Wie geht es dir heute?")
wordsFiltered = []

for w in words:
    if w not in stopWords:
        wordsFiltered.append(w)

['Wie', 'geht', 'heute', '?'] #Ergebnis
```

Listing 2.5: Füllwörter herausfiltern mit NLTK

Beim herausfiltern von Füllwörtern ist zu beachten, welche Wörter man verwendet. Sollte es passieren flasche Wörter als Füllwörter zu definieren, kann das Ergebnis des Textklassifizieres verfälscht werden. Beim folgenden Beispiel wird der Text "Vergiss die Milch nicht" auf Füllwörter überprüft und das Ergebnis ist "Vergiss Milch". Der Sinn ist somit über die entfernung des Wortes nicht, ein anderer. Deshalb ist zu beachten welche Füllwörter man verwendet.

```
import nltk
from nltk.corpus import stopwords
stopWords = set(stopwords.words('german'))
words = nltk.word_tokenize("Vergiss die Milch nicht")
wordsFiltered = []
```

```
6
7  for w in words:
8      if w not in stopWords:
9          wordsFiltered.append(w)
10
11 ['Vergiss', 'Milch']  #Ergebnis
```

Listing 2.6: Füllwörter herausfiltern mit NLTK

Diese Methoden werden für die meisten Textklassifizierer verwendet, um den Text vor dem trainieren und bei der Verwendung zu verarbeiten. Man muss diese Methoden nicht zwangsweise anwenden, es kommt immer darauf an wofür der Textklassifizierer verwendet wird und wie die Trainigsdaten aufgebaut sind.
Ein Computer würde den mit Sprachverarbeitungsmethoden verarbeiteten Text, jedoch noch nicht vergleichen können. Denn beim Textklassifizierern wird ein Vektor, mit Sprachverarbeitungsmethoden verarbeiteter Möglichkeiten und den dazugehörigen Features erstellt, um daraufhin ein Datensatz zuordnen zu können.
Wenn man bei einem Chatbot wissen möchte, ob der Benutzer Tanzen oder Singen möchte, hat man beispielsweise das Feature Tanzen und Singen. In den Folgenden Code Blöcken sieht man ein Beispiel für einen Textklassifizierer von der Trainierung bis zur Testung.

```
1  import nltk
2  from nltk.corpus import stopwords
3  from nltk.stem import WordNetLemmatizer
4  import numpy as np
5  import sklearn
6
7  def preprocessing(text):
8      # tokenize
9      tokens = [word for sent in nltk.sent_tokenize(text) for word in nltk.
           word_tokenize(sent)]
10     # remove stopwords
11     stop = stopwords.words('english')
12     tokens = [token for token in tokens if token not in stop]
13     # lower capitalization
14     tokens = [word.lower() for word in tokens]
15     # lemmatize
16     lemma = WordNetLemmatizer()
17     tokens = [lemma.lemmatize(token) for token in tokens]
18     preprocessed_text= ' '.join(tokens)
19     return preprocessed_text
```

Listing 2.7: Textverarbeitungsmethoden für die Vorverarbeitung festlegen

Für die Textverarbeitung wird in der preprocessing Funktion das Tokenization-Verfahren, Stopword-Verfahren, Normalisation-Verfahren und Lemmatization-Verfahren angewandt. Beim Normalisation-Verfahren wird der gesamte Text in Kleinbuchstaben umgewandelt.

```
1  labels = []
2  data = []
```

```
3
4  #train
5  data.append(preprocessing('Can you dance?'))
6  labels.append('intent_dance')
7  data.append(preprocessing('Can you sing?'))
8  labels.append('intent_sing')
9  data.append(preprocessing('Are you a good dancer?'))
10 labels.append('intent_dance')
11 data.append(preprocessing('Are you a good singer'))
12 labels.append('intent_sing')
13 data.append(preprocessing('Would you like to dance for me'))
14 labels.append('intent_dance')
15 data.append(preprocessing('Would you like to sing for me'))
16 labels.append('intent_sing')
17 data.append(preprocessing('Could you dance for me now?'))
18 labels.append('intent_dance')
19 data.append(preprocessing('Could you sing for me now?'))
20 labels.append('intent_sing')
21 data.append(preprocessing('What if we sing together?'))
22 labels.append('intent_sing')
23
24 #test
25 data.append(preprocessing('May I ask for a dance?'))
26 labels.append('intent_dance')
27 data.append(preprocessing('Lets dance'))
28 labels.append('intent_dance')
29 data.append(preprocessing('I love your dancing skills'))
30 labels.append('intent_dance')
31 data.append(preprocessing('May I ask you to sing?'))
32 labels.append('intent_sing')
33 data.append(preprocessing('Lets sing'))
34 labels.append('intent_sing')
35 data.append(preprocessing('I love your singing skills'))
36 labels.append('intent_sing')
```

Listing 2.8: Daten den jeweiligen Features zuweisen

Die Datensätze werden mit Text und den dazugehörigen Feature/Intent definiert und in einen Array gespeichert.

```
1 trainset_size = int(round(len(data)*0.60))
2
3 x_train = np.array(['.join(el) for el in data[0:trainset_size]])
4 y_train = np.array([el for el in labels[0:trainset_size]])
5
6 x_test = np.array(['.join(el) for el in data[trainset_size:len(data)]])
7 y_test = np.array([el for el in labels[trainset_size:len(labels)]])
```

Listing 2.9: Daten in Trainingsdaten und Testdaten teilen

Da man bei der Testung und der Trainierung des Textklassifizierers nicht die gleichen Datensätze verwenden soll, werden die Datensätze geteilt.

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=2, stop_words='english',
    strip_accents='unicode')


X_train = vectorizer.fit_transform(x_train)
X_test = vectorizer.transform(x_test)
```

Listing 2.10: Daten vektorisieren

Nachdem die Datensätze geteilt wurden, müssen die Daten vektorisiert werden. In Abbildung 2.1 ist zu sehen, dass "can dance" der Intention "feat_dance" zugewiesen wurde. Gleichzeitig ist zu sehen, dass "are good dancer"der Intention "feat_good" zugewiesen wird. Dabei ist zu beachten, dass "feat_good" nicht als Intention gebraucht wird und wegen der Textverarbeitung als Intent verwendet wird. Mithilfe Visualisierungen erkennt man schneller Fehler.

	feat_dance	feat_good	feat_like	feat_sing
0	1.000000	0.0	0.000000	0.00000
1	0.000000	0.0	0.000000	1.00000
2	0.000000	1.0	0.000000	0.00000
3	0.000000	1.0	0.000000	0.00000
4	0.656138	0.0	0.754641	0.00000
5	0.000000	0.0	0.793009	0.60921
6	1.000000	0.0	0.000000	0.00000
7	0.000000	0.0	0.000000	1.00000
8	0.000000	0.0	0.000000	1.00000

```
print(x_train)
```

```
['can dance ?' 'can sing ?' 'are good dancer ?' 'are good singer'
 'would like dance' 'would like sing' 'could dance ?' 'could sing ?'
 'what sing together ?']
```

Abbildung 2.1: Veranschaulichung des Ergebnisses nach dem vektorisieren

```
import pandas as pd

vectorizer.get_feature_names()

col = ['feat_'+ i for i in vectorizer.get_feature_names()]
table = pd.DataFrame(X_train.todense(), columns=col)
table
```

Listing 2.11: Features anzeigen. Ergebnis Abbildung 2.1

```
1 from sklearn.naive_bayes import MultinomialNB
2
3 clf = MultinomialNB().fit(X_train, y_train)
4 y_nb_predicted = clf.predict(X_test)
5 print(classification_report(y_test, y_nb_predicted))
```

Listing 2.12: Verwendung des Naive Bayes Classifier

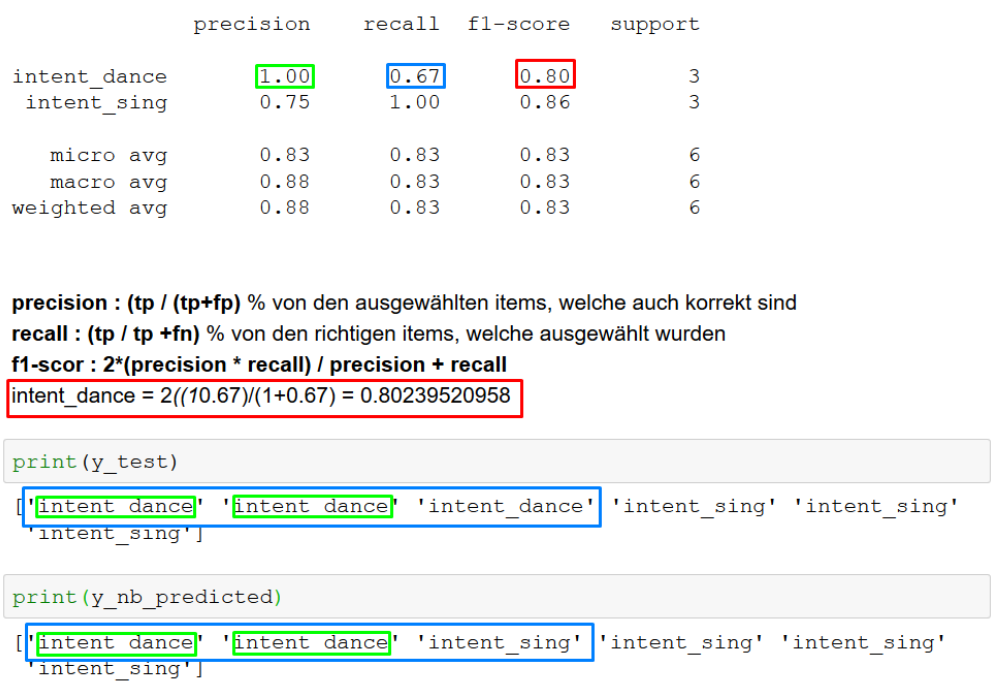


Abbildung 2.2: Veranschaulichung des Ergebnisses mit dem Naive Bayes Classifier

Für das Beispiel wurde der Naive Bayes Classifier verwendet. Sollte das Ergebnis (Abbildung 2.2) nicht ausreichen, kann man versuchen das Ergebnis zu verbesseren mit anderen Sprachverarbeitungsmethoden, Trainingsdaten oder auch einen anderen Classifier verwenden [5].

Entity

Intents können auch gewisse Metadaten aufweisen, welche dann meist "Entities" genannt werden. Wenn ein User 5 Äpfel bestellen möchte mit "Bestelle mir 5 Äpfel", soll das System die Quantität 5 und die Ware Äpfel als Entitäten behandeln, welche dann auch an Folgeprozesse weitergeleitet werden können, wie auch im Beispiel Code 2.1.3 mit "date" zu sehen als Datum.

```
1 import spacy
2 from spacy import displacy
3 from collections import Counter
4
5 nlp = spacy.load("en_core_web_sm")
6 displacy.render(
7     nlp('European authorities fined Google a record $5.1 billion on
          Wednesday for abusing its power in the mobile phone market and
          ordered the company to alter its practices'),
8     jupyter=True, style='ent')
```

Listing 2.13: Entity Extraction Beispiel mit spacy [34]

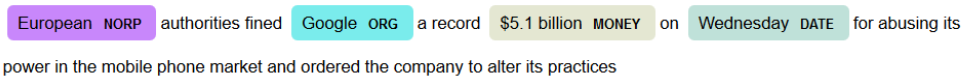


Abbildung 2.3: Veranschaulichung des Ergebnisses spacy [34]

Chatbots werden heute schon häufig eingesetzt, sei es beim Bestellen neuer Kleidung oder beim Support seiner Reisebuchung. Denn meist ist das Ziel, einen Prozess der zuvor von einem Mitarbeiter eines Unternehmens abgearbeitet wurde mithilfe eines Computersystemes zu automatisieren. Doch nicht nur bei Onlineshops oder in Form kleiner Büroklammern, bei Microsoft Produkten (Microsoft Office für Windows: Version 97 bis 2003, Microsoft Publisher: Version 98 bis 2003, Microsoft Office für Mac: Version 98 bis 2004) Anwendungen üblich, als Hilfestellung für den User bei Standardsoftware werden Chatbots eingesetzt. [19]

Abbildung 2.4: Veranschaulichung des Assistenten Karl Klammer [19]

Gerade auch beim ansteuern internetfähiger Glühbirnen die nicht über einen standardmäßigen Lichtschalter einschalten werden oder online über ein Webportal bestellen. Man verwendet heute dafür seine Stimme und kann so einige übliche zeitkostende Abläufe aus dem Alltag optimieren. Dafür wird am Markt ein weites Spektrum an Sprachassistenten angeboten. Die meisten im deutschsprachigen Raum kennen Amazons Alexa durch ihre Werbespots und einige Haushalte haben sie bereits im Wohnzimmer als Unterstützung.

Hier kommt das System Leo-Chat ins Spiel. Es ist der HTBLA-Leonding ein Anliegen Besucher zu unterstützen und auch Schüler beim Schulalltag helfen zu können. Durch die Verwendung eines Sprachassistenten soll dieses Ziel erreicht werden.

2.1.2 Kriterien

- Deutsche Sprache
- Einfache Erweiterung
- Einfache Handhabung
- Einfache Konfiguration
- Kostenlos
- Einfache Teamintegration

2.1.3 Alternativen

Kriterien	Dialogflow	Amazon Alexa
Deutsche Sprache	X	X
Einfache Erweiterung	X	X
Einfache Handhabung	X	X
Einfache Konfiguration	X	X
Kostenlos	X	X
Einfache Teamintegration	X	

Dialogflow

Dialogflow wird für die Erstellung von Conversational Interfaces verwendet und ist mit einem gut trainierten NLU (Natural Language Understanding) ausgestattet. Mit dem gut und durchdachten Benutzerinterface, lassen sich schnell und einfach Intents erstellen. Durch festgelegte Intents kann das System durch übereinstimmung erkennen, wie es auf gewisse Inputs reagieren soll und was es dem Benutzer antworten soll. Wie auf der

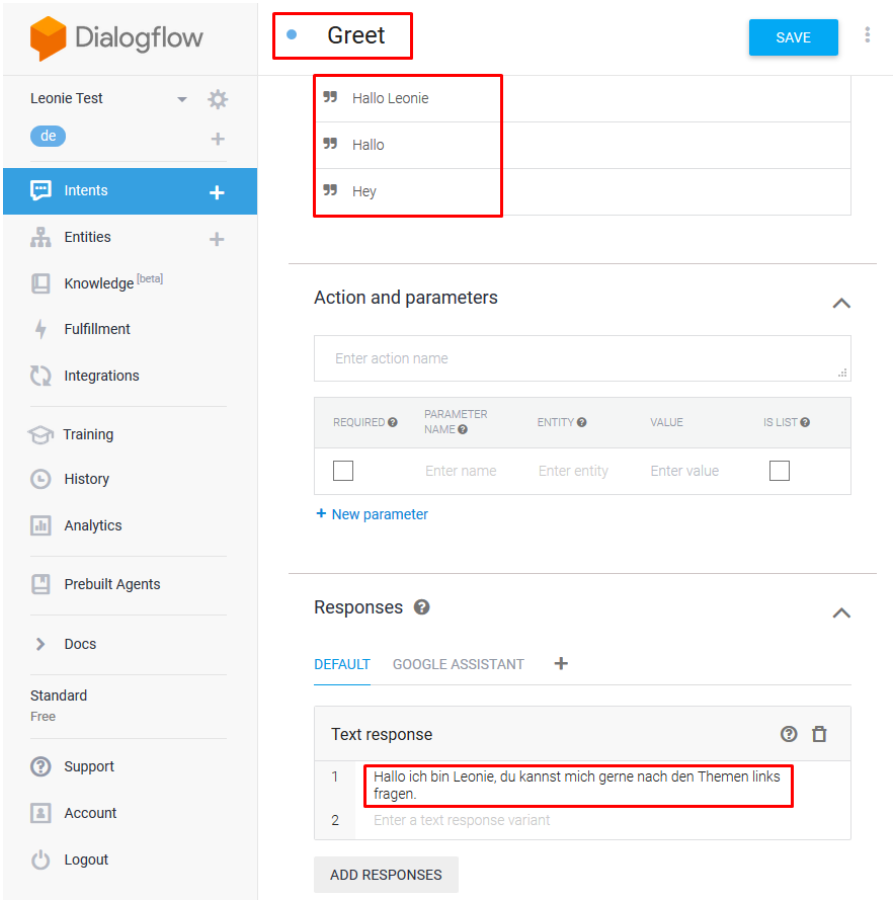


Abbildung 2.5: Veranschaulichung eines Beispiel Intents [28]

Abbildung 2.5 zu sehen, heißt dieser erstellte Intent "Greet". Die Phrasen "Hallo Leonie", "Hallo" oder "Hey" sollen diesen Intent auslösen. Wenn also der Input "Hallo Leonie" an das System gesendet wird, soll das System reagieren. In diesem Beispiel (Abbildung 2.5), antwortet das System mit einer statischen Antwort "Hallo ich bin Leonie, du kannst mich gerne nach den Themen links fragen.". Hat man einen weiteren Intent mit der Auslöse Phrase "Hallo Leonie, wie geht es dir?" erstellt, soll das System bei einem Input von "Hallo Leonie" dieses auch unterscheiden können. Hierfür wird mit Wahrscheinlichkeiten gearbeitet.

```
{
    ...
    "queryResult": {
    "queryText": "Hallo", //input
    ...
    "fulfillmentText": "Hallo ich bin Leonie, du kannst mich gerne
    nach den Themen links fragen.", //output
    ...
    "intent": {
       ...
      "displayName": "Greet"
    },
    "intentDetectionConfidence": 1,
    "languageCode": "de"
  }
}
```

Listing 2.14: Beispiel JSON output von Dialogflow

Hier ist das Ergebnis in Form eines JSON. Es ist zu sehen das das System den Intent zu einer 100 prozentigen Wahrscheinlichkeit getroffen hat und so dem User den "fullfillmentText" "Hallo ich bin Leonie, du kannst mich gerne nach den Themen links fragen." zurückgibt.

```
    "intentDetectionConfidence": 1
```

Dialogflow erlaubt es einem auch die Antwort von Input variablen abhängig zu machen. Dieser Intent heißt "DayOfTheWeek" wird über die Eingabe "Welcher Wochentag

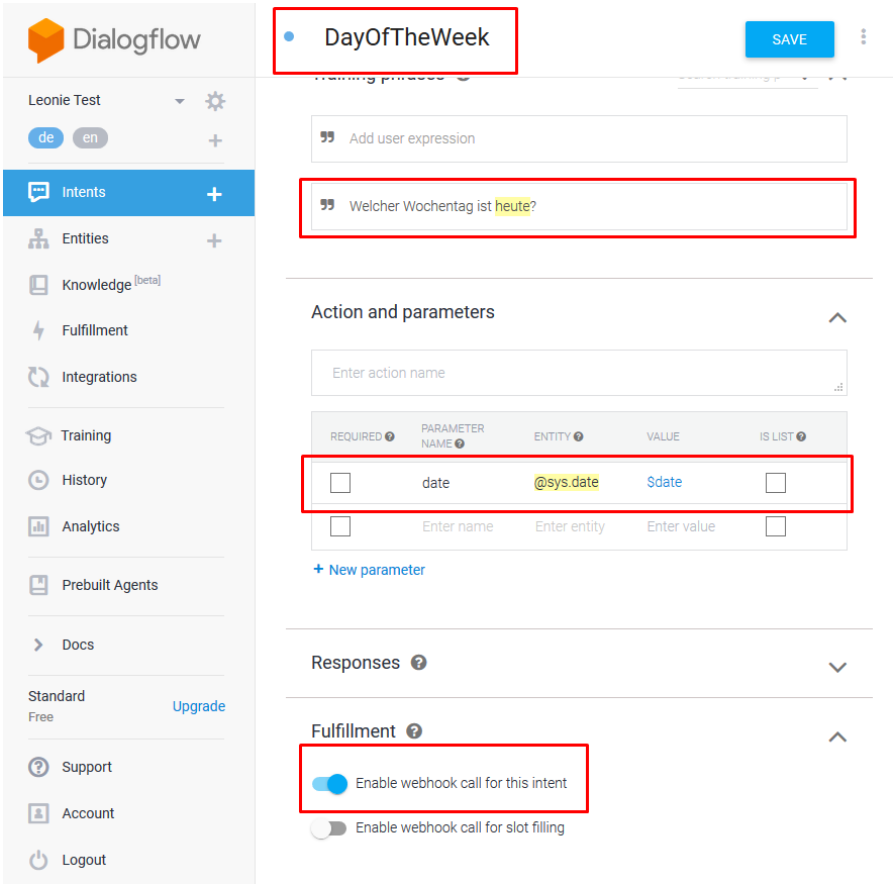


Abbildung 2.6: Veranschaulichung eines Beispiel Intents mit Parameter [28]

ist heute?" ausgelöst und kann mithilfe von Entity Extraction das Datum von dem angegebenen Tag herausfiltern. Sollte der User also fragen "Welcher Wochentag ist heute?", erkennt das System über heute das benötigte Datum. "Welcher Wochentag ist morgen?" würde das System auch erkennen und das Datum für den nächsten Tag zurückliefern.

Da hier eine statischer Rückgabetext keine Sinnhafte Antwort wäre, muss man nach erfolgreichen Auslösen des Intents das Datum an eine Schnittstelle weitergeben, sodass das Datum verarbeitet werden kann und der gewünschte Wochentag an den User weitergeben wird. Dafür wurde die Dialogflow Weboberfläche mit einem guten Editor ausgestattet, der es Erstellern ermöglicht die Umgebung nicht verlassen zu müssen.

