



HTL Dornbirn

Höhere Lehranstalt für Wirtschaftsingenieurwesen

Ausbildungsschwerpunkt Betriebsinformatik

Diplomarbeit

Entwicklung eines „Chatbot“ als Informationsquelle für jugendrelevante Themen in Lustenau

Ausgeführt im Schuljahr 2017/18 von:

Jakob Ott

5cWI

Matteo Kofler

5cWI

Manuel Waibel

5cWI

Betreuer:

Dipl.-Ing. Diethard Kaufmann

Ing. Benno Kofler

Dornbirn, am 05.04.2018

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

Diplomanden:

Datum und Unterschrift:

Jakob Ott

Matteo Kofler

Manuel Waibel

Abstract

The aim of the diploma project lustenauBot was to develop a chatbot for Facebook Messenger, which answers the user's questions about youth relevant subjects of Lustenau. The diploma project was developed in cooperation with the market municipality of Lustenau.

The project idea originated within our team. We also stood in contact with several other companies during our search for a suitable diploma project. This future-oriented topic offered an exciting challenge for us. During the development phase, we mastered many interesting challenges, which confirmed our decision, that this is the best project for us. After a successful conclusion of the project, a takeover of the chatbot by the market municipality for the official release is planned.

A special enrichment for us was to apply the program and project management abilities acquired in school lessons within the scope of a big project practically to extend them at the same time.

During the search and the beginning of our diploma project we depended on the help and the support of different people. At this point we would like to thank them for their support.

Special thanks are valid for our project responsible people, Prof. Diethard Kaufmann as a school responsible person and Mr Benno Kofler as representative of the market municipality of Lustenau, who stood to us aside.

Vorwort & Danksagung

Die Diplomarbeit lustenauBot hatte das Ziel, einen *Facebook Messenger* „Bot“ zu entwickeln, der dem User Fragen zu jugendrelevanten Themen der Marktgemeinde beantwortet. Das Projekt wurde in Zusammenarbeit mit der Marktgemeinde Lustenau entwickelt.

Die Projektidee entstand innerhalb unseres Teams, da wir uns über einen längeren Zeitraum mit der Suche eines geeigneten Diplomarbeitsthemas beschäftigten und dabei auch mit mehreren Firmen im Kontakt standen. Wir entschieden uns für die Umsetzung des „Chatbot“, da dieser für uns eine spannende Herausforderung bot. Während der Entwicklungsarbeit wurde unsere Entscheidung durch die vielen interessanten Teilbereiche bestätigt, die es zu meistern galt. Mit der Marktgemeinde Lustenau konnten wir ein Partnerunternehmen von unserer Idee überzeugen. Bei erfolgreichem Abschluss des Projekts ist eine Übernahme des „Chatbot“ durch die Marktgemeinde geplant. Somit könnte der „Chatbot“ der Lustenauer Jugend zur Verfügung gestellt werden.

Eine besondere Bereicherung für uns war, die im Unterricht erworbenen Programmier- und Projektmanagementfähigkeiten im Rahmen eines großen Projekts praktisch anzuwenden und gleichzeitig auch zu erweitern.

Während der Suche und der Umsetzung unseres Diplomarbeitsthemas waren wir auf die Hilfe und die Unterstützung unterschiedlichster Personen angewiesen. An dieser Stelle möchten wir uns deshalb recht herzlich für deren Unterstützung bedanken.

Ein besonderer Dank gilt unseren Projektbetreuern, Prof. Diethard Kaufmann als Schulbetreuer und Herrn Benno Kofler seitens der Marktgemeinde Lustenau, die uns stets zur Seite standen.

Hinweis zur geschlechtsneutralen Formulierung

Um eine geschlechtsneutrale als auch eine gut lesbare Formulierung zu gewährleisten, werden für geschlechtsspezifische Wörter Rollen definiert, die durchgehend als männlich bzw. weiblich verwendet werden.

In folgender Tabelle werden die verwendeten Rollen aufgelistet.

männlich	weiblich
Benutzer	Anwenderin
Entwickler	Programmiererin

Hinweis zur Textformatierung

Die verschiedenen Textformatierungen in diesem Dokument werden einheitlich verwendet.

In folgender Tabelle werden die unterschiedlichen Formatierungen erläutert:

Formatierung	Erklärung
<i>Kursiv</i>	Begriffe, die im Glossar erklärt werden.
Abkürzung (Abk.)	Abkürzungen werden bei erster Verwendung ausgeschreiben, gefolgt von der Abkürzung in Klammer. Abkürzungen sind im Abkürzungsverzeichnis aufgelistet.
„Anführungszeichen“	Anführungszeichen werden für technische Ausdrücke wie „ <i>Chatbot</i> “ oder „ <i>API</i> “ verwendet.
(Autor, Jahr)	Im Textverlauf werden Quellen im Format (Autor, Jahr) angegeben. Die Quellen sind im Literaturverzeichnis detaillierter angeführt.

Hinweis: Die verwendeten Hilfsmittel wurden in den folgenden Texten nicht speziell hervorgehoben, diese sind in Kapitel 10 aufgelistet.

Inhaltsverzeichnis

1	Impressum	1
1.1	Projektteam.....	1
1.1.1	Jakob Ott	1
1.1.2	Matteo Kofler	1
1.1.3	Manuel Waibel.....	2
1.2	Projektbetreuer	3
1.2.1	Dipl.-Ing. Diethard Kaufmann	3
1.2.2	Ing. Benno Kofler.....	3
2	Marktgemeinde Lustenau.....	4
3	Projektmanagement.....	5
3.1	Allgemein	5
3.2	Projektauftrag.....	6
3.3	Projektzieleplan.....	7
3.3.1	Hauptziele	7
3.3.2	Zusatzziele	7
3.3.3	Nichtziele	8
3.4	Projektorganisation und Projektorganigramm	9
3.5	Projektstrukturplan	11
3.6	Projektterminplan	12
3.7	Meilensteinplan	13
3.8	Risikoanalyse.....	14
3.9	Projektumweltanalyse	16
3.10	Projektabschlussbericht.....	18
3.10.1	Projektverlauf	18
3.10.2	Zielerreichung	18

IustenauBot

3.10.3	Projektabschlussmeeting	18
3.10.4	Ausblick (Zeit nach dem Projekt).....	19
4	Analysephase	20
4.1	Softwarearchitektur	20
4.1.1	Begriffsdefinition	20
4.1.2	Benutzeroberfläche	21
4.2	Facebook Messenger.....	21
4.3	Textverarbeitung	21
4.3.1	Dialogflow (ehemals API.ai)	22
4.3.2	Wit.ai	22
4.3.3	Microsoft Bot Framework.....	22
4.3.4	Vergleich zwischen den Plattformen.....	23
4.3.5	WebHook.....	23
4.4	Konzeption	24
4.4.1	Allgemein.....	24
4.4.2	Themenabdeckung des „Chatbot“	24
4.4.3	Konversationsgestaltung	25
4.5	Use-Case-Diagramm	26
4.6	Datenquellen.....	27
4.6.1	Statische und dynamische Daten	27
4.6.2	Abfahrtszeiten.....	27
4.6.3	Veranstaltungen	28
4.6.4	Wetterinformationen	28
4.6.5	Andere Datenquellen.....	29
5	Einarbeitung.....	30
5.1	Messenger API	30
5.2	Dialogflow	31

IustenauBot

5.2.1	Verbinden von Dialogflow mit der Facebook Messenger API.....	31
5.2.2	Intents und Zuweisung einer Rückgabe	32
5.2.3	Intents mit Kontext.....	32
5.2.4	Namenskonvention von Intents	33
5.2.5	Erstellen eines Intents	34
5.2.6	WebHook.....	36
5.2.7	Entities.....	37
5.3	Git	38
5.4	Python.....	40
5.4.1	Struktur.....	40
5.4.2	Python-Lauffähigkeit.....	41
5.4.3	Web-App-Konfiguration mit Flask	42
5.4.4	Von der Anfrage zur Antwort	43
5.5	Facebook Messenger.....	45
5.5.1	Aufsetzen einer Facebook-Seite.....	45
5.5.2	Facebook Developer Console	45
5.6	Heroku	46
5.6.1	Prinzip	46
5.6.2	Implementierung.....	47
6	Design.....	49
6.1	Mockups.....	49
6.2	User Interface	50
6.3	Logo	50
6.4	Facebook Seite	51
6.5	User Experience	52
6.5.1	Onboarding.....	52
6.5.2	Buttons	53