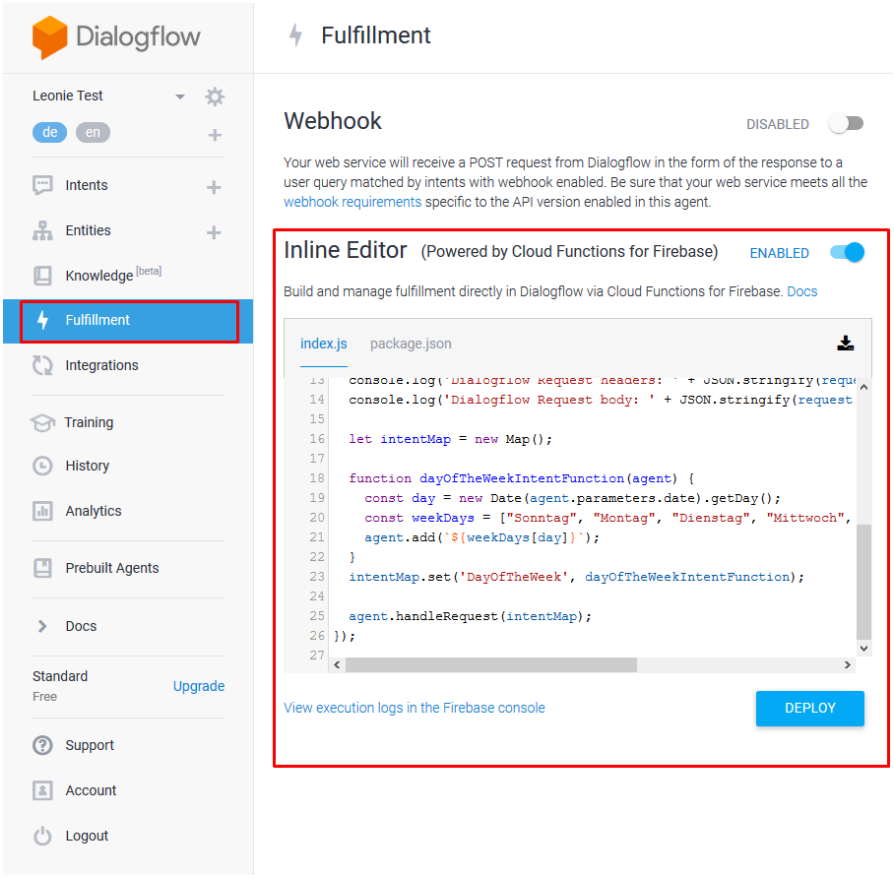


Abbildung 2.7: Dialogflow Inline Editor [28]

Wie in Abbildung 2.7 zu sehen kann man direkt in der Weboberfläche seine Intenterweiterungen programmieren. Wenn man direkt im sogennaten "Inline Editor" programmieren möchte muss man die Programmiersprache Javascript verwenden. Hierfür wird das NodeJS Package dialogflow-fullfillment (https://www.npmjs.com/package/dialogflow-fulfillment) verwendet. Möchte man nicht den Clouddienst Firebase als Deploymentsort verwenden, welcher automatisch über den "Inline Editor" gewählt wird, kann man auch seinen eigenen Server eintragen. Hierbei ist zu beachten, dass der Server über ein SSL Zertifikat abgesichert sein muss (genauere Konfigurationsinformationen findet man unter https://dialogflow.com/docs/fulfillment/configure), jedoch bringt die Verwendung den großen Vorteil mit sich bringt die Programmiersprache frei wählen zu können.

Für den Beispiel Intent "DayOfTheWeek" wird in der Funktion "dayOfTheWeekIntentFunction" in den folgenden Codezeilen der Parameter "date" in ein Datum umgewandelt und über die Wochentagsnummer in einen gültigen Wochentag Text umgewandelt und Benutzer zurückgegeben.

```
const functions = require('firebase-functions');
const {WebhookClient} = require('dialogflow-fulfillment');
const {Card, Suggestion} = require('dialogflow-fulfillment');

process.env.DEBUG = 'dialogflow:debug'; // enables lib debugging statements

exports.dialogflowFirebaseFulfillment = functions.https.onRequest((request, response) => {
    const agent = new WebhookClient({ request, response });
    console.log('Dialogflow Request headers: ' + JSON.stringify(request.headers));
    console.log('Dialogflow Request body: ' + JSON.stringify(request.body));

    let intentMap = new Map();

    function dayOfTheWeekIntentFunction(agent) {
        const day = new Date(agent.parameters.date).getDay();
        const weekDays = ["Sonntag", "Montag", "Dienstag", "Mittwoch", "Donnerstag", "Freitag", "Samstag"];
        agent.add(`${weekDays[day]}`);
    }
    intentMap.set('DayOfTheWeek', dayOfTheWeekIntentFunction);

    agent.handleRequest(intentMap);
});
```

Listing 2.15: AWS Codebeispiel

Durch den gesetzten Haken bei Abbildung 2.6 zu sehen "Enable webhook call for this intent". Werden immer wenn dieser Intent ausgelöst wird, die obengezeigten Codezeilen ausgeführt und dem Benutzer der Wochentag zurückgegeben.

Google hat bei Dialogflow hat die Testumgebung in die Weboberfläche ohne Zusatzseite eingebaut. So kann man seine zuvor erstellten Intents direkt auf der rechten Seite der Weboberfläche testen (Abbildung 2.8), was den Erstellungsprozess erleichtert.

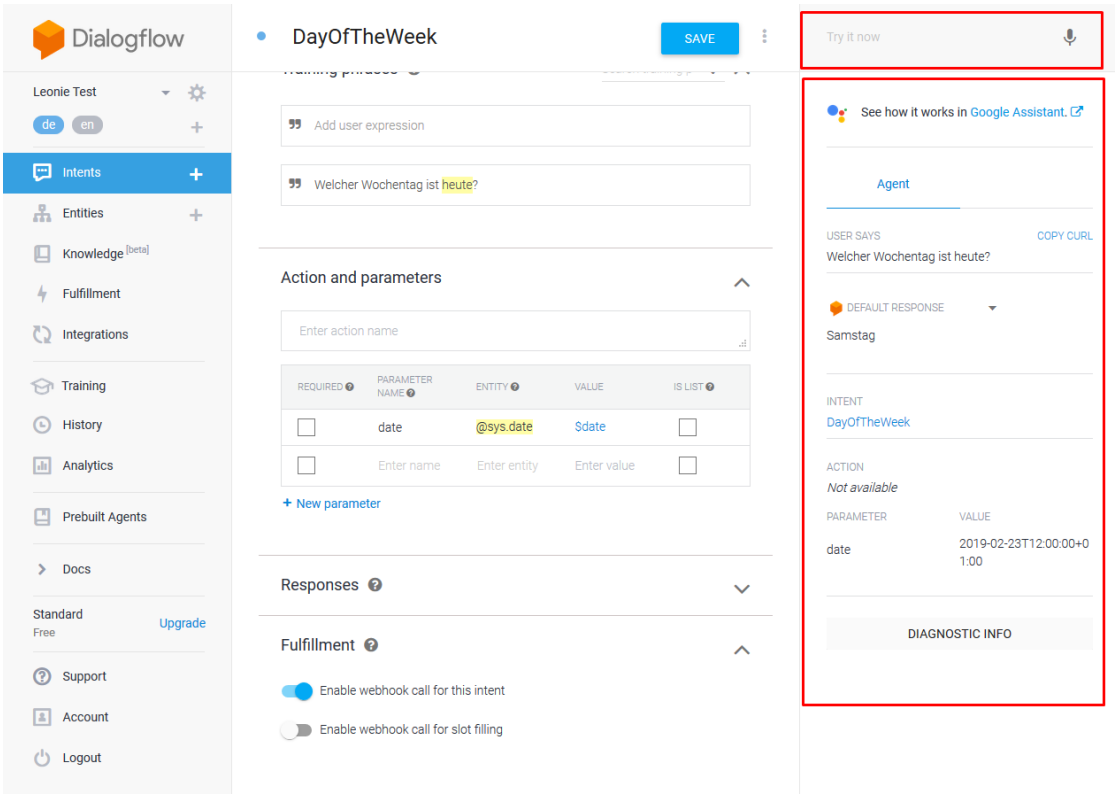


Abbildung 2.8: Dialogflow Testumgebung [28]

Alexa Skill

Amazon Echo ist ein Smart Speaker und sprachgesteuerter, internetbasierter Intelligenter persönlicher Assistent des US-amerikanischen Unternehmens Amazon.com. Er greift auf diverse herstellereigene Dienste sowie Dienste von Drittanbietern zu. Das Gerät wird seit dem 23. Juni 2015 in den Vereinigten Staaten und seit dem 26. Oktober 2016 in Deutschland verkauft. [3]

Abbildung 2.9: Amazon Echo (2.Gen) [2]

Das System kann mit eigenen Skills erweitert werden. Bei Googles System erstellt man eigene Skills mit Dialogflow und bei Alexa mit Alexas Skill Konsole. Die beiden Oberflächen sind sehr ähnlich aufgebaut und führen beide zu einem schnellen Ergebnis. Beim Google System muss man, jedoch noch eine eigene Google Action erstellen falls man es mit seinem Google Home integrieren möchte und bei Alexa kann man direkt nach dem Erstellen und der erfolgreichen Verbindung den Skill in der App auswählen und für seine verbunden Echos aktivieren.

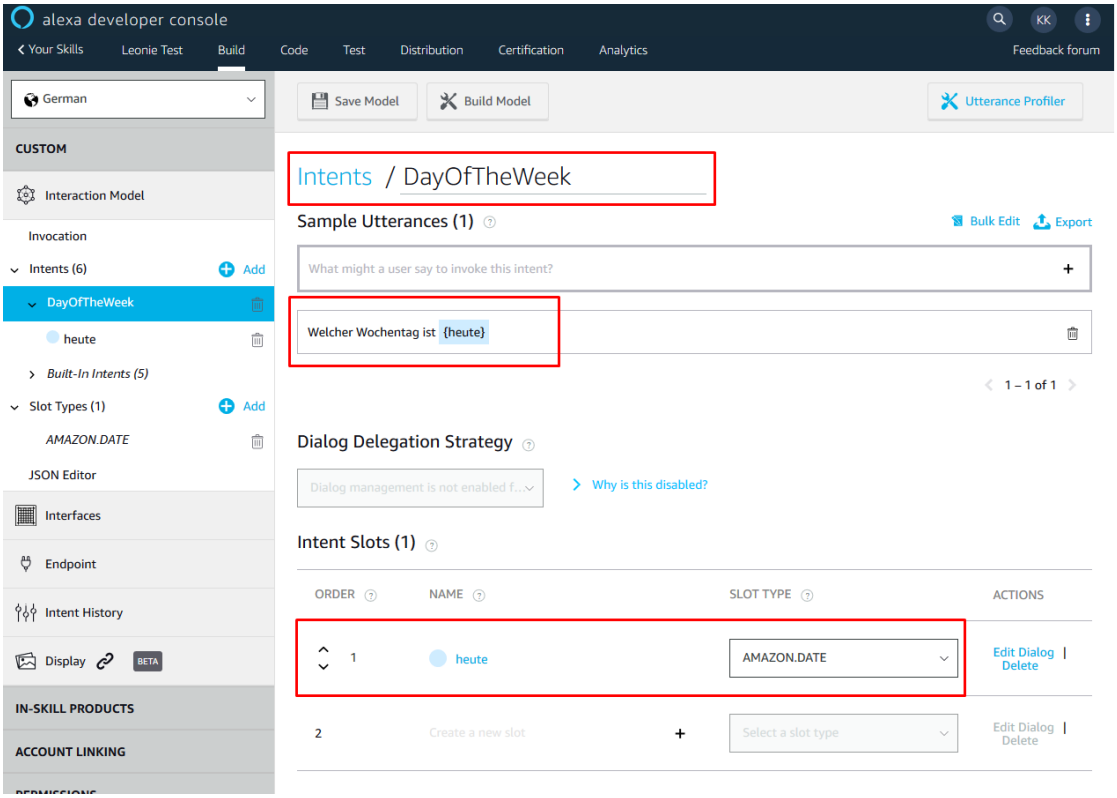


Abbildung 2.10: Veranschaulichung eines Beispiel Intents bei Alexas Skill Console [1]

Bei diesem Beispiel möchte man wie bei dem Beispiel für Dialogflow 2.1.3 zuvor, den Intent "DayOfTheWeek" mit der Phrase "Welcher Wochentag ist heute" auslösen. Der "Intent Slot" mit dem Namen heute wird bei Abbildung 2.10 mit dem Typen "AMAZON.DATE" definiert. So weiß die Entity-Extraction auf welchen Typen es achten soll. Wenn der User somit die Frage stellt "Welcher Wochentag ist heute?" wird dieser Parameter an eine Schnittstelle weitergegeben die das Datum in den Wochentag umgewandelt. Bei eigens erstellten Alexa Skills kann man den Amazon Cloud Service AWS verwenden.

Es kann auch ein eigener Server erstellt werden. Dafür stellt amazon auch ihre Alexa Skill Kit SDK für NodeJS, Java und Python bereit (https://github.com/alexa/alexa-skills-kit-sdk-for-nodejs). Hierbei ist wieder zu beachten, dass die Vereinbarungen für den eigenen Server eingehalten werden (https://developer.amazon.com/de/docs/custom-skills/host-a-custom-skill-as-a-web-service.html).

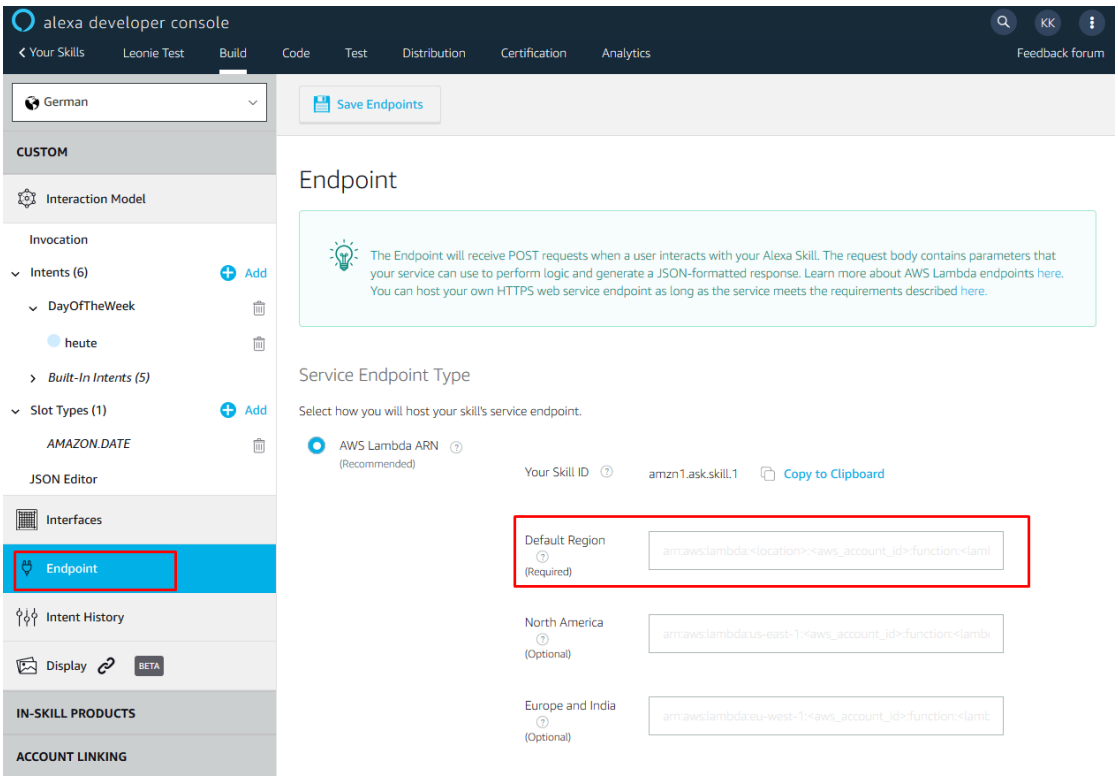


Abbildung 2.11: Alexa AWS Lambda Endpoint [1]

2.1.4 Entscheidung

Dialogflow

2.1.5 Begründung

Dialogflow bietet bessere Features für Teams. Alexas Developer Console erlaubt es einem den gesamten Skill als JSON zu exportieren, jedoch muss man die Änderungen mit z.B. Versionierungssoftware wie GIT für alle Team Member zugänglich machen. Da es wichtig ist, dass das System jeder in kurzer Zeit versteht und erweitern kann, liefert Dialogflow mit seinem Feature die Umgebung mit anderen Team Member zu teilen und so jeder im Team die Änderungen sofort sieht einen großen Vorteil (Abbildung 2.12).

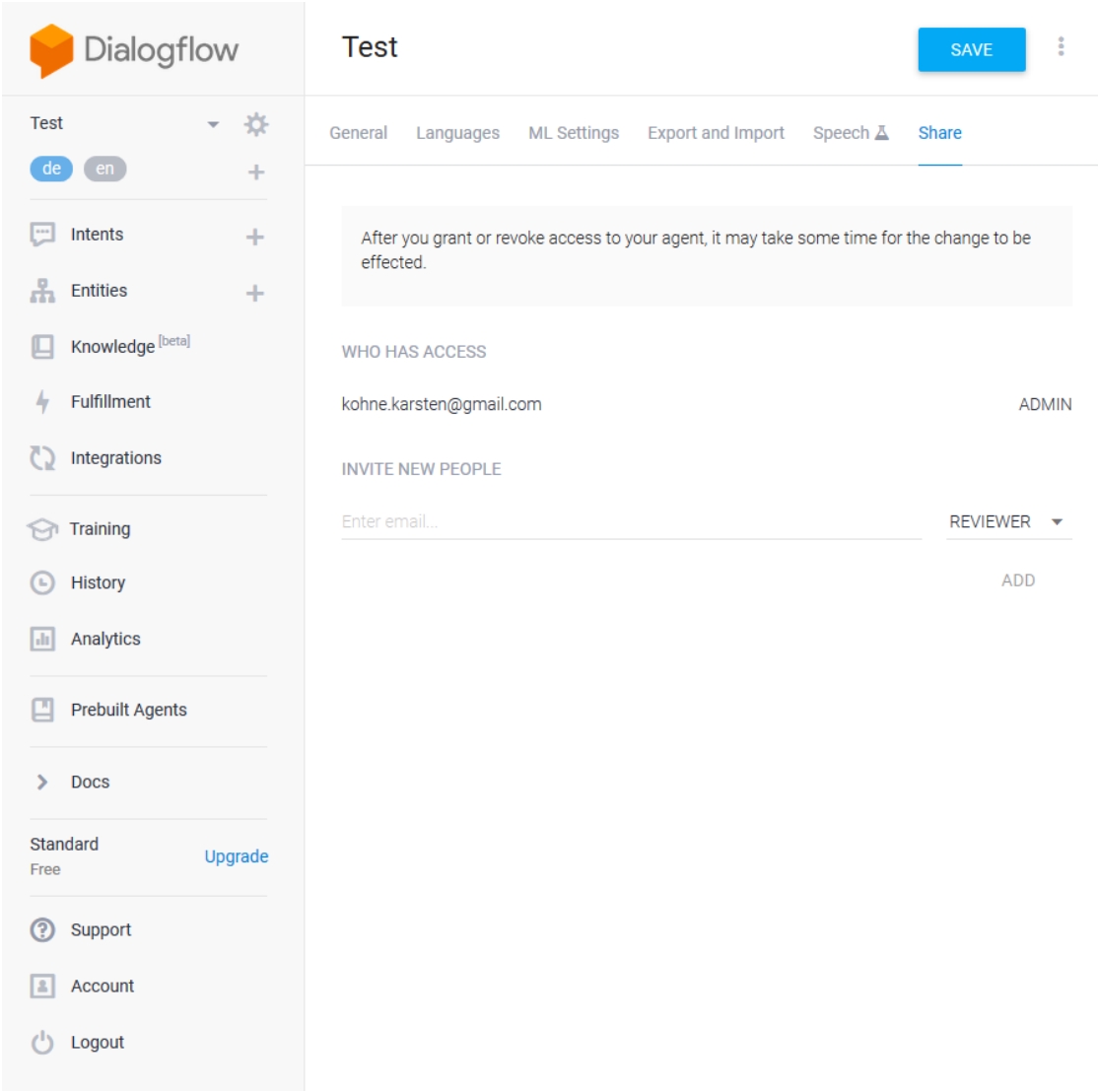


Abbildung 2.12: Dialogflow Umgebung teilen [28]

Vorallem aber ist es wichtig, dass das System nicht nur mit dem Standard Erkennungswort "Alexa" oder "Computer" gestartet werden kann. Es soll über ein eigenes Wakeword gestartet werden können, somit wird eine eigene Hotword Detection benötigt.

Die standard Funktionalitäten von Alexa sind bereits sehr gut und man kann sie bereits fast alles fragen, jedoch wenn man seinen eigenen Skill starten möchte muss man "Alexa, starte Leonie" sagen. "Leonie" wäre in diesem Fall der "Invocationname" (Aufrufename, welcher in der Alexa Developer Console festgelegt werden muss). Des weiteren kann man auch direkt den Skill ansprechen mit "Alexa, frag Leonie welcher Wochentag ist heute?". Wobei hier wieder das Erkennungswort Alexa gesagt werden muss und der beim Skill festgelegt Aufrufname für den eigenen Skill "Leonie", welches für das System Leo-Chat nicht geeignet ist.

Bei Dialogflow kann wie gezeigt in Abbildung 2.7 zu sehen direkt der Server programmiert werden, welcher bei erfolgreicher Auslösung eines Intents zur Ausführung der festgelegten Funktion genutzt wird, falls er aktiviert wurde. Für einfache Chatbots ist das ein großer Vorteil, da man so keinen eigenen Server konfigurieren und absichern muss, jedoch wenn man die Umgebung der Firebase Cloud Dienste und somit keinen Google Dienst verwenden möchte führt das zu einem Problem, denn Google erlaubt bei Firebase keine externen Aufrufe ohne ein Bezahlmethode hinterlegt zu haben (Abbildung 2.13).

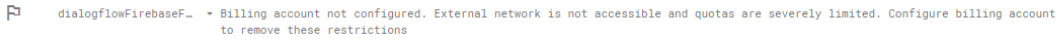


Abbildung 2.13: Firebase Fehlermeldung beim Aufrufen eines externen Dienstes

Dialogflow schafft viele Möglichkeit, den erstellten Chatbot zu nutzen. Dabei stellt Dialogflow auch eine API bereit, mit der man Dialogflow in sein eignes System integrieren kann ohne für den Spracherkennungsdienst von Google bezahlen zu müssen (Spracherkennung). Es erleichtert die Verbindung von Dialogflow mit der Spracherkennung sehr. Auch wenn die leicht zu integrierende Schnittstelle von dem Dialogflow Agent zur Webhook mit Firebase einfach verwendet werden könnte. Für die Kommunikation und die Weiterleitung der Daten wird das eigene Backend genutzt, welches somit keine Kosten bei Google erzeugt. Deshalb wird nur die Fähigkeit ein Chatbot mit Dialogflow schnell und einfach zu erstellen genutzt.

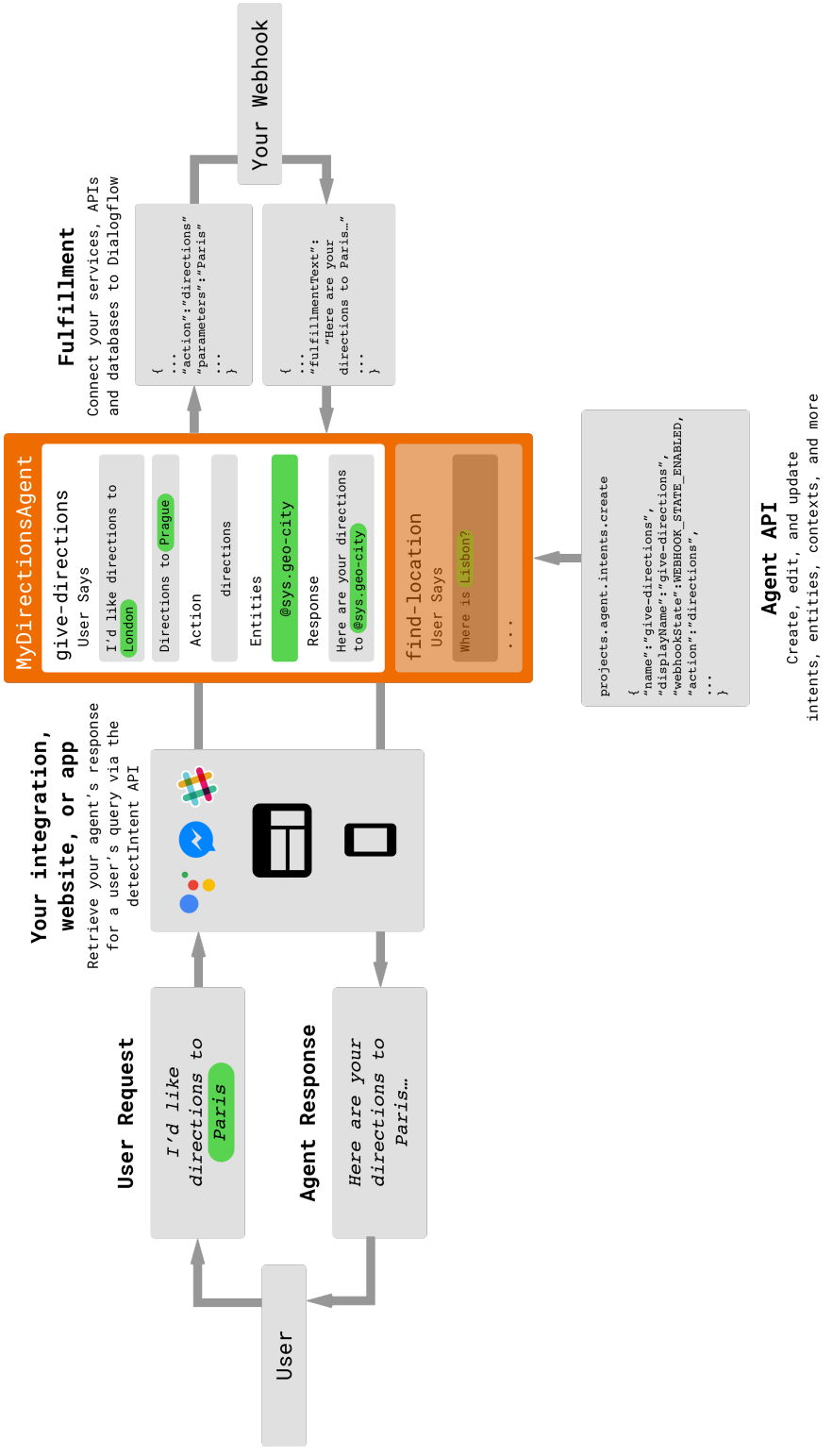


Abbildung 2.14: Dialogflow Ablauf

2.2 Spracherkennung

Der wichtigste Gegenstand der Sprachverarbeitung ist das Sprachsignal. Es entsteht, wenn eine Person etwas spricht und die produzierten Schallwellen über einen elektroakustischen Wandler (Mikrofon) in ein elektrisches Signal umgewandelt werden. Das Sprachsignal wird also durch das, was die Person sagt(Aussage), geprägt. [55]

Abbildung 2.15: Veranschaulichung eines Sprachsignals. Text = "Sprachassistenten kommen immer öfter in Einsatz"

Dem Menschen fällt das Verstehen von Sprache im Allgemeinen leicht. Er kann sogar unter schwierigen Umständen relativ mühelos kommunizieren und gewöhnt sich schnell an eine außergewöhnliche Stimme oder Sprechweise. Auch in einer ungünstigen akustischen Umgebung ist eine Verständigung möglich, also wenn es beispielsweise lärmig ist oder die Sprache über eine schlechte Telefonverbindung übertragen wird. Angesichts der menschlichen Fähigkeiten ist man geneigt anzunehmen, dass auch die maschinelle Spracherkennung kein so großes Problem sein sollte. [55]

Bei einem Sprachassistenten ist die Verarbeitung der Sprache, welches auch im englische Speech Recognition/Speech-To-Text genannt wird, ein wichtiges Kriterium, damit auf die Fragen des Benutzers eine gültige Antwort geliefert werden kann. Sollte das System die Sprache des Benutzers nicht richtig umwandeln, würde so das Ergebnis der anschließenden Textverarbeitung etwas liefern, was der Benutzer nicht wissen wollte und würde so keine Antwort auf die von ihm gestellte Frage bekommen.

Dabei sind viele Punkte zu beachten. Vor allem aber die Genauigkeit der Verwandlung von der Sprache in den Text.

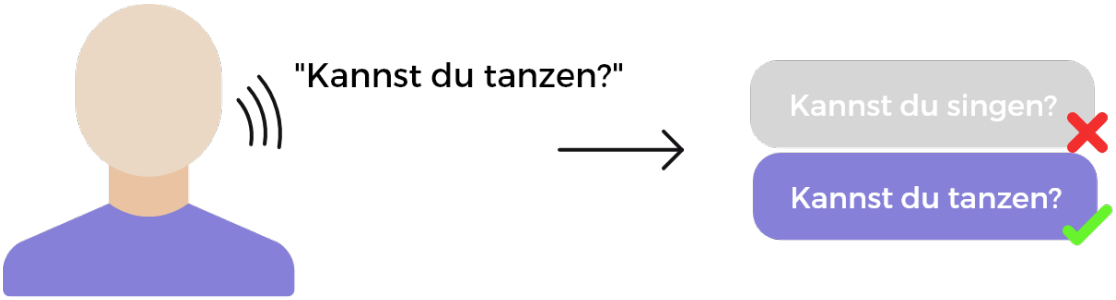


Abbildung 2.16: Genauigkeit Verwandlung von der Sprache in den Text

Da die Spracherkennung bei einem Sprachassistenten nicht nur vordefinierte Wörter erkennen soll, sondern man nicht weiß wie der User die Fragen stellen wird, muss die Spracherkennung über den nötigen Wortschatz verfügen, um den Sprachinput auch richtig zu verstehen.

Doch nicht nur die Genauigkeit der Verwandlung von der Sprache in den Text ist ausschlaggebend, sondern selbst die durch Geräusch veränderte Umgebungslautstärke spielt eine große Rolle.

Denn gerade bei der Erkennung von Homophonen (Wörter mit unterschiedlicher Bedeutung und gleicher Aussprache) fällt es Spracherkennungssystem schwer diese zu unterscheiden. Kommt dann noch die Lautstärke Kulisse eines Raumes mit vielen sich unterhaltenden Personen oder lauter Musik hinzu, sind Systeme ohne weiterer Verarbeitung des Audiosignals mithilfe von Lärmunterdrückungs Software oder anderen Hilfen kaum noch zu verwenden.

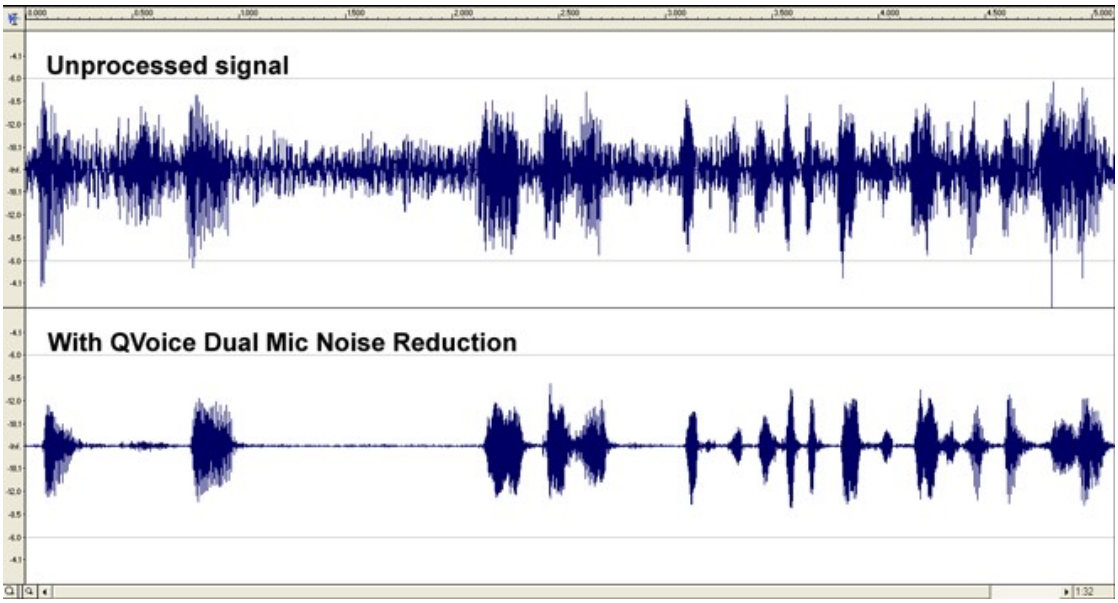


Abbildung 2.17: Veranschaulichung von Lärmunterdrückungs Software

Auch Pegeleinstellungen bei Mikrofonen können zu einem rauschfreieren Ergebnis führen. Da so die meist schlecht verbauten Mikrofonverstärker nicht unter voller Spannung stehen und somit die Aufnahme leiser ist.

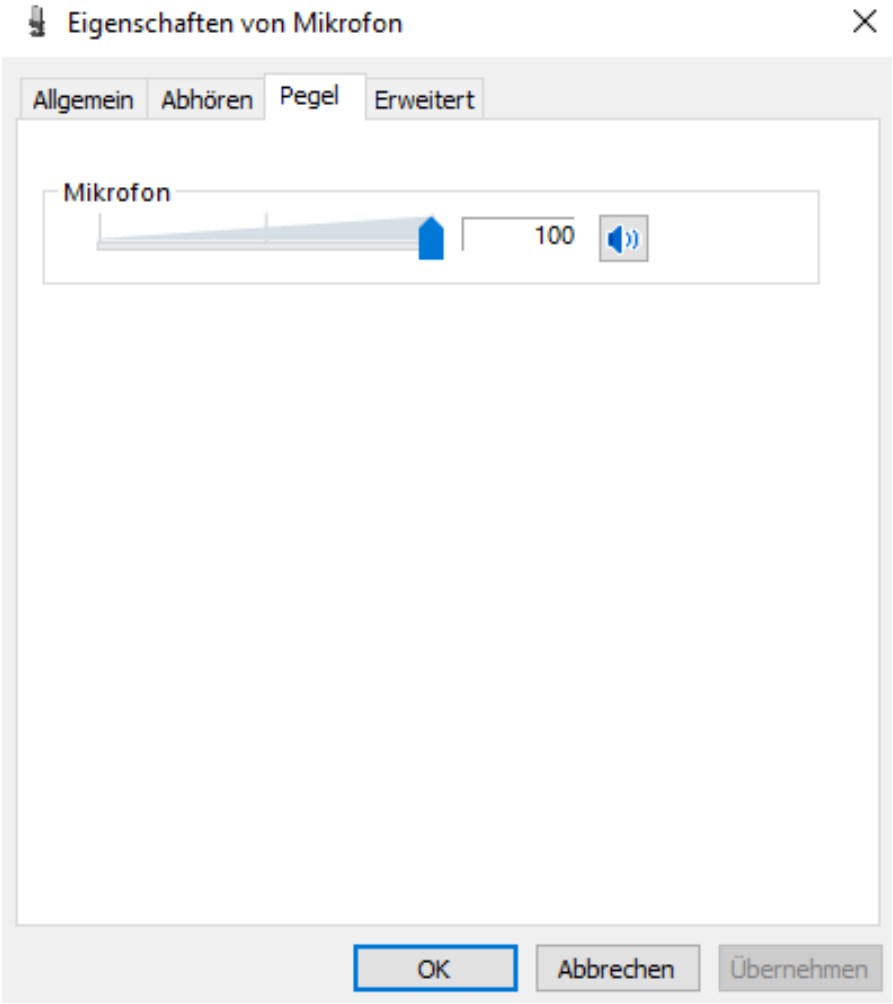


Abbildung 2.18: Veranschaulichung von Pegeleinstellungen beim Windows

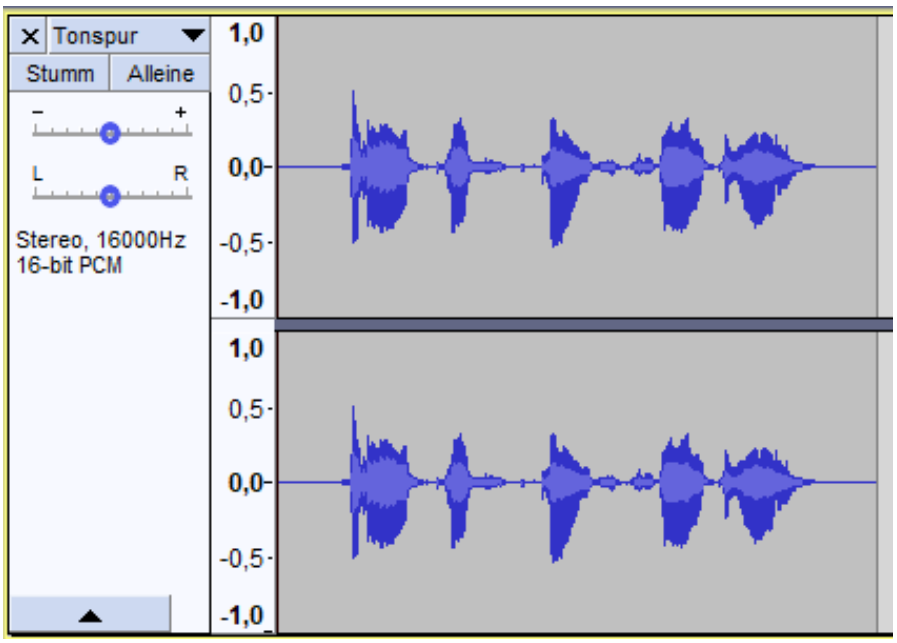


Abbildung 2.19: Veranschaulichung des Ergebnisses bei Pegeleinstellung 100

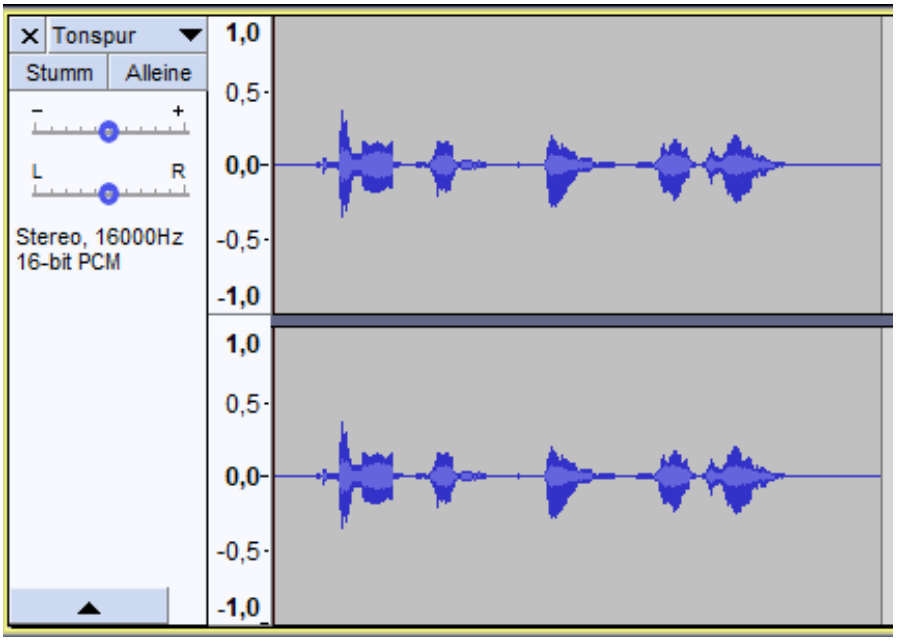


Abbildung 2.20: Veranschaulichung des Ergebnisses bei Pegeleinstellung 80

2.2.1 Kriterien

- Deutsche Sprache
- Großer Wortschatz
- Gute Spracherkennung
- Kostenlos

2.2.2 Alternativen

Spracherkennung	Deutsche Sprache	Großer Wortschatz	Gute Spracherkennung	Kostenlos
Web Speech API	X	X	X	X
Azure: STT	X	X	X	
Watson: STT	X	X	X	

STT - Speech to text

Die Abkürzung STT steht für Spracherkennung [35]

Web Speech API

Die W3C standardisierte Web-Speech-API soll es Webentwicklern ermöglichen, in einem Webbrowser Funktionen für Spracheingabe und Text-zu-Sprache-Ausgabe bereitzustellen, die normalerweise nicht verfügbar sind, wenn Standard-Spracherkennungs- oder Bildschirmleseprogramme verwendet werden. Die API ermöglicht sowohl eine kurze (einmalige) Spracheingabe als auch eine kontinuierliche Spracheingabe.

Für die Verwendung der Spracherkennung auf einer Webseite ist in Chrome ein serverbasiertes Erkennungsmodul erforderlich. Der Audio Input wird zur Erkennungsverarbeitung an einen Webdienst von Google gesendet, somit funktioniert es nicht offline. Die Web-Speech-API kann jedoch kostenlos verwendet werden. [57,]

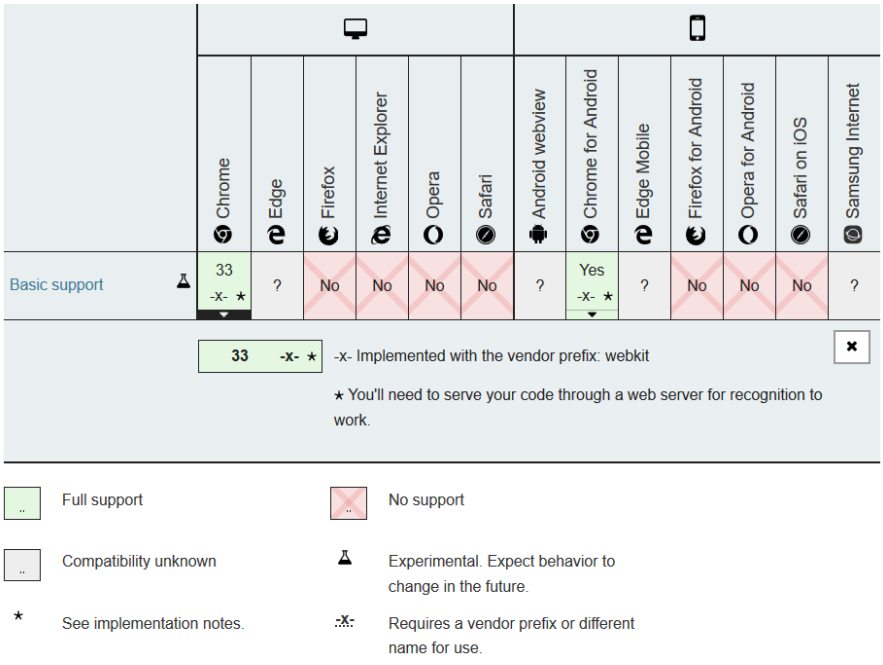


Abbildung 2.21: Web Speech API Browser Kompatibilität [36]

Zur Verwendung in Chrome kann wie folgt vorgegangen werden. [58]

```
window.SpeechRecognition = window.webkitSpeechRecognition || window.
    SpeechRecognition;

if ('SpeechRecognition' in window) {
    // speech recognition API supported
} else {
    // speech recognition API not supported
}

const recognition = new window.SpeechRecognition();

recognition.onresult = (event) => {
    console.log(event.results[0][0])
}

recognition.start();
```

Listing 2.16: Verwendung von SpeechRecognition in Chrome

Microsoft Azure: Speech to Text

Der Speech-Dienst basiert genau wie die anderen Azure-Sprachdienste auf den Sprachtechnologien aus Produkten wie Cortana und Microsoft Office. Der Speech-Dienst vereint in sich die Azure-Sprachfeatures, die zuvor über die Dienste Bing-Spracheingabe-API, Sprachübersetzung, Custom Speech und Custom Voice verfügbar waren. Nun kann über ein einzelnes Abonnement auf alle diese Funktionen zugegriffen werden. [38,]

Der Spracherkennungsdienst von Microsoft kann über eine SDK (Software Development Kit) oder auch über eine REST-Schnittstelle angesprochen werden. So kann man auch eine kurze wie auch kontinuierliche Spracheingabe auswerten. Dieser Service ist jedoch bei häufiger Verwendung nicht kostenlos (Preistabelle).