IustenauBot

6.5.3	Interaktionsmöglichkeiten	53
6.6	Sprache.....	54
7	Entwicklung und Umsetzung.....	56
7.1	Wetterbericht.....	56
7.1.1	YahooWeatherAPI.....	57
7.1.2	OpenWeatherMap	58
7.2	Jugendveranstaltungen.....	63
7.2.1	Probleme	63
7.2.2	Übersicht der relevanten Tabellen.....	64
7.2.3	Select-Statement für erfolgreiche Selektion der Daten.....	66
7.2.4	Aufbereitung der selektierten Daten	68
7.2.5	Beispiel für die Anfrage einer Veranstaltung	68
7.3	Freizeitangebote	70
7.3.1	Sport.....	70
7.3.2	Kino	71
7.3.3	Gastronomie.....	72
7.3.4	Relaxen	74
7.4	Abfahrtszeiten	75
7.4.1	Programmierung.....	75
7.5	Kontakt zur Gemeinde	79
7.6	Smalltalk mit „Chatbot“	79
7.6.1	Erzählen von Witzen.....	79
7.6.2	Grußformel	81
7.6.3	Vulgärsprache	81
7.6.4	Allgemeine Informationen.....	82
8	Testphase	84
8.1	Tag der offenen Tür der HTL Dornbirn.....	84

Iustenaubot

8.2	Laufende Tests	84
9	Fazit	86
10	Hilfsmittel.....	87
11	Glossar	94
12	Autorenverzeichnis.....	96
13	Abbildungsverzeichnis.....	98
14	Tabellenverzeichnis.....	103
15	Abkürzungsverzeichnis	104
16	Literaturverzeichnis	105
17	Anhang.....	109
17.1	Projektstatusbericht	109
17.1.1	Momentaufnahme des Projekts.....	109
17.1.2	Statuszusammenfassung	109
17.1.3	Zeiterfassung Diplomarbeit	109

1 Impressum

Das Impressum des Projekts beinhaltet alle Personen, die am Projekt mitgearbeitet bzw. das Projekt betreut haben.

1.1 Projektteam

Das Projektteam bestand aus drei Personen, zwei davon waren zum größten Teil mit der Erstellung des „Chatbot“ beauftragt. Die dritte Person arbeitete am Projektmanagement. Die Projektteammitglieder besuchen den 5. Jahrgang der HTL Dornbirn im Bereich Wirtschaftsingenieurwesen mit Schwerpunkt Betriebsinformatik.

1.1.1 Jakob Ott



Abb. 1 Jakob Ott

Jakob übernahm als Projektleiter (PL) die Aufgaben des Projektmanagements und die Aufgabenverteilung. Er beschäftigte sich unter anderem mit der Terminplanung und sorgte somit für einen reibungslosen Ablauf des Projekts, außerdem war er für die Erstellung der gesamten Projektpläne und für die Kommunikation mit dem Projektbetreuer zuständig.

1.1.2 Matteo Kofler



Abb. 2 Matteo Kofler

Matteo beschäftigte sich zu Beginn des Projekts mit der Entwicklung des Konzepts und anschließend mit der Auswahl der verwendeten *Frameworks*. Im weiteren Verlauf war er mit der Programmierung des „Chatbot“, im speziellen mit der Entwicklung des Wetterberichts und der Abfahrtszeiten beschäftigt. Auch das Führen von Smalltalk mit dem „Chatbot“ zählte zu Matteos Aufgabenbereich.

1.1.3 Manuel Waibel



Manuel beschäftigte sich in der Startphase des Projekts hauptsächlich mit dem Entwurf der einzelnen Teilbereiche, welche der „Chatbot“ abdecken soll. Im weiteren Verlauf war er mit der Programmierung des „Chatbot“, im speziellen mit der Entwicklung der Freizeitangebote, der Jugendveranstaltungen und dem Kontakt zur Gemeinde beschäftigt. Manuel war durch das Einbringen neuer Ideen stets eine Bereicherung für uns.

Abb. 3 Manuel Waibel

1.2 Projektbetreuer

Dem Projektteam standen zwei Projektbetreuer zur Seite. Als Schulbetreuer gab uns Herr Diethard Kaufmann bei Problemen und Fragen wertvolle Hilfestellung. Seitens der Marktgemeinde Lustenau stand uns Herr Benno Kofler bei Fragen gerne zur Verfügung. Herr Kaufmann war einverstanden, unser Projekt seitens der Schule zu unterstützen, da er bereits einen Teil der Klasse in Softwareentwicklung und Projektmanagement (SWP) unterrichtete.

1.2.1 Dipl.-Ing. Diethard Kaufmann



Abb. 4 Diethard Kaufmann

Herr Kaufmann betreute unsere Diplomarbeit seitens der HTL Dornbirn. Er unterrichtet an der HTL im Bereich Betriebsinformatik. Weiters arbeitet er als Java-Entwickler bei Inet-logistics in Dornbirn. Seine Erfahrung im Bereich Projektmanagement und seine Hilfe bei der Terminplanung trug positiv zum Erfolg des Projektes bei.

1.2.2 Ing. Benno Kofler



Abb. 5 Benno Kofler

Herr Kofler arbeitet in der Abteilung Informatik der Marktgemeinde Lustenau. Zu seinen Aufgaben zählt die Betreuung der Informatik der Gemeinde und des geografischen Informationssystems. Er wurde dem Projektteam als Betreuer der Marktgemeinde Lustenau zur Seite gestellt. Er stand uns als Kontakt mit der Marktgemeinde zur Verfügung.

2 Marktgemeinde Lustenau

Lustenau ist mit über 23.000 Einwohnern die einwohnerreichste Marktgemeinde Österreichs. Da Lustenau einen hohen Anteil an Jugendlichen aufweist, war es für die Marktgemeinde eine klare Sache, den lustenauBot zu unterstützen. In Lustenau hat die Jugend schon seit vielen Jahren einen hohen Stellenwert, so wurde bereits eine Vielzahl an Jugendprojekten realisiert.

Dies sind einige Projekte der letzten Jahre:

- Offene Jugendarbeit
- Jugendplatz Habedere!
- Sportpark
- Jugendtreff Oase
- Culture Factor Y (mit Jugendcafé)
- Interkulturelles Mädchencafé

weitere Infos finden Sie auf der Homepage der Marktgemeinde (www.lustenau.at).

3 Projektmanagement

Das Projektmanagement (PM) stellte während der Projektlaufzeit die Erreichung der Projektziele sicher und sorgte für ein termingerechtes Zeitmanagement.

3.1 Allgemein



Im Vorfeld der Entwicklung des „*Chatbot*“, wurden Projektpläne erstellt. Diese dienten der Sicherung des Projekterfolgs, welcher durch die Erfüllung der Ziele erreicht wurde. Durch die Erstellung der Projektpläne zu Beginn des Projekts konnten etwaige Risiken und Terminkollisionen bereits früher erkannt werden.

Die Projektpläne wurden im Laufe der Projektphasen ständig aktualisiert und überprüft, um die festgelegten Termine im Auge zu behalten. Während der gesamten Projektlaufzeit wurde jeweils zu Beginn eines jeden Monats ein Projektstatusbericht angefertigt (siehe Kapitel 17.1).

Die Projektpläne wurden größtenteils mit Microsoft Visio 2013 und Microsoft Word 2016 erstellt und waren während des gesamten Projekts via Google Drive für das Projektteam einsehbar.

3.2 Projektauftrag

Im Projektauftrag wurden die Projektziele und die geplanten Start- und Endtermine für das Projekt festgelegt. Außerdem wurden die Rollen innerhalb des Projektteams, der Auftraggeber und der Betreuer des Projekts schriftlich festgehalten. Mit der Unterschrift des Projektauftraggebers und des Projektleiters ist das Projekt offiziell gestartet worden.