INSTANZ	GLEICHZEITIGE ANFORDERUNG	MERKMALE	PREIS
Free	1 gleichzeitige Anforderung	Sprachübersetzung	5 Stunden kostenlos pro Monat
		Spracherkennung	5 Stunden kostenlos pro Monat
		Spracherkennung mit benutzerdefinierten Sprachmodellen	5 Stunden kostenlos pro Monat
		Speech-Endpunkthosting[1]	1 kostenloses Modell pro Monat
		Sprachsynthese mit Standardstimmen	5 Millionen kostenlose Zeichen pro Monat
		Sprachsynthese mit benutzerdefinierten Voicefonts	5 Millionen kostenlose Zeichen pro Monat
		Sprachsynthese mit neuronalen Stimmen	0,5 Millionen kostenlose Zeichen pro Monat
		Hosting benutzerdefinierter Voicefonts[1]	1 kostenloses Modell pro Monat
Standard	20 gleichzeitige Anforderungen	Sprachübersetzung	€2,109 pro Stunde
		Spracherkennung	€0,844 pro Stunde
		Spracherkennung mit benutzerdefinierten Sprachmodellen	€1,181 pro Stunde
		Speech-Endpunkthosting	€33,74/Modell/Monat
		Sprachsynthese mit Standardstimmen	€3,374 pro 1 Mio. Zeichen
		Sprachsynthese mit benutzerdefinierten Voicefonts	€5,06 pro 1 Mio. Zeichen
		Sprachsynthese mit neuronalen Stimmen	Nicht zutreffend pro 1 Mio. Zeichen[2]

Abbildung 2.22: Microsoft Azure: Speech to Text Preise [38]

IBM Watson: Speech to Text

Der IBM Spracherkennungsdienst kann über eine REST Schnittstelle oder auch über ein Websocket Interface abgerufen werden. Somit ist eine kurze wie auch kontinuierliche Spracheingabe möglich. Der Dienst nutzt maschinelles Lernen, um Grammatikkenntnisse, Sprachstrukturen und die Zusammenstellung von Audio- und Sprachsignalen, um die menschliche Stimme genau zu transkribieren. Es aktualisiert und verfeinert kontinuierlich seine Transkription, wenn es mehr Sprache erhält. [52,] Dieser Service ist jedoch nicht kostenlos. Man bezahl pro Minute 0.02 $ [30]

2.2.3 Entscheidung

Web Speech API

2.2.4 Begründung

Da das System ohne zusätzlichen Aufwand im Webbrowser auch funktionieren soll und die Web Speech API als Spracherkennung dafür geeignet ist und dabei ein sehr gutes Ergebnis liefert, fiel die Wahl nicht schwer. Der größte Vorteil ist, dass die Web Speech API kostenlos ist und keine weitere Anmeldung bei einem Cloud Dienst wie z.B. bei der Microsoft Azure: Speech to Text benötigt wird.

Die Web Speech API als Spracherkennung wird jedoch nur im Chrome Browser unterstützt (Abbildung 2.21) und kann nur über einen Webserver, welcher über SSL gesichert ist abgerufen werden.

2.3 Sprachausgabe

"Seit Jahrhunderten waren Menschen von der Idee fasziniert, eine Maschine zu konstruieren, mit welcher künstliche Lautsprache erzeugt werden kann. Wissenschaftliche Grundlagenarbeiten und erste Konstruktionen von akustischen Versuchsapparaten in dieser Richtung fanden in der zweiten Hälfte des 18. Jahrhunderts statt. Am berühmtesten ist wohl die "sprechende Maschine" von Wolfgang von Kempelen, mit der nicht nur einzelne Laute, sondern auch Wörter und kürzere Sätze erzeugt werden konnten. Die wichtigsten Komponenten dieser Maschine waren ein Blasebalg, ein vibrierendes Stimmblatt und ein verformbares Lederrohr. Die Maschine war jedoch kein Automat, sondern eher ein Instrument, mit dem ein fingerfertiger Spieler mit viel Übung gewisse sprachähnlichen Lautabfolgen zu produzieren vermochte. Von Kempelen hat mit seinen Arbeiten insbesondere gezeigt, dass der Vokaltrakt der zentrale Ort der Artikulationist. Zuvor war man der Ansicht, dass die Sprachlaute vom Kehlkopf erzeugt würden." [55]

Abbildung 2.23: Nachahmung einer Kempelens Sprachmaschine [45]

Mit dem Begriff Sprachsynthese ist hier die Umsetzung geschriebener Sprache(orthographischer Text) in Lautsprache, also in Sprachsignale, gemeint. Diese Umsetzung steht in Analogie zu einer Person, die vorliest. Die Aufgabe der Sprachsynthese ist somit nichts Geringeres als das zu leisten, was diese Person leistet. [55]

Wenn man an ein Sprachassistent-Systeme denkt, sei es am Smartphone, am Computer oder eine Box im Wohnzimmer. Denkt man meist an eine angenehme Stimme die einem eine Frage in schneller Zeit beantwortet. Es ist wichtig eine angenehme und klare Stimme für einen Sprachassistent zu wählen, damit der Benutzer das System auch gerne verwendet und die Antwort somit verstehen kann.

2.3.1 Kriterien

- Deutsche Sprache
- Klare Stimme
- Frauen und Männerstimme
- Kostenlos

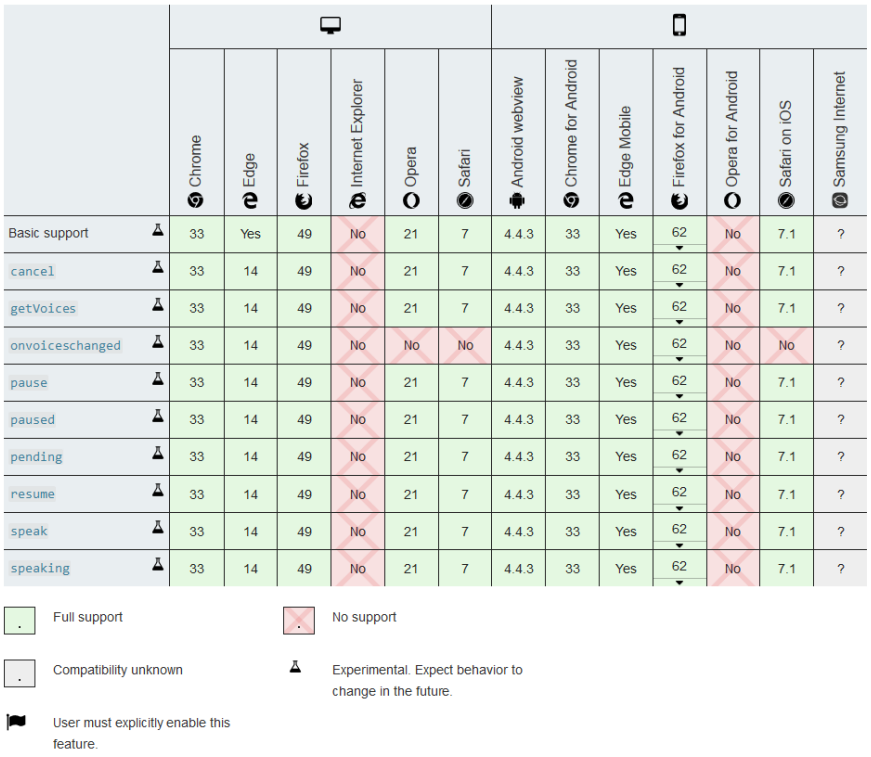
2.3.2 Alternativen

Sprachausgabe	Deutsche Sprache	Klare Stimme	Frauen und Männer Stimme	Kostenlos
Web Speech API	X	X	X	X
Azure: TTS	X	X	X	
Watson: TTS	X	X	X	
Google: TTS	X	X	X	

Web Speech API

Die W3C standardisierte Web-Sprach-API soll es Webentwicklern ermöglichen, in einem Webbrowser Funktionen für Spracheingabe und Text-zu-Sprache-Ausgabe bereitzustellen, die normalerweise nicht verfügbar sind, wenn Standard-Spracherkennungs- oder Bildschirmleseprogramme verwendet werden.

Die SpeechSynthesis ist nicht wie die SpeechRecognition nur für Chrome verfügbar, sondern funktioniert auch mit anderen Browsern.



	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	Android webview	Chrome for Android	Edge Mobile	Firefox for Android	Opera for Android	Safari on iOS	Samsung Internet
Basic support	33	Yes	49	No	21	7	4.4.3	33	Yes	62	No	7.1	?
cancel	33	14	49	No	21	7	4.4.3	33	Yes	62	No	7.1	?
getVoices	33	14	49	No	21	7	4.4.3	33	Yes	62	No	7.1	?
onvoiceschanged	33	14	49	No	No	No	4.4.3	33	Yes	62	No	No	?
pause	33	14	49	No	21	7	4.4.3	33	Yes	62	No	7.1	?
paused	33	14	49	No	21	7	4.4.3	33	Yes	62	No	7.1	?
pending	33	14	49	No	21	7	4.4.3	33	Yes	62	No	7.1	?
resume	33	14	49	No	21	7	4.4.3	33	Yes	62	No	7.1	?
speak	33	14	49	No	21	7	4.4.3	33	Yes	62	No	7.1	?
speaking	33	14	49	No	21	7	4.4.3	33	Yes	62	No	7.1	?

Abbildung 2.24: Unterstützte Browser für die SpeechSynthesis [37]

```
1 const synth = window.speechSynthesis;
2 synth.getVoices();
```

Listing 2.17: Verwendung von SpeechSynthesis in Chrome

Falls man alle verfügbaren Stimmen eines System anzeigen möchte kann man über die Browser Konsole diese Befehle eingeben.

Microsoft Azure: Text to Speech

Die Microsoft Azure Text-to-Speech-API konvertiert Eingabetext in natürlich klingende Sprache (dies wird auch als Sprachsynthese bezeichnet). Zur Generierung von Sprache sendet Ihre Anwendung HTTP POST-Anforderungen an die Text-to-Speech-API. Dort wird Text in menschliche Sprache synthetisiert und als Audiodatei zurückgegeben. Es werden viele Stimmen und Sprachen unterstützt.

Der Text-to-Speech-Dienst von Microsoft bietet mehr als 75 Stimmen in mehr als 45 Sprachen und Gebietsschemas. Wenn Sie diese standardmäßigen „Voicefonts“ verwenden möchten, müssen Sie beim Aufrufen der REST-API nur den Namen der Stimme mit einigen anderen Parametern angeben. [49]

IBM Watson: Text to Speech

Der IBM Text-to-Speech-Dienst konvertiert geschriebenen Text in natürlich klingende Sprache, um Sprachsynthese Funktionen für Anwendungen bereitzustellen. [12]

Kann auch einfach für alle verschiedenen Sprachen getestet werden über https://text-to-speech-demo.ng.bluemix.net/

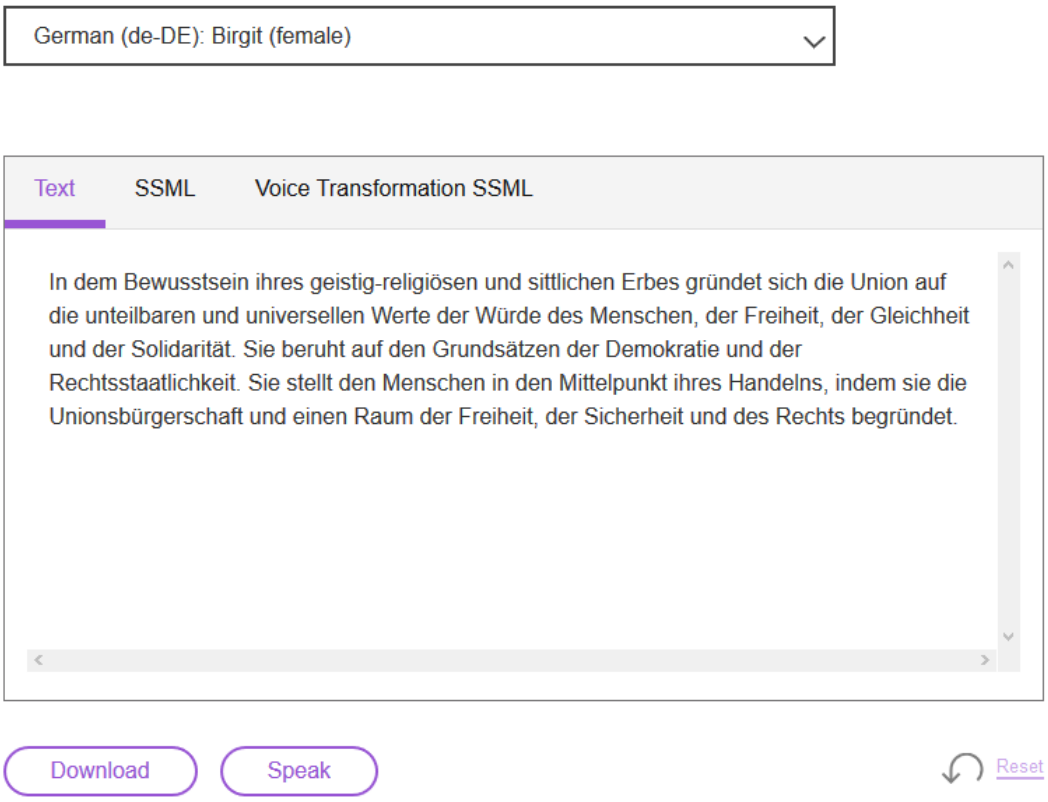


Abbildung 2.25: IBM Watson: Text to Speech Testen

Google: Text to Speech

Mit Google Cloud Text-to-Speech können Entwickler natürlich klingende Sprache mit 30 Stimmen synthetisieren, die in mehreren Sprachen und Sprachvarianten verfügbar sind. Es werden die bahnbrechende Forschung von DeepMind in WaveNet und die leistungsstarken neuronalen Netzwerke von Google angewendet, um erstklassige Audiodateien zu erhalten. Mit dieser benutzerfreundlichen API können Sie naturgetreue Interaktionen mit Ihren Nutzern in vielen Anwendungen und Geräten erstellen. [6]

Kann auch einfach getestet werden über https://cloud.google.com/text-to-speech/

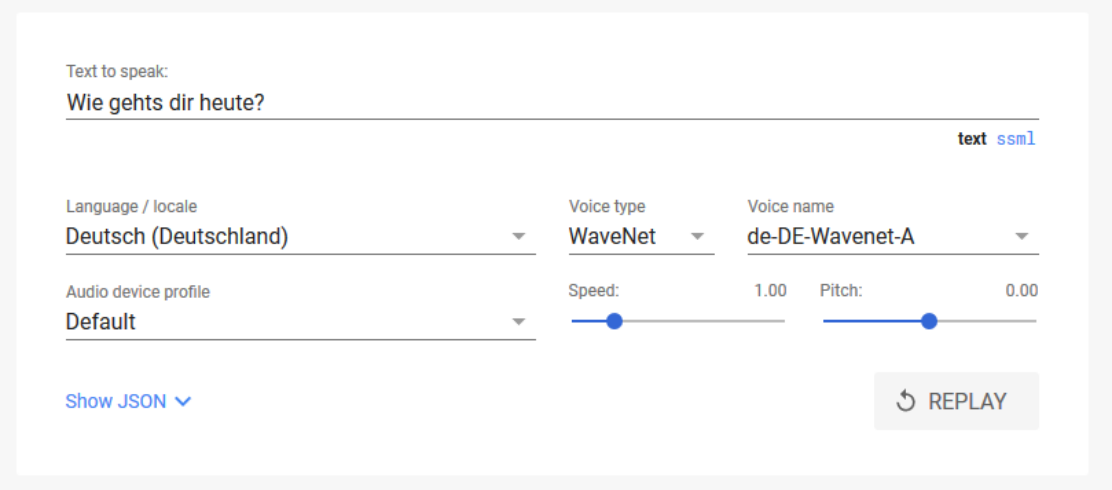


Abbildung 2.26: Google: Text to Speech Test

2.3.3 Entscheidung

Web Speech API

2.3.4 Begründung

Die Web Speech API liefert sehr gute Ergebnise und darf auch laut der gewählen Lizenzvereinbarung (https://github.com/mdn/web-speech-api/blob/master/LICENSE) kommerziell genutzt werden. Alle anderen Alternativen wären mit Kosten verbunden und liefern zwar teilweise ein besseres Ergebnis, aber stehen durch die damit verbundenen Kosten nicht in Relation. Demo https://github.com/mdn/web-speech-api/tree/master/speak-easy-synthesis

2.4 Hotword Detection

"Alexa, spiel meine Fitness playlist!", viele werden diese Phrase aus der Werbung von Amazons Alexa schon gehört haben. Hierbei ist gut zu erkennen wie in der Werbung der Sprachassistent von Amazon gestartet wurde. Denn immer wenn man Befehle an seine Alexa erteilen möchte oder Fragen hat, muss man seine Frage/Befehle mit "Alexa" beginnen (Sollte das standardmäßige Hotword/Wakeword nicht geändert worden sein). So signalisiert man dem System, dass es jetzt zuhören soll und die Frage beantworten soll.

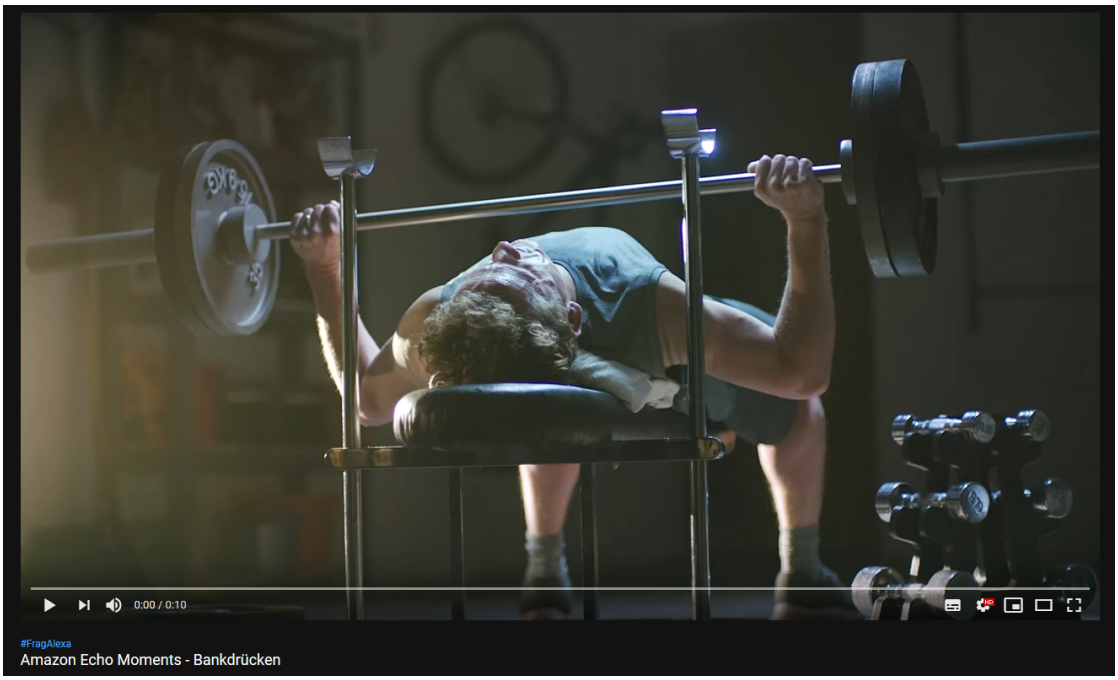


Abbildung 2.27: Amazon Alexa Werbung www.youtube.com/watch?v=71YJyRu4p00

Die meisten Konkurrenzprodukte zur Alexa wie z.B. der Google Assistant oder Siri von Apple haben auch ihre eigenen Erkennungsworte (Hotword) mit denen sie gestartet werden können. Bei Googles Assistant ist es "Hey Google"' und bei Apples Siri ist es "Hey Siri". Gerade bei Amazons Alexa ist es schwierig nicht den Assistenten zu starten, da man schnell einmal über seine neue Errungenschaft "Alexa" mit seinen Freunden sprichten und so den Sprach Assistent startet. Deshalb ist es gerade für die Platzierung von Sprachassistenten im öffentlichen Raum leichter ein Hotword zu wählen, welches nur durch bewusstes ansprechen des Assistenten zum Starten bringt. Hierbei kann man sich Googles gewähltes Erkennungswort als Beispiel nehmen. Denn das man "Hey Google" wortwörtlich in einem Gespräch sagt, ist sehr unwahrscheinlich. Deshalb auch die klare Wahl der Ansprache mit "Hey Google".

Natürlich heißt das im weiteren Prozess der Entwicklung, dass die Erkennung ein "Hey Google" nicht mit einem "Ho Google" verwechseln darf. Zudem kommt noch hinzu, dass viele eine andere Aussprache wählen und so ein "Hey Google" nicht nur durch die unterschiedlichen Stimmen schwer zu unterscheiden ist, sondern auch durch die verschiedenen Aussprachen erschwert wird.

Für das System "Leo-Chat" wurde als Hotword "Hey Leonie" gewählt.

Wie bei der Spracherkennung auch ist die Wahl der richtigen Hardware ausschlaggebend für das Ergebnis. Denn für die Erkennung des Hotwords muss ein Mikrofon bis zur schlussendlichen Erkennung dauerhaft aufnehmen und dieses Signal für die weitere Verarbeitung an den angeschlossenen Computer und Mikrocontroller weiter senden. Eine hohe Umgebungslautstärke führt hierbei wieder zu einem großen Problem, da so das Audiosignal ohne weiterer Filter kaum zu verwenden ist.

Für die Wahl der richtigen Hardware wurde unter anderem der Benchmarktest von Charles Rouchon [56] genommen, um die Qualität der Erkennung der meist verwendetsten Mikrofone für die Erkennung von Hotwords und deren digitale Signalverarbeitungs prozessoren zu vergleichen.

Bei dem Test wird zum einen die Rate an erfolgreich erkannten Erkennungsworten, mit einer unterschiedlichen Distanz von 0,5 m bis 5 m, in einem Intervall von 3 Sekunden, 25 mal wiederholt gesagt.

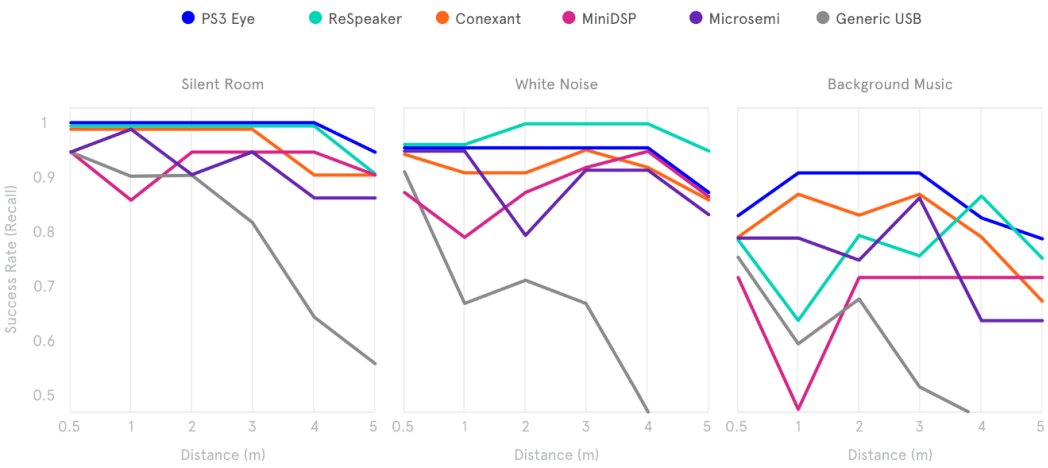


Abbildung 2.28: Ergebnis des ersten Experiments bei unterschiedlicher Distanz

Beim zweiten Test wurde die Rate an erfolgreich erkannten Erkennungsworten bei unterschiedlichen Winkel der Audioquelle, welche das Erkennungswort wiederholt sagt getestet. Hierbei war der Lautsprecher immer 1,5m entfernt.

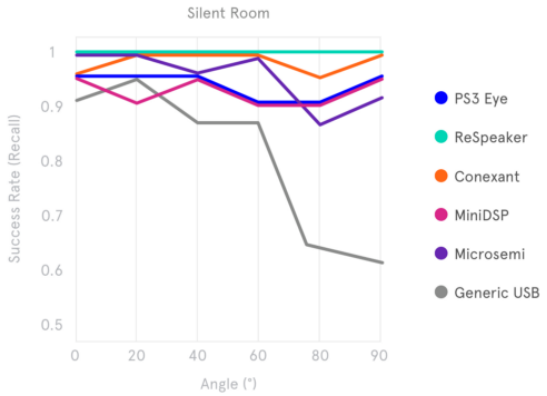


Abbildung 2.29: Ergebnis des zweiten Experiments bei unterschiedlichen Winkeln

Bei den Tests ist zu sehen, dass die Playstation 3 Eye Kamera, welche für eine Videospiel entwickelt wurde, mit einem guten Mikrofon ausgestattet ist und so meist ein gutes Ergebnis liefert. Vergleicht man nun noch die Preise der einzelnen Geräte erkannt man, dass die die Playstation 3 Eye nicht nur mit seiner guten Erkennungsrate glänzen kann, sondern auch mit einem unschlagbaren Preis von ungefähr 10 Euro (Stand Februar 2019). Im Vergleich liegen die anderen getesteten Geräte bei 120 Euro. Hier ist jedoch das "Conexant 4-mic Development Kit" der große Ausreißer mit 350 Euro. Der verbaute Audio Chip "Conexant CX20924" ist einer, welcher auch in der Alexa zu finden ist und hat auch das Hotword der Alexa bereits auf dem Chip für die besste Erkennung bei Alexa Systemen.

Desweiteren haben wir auch weitere Webcams getestet, da die Gesichtserkennung den Audiostream nicht benötigt und so das Systemminimaler werden würde, da man nicht viel Hardware benötigen würde. Es wurde die Logitech BRIO [20] getestet, welche für die Verwendung bei niedriger Umgebungslautstärke gut geeignet ist. Sie schafft es jedoch leider nicht bei sehr hohen Umgebungslautstärke ein deutliches Audiosignal für die Erkennung des Hotwords zu liefern.

2.4.1 Kriterien

- Eigenes Hotword
- Gute Erkennung
- Offline
- Kostenlos
- Einfaches Trainieren

2.4.2 Alternativen

Hotword Detection	Eigenes Hotword	Gute Erkennung	Offline	Kostenlos
Picovoice: Porcupine	X	X	X	X
KITT.AI: Snowboy	X	X	X	X

2.4.3 Picovoice Porcupine

" Made in Vancouver, Canada by Picovoice Porcupine is a self-service, highly-accurate, and lightweight wake word (voice control) engine. It enables developers to build always-listening voice-enabled applications/platforms. Porcupine is

- self-service. Developers are empowered to choose any wake word and build its model within seconds.
- using deep neural networks trained in real-world situations.
- compact and computationally-efficient making it suitable for IoT applications. It can run with as low as 20 KB of RAM.
- cross-platform. It is implemented in fixed-point ANSI C. Currently Raspberry Pi, Android, iOS, watchOS, Linux, Mac, Windows, and web browsers are supported. Furthermore, Support for various ARM Cortex-A and ARM Cortex-M processors and a growing number of DSPs is available.
- scalable. It can detect tens of commands concurrently with no added CPU/memory footprint.
- works offline

" [26]

Die Erstellung eines eigenen Hotwords funktioniert mithilfe des Porcupine optimizers (https://github.com/Picovoice/Porcupine/tree/master/tools/optimizer). Wenn das Repository geklont wurde, kann man einfach ein eigenes Hotword erstellen.

```
pv_porcupine_optimizer
    -r resource_directory
    -w keyword
    -p platform
    -o output_directory

-r Absolute path to resource directory that contains data needed
   by optimizer.
-w Keyword.
-p Target platform for running keyword spotting module.
   Valid options are: linux, mac, and windows.
-o Absolute path to output directory where keyword file will be stored.
```

Listing 2.18: Verwendung von Porcupines optimizer

Das erstellte Hotword kann direkt verwendet werden und muss nicht weiter trainiert werden. Im geklonten Repository findet man auch ein demo, mit dem man das zuvor erstellte Hotword gleich testen kann.

2.4.4 KITT.AI Snowboy

" Snowboy is a customizable hotword detection engine for you to create your own hotword like "OK Google" or "Alexa". It is powered by deep neural networks and has the following properties:

- highly customizable: you can freely define your own magic phrase here – let it be "open sesame", "garage door open", or "hello dreamhouse", you name it.
- always listening but protects your privacy: Snowboy does not use Internet and does not stream your voice to the cloud.
- light-weight and embedded: it even runs on a Raspberry Pi and consumes less than 10 percent CPU on the weakest Pi (single-core 700MHz ARMv6).
- Apache licensed!
- scalable. It can detect tens of commands concurrently with no added CPU/memory footprint.

Currently Snowboy supports:

- all versions of Raspberry Pi (with Raspbian based on Debian Jessie 8.0)
- 64bit Mac OS X
- 64bit Ubuntu 14.04
- iOS

- Android
- ARM64 (aarch64, Ubuntu 16.04)

" [7] Für die Erstellung des eigenes Hotwords bei Snowboy benötigt man einen Account auf https://snowboy.kitt.ai. Man kann sich zurzeit mit GitHub, Google oder Facebook einloggen. Nach der erfolgreichen Authentifizierung, kann man ein eigenes Hotword erstellen.

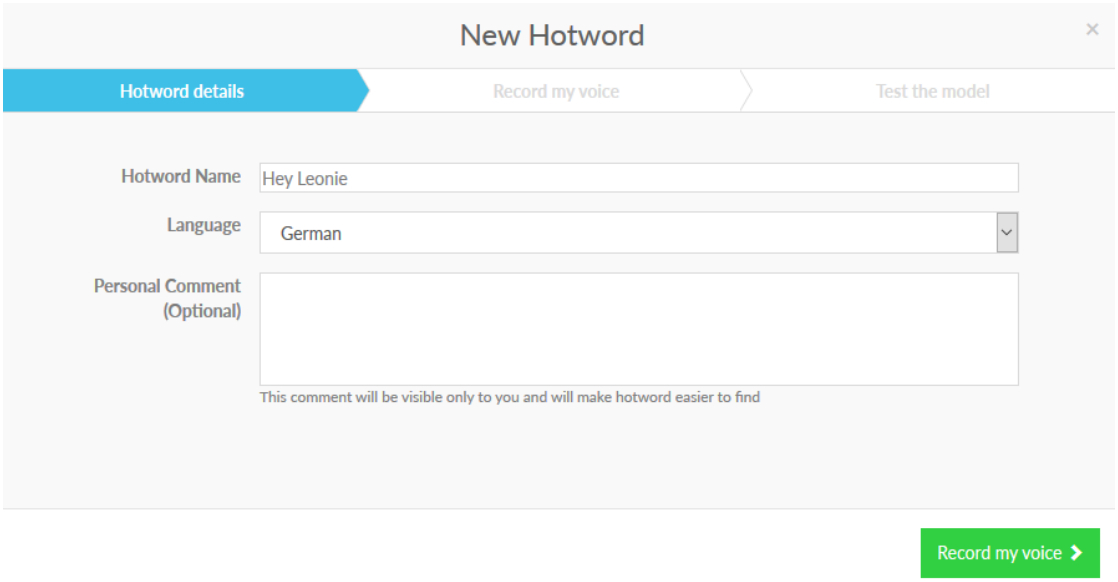


Abbildung 2.30: Eigenes Hotword auf Snowboy.kitt.ai erstellen

Wie in der Abbildung 2.30 zu sehen ist der Hotword Name "Hey Leonie", welches auch gleichzeitig für das Erkennungswort steht, mit dem man das Hotword trainieren muss. Nachdem der Name und die Sprache festgelegt wurde muss man das Hotword auch schon trainieren. Jetzt kann man das Hotword über eine REST-Schnittstelle oder

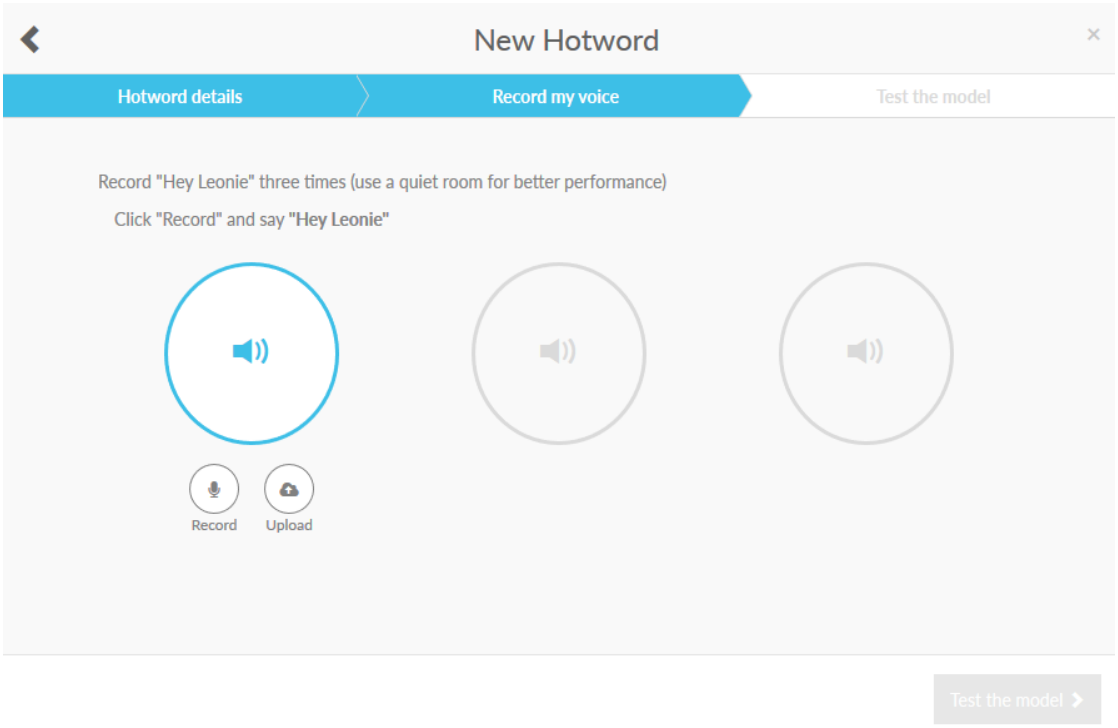


Abbildung 2.31: Eigenes Hotword trainieren

als registrierter User auf Snowboy.kitt.ai trainieren

2.4.5 Entscheidung

Picovoice Porcupine

2.4.6 Begründung

Da beide Libraries die gesuchten Kriterien erfüllten, jedoch Provoice mit Porcupine eine leichtere Handhabung und eine bessere Erkennung mit einer Logitech BRIO erzielten, fiel die Auswahl auf Porcupine. Gerade aber auch durch die leichte Handhabung, man muss zwar nach 90 Tagen den Erstellungsprozess des Hotwords wiederholen, jedoch hat man den Vorteil, dass man das Hotword nicht trainieren muss und die Erkennungsrate immer noch sehr gut ist.

2.5 Gesichtserkennung (Face-Detection)

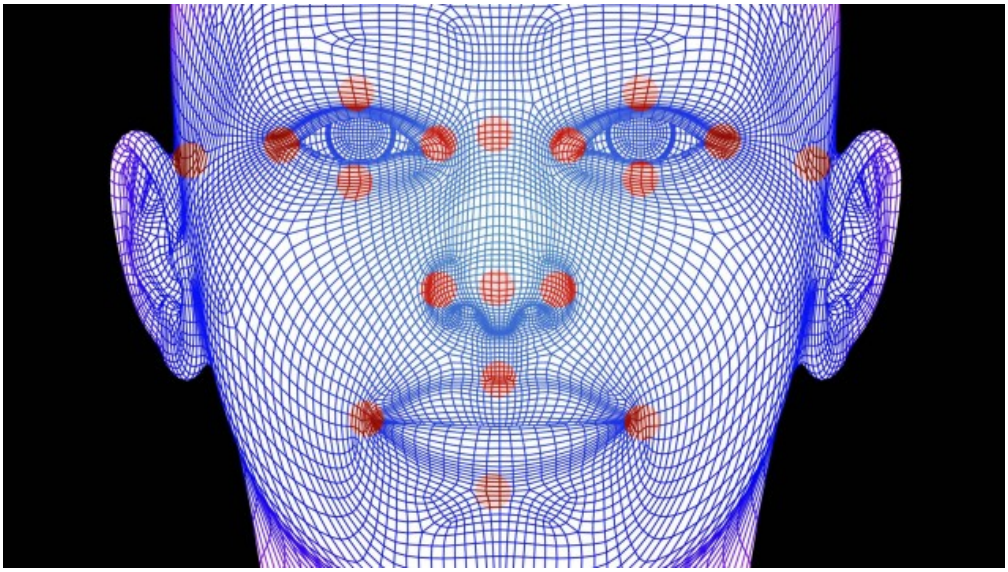


Abbildung 2.32: Erkennungspunkte eines Gesichtes [11]

Die Gesichtserkennung bezeichnet grundsätzlich die Erkennung eines Gesichtes, von vorne, durch ein technisches Hilfsmittel. In 1964 und 1965 wurde von 3 Wissenschaftlern erstmals daran gearbeitet ein System am Computer zu entwickeln, welches Gesichter automatisch erkennt. Für die Gesichtserkennung gibt es verschiedene Ansätze, so gibt es den traditionellen"Weg der auf bestimmte Bereiche im Gesicht überprüft, das wären z.B. Augen, Mund, Nase. Während bei der "3-DimensionalenMethode auf ein Gesicht im 3 Dimensionalen Raum überprüft wird, das verbessert die Genauigkeit der Erkennung ungemein. Dies funktioniert auch noch auf größere Distanzen und wird vom Licht nicht beeinflusst. Ein weitere Methode welche immer mehr verwendet wird, ist die "Haut-TexturAnalyse, diese Methode überprüft auf spezielle visuelle Details des Gesichtes und rechnet diese in einen mathematischen Raum um. Man kann natürlich auch diese Methoden miteinander kombinieren um so ein besseres Ergebnis zu bekommen. Eine interessante weitere Möglichkeit die sehr wenig verwendet wird, ist das erkennen eines Gesichtes über eine Wärmebildkamera. Eine weitere Methode ist die "Viola-JonesMethode diese beruht auf mathematischen Verfahren zur Mustererkennung in digitalen Bildern. [10]

Heutzutage wird die Gesichtserkennung schon in vielen Bereichen eingesetzt, diese wären im Bereich der "Social Media" im Bereich von "SnapChat". Diese App verwendet die Gesichtserkennung um verschiedene "Filter" auf ein Gesicht zu legen.

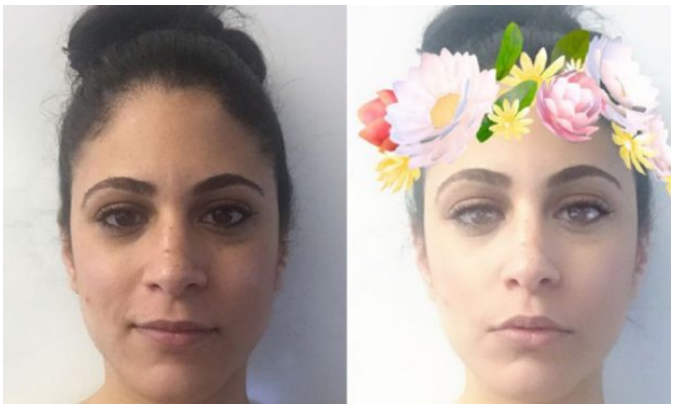


Abbildung 2.33: Gesichtsfilter von SnapChat [33]

Die Gesichtserkennung wird aber nicht nur zur Manipulation der Fotos verwendet sondern auch nur zum erkennen, so wie das bei der Äpple FaceID"der Fall ist. Diese entsperrt nachdem ein Gesicht erkannt wurde das Smartphone. Das soll die Benutzung des Smartphones vereinfachen und so auch Zeit einsparen.

Abbildung 2.34: Apple FaceID zum entsperren vom iPhone X [41]

Auch im Bereich der Sicherheit wird immer öfter Gesichtserkennungs-Software eingesetzt. So verwendet zum Beispiel der Australische Grenzschutz eine Software namens "SmartGate" welche mittels Face Recognition und eines "e-Passport"-Microchips erkennt ob diese Person einreisen darf.

2.5.1 Festlegung der Kriterien

Die Gesichtserkennung muss ein Gesicht bei einem Abstand von 2m mit einer Logitech Brio 4K Kamera problemlos erkennen können. Außerdem muss der Code der Gesichtserkennung so klein wie möglich sein. Das Video der Gesichtserkennung muss in mindestens 25 FPS zu sehen sein und es sollte kein Lag vorhanden sein. Die Gesichtserkennung sollte ein bearbeitetes Video mit Rechtecken oder Kreisen liefern die das Gesicht umfassen. Das Video darf höchstens 2s hinter der Live-Recording liegen. Die Erkennung soll in 90% der Fälle, in denen ein Gesicht sichtbar ist, erkennen das sich ein Gesicht vor dem System befindet.