PROJEKTAUFTRAG	
Projektstarttermin <ul style="list-style-type: none"> 11.09.2017 	Projektendtermin: <ul style="list-style-type: none"> 01.05.2018
Projektziele: <ul style="list-style-type: none"> Bereitstellen von Sportplätzen, Öffnungszeiten und Infos über Freizeitangebote in Lustenau Jugendveranstaltungen der kommenden Woche in Lustenau können abgerufen werden. Der aktuelle Wetterbericht inklusive Vorhersage für Lustenau kann abgefragt werden. Die aktuellen Abfahrtszeiten der Landbusse in Lustenau können abgerufen werden. 	
Zusatzziele: <ul style="list-style-type: none"> Kontakte zu jugendrelevanten Services der Gemeinde mit Telefonnummern können abgerufen werden. Integration des LustenauBot auf der Homepage der Marktgemeinde Lustenau Bereitstellung von zwischenmenschlichen Antworten für einfachen Smalltalk 	
Nichtziele: <ul style="list-style-type: none"> Entwicklung einer eigenen App Das Erstellen eines Updates bzw. <i>Postings</i> im eigenen Facebook-<i>Profil</i> Betreuung des Bots nach der Projektabnahme 	
Hauptaufgaben (Projektphasen): <ul style="list-style-type: none"> Vorprojektphase Programmierung Bot Dokumentation Testphase und Nachbearbeitung Projektabschluss, Abnahme 	
ProjektauftraggeberIn: Marktgemeinde Lustenau Betreuer: Benno Kofler	ProjektleiterIn: Jakob Ott
Projektteam: Matteo Kofler Manuel Waibel	Projektbetreuer: Diethard Kaufmann
 Vorname Nachname, (ProjektauftraggeberIn)	 Vorname Nachname, (ProjektleiterIn)

Tab. 1 Projektauftrag

3.3 Projektzieleplan

Mithilfe des Projektzieleplans wurden die Projektziele festgelegt, diese sind untergliedert in Haupt-, Zusatz- und Nichtziele. Das Erfüllen der Hauptziele ist für einen erfolgreichen Projektabschluss entscheidend. Nebenziele sind optional und müssen nicht unbedingt erfüllt werden, sind aber als positiver Zusatzaspekt zu sehen. Mithilfe der Nichtziele wurden die Projektgrenzen klar definiert.

3.3.1 Hauptziele

- Bereitstellen von Sportplätzen, Öffnungszeiten und Infos über Freizeitangebote in Lustenau
 - Standorte der beliebtesten Sportplätze in Lustenau werden angezeigt, das Angebot wird durch weitere nützliche Informationen zu verschiedensten Freizeitangeboten ergänzt.
- Jugendveranstaltungen der kommenden Woche in Lustenau können abgerufen werden.
 - Veranstaltungen in Lustenau wie etwa die Kilbi oder Sommer.Lust am Platz können direkt im Bot angezeigt werden.
- Der aktuelle Wetterbericht inklusive Vorhersage für Lustenau kann abgefragt werden.
 - Der Wetterbericht von heute inklusive der aktuellen Temperatur in Lustenau kann abgerufen werden, außerdem ist eine Vorhersage für den Folgetag vorhanden.
- Die aktuellen Abfahrtszeiten der Landbusse in Lustenau können abgerufen werden.
 - Durch die Eingabe einer Bushaltestelle in Lustenau werden die nächsten dort abfahrenden Busse inklusive Uhrzeit angezeigt.

3.3.2 Zusatzziele

- Kontakte zu jugendrelevanten Services der Gemeinde mit Telefonnummern können abgerufen werden.
 - Statische Informationen wie Telefonnummern und Emailadressen können abgefragt werden.
- Integration des lustenauBot auf der Homepage der Marktgemeinde Lustenau.
 - Durch die Integration des „Chatbot“ auf der Homepage, kann der „Chatbot“ direkt über ein integriertes Chatfenster verwendet werden.
- Bereitstellung von zwischenmenschlichen Antworten für einfachen Smalltalk.
 - Der „Chatbot“ kann Witze erzählen und einfache Konversationen führen.
- Aufzeigen der bekanntesten Sehenswürdigkeiten in Lustenau
 - Einige der bedeutendsten Sehenswürdigkeiten können abgerufen werden, diese werden mit einigen Hintergrundinformationen angezeigt.

- Auflistung einiger markanter Fakten über Lustenau
 - Die wichtigsten Fakten über Lustenau werden aufgelistet.

3.3.3 Nichtziele

- Entwicklung einer eigenen App.
 - Eine Applikation für mobile Geräte (Android oder IOS) wird nicht entwickelt. Da der lustenauBot über den Facebook-Messenger erreicht wird, wäre eine eigene App obsolet.
- Das Erstellen eines Updates bzw. *Postings* im eigenen Facebook-*Profil*
 - Das Erstellen eines Updates bzw. *Posting* in der eigenen Facebook-Timeline durch den „*Chatbot*“ wird nicht entwickelt.
- Betreuung des Bots nach der Projektabnahme
 - Die Betreuung des Bots nach der Projektabnahme ist nicht vorgesehen, da das Projektteam nach Abnahme des Projekts aufgelöst wird.

3.4 Projektorganisation und Projektorganigramm

In der folgenden Tabelle wird die Organisation des Projekts dargestellt. Es wird der Betreuer, der Auftraggeber, der Projektleiter und die Teammitglieder zusammen mit ihren jeweiligen Aufgabenbereichen aufgelistet.

Projektorganisation		
Projektrolle	Aufgabenbereich	Name
Projektauftraggeber	<ul style="list-style-type: none"> • Ansprechpartner der Gemeinde Lustenau 	Benno Kofler
Projektbetreuer	<ul style="list-style-type: none"> • Hilfestellung bei Problemen • Inhaltliche Korrektur der Dokumentation • Hilfe bei Fragen zum Projektmanagement 	Diethard Kaufmann
Projektleiter	<ul style="list-style-type: none"> • Koordination und Leitung des Projekts • Projektmanagement • Design und Aufbau der Dokumentation • Organisation von Statusmeetings • Kommunikation mit Projektbetreuer und Auftraggeber • Schreiben der Dokumentation • Testen des „<i>Chatbot</i>“ 	Jakob Ott
Projektteammitglieder	<ul style="list-style-type: none"> • Analyse der <i>Frameworks</i> • Erarbeiten des Konzepts • Entwicklung „<i>Chatbot</i>“ • Schreiben der Dokumentation • Testen des „<i>Chatbot</i>“ 	Matteo Kofler Manuel Waibel

Tab. 2 Projektorganisation

In der untenstehenden Abbildung (Abb. 6) ist der hierarchische Aufbau des Projekts grafisch dargestellt:

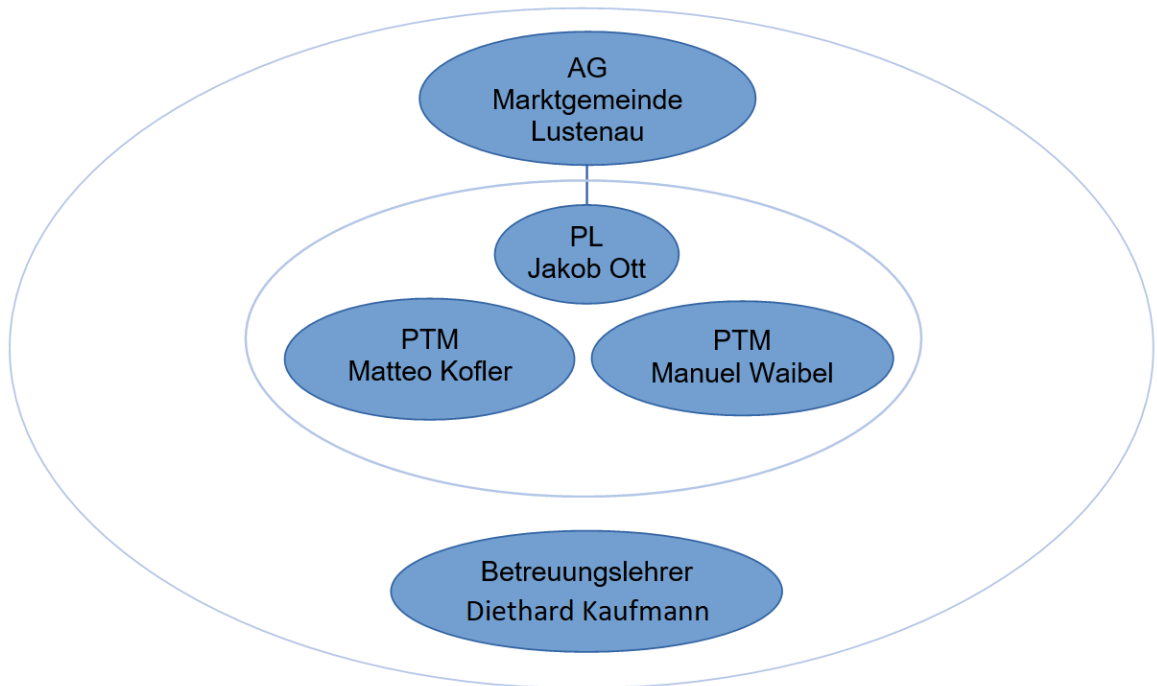


Abb. 6 Projektorganigramm

3.5 Projektstrukturplan

Im Projektstrukturplan (PSP) wurden die einzelnen Arbeitsschritte des Projekts in Phasen eingeteilt, welche auch als Arbeitspakete bezeichnet werden können (siehe Abb. 7). Die einzelnen Phasen wurden meist chronologisch abgearbeitet, allerdings kam es für gewöhnlich vor, dass mehrere Arbeitspakete parallel abgearbeitet wurden.

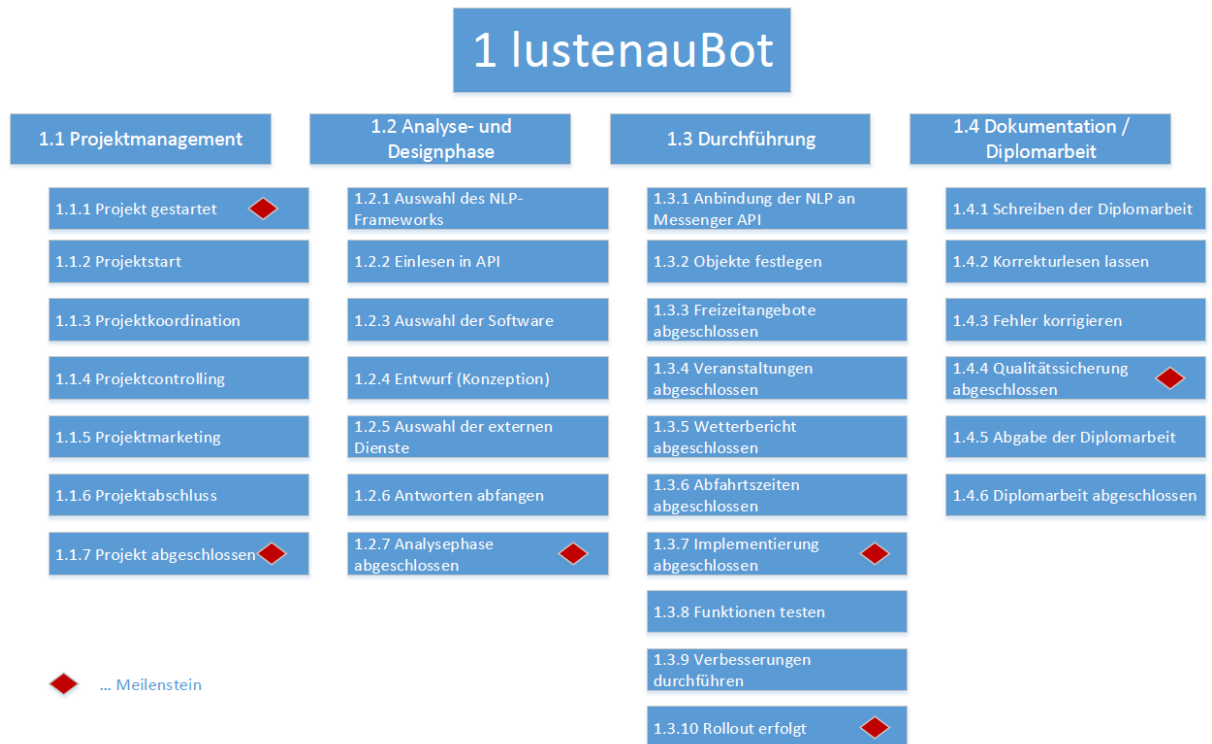


Abb. 7 Projektstrukturplan

Besondere Arbeitspakete, die das Erreichen eines wichtigen Ereignisses darstellen, werden als Meilensteine (oben als rote Raute dargestellt) bezeichnet. Meilensteine werden zur Signalisierung eines wichtigen Start- oder Abschlussprozesses in einem Projekt verwendet.

Der PSP dient als Basis für den Projektterminplan (PTP), die einzelnen Phasen werden direkt in den Terminplan übernommen.

Der PSP stellte während des Projekts eine Hilfe bei der Organisation der Aufgabenverteilung dar. Bei einem Projekt in dieser Größenordnung ist es wichtig, dass eine gewisse Struktur eingehalten wird, um den ständigen Überblick über das Projekt zu behalten.

3.6 Projektterminplan

Der Projektterminplan zeigt den Anfang und das Ende der jeweiligen Arbeitspakete auf. Der PTP diente der groben Planung der Termine, welche im Laufe des Projektes nicht immer eingehalten werden konnten, da einzelne Arbeitspakete teilweise vor oder nach dem Endtermin abgeschlossen wurden.

PSP-Code	Vorgangsname	Dauer	Anfang	Ende
1.1	Projektmanagement	206 Tage	12.09.2017	05.04.2018
1.1.1	Projekt gestartet	0 Tage	12.09.2017	12.09.2017
1.1.2	Projektstart	0 Tage	12.09.2017	12.09.2017
1.1.3	Projektkoordination	203 Tage	15.09.2017	05.04.2018
1.1.4	Projektcontrolling	203 Tage	15.09.2017	05.04.2018
1.1.5	Projektmarketing	203 Tage	15.09.2017	05.04.2018
1.1.6	Projektabschluss	0 Tage	05.04.2018	05.04.2018
1.1.7	Projekt abgeschlossen	0 Tage	05.04.2018	05.04.2018
1.2	Analyse- und Designphase	19 Tage	12.09.2017	30.09.2017
1.2.1	Auswahl des NLP-Frameworks	19 Tage	12.09.2017	30.09.2017
1.2.2	Einlesen in API	19 Tage	12.09.2017	30.09.2017
1.2.3	Auswahl der Software	19 Tage	12.09.2017	30.09.2017
1.2.4	Entwurf (Konzeption)	19 Tage	12.09.2017	30.09.2017
1.2.5	Auswahl der externen Dienste	19 Tage	12.09.2017	30.09.2017
1.2.6	Antworten abfangen	19 Tage	12.09.2017	30.09.2017
1.2.7	Analysephase abgeschlossen	0 Tage	30.09.2017	30.09.2017
1.3	Durchführung	92 Tage	01.10.2017	31.12.2017
1.3.1	Anbindung der NLP an Messenger API	5 Tage	01.10.2017	05.10.2017
1.3.2	Objekte festlegen	87 Tage	06.10.2017	31.12.2018
1.3.3	Freizeitangebote abgeschlossen	87 Tage	06.10.2017	31.12.2018
1.3.4	Veranstaltungen abgeschlossen	87 Tage	06.10.2017	31.12.2018
1.3.5	Wetterbericht abgeschlossen	87 Tage	06.10.2017	31.12.2018
1.3.6	Abfahrtszeiten abgeschlossen	87 Tage	06.10.2017	31.12.2018
1.3.7	Implementierung abgeschlossen	0 Tage	31.12.2017	31.12.2017
1.3.8	Funktionen testen	92 Tage	01.11.2017	31.01.2018
1.3.9	Verbesserungen durchführen	29 Tage	01.02.2018	01.03.2018
1.3.10	Rollout erfolgt	0 Tage	01.03.2018	01.03.2018
1.4	Dokumentation / Diplomarbeit	126 Tage	01.12.2017	05.04.2018
1.4.1	Schreiben der Diplomarbeit	62 Tage	01.12.2017	31.01.2018
1.4.2	Korrekturlesen lassen	9 Tage	31.01.2018	08.02.2018
1.4.3	Fehler korrigieren	21 Tage	09.02.2018	01.03.2018
1.4.4	Qualitätssicherung abgeschlossen	0 Tage	01.03.2018	01.03.2018
1.4.5	Abgabe der Diplomarbeit	0 Tage	05.04.2018	05.04.2018
1.4.6	Diplomarbeit abgeschlossen	0 Tage	05.04.2018	05.04.2018

Tab. 3 Projektterminplan

3.7 Meilensteinplan

Der Meilensteinplan zeigt in chronologischer Abfolge die zentralen Ereignisse im Projekt (Meilensteine), die erreicht bzw. abgeschlossen wurden. Zur besseren Veranschaulichung wurden die Meilensteine mit dem jeweiligen PSP-Code und dem geschätzten Plantermin in der untenstehenden Tabelle dargestellt.

Meilensteinplan		
PSP-Code	Meilenstein	Plantermin
1.1.1	Projekt gestartet	12.09.2017
1.2.7	Analysephase abgeschlossen	30.09.2017
1.3.7	Implementierung abgeschlossen	31.12.2017
1.3.10	Rollout erfolgt	01.03.2018
1.4.4	Qualitätssicherung abgeschlossen	01.03.2018
1.1.7	Projekt abgeschlossen	05.04.2018

Tab. 4 Meilensteinplan

3.8 Risikoanalyse

Zu Beginn des Projekts wurde eine Risikoanalyse erstellt, die die Identifizierung von potenziellen Risiken unterstützt. Die einzelnen Risiken wurden in einer Matrix nach Eintreffen und Auswirkung auf das Projekt eingestuft (siehe Abb. 8). Um das Entstehen der projektgefährdenden Risiken zu verhindern, wurden parallel zu Projektbeginn Lösungsmaßnahmen entwickelt.

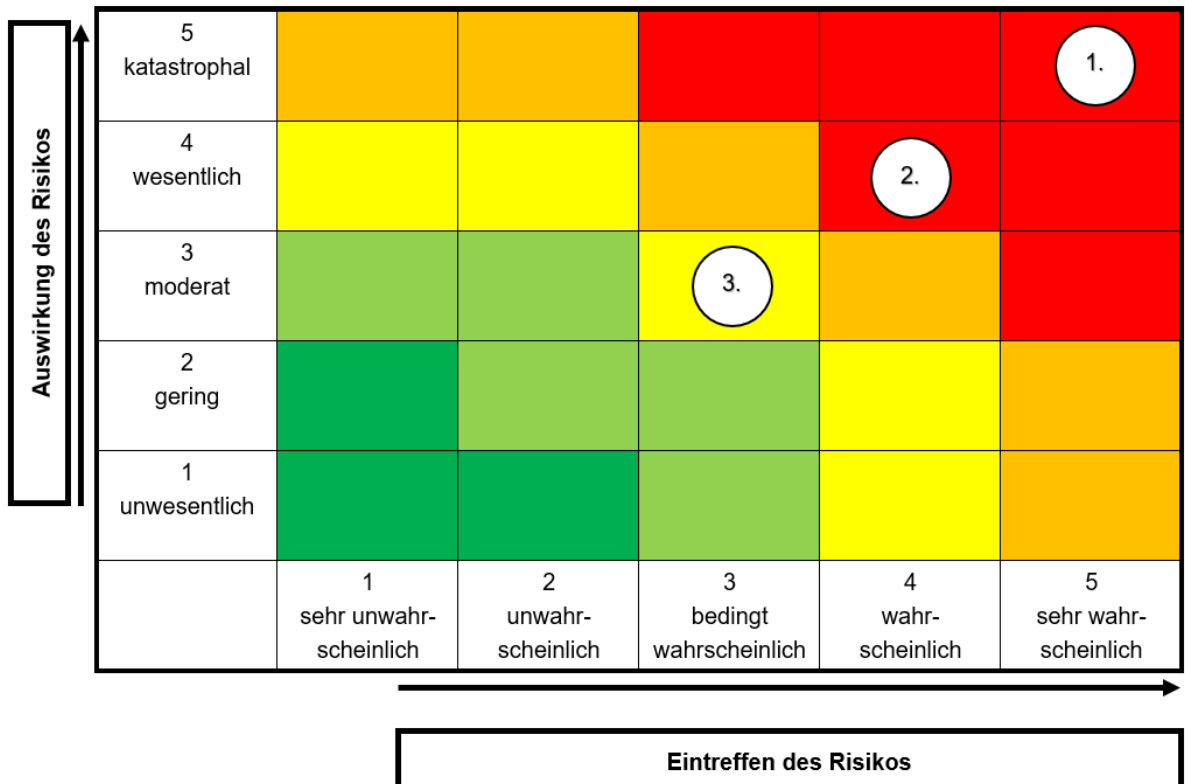


Abb. 8 Risikoanalyse

(BBK Deutschland, 2018)

In der folgenden Tabelle werden die oben gekennzeichneten Risiken mit der jeweiligen Nummer aufgelistet. Des Weiteren wurden die einzelnen Risiken auf ihre Gefahr hin bewertet und Maßnahmen zur Lösung beschrieben.

Nr.	Risiko	Gefahr	Maßnahme(n)
1	Falsche Einschätzung des zeitlichen Aufwands der einzelnen Arbeitspakete.	hoch	<ul style="list-style-type: none">• Einplanen von Pufferzeiten• rechtzeitiger Beginn mit der Abarbeitung der Arbeitspakete
2	Fehlendes Fachwissen im Umgang mit <i>Dialogflow</i>	hoch	<ul style="list-style-type: none">• Rechtzeitiges Einlesen in <i>Dialogflow</i>
3	Fehlende Daten von Drittanbietern	mittel	<ul style="list-style-type: none">• Rechtzeitige Abklärung mit Drittanbietern bezüglich Zugriff auf Datenbanken

Tab. 5 Risikoanalyse

3.9 Projektumweltanalyse

Bei der Projektumweltanalyse wurden alle für das Projekt relevanten Umwelten (Personen, Personengruppen, ...), auf interner und externer Ebene erfasst. (siehe Abb. 9) Im nächsten Schritt wurden die kritischen Umwelten, das bedeutet jene Personen bzw. Personengruppen, die dem Projekt oder einem Teilbereich des Projekts gegenüber kritisch eingestellt sind, näher betrachtet. Um diesen kritischen Beziehungen positiv entgegen zu wirken, wurden Maßnahmen entwickelt, die in der untenstehenden Tabelle angeführt sind.