2.5.2 pico.js

pico.js ist eine JavaScript Library mit nur 200 Zeilen Code. Diese Library bezieht sich auf eine Methode aus 2013 von Markuš et al. Diese Version aus 2013 wurde in C geschrieben. Die Gesichtserkennung funktioniert in Real-Time und rendert die Videoausgabe innerhalb weniger Sekunden. Wenn ein Gesicht vor dem System ist, wird es von pico.js in ca. 95% der Fälle erkannt. Der Ressourcenverbrauch der Face-Detection ist sehr gering, sie ist auf allen Geräten durch einen Web-Browser lagfrei abrufbar. [29]

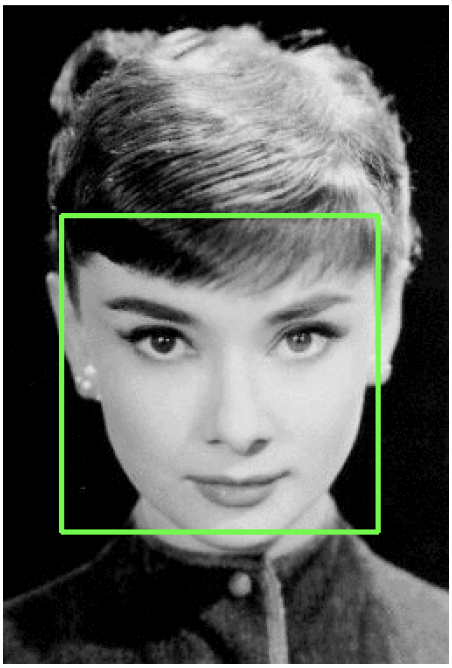


Abbildung 2.35: Gesichtserkennung mit der "Viola-Jones"-Methode mit pico.js [29]

2.5.3 ageitgey/facerecognition

Die Face-recognition von ageitgey(Github-Name) umfasst mehr als nur die benötigte Face-Detection. Diese Library verfügt auch über die Möglichkeit Bilder in einer Datenbank zu speichern und dann jedes einzelne Gesicht erkennt. Hier wird auch ein Live-Time Video gerendert, dieses läuft nicht ganz laggfrei und ist um ca. 3s von den aktuellen Aufnahmen abweichend. Der Code basiert auf der Sprache Python und ist nicht größer als 25 MB. Die Face-Recognition hat laut dem "Labeled Faces in the Wild"-Test der "University of Massachusetts" eine Genauigkeit von 99,38%. Die Installation dieser Library ist sehr aufwendig, auf Windows ist sie offiziell nicht unterstützt. Auf Linux/MacOS ist die Bibliothek unterstützt, aber auch nur nach mehreren Stunden lauffähig. Die Library ist sehr ressourcen verbrauchen und somit nicht auf jedem handelsüblichen PC einsetzbar. Gesichter von Personen die vor dem System stehen werden in ca. 90% der Fälle erkannt. [51]

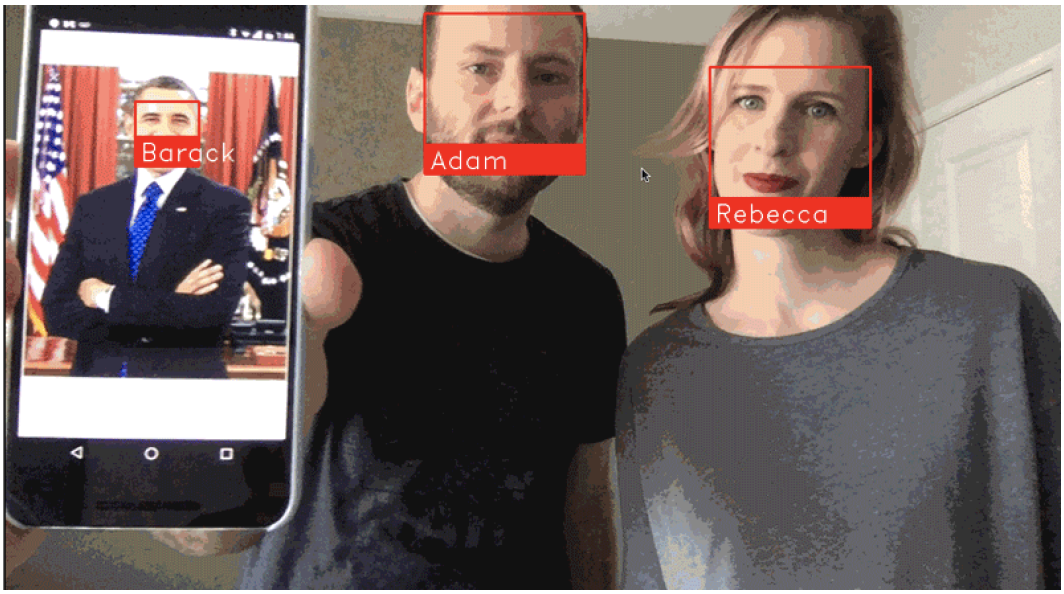


Abbildung 2.36: Gesichtserkennung mit ageitgey/face-recognition [51]

2.5.4 opencv.js

OpenCV existiert seit 1999 und ist der "Standard" in Bereich Face-Detection. Diese Software ist in vielen Sprachen einsetzbar wie z.B. C++, Python, JavaScript und etc. , im speziellen die JavaScript Version die hier verwendet wurde verfügt nicht über alle Funktionen von OpenCV im nativen Einsatz. OpenCV wurde von Intel ins Leben gerufen. Diese JavaScript Library ist auf jedem PC einsetzbar und braucht zur Ausführung nur einen Browser. Der Code ist sehr klein und das Live Time Video wird schnell gerendert. Dieses läuft aber nicht lagfrei und ist nicht wirklich als Video anzusehen da es mit ca. 15 FPS läuft. Gesichter werden zu ca. 85% vor dem System erkannt. Es werden auch andere Objekte als Gesicht erkannt. [27]

Abbildung 2.37: Gesichtserkennung mit opencv.js

2.5.5 Entscheidung

Um die 3 Alternativen zu vergleichen wurde eine Matrix erstellt die alle Vor- und Nachteile der Alternativen auffasst. Hier sieht man welche Alternative die beste ist. pico.js wurde schlussendlich ausgewählt, da es alle Punkte erfüllt im Gegensatz zu ageitgey/face-recognition welches auch alle Punkte erfüllt aber nicht so stark. opencv.js fällt ganz aus der Entscheidung heraus, da es 2 Punkt nicht erfüllen kann.

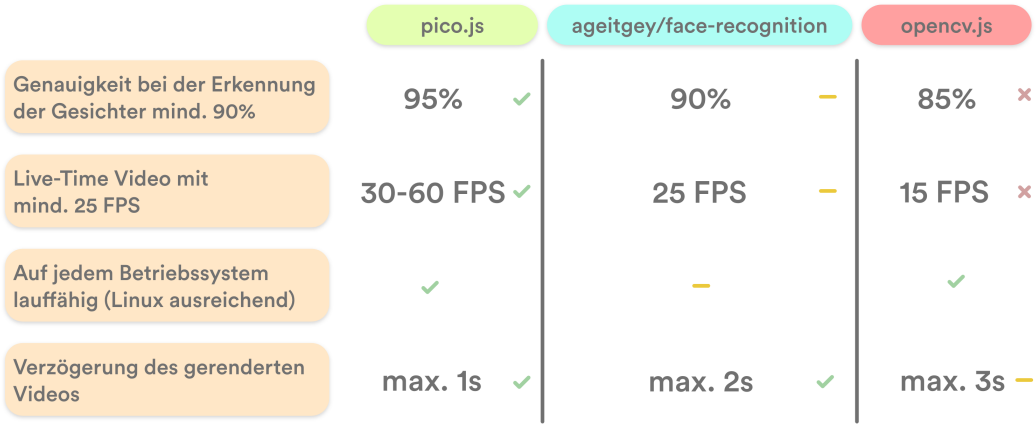


Abbildung 2.38: Vergleichsmatrix zur Entscheidung der Face-Detection

2.6 Backend

"Backend ist ein System, welches das Netzwerk im Hintergrund mit einer Dienstleistung versorgt, also meist ein Server im weiteren Sinn des Begriffs; es besteht kein Zusammenhang mit der Zubereitung von Hefespeisen..." [50]

Leonie ist ein komplexes System, die aus vielen Komponenten besteht. Dazu gehören der Chatbot, das Hologramm, der Show und andere Teile. Bei manchen Anfragen ist es auch notwendig auf äußere Schnittstellen zuzugreifen, zum Beispiel auf LinzAG Linien API oder OpenWeatherMap API. Damit die Leonie als Gesamtsystem funktioniert müssen alle Komponenten miteinander kommunizieren können. Wenn man sie anspricht muss nicht nur ein verbales Antwort kommen, sondern auch die Hologramm muss sich bewegen. Wenn man die Leonie fragt ob sie tanzen kann muss sie das nicht nur positiv beantworten, sondern auch vorzeigen. Und wenn man fragt, wann der nächste Bus von Mozartkreuzung bis Hauptbahnhof fährt müssen aktuelle Daten geholt werden, die beides sprachlich und grafisch auf den nebenstehenden Bildschirm ausgegeben werden. Um die Kommunikation zu erlauben und sicherzustellen, dass alle Teilsysteme auf eine Anfrage richtig reagieren, steht ein Backend-Server im Center des Systems.

Nicht jedes System kann auf Frontend und Backend geteilt werden, jedoch gehört die Leonie eindeutig nicht zu diese Ausnahmen dazu. In IT-Bereich gelten allgemein die zwei Begriffe. Als Frontend"wird normalerweise die Benutzeroberfläche beschrieben. Hier kommuniziert der Benutzer mit System. Im Frontend, der möglichst angenehm und benutzerfreundlich gestaltet werden muss, werden die Daten des Benutzers aufgefasst und andere Daten ausgegeben.

Der Platz wo die aufgeführten Daten hinkommen, wo sie bearbeitet werden und wo die Ergebnisse erstellt werden bezeichnet man als Backend. Backend kann als "Backstageëines System gesehen werden. Der Benutzer bekommt nicht mit was hier passiert, aber ohne den Backend würde Frontend nur eine schöne Dekoration sein.

Die Leonie kommuniziert auf vielen Arten mit den Benutzer und hat somit mehrere miteinander verbundene Frontend-Schichten. Spracheingabe und -ausgabe, Hologramm, Kamera und Infobildschirm müssen perfekt miteinander arbeiten und dazu benötigt das System einen schnellen, stabilen Backend.

2.6.1 Kriterien

- Stabilität
- Gutes Performance
- Open Source
- Plattformunabhängig
- Einfach zu bedienen
- Einfaches Deployment

2.6.2 Alternativen

Kriterien	JavaEE	Microsoft .NET
Stabilität	X	X
Gutes Performance	X	X
Open Source	X	
Plattformunabhängig	X	
Einfach zu bedienen	X	X
Einfaches Deployment	X	X

JavaEE

Abbildung 2.39: JavaEE Logo [54]

"Java Platform, Enterprise Edition (Java EE) ist die Spezifikation einer Softwarearchitektur für die transaktionsbasierte Ausführung von in Java programmierten Anwendungen. In der Spezifikation werden Softwarekomponenten und Dienste definiert, die hauptsächlich in der Programmiersprache Java erstellt werden. Die Spezifikation dient dazu, einen allgemein akzeptierten Rahmen zur Verfügung zu stellen, um auf dessen Basis aus modularen Komponenten verteilte, mehrschichtige Anwendungen entwickeln zu können." [16]

Java EE ist eine von Oracle entwickelte Erweiterung für Java SE (Standard Edition). Derzeit basiert es auf Java SE 8, also nicht auf der neuesten Java Version Java 11, kann aber auch mit höheren Java Versionen arbeiten.

Zusätzlich zu Standard Edition Möglichkeiten spezifiziert Enterprise Edition, wie der Name sagt, Betriebsfunktionen die es erlauben Verteilte Systeme zu erstellen. Zu Java EE APIs gehören unter anderem:

- Servlets - erlauben es HTTP-Requests dynamisch zu behandeln.
- WebSockets - Alternative zu HTTP, erlauben eine stetige, zweiseitige Verbindung zwischen den Client und den Server.
- JAX-RS - Java API for RESTful Web Services, erlaubt es einfach Webservice nach REST-Prinzip zu erstellen.
- APIs für einfache Bearbeitung und Konvertierung von JSON-Informationen.
- Dependency Injection - erlaubt es während Laufzeit Abhängigkeiten an Objekte zu übergeben anstatt neue Objekte zu generieren.
- EJB - Enterprise JavaBean, standardisierte Komponente eines Java EE Servers.
- JPA - Java Persistence API, erlaubt einfache Verbindung zwischen Datenbank und Java Klassen.

In der Regel laufen Java EE Anwendungen auf Anwendungsservern, die Nebenläufigkeit, Transaktionen und Sicherheit übernehmen. Heute gibt es mehrere solche Anwendungsservern auf den Markt. Es werden beides Kostenpflichtige (IBM WebSphere Application Server, WebOTX) und freie (GlassFish, WildFly, Apache TomEE) angeboten.

Folgende Codebeispiel zeigt die Implementierung eines Data Access Objects auf Java EE, mit Hilfe von JPA, der benutzt werden kann, um Objekte in Datenbank zu persistieren: ”

```
@Stateless
public class UserDAO {

    @PersistenceContext
    private EntityManager entityManager;

    public void save(User user) {
        entityManager.persist(user);
    }

    public void update(User user) {
        entityManager.merge(user);
    }

    public List<User> getAll() {
        return entityManager.createNamedQuery("User.getAll", User.class)
                            .getResultList();
    }
}
```

” [?]

Microsoft .NET Framework

Abbildung 2.40: Microsoft .NET Logo [22]

”.NET Framework ist ein Teil von Microsofts Software-Plattform .NET und dient der Entwicklung und Ausführung von Anwendungsprogrammen. Das .NET Framework besteht aus einer Laufzeitumgebung (Common Language Runtime), in der die Programme ausgeführt werden, sowie einer Sammlung von Klassenbibliotheken, Programmierschnittstellen und Dienstprogrammen (Services).” [25]

In gewissen Maßen ist die .NET Plattform ein Gegenbild von Java EE. Java EE ist open Source und modular, wird also von mehreren von Oracle unabhängigen Programmierern entwickelt. Microsoft hält den Quellcode versteckt und entwickelt ein monolithes Framework. Java EE basiert fast vollständig auf Java und dank der JVM können die Programme fast ohne Änderungen auf jeder Plattform ausgeführt werden. .NET unterstützt mehrere Programmiersprachen wie C Sharp und C++, die auch in ein Programm gleichzeitig benutzt werden können, ist dafür nur auf Windows ausgerichtet. Ein neues Framework von Microsoft, .NET Core, versucht diese Probleme zu beheben, ist aber noch zu jung und verfügt nicht über sehr viele Möglichkeiten.

.NET Framework beinhaltet ein großes Klassenbibliothek namens Framework Class Library. Diese implementiert unter anderem folgende grundlegende Funktionalitäten:

- Windows Forms - erlaubt es einfach grafische Benutzeroberflächen für Desktop-Anwendungen zu erstellen.
- ASP.NET - erlaubt dynamische Webseiten zu erstellen.
- WPF - erlaubt schwierigere und mächtigere GUIs zu erstellen.
- Funktionen für einfache Verbindung zwischen Klassen und Datenbanktabellen.
- Funktionen für Kommunikation zwischen Server und Client.

Die .NET Framework Anwendungen werden in einem Software-Umgebung namens Common Language Runtime (CLR) ausgeführt, also auf eine virtuelle Maschine, die für Sicherheit, RAM-Verwaltung und Ausnahmebehandlung sorgt.

Folgende Code-Beispiel zeigt eine generische Repository auf CSharp, die benutzt werden kann, um Objekte in Datenbank zu persistieren:

```
1  public class Repository<T, Ctx> : IRepository<T>
2          where T : class
3          where Ctx : DbContext, new()
4      {
5          public void Create(T entity)
6          {
7              using (Ctx db = new Ctx())
8              {
9                  db.Set<T>().Add(entity);
10                 db.SaveChanges();
11             }
12         }
13
14         public void Update(T entity)
15         {
16             using (Ctx db = new Ctx())
17             {
18                 db.Set<T>().Attach(entity);
19                 db.Entry<T>(entity).State = EntityState.Modified;
20                 db.SaveChanges();
21             }
22         }
23
24         public void Delete(T entity)
25         {
26             using (Ctx db = new Ctx())
27             {
28                 db.Set<T>().Attach(entity);
29                 db.Set<T>().Remove(entity);
30                 db.SaveChanges();
31             }
32         }
33
34         public IEnumerable<T> Get(Expression<Func<T, bool>> filter = null
               ,
35             Func<IQueryable<T>, IOrderedQueryable<T>> order = null,
36             string include = "")
37         {
38             using (Ctx db = new Ctx())
39             {
40                 var result = db.Set<T>() as IQueryable<T>;
41                 if (filter != null)
42                     result = result.Where(filter);
43
44                 foreach (string prop in include.Split(
45                     new char[] { ',' }, StringSplitOptions.
                           RemoveEmptyEntries))
46                 {
47                     result = result.Include(prop);
48                 }
49
```

```
50                  if (order != null)
51                      result = order(result);
52
53                  return result.ToList();
54              }
55          }
```

2.6.3 Entscheidung

Java EE

2.6.4 Begründung

Beide Varianten liefern eine effiziente und stabile Lösung für Leonie-Backend, beide Servern können relativ leicht bedient und deployed werden. Java EE Server kann jedoch auf beliebigen Betriebssysteme gestartet werden, was bei der Wahl der entscheidende Punkt war. Vor allem in der Entwicklungsphase wurde die Systemarchitektur mehrmals geändert. Das System wurde mehrmals übertragen und auf verschiedenen Rechnern gleichzeitig gestartet. Deswegen ist für diese Problemstellung die flexible Java EE Lösung besser geeignet, als Microsoft .NET Framework. Ein weitere wichtiger Punkt ist die öffentlichkeit des Quellcodes. Java EE erlaubt es viel einfacher bereits von anderen Benutzern implementierte Bibliotheken zu benutzen, was den Entwicklungsprozess einfacher und flexibler macht. Außerdem kann man aus Codebeispiele auslesen, dass Java EE gewisse Probleme einfacher und schöner lösen lässt.

Kapitel 3

Ausgewählte Aspekte und Probleme

3.1 Gesprächsinitierung

Bei der Gesprächsinitierung ist der erste Kontakt mit dem System sehr wichtig. Dieser sollte sehr einfach und für jeden möglich sein. Daher kommt hier eine Gesichtserkennung zum Einsatz, diese ermöglicht ein möglichst einfaches beginnen der Konversation. Diese erkennt ein Gesicht wenn sich ein Benutzer in unmittelbarer nähe befindet. Es zeigt dann am Nebenbildschirm, dass ein Gesicht erkannt wurde.

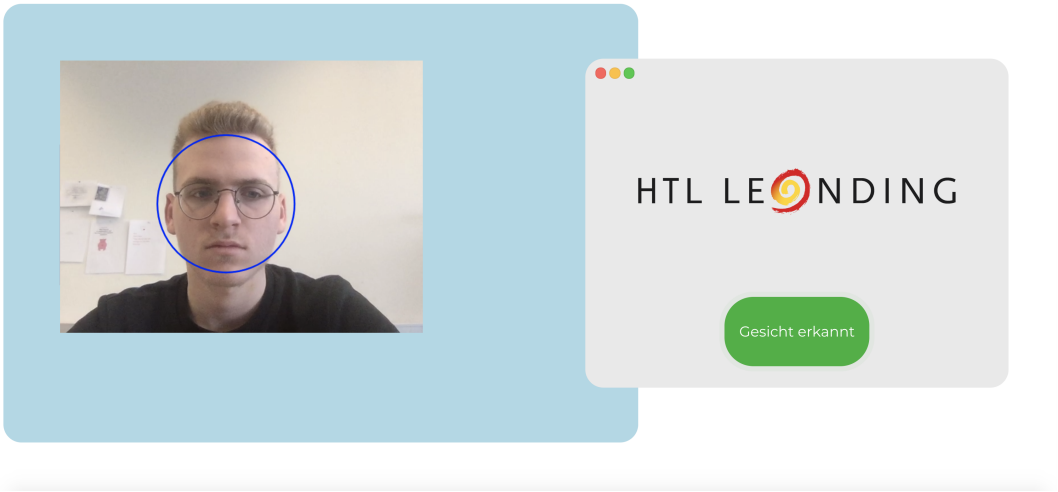


Abbildung 3.1: Nebenbilschirm von "Leonie" nach der Gesichtserkennung

Nach der Erkennung des Gesichtes wird der Benutzer der davor steht, durch Leonie"begrüßt. Das Hologramm reagiert dann mit: "Hallo, mein Name ist Leonie! Du kannst mich nach den Themen links fragen!". Nach dieser Phrase von "Leonie" hat der Benutzer 30s Zeit mit dem Gespräch fortzufahren. Wird der Benutzer in dieser Zeit nicht aktiv, so erscheint auf dem Bildschirm ein Fenster mit dem Text: "Starte mich mit: Hey Leonie!". Dieses Fenster bleibt für ebenfalls für 30s stehen danach begibt sich "Leonie" wieder in den Startzustand.

3.2 Konstruktion und Bau von Prototyp

Beim Bau und der Konstruktion des Prototyp-Systems war es sehr wichtig, so wenig externe Dienstleistungen wie möglich zu benutzen. Für die Konstruktion eines Prototypen braucht man als aller erst einen Plan und eine grundsätzliche Materialvorstellung. Am Anfang waren nur die Maße des Monitors bekannt, somit mussten noch die Höhe vom Hologramm und die Größe der Pyramide bestimmt werden. Das Plexiglas bzw. Glas brauchte eine bestimmte Brechung und eine bestimmte Größe.

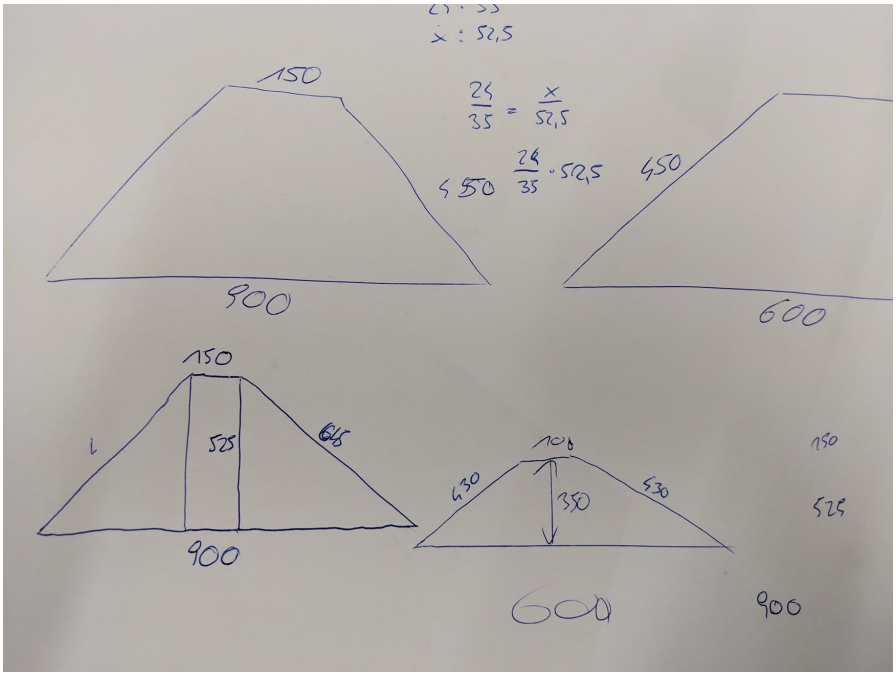


Abbildung 3.2: Grobe 1.Skizze des 1. Pyramiden Prototyps

Abbildung 3.3: 1. Prototyp der Pyramide für das Hologramm

Nach dem Testen des 1. Prototyp wurde klar, dass die Pyramide etwas zu klein und die Spiegelung etwas zu schlecht war. Nach vielen weiteren probieren wurde einen Folie für Plexiglas gefunden, und ein Glas welches den Standards die gewollt waren entsprach. Es wurde eine Pyramide aus Plexiglas mit spezieller Folie und eine Glaspyramide angefertigt. Auch das Gestell wurde in 3 Versionen gebaut, 2 wurden im Haus entwickelt und gebaut, eines wurde extern geplant und auch gebaut. Beim Bau des Grundgestell waren ein paar Punkte sehr wichtig. Zum einen sollte es besonders leicht und zum anderen auch besonders günstig sein. Der 1. Prototyp bestand rundherum aus 3mm dicken Winkeleisen aus Profilstahl welche in Schwarz lackiert wurden. Diese Winkeleisen wurden zusammengeschweißt, sodass man am Schluss ein Gestell mit einer Holzplatte in der Mitte für die Pyramide und einen Rahmen für den Monitor hatte.