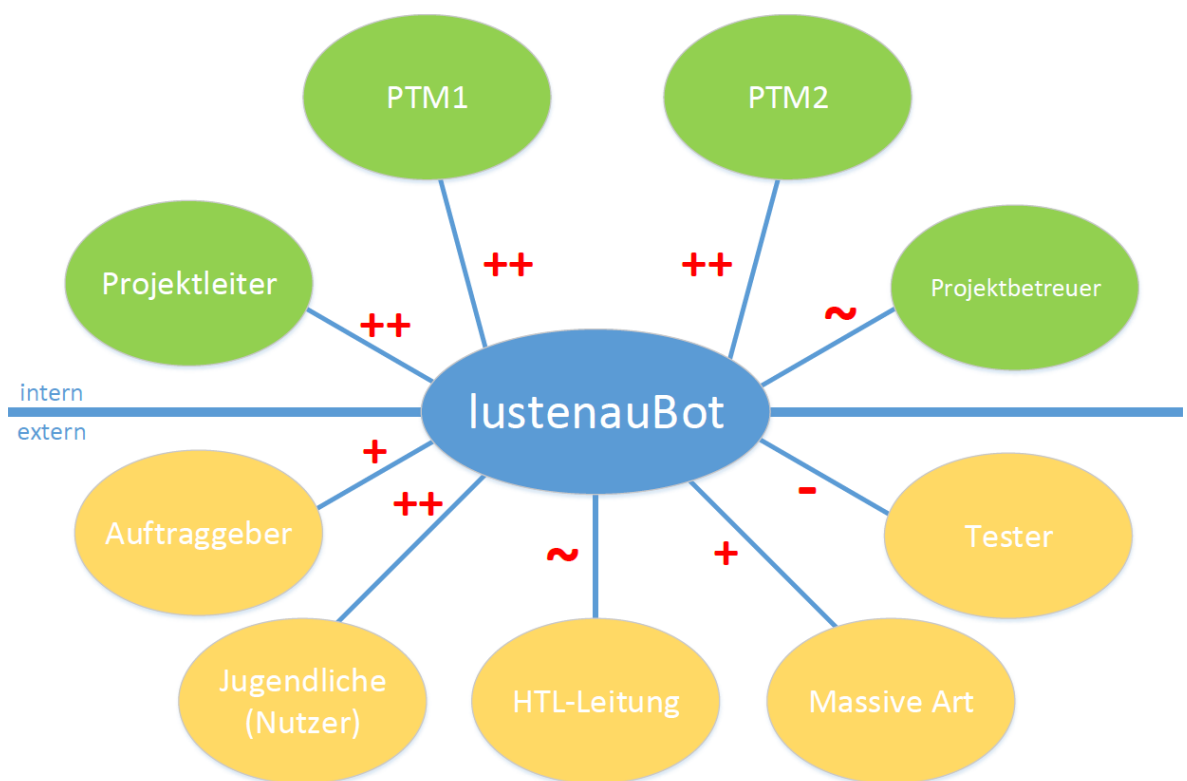


Abb. 9 Projektumweltanalyse

In der folgenden Tabelle werden die Maßnahmen, die den kritischen Umwelten positiv entgegenwirken sollten, beschrieben.

Projektumweltbeziehungen				
Umwelten	Beziehung	Konflikt	Maßnahmen	PSP-Code
Projektbetreuer	Bewertung der Diplomarbeit	Die vom Betreuer erwartete Qualität wird nicht erreicht.	Der Betreuer bekommt die Diplomarbeit vorab, um diese inhaltlich Korrektur zu lesen.	1.4.2
Tester	Testen der einzelnen Funktionen des „Chatbot“	Der „Chatbot“ kann nicht getestet werden, da die erste Entwicklungsphase noch nicht abgeschlossen ist.	Um einen rechtzeitigen Abschluss der Entwicklungsphase sicherstellen zu können, wird ein zeitlicher Puffer eingeplant.	1.3.8
HTL-Leitung	Abgabe der fertigen DA	Verspätete Abgabe der DA → negative Bewertung	Durch regelmäßiges Projektcontrolling (überprüfen der Zeitpläne), soll eine rechtzeitige Abgabe der DA sichergestellt werden.	1.1.4

Tab. 6 Projektumweltbeziehungen

3.10 Projektabschlussbericht

Zur Erstellung des Projektabschlussberichtes wurde am Ende des Projekts ein Projektabschlussmeeting abgehalten, bei welchem die Projektziele auf deren Erreichung bewertet wurden.

3.10.1 Projektverlauf

Das Projekt verlief aus Sicht des Projektteams und des Auftraggebers sehr gut. Bereits zu Beginn des Projektes gab es rasche Fortschritte im Bereich des Projektmanagements und des praktischen Teils. Auch im weiteren Verlauf des Projekts verlief alles nach Plan. Es traten im Projektverlauf keine unvorhergesehenen Ereignisse ein.

3.10.2 Zielerreichung

Es wurden alle Hauptziele erreicht, der Funktionsumfang wurde sogar noch erweitert. Einzig das Zusatzziel „Integration des lustenauBot auf der Homepage der Marktgemeinde Lustenau“ wurde nicht umgesetzt. Nach Absprache mit dem AG wurde beschlossen, dass die Integration erst nach dem Projektabschluss möglich sein wird.

Das Projekt war ein Erfolg, da der „*Chatbot*“ in Zukunft wahrscheinlich veröffentlicht wird. Des Weiteren konnten die Fähigkeiten des Projektteams besonders im Ausführen eines mittleren Projekts ausgebaut werden. Außerdem kann das Projekt als Referenz bei zukünftigen Bewerbungen angegeben werden.

3.10.3 Projektabschlussmeeting

Am Montag den 26.02.2018 fand die Abschlusspräsentation und das Abschlussmeeting mit dem Projektauftraggeber statt. Neben dem AG (Benno Kofler), war auch der Lustenauer Bürgermeister Dr. Kurt Fischer und zwei seiner MitarbeiterInnen bei der Präsentation anwesend. Die genannten Personen waren sichtlich begeistert vom Ergebnis der Diplomarbeit und freuen sich auf die Veröffentlichung des „*Chatbot*“, die für Herbst 2018 geplant ist.

3.10.4 Ausblick (Zeit nach dem Projekt)

Voraussichtlich wird die Marktgemeinde Lustenau den „Chatbot“ übernehmen und im Herbst 2018 für die Lustenauer Jugend öffentlich über Facebook zugänglich machen. Bis dahin bedarf es noch einer genauen Aufstellung, mit welchem zeitlichen bzw. finanziellen Aufwand bei einer Veröffentlichung zu rechnen ist. Außerdem wäre eine Verwendung des „Chatbot“ laut Bürgermeister Kurt Fischer auch für andere Themengebiete denkbar.

Auf dem unten abgebildeten Screenshot ist der Projektabschlussbericht dargestellt:

PROJEKTABSCHLUSSBERICHT		
Gesamteindruck		
<ul style="list-style-type: none">• Projekt wurde erfolgreich abgeschlossen.• Die Projektziele wurden zur vollsten Zufriedenheit des AG erfüllt.		
Reflexion: Realisierung der Projektziele		
<ul style="list-style-type: none">• Projekthauptziele wurden in ausreichendem Maße erfüllt.• Projektnebenziele wurden zum größten Teil erfüllt.• Projektnichtziele wurden eingehalten.		
Zusammenfassende Erfahrungen für andere Projekte		
Die wohl größte Bereicherung für das Projektteam stellte das Planen von Terminen dar. Dabei wurde uns bewusst, wie essentiell ein ausgefeiltes Projektmanagement zum Erfolg beiträgt. Weiters konnten die Projektteammitglieder vertiefende Programmierkenntnisse erwerben, welche bei zukünftigen Projekten von großem Wert sein werden. Alles in allem konnten wir durch die Diplomarbeit eine Vielzahl an Fähigkeiten erwerben bzw. erweitern, die uns auch bei zukünftigen Projekten zugutekommen werden.		
Version: 1	Datum: 26.02.2018	Ersteller: Jakob Ott

Abb. 10 Projektabschlussbericht

4 Analysephase

Im Verlauf der Analysephase wurden die benötigten Softwarekomponenten miteinander verglichen, um auf deren Grundlage eine Auswahl zu treffen.

4.1 Softwarearchitektur

Im folgenden Punkt wird die Softwarearchitektur eines „Chatbot“ näher erläutert.

4.1.1 Begriffsdefinition

Bevor mit der Programmierung und *Implementierung* der Software begonnen werden konnte, musste eine Softwarearchitektur überlegt werden. Grundlegend beschreibt eine Softwarearchitektur die Verbindungen und das Zusammenspiel einzelner Komponenten eines Software-Systems (siehe Abb. 11). (Franco, 2017)

Eine geeignete Architektur ist essentiell für den Erfolg eines Projektes, ist sie doch die Basis für jedes Softwareprojekt. Eine typische Softwarearchitektur eines „Chatbot“ für den Facebook Messenger ist in drei Teile gegliedert (siehe Bild unten):

- Benutzeroberfläche, auf welcher der Benutzer mit dem „Chatbot“ interagiert (User und App/Device)
- Textverarbeitung, welche die Eingaben der Anwenderin für die Weiterverarbeitung übersetzt (API.AI Plattform)
- *WebHook*, welcher externe Dienste anbindet und passende Antworten generiert (Fulfillment)

Weiters wird die Funktion eines „Chatbot“ im Kapitel 4.1.1 beschrieben.