Abbildung 3.4: 1. Prototyp des Gestells für das System

Nach Berücksichtigung aller Fehler bzw. Verbesserung die gefunden wurden, ging es an den Bau des 2. Prototypen des Systems. Das Material blieb gleich, die Maße änderten sich und die Funktionalität wurde verbessert. So hat der 2. Prototyp Räder um die Transportabilität zu erhöhen. Das System hat außerdem einen Stauraum im inneren um alle Komponenten unterzubringen, wie z.B. PC, Lautsprecher und Stromverteiler. Das Gestell ist in der letzten Version auch 2-teilig, das heißt der obere Teil ist alleine verwendbar und transportierbar. Dieser Schritt brachte viele Vorteile mit sich. Nun ist es möglich das System ohne den unteren Teil völlig einsatzbereit. Hier war es wichtig, dass das System ohne den unteren Teil in einem handelsüblichen Mittelklasse Fahrzeug transportierbar ist.

Abbildung 3.5: 2. Prototyp des Gestells / Grundgerüst

Es war wichtig das Gestell in 2 Teile aufteilen zu können, und zusätzlich sehr leicht zu machen. Das Gestell verfügt über 2 Holzplatten in den jeweiligen Einzelteilen, so kann

das Gestell auch ohne den unteren Teil verwendet werden. Im oberen Teil des Gestells ist Platz für die Pyramide die vorzugsweise nur 3 Seiten haben sollte. Im unteren Teil des Gestells ist Platz für jegliche Hardware die gebraucht wird, um den Avatar "Leonie" als Hologramm darzustellen. Das Gestell wurde an den Außenseiten mit 3mm starken Dibond verkleidet, dieses Material ist sehr leicht und durch die Beschichtung an der Oberfläche bietet es eine sehr wertvolle Anmutung. Diese Platten aus Dibond wurden in der Farbe Weiß gewählt, da dies ein sehr guter Kontrast zum restlichen Gestell, welches Schwarz ist stellt.

Abbildung 3.6: 2. Prototyp des Gestells / unterer Teil des Gestells

Beim 2. Prototyp war es auch wichtig das der Monitor eine Abdeckung hat. Die Abdeckung besteht aus schwarz folierten Dibond, welches 3mm dick ist. Diese Abdeckung ist auch dafür da, die Lautsprecher zu halten. Da der Monitor sehr wenig Hitze produziert ist die Abdeckung fast komplett geschlossen, nur die Lautsprecher Einlässe sind offen.

Abbildung 3.7: 2. Prototyp des Gestells / oberer Teil des Gestells

Abbildung 3.8: 2. Prototyp des Gestells / komplettes Gestell

Kapitel 4

Toolstack

4.1 Sprachen

Mar 2019	Mar 2018	Change	Programming Language	Ratings	Change
1	1		Java	14.880%	-0.06%
2	2		C	13.305%	+0.55%
3	4	↑	Python	8.262%	+2.39%
4	3	↓	C++	8.126%	+1.67%
5	6	↑	Visual Basic .NET	6.429%	+2.34%
6	5	↓	C#	3.267%	-1.80%
7	8	↑	JavaScript	2.426%	-1.49%
8	7	↓	PHP	2.420%	-1.59%
9	10	↑	SQL	1.926%	-0.76%
10	14	⇈	Objective-C	1.681%	-0.09%
11	18	⇈	MATLAB	1.469%	+0.06%
12	16	⇈	Assembly language	1.413%	-0.29%

Abbildung 4.1: Die 12 meistbenutzen Programmiersprachen laut dem TIOBE-Index [40]

Laut dem TIOBE-Index ist JavaScript und Python am aufsteigenden Ast von März 2018 bis 2019. Java befindet sich unangefochten auf Platz 1.

4.1.1 Java

Abbildung 4.2: Logo von Java [16]

”Java ist eine objektorientierte Programmiersprache und eine eingetragene Marke des Unternehmens Sun Microsystems, welches 2010 von Oracle aufgekauft wurde. Die Programmiersprache ist ein Bestandteil der Java-Technologie – diese besteht grundsätzlich aus dem Java-Entwicklungswerkzeug (JDK) zum Erstellen von Java-Programmen und der Java-Laufzeitumgebung (JRE) zu deren Ausführung. Die Laufzeitumgebung selbst umfasst die virtuelle Maschine (JVM) und die mitgelieferten Bibliotheken. Java als Programmiersprache sollte nicht mit der Java-Technologie gleichgesetzt werden; Java-Laufzeitumgebungen führen Bytecode aus, der sowohl aus der Programmiersprache Java als auch aus anderen Programmiersprachen wie Nice und Groovy kompiliert werden kann. Im Prinzip könnte jede Programmiersprache als Grundlage für Java-Bytecode genutzt werden, meistens existieren aber keine entsprechenden Bytecode-Compiler. Die Programmiersprache Java dient innerhalb der Java-Technologie vor allem zum Formulieren von Programmen. Diese liegen zunächst als reiner, menschen verständlicher Text vor, dem sogenannten Quellcode. Dieser Quellcode ist nicht direkt ausführbar; erst der Java-Compiler, der Teil des Entwicklungswerkzeugs ist, übersetzt ihn in den maschinenverständlichen Java-Bytecode. Die Maschine, die diesen Bytecode ausführt, ist jedoch typischerweise virtuell – das heißt, der Code wird meist nicht direkt durch Hardware (etwa einen Mikroprozessor) ausgeführt, sondern durch entsprechende Software auf der Zielplattform. Zweck dieser Virtualisierung ist Plattformunabhängigkeit: Das Programm soll ohne weitere Änderung auf jeder Rechnerarchitektur laufen können, wenn dort eine passende Laufzeitumgebung installiert ist. Oracle selbst bietet Laufzeitumgebungen für die Betriebssysteme Linux, macOS, Solaris und Windows an. Andere Hersteller lassen eigene Java-Laufzeitumgebungen für ihre Plattform zertifizieren. Auch in Autos, HiFi-Anlagen und anderen elektronischen Geräten wird Java verwendet. Um die Ausführungsgeschwindigkeit zu erhöhen, werden Konzepte wie die Just-in-time-Kompilierung und die Hotspot-Optimierung verwendet. In Bezug auf den eigentlichen

Ausführung Vorgang kann die JVM den Bytecode also interpretieren, ihn bei Bedarf jedoch auch kompilieren und optimieren. Java ist eine der populärsten Programmiersprachen. In dem seit 2001 veröffentlichten TIOBE-Index lag Java, konkurrierend mit C, stets auf Platz 1 oder 2 des Rankings. Nach dem RedMonk-Programmiersprachen Index 2018 liegt Java knapp auf Platz zwei nach JavaScript, nach dem „Programming Language Popularity"-Index liegt Java ebenfalls auf Platz zwei, diesmal aber nach C." [16]

Java bildet in dieser Arbeit den Grundbestandteil des Backends. Mithilfe der RESTful-API können Datenbanken abgefragt werden und dann über Leonie sprachlich ausgegeben werden.

4.1.2 Javascript

Abbildung 4.3: unofizelles Logo von Javascript [17]

"JavaScript (kurz JS) ist eine Skriptsprache, die ursprünglich 1995 von Netscape für dynamisches HTML in Webbrowsern entwickelt wurde, um Benutzerinteraktionen auszuwerten, Inhalte zu verändern, nachzuladen oder zu generieren und so die Möglichkeiten von HTML und CSS zu erweitern. Heute findet JavaScript auch außerhalb von Browsern Anwendung, so etwa auf Servern und in Mikrocontrollern. Der heutige Name der ursprünglich LiveScript genannten Sprache entstand 1996 aus einer Kooperation von Netscape mit Sun Microsystems. Deren Java-Applets, erstellt mit der gleichfalls 1995 veröffentlichten Programmiersprache Java, wurden mithilfe von LiveScript in den Netscape Navigator integriert. Um die Popularität von Java zu nutzen, wurde LiveScript in JavaScript umbenannt, obwohl sich die beiden Sprachen stark voneinander unterscheiden. Der als ECMAScript (ECMA 262) standardisierte Sprachkern von JavaScript beschreibt eine dynamisch typisierte, objektorientierte, aber klassenlose Skriptsprache. Sie wird allen objektorientierten Programmierparadigmen unter anderem auf der Basis von Prototypen gerecht, deren Deklaration ab ECMAScript 6 mit einer Syntax ermöglicht wird, wie sie ähnlich auch bei klassen basierten Programmiersprachen üblich ist. In JavaScript lässt sich je nach Bedarf objektorientiert, prozedural oder funktional programmieren." [17]

Mit Javascript wurde in dieser Arbeit die Gesichtserkennung implementiert. Das verwenden dieser Sprache ermöglicht es, das der Algorithmus der Gesichterkennung auf nahezu jedem Mobilegerät, Notebook oder eines Gerätes mit Internetzugang aufzurufen ist.

4.1.3 Python

Abbildung 4.4: Logo von Python [31]

"Python ist eine universelle, üblicherweise interpretierte höhere Programmiersprache. Sie hat den Anspruch, einen gut lesbaren, knappen Programmierstil zu fördern. So werden beispielsweise Blöcke nicht durch geschweifte Klammern, sondern durch Einrückungen strukturiert. Wegen seiner klaren und übersichtlichen Syntax gilt Python als einfach zu erlernen. Python unterstützt mehrere Progrgit sammierparadigmen, z. B. die objektorientierte, die aspektorientierte und die funktionale Programmierung. Ferner bietet es eine dynamische Typisierung. Wie viele dynamische Sprachen wird Python oft als Skriptsprache genutzt. Die Sprache weist ein offenes, gemeinschaftsbasiertes Entwicklungsmodell auf, das durch die gemeinnützige Python Software Foundation gestützt wird, die de facto die Definition der Sprache in der Referenzumsetzung CPython pflegt." [31]

Python ermöglicht in dieser Arbeit die Hotword-Detection, durch ein leichtgewichtiges Framework wird dieser Code auf dem System gestartet und erkennt so wenn "Hey Leonie" oder "Ok Leonie" gesagt wurde.

4.2 Technologien

4.2.1 REST Service

Representational State Transfer (REST) Service ermöglicht es ganz einfach eine Schnittstelle aufzubauen, über die Daten zwischen Client und Server ausgetauscht werden können.

Abbildung 4.5: REST Logo [32]

Eine REST-Schnittstelle wird für kommunikation zwischen Java EE Server der Leonie und alle anderen Teilsysteme verwendet. Die Kommunikation über REST basiert auf zustandslosen Requests mit CRUD-Operatoren:

- Create (POST)
- Read (GET)
- Update (PUT)
- Delete (DELETE)

Um einen REST-Service auf Java EE Server aufzubauen benötigt man eine von Application abgeleitete Klasse namens RestConfig:

```
@ApplicationPath("api")
public class RestConfig extends Application { }
```

Abbildung 4.6: RestConfig

Einzelne Methoden, die über REST URLs aufrufbar sein sollen, werden in Klassen die als Endpoints bekannt sind implementiert. Diese sind durch entsprechende Annotationen gekennzeichnet:

```java
@Path("diagram")
public class DiagramsEndpoint {
    @POST
    @Produces(MediaType.APPLICATION_JSON)
    @Consumes(MediaType.APPLICATION_JSON)
    public Response changeDiagram(JsonObject digram) throws JsonParsingException {
        Event event = new Gson().fromJson(digram.toString(), Event.class);
        String eventJson = new Gson().toJson(event);
        ClientSocket.broadcast(eventJson, ClientSocket.LEONIE_SHOW);
        return Response.ok(eventJson).build();
    }
}
```

Abbildung 4.7: Endpoint

Im Fall der Leonie dienen die Endpoints dazu Events von verschiedenste Schnittstellen zu empfangen, bearbeiten und an andere Schnittstellen weiterzuleiten:

```java
public class Event {
    private Long deviceid;
    private String sourcetype;
    private String source;
    private String eventname;
    private boolean eventstate;
    private Long timestamp;
    private EventData eventdata;

    public Event() {}

    public Event(Long deviceid, String sourcetype, String source, String eventName, boolean eventState, Long timestamp, EventData eventdata) {
        this.deviceid = deviceid;
        this.sourcetype = sourcetype;
        this.source = source;
        this.eventname = eventName;
        this.eventstate = eventstate;
        this.timestamp = timestamp;
        this.eventdata = eventdata;
    }
```

Abbildung 4.8: Event

4.2.2 Websockets

Websockets erlauben es eine stetige Verbindung zwischen Clients und den Server zu erstellen. Über diese Verbindung können in beide Richtungen Daten ausgetauscht werden. Um einen Websocket in Java EE zu implementieren benötigt man eine Socket-Klasse

Abbildung 4.9: WebSocket Logo [44]

mit @ServerEndpoint Annotation, wo der Socket-Pfad steht und @OnOpen und @OnMessage Methoden, die entsprechend aufgerufen werden, wenn eine Verbindung erstellt wird oder eine Nachricht geschickt wird:

```java
@ServerEndpoint(value = "/client/{clientName}")
public class ClientSocket {
    /* Add ClientId or API KEY for futere referencing */
    /* Client names */
    public static String LEONIE_HOLO = "holo";
    public static String LEONIE_SHOW = "show";
    public static String LEONIE_HARDWARE = "hardware";

    private Session session;
    private static HashMap<Session, ClientSession> clientSessions = new HashMap<~>();

    @OnOpen
    public void onOpen(Session session, @PathParam("clientName") String clientName) {
        this.session = session;
        clientSessions.put(session, new ClientSession(session, clientName));
    }

    @OnClose
    public void onClose(Session session) { clientSessions.remove(session); }

    /**
     * Broadcast sends messages to all specified connected sockets
     * @param msg Message to send
     * @param clientName Specified clientName were message should be sent
     *        possibilities: LEONIE_ANIMATION | LEONIE_SHOW
     */
    public static void broadcast(String msg, String clientName) {
        clientSessions.forEach((session, clientSession) -> {
            if (clientSession.getClientName().equals(clientName)) {
                try {
                    session.getBasicRemote().sendText(msg);
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        });
    }
```

Abbildung 4.10: ClientSocket von Leonie Backend Server

Danach kann eine Verbindung Session aufgebaut werden. Die Session ist ein in Java EE vorhandene Klasse, diese kann jedoch für eigene Zwecke erweitert werden:

```java
public class ClientSession {
    private Session session;
    private String clientName;

    public ClientSession() {}

    public ClientSession(Session session, String clientName) {
        this.session = session;
        this.clientName = clientName;
    }

    public Session getSession() { return session; }

    public void setSession(Session session) { this.session = session; }

    public String getClientName() { return clientName; }

    public void setClientName(String clientName) { this.clientName = clientName; }
}
```

Abbildung 4.11: ClientSession

4.2.3 WildFly

Der WildFly Application Server ist ein von Red Hat nach dem Java-EE-Standard entwickeltes Application Server. Es wird unter der GNU Lesser Public License verbreitet. Das heißt, dass der Server inklusive Quellcode gratis und öffentlich verfügbar ist. WildFly ist einfach einzurichten und zu bedienen, gratis und verfügt über alle nötige Funktionalitäten und wird somit als Application Server für Java EE Backend von Leonie verwendet.

Abbildung 4.12: WildFly Logo [18]

4.2.4 Containervirtualisierung

"Virtualisierung bezeichnet in der Informatik die Nachbildung eines Hard- oder Software-Objekts durch ein ähnliches Objekt vom selben Typ mit Hilfe eines Abstraktions-Layers. Dadurch lassen sich virtuelle (d. h. nicht-physische) Geräte oder Dienste wie emulierte Hardware, Betriebssysteme, Datenspeicher oder Netzwerkressourcen erzeugen." [42] Es gibt grundsätzlich zwei Ansätze bei der Software Virtualisierung: Containervirtualisierung und Virtualisierung mittels eines Hypervisors (Virtual Machine).

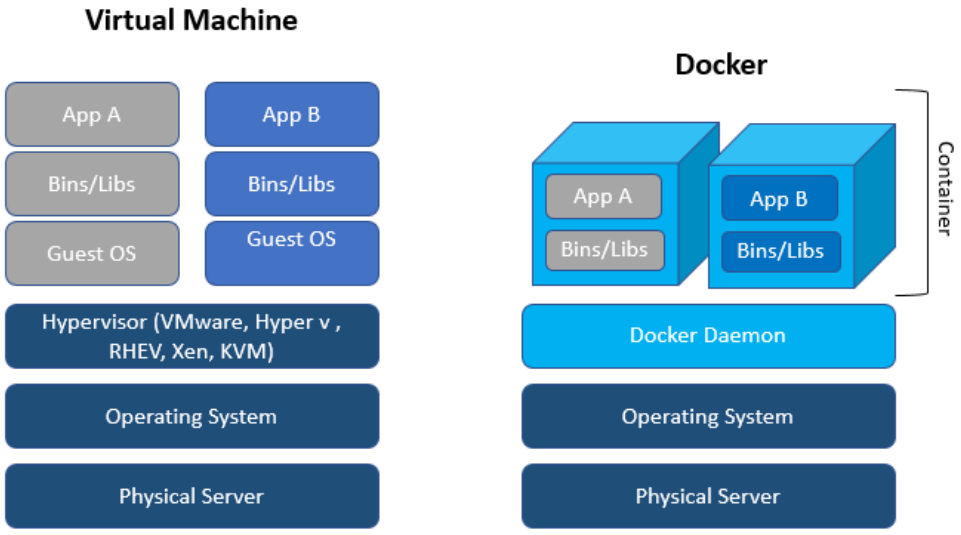


Abbildung 4.13: Vergleich der Software Virtualisierungsmethoden [8]

Wie man auf der Abbildung sehen kann, wird bei der Benutzung einer Virtuelle Maschine eine zusätzliche Abstraktionsebene angelegt. Das passiert weil die VM eine eigene Betriebssystem braucht. Diese wird über die auf das tatsächliche Hardware bereits vorhandene Betriebssystem installiert. Somit laufen bei der Verwendung einer Virtuelle Maschine zwei oder mehr Betriebssysteme gleichzeitig auf einem Rechner. Das verbraucht sehr viele Ressourcen.
Im gegensatz dazu verwenden die Container bei der Containervirtualisierung (in dem Fall mit Docker) die auf den Rechner installierte Betriebssystem. Das ist nicht nur ressourcenschonend, sondern erlaubt auch im Vergleich sehr schnell die virtualisierte Anwendungen zu starten und wieder herunterzufahren. Die Prozesse werden von den System als native gesehen, wodurch CPU und RAM auslastung relativ klein gehalten werden kann. Ein weiteres Vorteil des Containervirtualisierung ist es, dass man kein umfangreiches und nicht intuitives Software braucht. Die Containers werden von einem Daemon kontrolliert, der über Commandline gesteuert werden kann.
In dieser Arbeit wurden Systemteile mittels Docker in Containers virtualisiert,um das Speichern und die Verwaltung zu erleichtern.

4.3 Werkzeuge

4.3.1 GitHub

"GitHub ist ein Onlinedienst, der Software-Entwicklungsprojekte auf seinen Servern bereitstellt (Filehosting). Namensgebend war das Versionsverwaltungssystem Git. Die GitHub, Inc. hat ihren Sitz in San Francisco in den USA." [13]

Abbildung 4.14: Logo von GitHub [13]

In dieser Arbeit wurde Github für die Versionsverwaltung aller Anwendungen und Programmen verwendet. So ist es möglich zu jeder Zeit eine beliebige Versionen der Software zu erhalten.

4.3.2 YouTrack

YouTrack ist ein proprietärer, kommerzieller browserbasierter Bug-Tracker, ein Issue-Tracking-System und eine von JetBrains entwickelte Projektmanagement-Software.

Abbildung 4.15: Logo von YouTrack [47]

YouTrack wurde in dieser Arbeit dazu verwendet, stets einen Überblick über alle noch offenen Arbeiten zu haben. So konnte jederzeit gesehen werden wer welche Aufgaben zu absolvieren hatte.

4.3.3 Maven

Maven ist ein Build-Management-Tool für Java. Maven erlaubt es standardisierte Java-Anwendungen zu erstellen

Abbildung 4.16: Maven Logo [21]

Dabei sind die Einstellungen wie Dependencies auf bereits existierende Libraries in eine XML Datei namens pom.xmlëingeschrieben.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.leonie</groupId>
    <artifactId>backend</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>war</packaging>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <failOnMissingWebXml>false</failOnMissingWebXml>
    </properties>

    <dependencies>
        <dependency>
            <groupId>javax</groupId>
            <artifactId>javaee-web-api</artifactId>
            <version>8.0</version>
        </dependency>
        <dependency>
            <groupId>com.google.code.gson</groupId>
            <artifactId>gson</artifactId>
            <version>2.8.5</version>
        </dependency>
    </dependencies>

    <build>
        <finalName>leonie-backend</finalName>
    </build>

</project>
```

Abbildung 4.17: pom.xml

4.3.4 IntelliJ IDEA

"IntelliJ IDEA ist eine integrierte Entwicklungsumgebung (IDE) des Softwareunternehmens JetBrains für die Programmiersprachen Java, Kotlin, Groovy und Scala." [14]

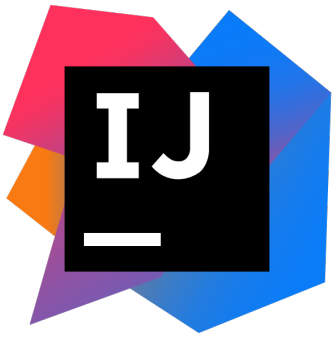


Abbildung 4.18: IntelliJ IDEA Logo [15]

Vor allem bei der Backend-Entwicklung wurde das IntelliJ als Hauptumgebung verwendet. Von JetBrains entwickelte IDE ist schnell, gemütlich und funktionsreich, daher unersetzlich für Entwicklung von Java-Anwendungen.

4.3.5 Visual Studio Code

"VS Code ist ein freier Quelltext-Editor von Microsoft. Visual Studio Code ist plattformübergreifend für die Betriebssysteme Windows, macOS und Linux verfügbar. Visual Studio Code basiert auf dem Framework Electron und ermöglicht u. a. Syntaxhervorhebung, Code-Faltung, Debugging, Autovervollständigung und Versionsverwaltung." [43]

Abbildung 4.19: Visual Studio Code Logo [43]

Von Microsoft entwickelter freier Editor ist vor allem bei "kleinerenProgrammiersprachen wie JavaScript und TypeScript gebräuchlich. Er wurde vor allem bei Frontend-Entwicklung verwendet.

4.3.6 Docker

Docker ist eine von Docker, Inc entwickeltes und betriebenes Software, die zur Virtualisierung auf Betriebssystemebene dient.

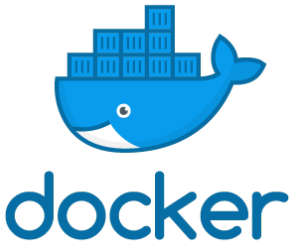


Abbildung 4.20: Docker Logo [9]

Beim Virtualisierung mit Docker wird Software in die sogenannten Containers verpackt. Diese enthalten den Code so wie alle zur Ausführung benötigten Werkzeuge und Bibliotheken. Die Docker-Containers können als Dateien - Images - leicht transportiert und auf einem beliebigen Rechner installiert werden.
Der Docker besteht aus folgenden Komponenten:

- Docker Daemon, ein im Hintergrund laufender Prozess, der für Verwaltung der Docker-Containers zuständig ist.
- Docker Container ist eine verschachtelte Laufzeitumgebung, wo eine Anwendung ausgeführt wird.
- Docker Images sind Abbildungen der Containers. Diese können als Files gespeichert, geschickt und geteilt werden. Ein Image besteht aus unveränderliche Layern.
- Layer enthält genau eine Datei oder einen Befehl aus Dockerfile, der ein Teil des Images ist.

Neben den Docker selbst betreibt Docker, Inc auch ein Online-Dienst Docker Hub, eine Registry für Docker-Images. Jeder Benutzer kann seine eigene Images auf Docker Hub hochladen und diese somit öffentlich zur Verfügung stellen. Es gibt auch einen privaten Teil, der zum Beispiel zur gruppeninterne verteilung dienen kann. Außerdem bietet Docker Hub Versionsverwaltungssystem und ein API an, die es erlaubt Images aus Git-Repositories automatisch zu erstellen.
Ein weiteres Tool, Docker Compose, erlaubt es mehrere Containers mittels einen YAML-File zu verbinden und somit gemeinsam zu bedienen.
In dieser Arbeit wurde Docker für das erleichtertes und übersichtliches Verwaltung von verschiedene Systemteile, hauptsächlich den Backend-Server, verwendet.

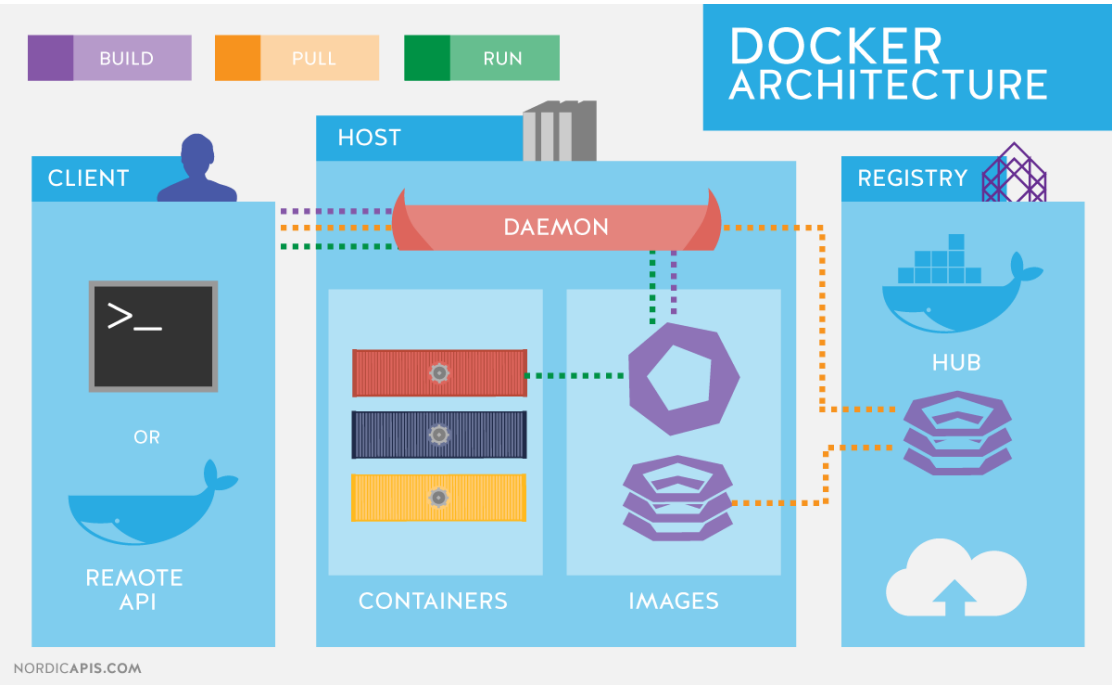


Abbildung 4.21: Docker Architektur [48]

Literaturverzeichnis

[1] Amazon alexa (abgerufen am: 2019-04-05). URL: https://developer.amazon.com/de/alexa.

[2] Amazon echo - bluetooth lautsprecher mit alexa (abgerufen am: 2019-04-05). URL: https://www.amazon.de/Amazon-Echo-Intelligenter-Lautsprecher-Alexa/dp/B06ZXQV6P8.

[3] Amazon echo (abgerufen am: 2019-02-23). Page Version ID: 185650676. URL: https://de.wikipedia.org/w/index.php?title=Amazon_Echo&oldid=185650676.

[4] Chatbot (abgerufen am: 2019-02-23). Page Version ID: 184275634. URL: https://de.wikipedia.org/w/index.php?title=Chatbot&oldid=184275634.

[5] Classifier comparison — scikit-learn 0.20.3 documentation (abgerufen am: 2019-04-05). URL: https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html.

[6] Cloud text-to-speech – sprachsynthese | cloud text-to-speech API (abgerufen am: 2019-02-21). URL: https://cloud.google.com/text-to-speech/?hl=de.

[7] DNN based hotword and wake word detection toolkit. contribute to kitt-AI/snowboy development by creating an account on GitHub (abgerufen am: 2019-02-22). original-date: 2016-05-10T00:54:30Z. URL: https://github.com/Kitt-AI/snowboy.

[8] Docker containerization technology for DevOps | accenture (abgerufen am: 2019-04-05). URL: https://www.accenture.com/us-en/blogs/blogs-reshma-shinde-docker-containerization-devops.

[9] Docker legal terms | docker (abgerufen am: 2019-04-05). URL: https://www.docker.com/legal.

[10] Facial recognition system (abgerufen am: 2019-03-06). Page Version ID: 885926857. URL: https://en.wikipedia.org/w/index.php?title=Facial_recognition_system&oldid=885926857.

[11] Gesichtserkennung - hilft nur noch ein balken vor dem gesicht? (abgerufen am: 2019-03-13). URL: https://www.deutschlandfunkkultur.de/gesichtserkennung-hilft-nur-noch-ein-balken-vor-dem-gesicht.2156.de.html?dram:article_id=362704.

[12] Getting started tutorial (abgerufen am: 2019-02-21). URL: https://cloud.ibm.com/docs/services/text-to-speech?topic=text-to-speech-gettingStarted&locale=de#gettingStarted.

[13] GitHub (abgerufen am: 2019-03-19). Page Version ID: 184946013. URL: https://de.wikipedia.org/w/index.php?title=GitHub&oldid=184946013.

[14] IntelliJ IDEA (abgerufen am: 2019-03-27). Page Version ID: 185552351. URL: https://de.wikipedia.org/w/index.php?title=IntelliJ_IDEA&oldid=185552351.

[15] IntelliJ IDEA: The java IDE for professional developers by JetBrains (abgerufen am: 2019-04-05). URL: https://www.jetbrains.com/idea/.

[16] Java platform, enterprise edition (abgerufen am: 2019-03-27). Page Version ID: 186577433. URL: https://de.wikipedia.org/w/index.php?title=Java_Platform,_Enterprise_Edition&oldid=186577433.

[17] JavaScript (abgerufen am: 2019-03-20). Page Version ID: 185803185. URL: https://de.wikipedia.org/w/index.php?title=JavaScript&oldid=185803185.

[18] JBoss.org UI design (abgerufen am: 2019-04-05). URL: http://design.jboss.org/wildfly/index.htm.

[19] Karl klammer (microsoft) (abgerufen am: 2019-03-27). Page Version ID: 186576451. URL: https://de.wikipedia.org/w/index.php?title=Karl_Klammer_(Microsoft)&oldid=186576451.

[20] Logitech BRIO webcam mit 4k ultra HD-video und RightLight 3 mit HDR (abgerufen am: 2019-02-22). URL: https://www.logitech.com/de-at/product/brio.

[21] Maven – welcome to apache maven (abgerufen am: 2019-04-05). URL: https://maven.apache.org/.

[22] Microsoft .NET logo vector (.AI) free download (abgerufen am: 2019-04-05). URL: https://seeklogo.com/vector-logo/168315/microsoft-net.

[23] Natural language processing (abgerufen am: 2019-04-05). Page Version ID: 888277178. URL: https://en.wikipedia.org/w/index.php?title=Natural_language_processing&oldid=888277178.

[24] Natural language toolkit — NLTK 3.4 documentation (abgerufen am: 2019-04-05). URL: http://www.nltk.org/.

[25] .NET framework (abgerufen am: 2019-03-27). Page Version ID: 185858588. URL: https://de.wikipedia.org/w/index.php?title=.NET_Framework&oldid=185858588.

[26] On-device wake word detection powered by deep learning.: Picovoice/porcupine (abgerufen am: 2019-02-22). original-date: 2018-03-08T01:55:25Z. URL: https://github.com/Picovoice/Porcupine.

[27] OpenCV (abgerufen am: 2019-03-08). Page Version ID: 185993118. URL: https://de.wikipedia.org/w/index.php?title=OpenCV&oldid=185993118.

[28] Overview | dialogflow (abgerufen am: 2019-02-23). URL: https://dialogflow.com/docs/intro.

[29] pico.js: a face-detection library in 200 lines of JavaScript (abgerufen am: 2019-02-12). URL: https://tkv.io/posts/picojs-intro/.

[30] Pricing (abgerufen am: 2019-03-27). URL: https://console.bluemix.net/docs/services/speech-to-text/faq-pricing.html#faq-pricing.

[31] Python (programmiersprache) (abgerufen am: 2019-03-20). Page Version ID: 186502024. URL: https://de.wikipedia.org/w/index.php?title=Python_(Programmiersprache)&oldid=186502024.

[32] REST schnittstellen und TYPO3 (abgerufen am: 2019-04-05). URL: http://www.oliver-weiss.com/blog/einzelansicht/article/rest-schnittstellen-und-typo3/News/detail/.

[33] Snapchat dysmorphia: How selfie filters are driving people to get plastic surgery (abgerufen am: 2019-03-08). URL: https://kslnewsradio.com/1892571/snapchat-dysmorphia/??

[34] spaCy · industrial-strength natural language processing in python (abgerufen am: 2019-04-05). URL: https://spacy.io/.

[35] Speech recognition (abgerufen am: 2019-03-27). Page Version ID: 889270640. URL: https://en.wikipedia.org/w/index.php?title=Speech_recognition&oldid=889270640.

[36] SpeechRecognition (abgerufen am: 2019-04-05). URL: https://developer.mozilla.org/en-US/docs/Web/API/SpeechRecognition.

[37] SpeechSynthesis (abgerufen am: 2019-02-21). URL: https://developer.mozilla.org/en-US/docs/Web/API/SpeechSynthesis.

[38] Spracherkennungs-API | microsoft azure (abgerufen am: 2019-02-20). URL: https://azure.microsoft.com/de-de/services/cognitive-services/speech-to-text/.

[39] Text segmentation (abgerufen am: 2019-04-05). Page Version ID: 884506547. URL: https://en.wikipedia.org/w/index.php?title=Text_segmentation&oldid=884506547.

[40] TIOBE index | TIOBE - the software quality company (abgerufen am: 2019-04-04). URL: https://www.tiobe.com/tiobe-index/.

[41] Use face ID on your iPhone or iPad pro (abgerufen am: 2019-03-06). URL: https://support.apple.com/en-us/HT208109.

[42] Virtualisierung (informatik) (abgerufen am: 2019-04-05). Page Version ID: 187221841. URL: https://de.wikipedia.org/w/index.php?title=Virtualisierung_(Informatik)&oldid=187221841.

[43] Visual studio code (abgerufen am: 2019-03-27). Page Version ID: 186694407. URL: https://de.wikipedia.org/w/index.php?title=Visual_Studio_Code&oldid=186694407.

[44] WebSocket for android - APK download (abgerufen am: 2019-04-05). URL: https://apkpure.com/websocket/uk.ing.websocket.

[45] Wolfgang von kempelen (abgerufen am: 2019-04-05). Page Version ID: 184442809. URL: https://de.wikipedia.org/w/index.php?title=Wolfgang_von_Kempelen&oldid=184442809.

[46] WordNet | a lexical database for english (abgerufen am: 2019-04-05). URL: https://wordnet.princeton.edu/.

[47] YouTrack: The issue tracking and project management tool for software teams (abgerufen am: 2019-04-04). URL: https://www.jetbrains.com/youtrack/.

[48] ApacheBooster. What is a docker container for beginners? (abgerufen am: 2019-04-05). URL: http://apachebooster.com/kb/what-is-a-docker-container-for-beginners/.

[49] erhopf. Informationen zur sprachsynthese – speech services - azure cognitive services (abgerufen am: 2019-02-21). URL: https://docs.microsoft.com/de-de/azure/cognitive-services/speech-service/text-to-speech.

[50] Peter Fischer and Peter Hofer. *Lexikon der Informatik*. Springer-Verlag. Google-Books-ID: VbooBAAAQBAJ.

[51] Adam Geitgey. The world's simplest facial recognition api for python and the command line: ageitgey/face_recognition (abgerufen am: 2019-03-08). original-date: 2017-03-03T21:52:39Z. URL: https://github.com/ageitgey/face_recognition.

[52] IBM. About (abgerufen am: 2019-02-20). URL: https://console.bluemix.net/docs/services/speech-to-text/index.html#about.

[53] Johner Institut. Blog vom johner institut (abgerufen am: 2019-04-04). URL: https://www.johner-institut.de/blog/iec-62304-medizinische-software/iso-9126-und-iso-25010/.

[54] von Martin Schindler am 6 März 2018 and 17:02 Uhr. Oracle verabschiedet sich von java enterprise edition (abgerufen am: 2019-04-05). URL: https://www.zdnet.de/88327817/oracle-verabschiedet-sich-von-java-enterprise-edition/.

[55] Beat Pfister and Tobias Kaufmann. *Sprachverarbeitung: Grundlagen und Methoden der Sprachsynthese und Spracherkennung (abgerufen am: 2019-02-21)*. Springer Vieweg, 2 edition. URL: https://www.springer.com/de/book/9783662528372.

[56] Charles Rouchon. Benchmarking microphone arrays: ReSpeaker (abgerufen am: 2019-02-20), conexant, MicroSemi AcuEdge (abgerufen am: 2019-02-20), matrix creator, MiniDSP... (abgerufen am: 2019-02-20). URL: https://bit.ly/2j7cp83.

[57] Glen Shires and Philip Jägenstedt. Web speech API (abgerufen am: 2019-02-20). URL: https://w3c.github.io/speech-api/.

[58] Kai Wedekind. HTML5 speech recognition API (abgerufen am: 2019-03-27). URL: https://codeburst.io/html5-speech-recognition-api-670846a50e92.

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang A

Arbeitsteilung

A.1 Karsten Köhne

A.1.1 Arbeitszeit

ca. 180h

A.1.2 Praktische Ausarbeitung

Chatbot
Sprachverarbeitung
Visualisierung des Avatars

A.1.3 Theoretische Ausarbeitung

Pflichtenheft

Systemarchitektur

Entwurfsentscheidungen

Chatbot
Hotword Detection
Sprachausgabe
Spracherkennung

A.2 Marcel Langoth

A.2.1 Arbeitszeit

ca. 180h

A.2.2 Praktische Ausarbeitung

Gesichtserkennung
Konzeption und Bau des Prototypen

A.2.3 Theoretische Ausarbeitung

Pflichtenheft

Ausgangslage
Istzustand
Problemstellung
Aufgabenstellung
Ziele
Funktionale Anforderungen
Nichtfunktionale Anforderungen

Entwurfsentscheidungen

Gesichtserkennung

Ausgewählte Aspekte und Probleme

Gesprächsinitierung
Konstruktion und Bau von Prototyp

Toolstack

Sprachen
Werkzeuge - GitHub, YouTrack

A.3 Denys Sheludchenko

A.3.1 Arbeitszeit

ca. 180h

A.3.2 Praktische Ausarbeitung

Backend
Externe APIs

A.3.3 Theoretische Ausarbeitung

Entwurfsentscheidungen

Backend

Toolstack

Technologien
Werkzeuge - IntelliJ, Visual Studio Code, Docker

Anhang B

Protokolle

B.0.1 Besprechung 23.07.2018

Teilnehmer

Prof. Mag. Dr. Thomas Stütz, Karsten Köhne, Denys Sheludchenko

Ort

HTL Leonding

Aufgaben

Projektiststand ansehen
Projektplan erstellen - Scrum
Visionen-Brainstorming

B.0.2 Besprechung 13.08.2018

Teilnehmer

Prof. Mag. Dr. Thomas Stütz, Karsten Köhne, Marcel Langoth, Denys Sheludchenko

Ort

HTL Leonding

Aufgaben

Aufgabenverteilung
Prototypplanung
Entscheidung der Implementierungssprachen

B.0.3 Besprechung 03.09.2018

Teilnehmer

Prof. Mag. Dr. Thomas Stütz, Karsten Köhne, Marcel Langoth, Denys Sheludchenko

Ort

HTL Leonding

Aufgaben

Alexa-Skill imlementieren
Vorbereitung auf Austellung
Zusammenbau Prototyp

B.0.4 Besprechung 08.10.2018

Teilnehmer

Prof. Mag. Dr. Thomas Stütz, Karsten Köhne, Marcel Langoth, Denys Sheludchenko

Ort

HTL Leonding

Aufgaben

Nachbesprechung Austellung

B.0.5 Besprechung 19.10.2018

Teilnehmer

Prof. Mag. Dr. Thomas Stütz, Karsten Köhne, Marcel Langoth, Denys Sheludchenko

Ort

HTL Leonding

Aufgaben

Geometrische Maße für neues Pyramiden Layout ermitteln
Systemdokumentation erstellen
Systemspezifikation erstellen

B.0.6 Besprechung 12.11.2018

Teilnehmer

Prof. Mag. Dr. Thomas Stütz, Karsten Köhne, Marcel Langoth, Denys Sheludchenko

Ort

HTL Leonding

Aufgaben

Besprechung weitere Vorgehensweise Sprachverarbeitung
Dialogflow-Test
Linz-AG API einlesen und abfragen

B.0.7 Besprechung 11.12.2018

Teilnehmer

Prof. Mag. Dr. Thomas Stütz, Karsten Köhne, Marcel Langoth, Denys Sheludchenko

Ort

HTL Leonding

Aufgaben

Benutzertest Dialogflow
Bau des 2. Prototypen
Materialanalyse Pyramiden
Mikrofontest

B.0.8 Besprechung 05.01.2019

Teilnehmer

Prof. Mag. Dr. Thomas Stütz, Karsten Köhne, Marcel Langoth

Ort

HTL Leonding

Aufgaben

Benutzertest Dialogflow
Bau des 2. Prototypen
Materialanalyse Pyramiden
Mikrofontest
Besprechung Aufbau Schlossmuseum

B.0.9 Besprechung 18.01.2019

Teilnehmer

Prof. Mag. Dr. Thomas Stütz, Karsten Köhne, Marcel Langoth

Ort

HTL Leonding

Aufgaben

Aufbau für Ball der Leondinger
Generalprobe von Leonie
Animation von Leonie mit Game-Engine

B.0.10 Besprechung 13.02.2019

Teilnehmer

Prof. Mag. Dr. Thomas Stütz, Karsten Köhne, Marcel Langoth

Ort

HTL Leonding

Aufgaben

Aufbau von Leonie in der Aula
Besprechung für den Aufbau de schriftlichen Arbeit

B.0.11 Besprechung 22.02.2019

Teilnehmer

Prof. Mag. Dr. Thomas Stütz, Karsten Köhne, Marcel Langoth

Ort

HTL Leonding

Aufgaben

Abgabe und Kontrolle der 1. schriftlichen Fassung