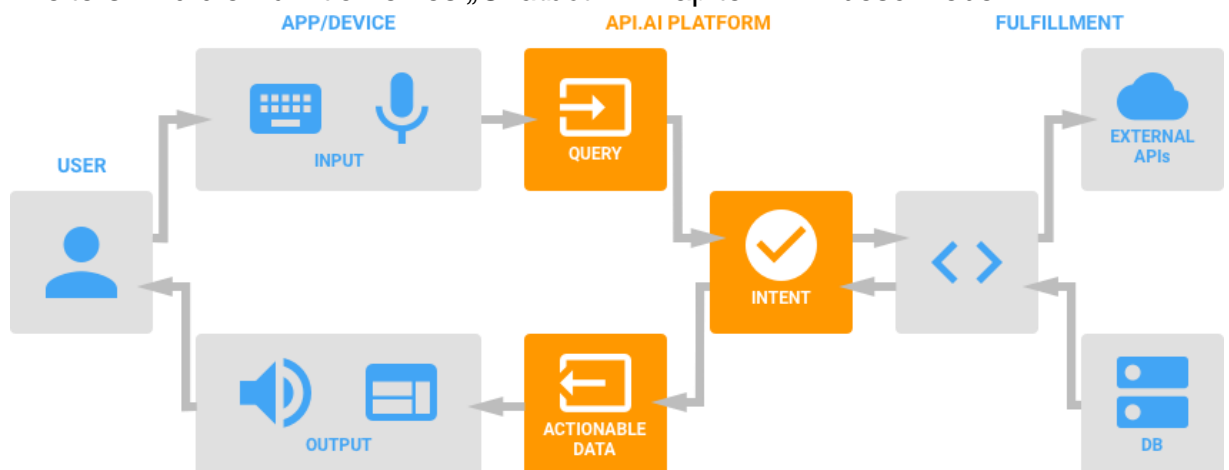


Abb. 11 Typische Softwarearchitektur eines Chatbot [Pluut Interaction B.V.]

4.1.2 Benutzeroberfläche

Aufgrund der Tatsache, dass der „*Chatbot*“ über den Facebook Messenger bereitgestellt und dementsprechend von Facebook *gehostet* wird, bedurfte es lediglich über einen Zugangspunkt zu diesem.

Facebook ermöglicht diesen über die Facebook *Developer Plattform*. Auf dieser wird der „*Chatbot*“ registriert und verwaltet. Die Plattform ist kostenlos verfügbar und ermöglicht den benötigten Zugang zum „*Chatbot*“ mittels der *Implementierung* eines sogenannten „*WebHook*“. (Imura, 2017); (Franco, 2017)

4.2 Facebook Messenger

Der Messenger ist für die Plattform Facebook fast unverzichtbar. Er wird für den Nachrichtenaustausch im sozialen Netzwerk verwendet. Weltweit hatte Facebook im Jahre 2016 rund 1,7 Milliarden Nutzer und der Messenger alleine rund eine Milliarde Nutzer. Im Januar 2017 verwendeten in Österreich rund 3,7 Millionen Nutzer diesen Dienst.

(Roth, 2018) (Statista, 2018)

Facebook stellt außerdem eine „*Developer Plattform*“ für den Messenger bereit. Mit dieser Plattform können „*Chatbots*“ integriert werden, welche auf Nachrichten, Bilder, und andere Medien reagieren können. (Debecker, 2018)

4.3 Textverarbeitung

Im nächsten Schritt wurden die verschiedenen Plattformen zur Textverarbeitung verglichen. Hierbei wurde nicht nach einem einfachen Framework, das selbstständig gehostet werden müsste, sondern nach einer Lösung, die sowohl das Hosting als auch die Textverarbeitung übernimmt, gesucht. Dies ermöglicht eine einfachere und unkompliziertere Entwicklung. Diese Plattformen wurden auf Funktionalität und Einsatzmöglichkeiten untersucht. Ziel dabei war es, ein geeignetes Werkzeug zur Analyse und Übersetzung von Benutzereingaben in Form von Text zu finden. Das Framework sollte selbstständig Benutzereingaben auf festgelegte Schlüsselwörter und Sätze filtern und diese mithilfe maschinellen Lernens richtig zuordnen können. Ziel der Textverarbeitung war es, den Text aus dem Facebook Messenger zu übernehmen, zu verarbeiten und dementsprechend eine Antwort zu generieren.

4.3.1 Dialogflow (ehemals API.ai)

Dialogflow ist eine von Google entwickelte Plattform für Textverarbeitung. Sie zeichnet sich durch einen hohen Funktionsumfang aus und bietet eine eigenes *User Interface* für die Programmiererin. Die Integration von externen Diensten (*WebHooks*) ist einfach und unkompliziert.

Die Plattform wird von Google in deren Cloud *gehostet*, ist jedoch kostenfrei verfügbar. Sie liefert Bibliotheken für alle gängigen Programmiersprachen und Betriebssysteme. Dies resultiert daraus, dass die Plattform die Daten im „Java Script Object Notation-Format“ (*JSON-Format*) retourniert.

4.3.2 Wit.ai

Wit.ai ist eine Plattform für Textverarbeitung, welche von Facebook entwickelt wird. Das Framework stellt ein eigenes *User Interface* für die Entwicklung zur Verfügung, das eine visuelle Darstellung der Konversationen ermöglicht. Darüber hinaus ermöglicht es eine einfache Integration in den Facebook Messenger.

Facebook stellt Entwicklern die Plattform, welche über 50 Sprachen unterstützt, kostenlos zur Verfügung.

4.3.3 Microsoft Bot Framework

Das Microsoft Bot *Framework* ist eine Plattform zur Erstellung eines „*Chatbot*“, in welche das Textverarbeitungswerkzeug Luis integriert ist. Dieses verfügt über ein eigenes User-Interface zur Entwicklung. Die Einbindung externer Dienste ist ebenso wie bei Wit.ai nur erschwert möglich, jedoch nützt Luis ein sehr genaues und treffsicheres Modell zur Textverarbeitung.

4.3.4 Vergleich zwischen den Plattformen

In der folgenden Tabelle werden drei unterschiedliche Bot-Plattformen aufgrund ihrer Vor- und Nachteile verglichen.

Lösung	Vorteile	Nachteile
Dialogflow	<ul style="list-style-type: none"> • <i>WebHook</i> von Textverarbeitung getrennt • Benutzerfreundlichkeit • starker <i>Algorithmus</i> 	<ul style="list-style-type: none"> • Diagnosewerkzeuge in Betaphase
Wit.ai	<ul style="list-style-type: none"> • Direkte Integration in Facebook Messenger • Visuelle Darstellung der Konversationen 	<ul style="list-style-type: none"> • Unzureichende Dokumentation
Microsoft Bot Framework (Luis)	<ul style="list-style-type: none"> • Hervorragende Diagnosewerkzeuge • starker <i>Algorithmus</i> 	<ul style="list-style-type: none"> • Unausgereiftes <i>User Interface</i> • Geringe Verbreitung

Tab. 7 Plattformenvergleich

(Kang, 2018); (Unbekannt, 2018); (Maruti Techlabs, 2018); (Nikita, 2018); (Ring, 2018)

4.3.5 WebHook

Als letzten Schritt bei der Architektur musste überlegt werden, wie externe Dienste abgefragt werden können. Das Prinzip dabei ist wie folgt: Der Benutzer schreibt etwas im Messenger, dieser Text wird dann durch den *WebHook*, welcher in der *Developer Plattform* registriert wurde, an die Textverarbeitung weitergeleitet. Diese wertet die Anfrage aus und schickt sie gegebenenfalls an den *WebHook* weiter. Dieser muss eigenständig gehostet werden. Aufgrund bereits vorhandener Erfahrungswerte ist die Entscheidung auf Heroku gefallen. Diese Plattform ermöglicht kostenloses *Hosting* für Python-Programme.

Heroku stellt ebenfalls ein direktes *Deployment* von *GitHub-Banches* bereit. Dies bedeutet, dass das Programm jedes Mal neu geladen wird, wenn sich der Code, der auf GitHub hochgeladen wird, ändert. Aus diesem Grund wurde GitHub für die Codeverwaltung ausgewählt. Ein weiterer Grund pro Heroku war die exzellente Dokumentation, die die Plattform Dialogflow für die Entwicklung eines *WebHook* bereitstellte. (Heroku, 2018); (xVir, 2018); (Mahanoor, 2018)

4.4 Konzeption

Um die Themengebiete, welche der „*Chatbot*“ abdecken sollte, abgrenzen zu können, wurde ein Konzept ausgearbeitet, in welchem die einzelnen Bereiche festgelegt wurden.

4.4.1 Allgemein

Das Prinzip des „*Chatbot*“ besteht darin, vom Nutzer gesendete Nachrichten/Anfragen zu verarbeiten und eine passende Rückgabe zu liefern. Anfragen werden zuerst vom *Textverarbeitungsframework* einem *Intent* zugewiesen. Im *Intent* wird anschließend differenziert, ob es sich um statische Daten handelt, d.h. ob die Rückgabe direkt vom *Intent* erfolgt, oder ob ein *WebHook* aufgerufen wird. Nötigenfalls kann ein *Intent* noch mit einem Kontext versehen werden. Ein *Intent* benötigt einen Kontext, wenn er durch einen anderen *Intent* aufgerufen werden kann.

4.4.2 Themenabdeckung des „*Chatbot*“

Die Themengebiete wurden auf jugendrelevante Themengebiete in Lustenau eingegrenzt. Im genaueren Sinne Themengebiete wie:

- Informationen und Öffnungszeiten zu Bars, Restaurants und Cafés
- Sport- und Freizeitangebote (Kino, Veranstaltungen)
- Informationen über das Rathaus und dortige Dienste
- Kontaktinformationen zur „Offenen Jugendarbeit in Lustenau“
- Wetterinformationen
- Abfahrtszeiten von Bussen
- Witze erzählen
- Informationen über das Parkbad in Lustenau (Öffnungszeiten und Eintrittspreise)
- Sehenswürdigkeiten in Lustenau
- Fakten über die Marktgemeinde Lustenau
- Die Fähigkeit, Smalltalk mit der Anwenderin führen zu können

Diese Themengebiete sollten dem Benutzer das Leben in Lustenau erleichtern. Ein Beispiel hierfür wäre die Suche nach Freizeitangeboten. Da im Internet nicht direkt nach Freizeitangeboten in Lustenau, sondern nur nach etwas Spezifischem gesucht werden kann, ist es durch den „lustenauBot“ möglich, sich dafür Vorschläge bringen zu lassen. Auch das Zusammensuchen von Informationen wird durch den „Chatbot“ um einen beträchtlichen Teil verringert. Es muss lediglich eine Frage an den „Bot“ gestellt werden, welcher diese Informationen, falls *implementiert*, durch eine einfache Rückgabe wiedergeben kann. Für genauere Daten wird meist noch die „Uniform Resource Locator“ (*URL*) einer entsprechenden Webseite vorgeschlagen, auf die sich der Nutzer begeben kann, wenn ihm die bereitgestellten Informationen nicht ausreichen.

4.4.3 Konversationsgestaltung

Die Konversation zwischen dem „Chatbot“ und der Anwenderin sollte möglichst nett und menschlich wirken. Da der „Bot“ vor allem für Jugendliche ausgelegt ist, sollte man sich auch auf einem persönlichen Level mit dem „Bot“ identifizieren können. Es wurde deshalb sehr darauf Wert gelegt, die Rückgaben des „Bots“ mit *Emoticons* auszuschnücken und auch die Sätze an sich auf eine lustige Art und Weise zu formulieren. So sollte ein Kontakt zum Benutzer hergestellt werden und nicht einfach nur eine Aufzählung nackter Fakten.

4.5 Use-Case-Diagramm

Das Use-Case-Diagramm verschafft dem Betrachter einen Überblick über die Funktionalität der Software. Es zeigt die Anwendungsfälle, die bei der Benützung der Software auftreten, sowie die einzelnen Akteure (die einzelnen Benutzerrollen), welche die einzelnen Anwendungsfälle betreffen (siehe Abb. 12).

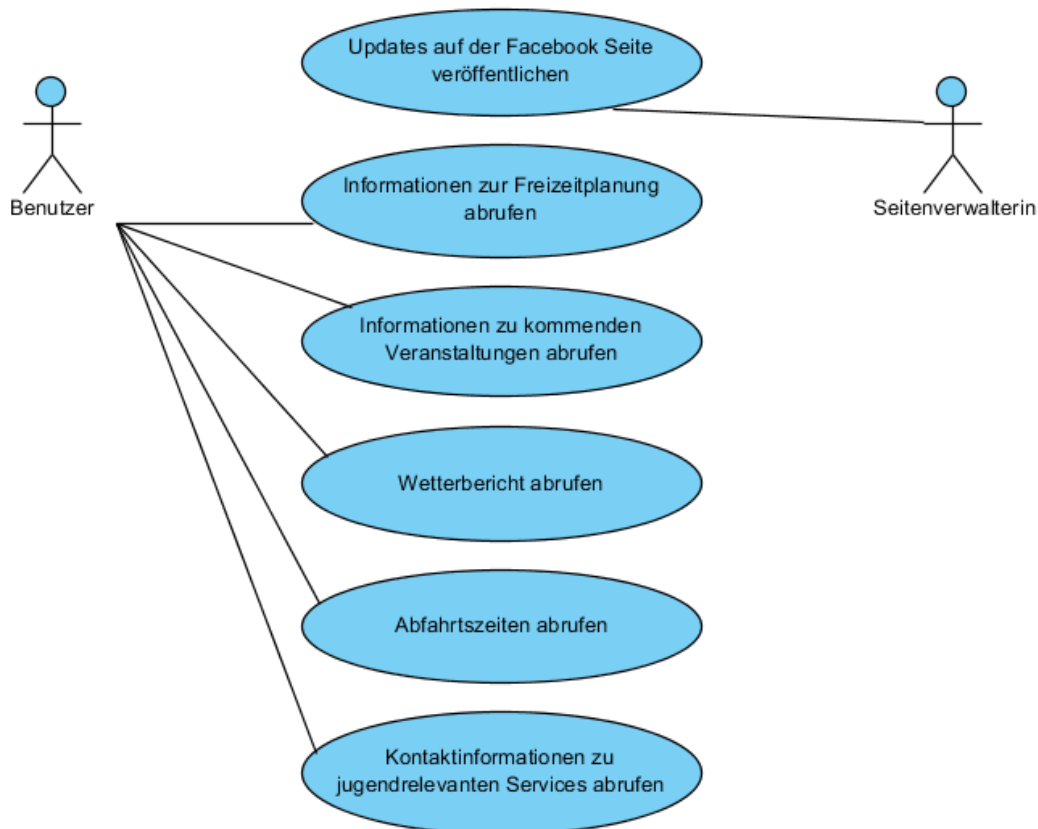


Abb. 12 Use-Case-Diagramm

(Unbekannt, Sparx Systems Central Europe, 2017)

4.6 Datenquellen

Für die Bereitstellung von sich häufig ändernden/dynamischen Informationen wurden externe Datenquellen, bzw. „API“ oder externe Datenbanken verwendet. Für statische Daten (z.B. Öffnungszeiten) wurden die nötigen Informationen von den entsprechenden Webseiten ausgelesen.

4.6.1 Statische und dynamische Daten

Grundsätzlich lässt es sich zwischen statischen und dynamischen Daten unterscheiden. Statische Daten verändern sich kaum bis gar nicht. Daher können sie „hart-codiert“, bzw. fix in das Repertoire des „Chatbots“ eingebaut werden. Ein Beispiel für solche statischen Daten wären Öffnungszeiten in der Gastronomie. Öffnungszeiten von Bars und Restaurants werden nur sehr selten bis gar nicht geändert und eignen sich daher für eine fixe *Implementierung*.

Im Gegensatz dazu stehen dynamische Daten. Dynamische Daten weisen eine hohe Änderungsrate auf und können daher nicht fix *implementiert* werden, da sie nach kurzer Zeit schon wieder veraltet wären. Um nun die Umstände zu umgehen, häufig sich ändernde Daten jedes Mal manuell ändern zu müssen, verwendet man *Implementierungen*, welche die Daten automatisch aus den zugehörigen Datenbanken selektieren. Dies kann durch einen direkten Zugang auf die Datenbank gewährleistet werden oder durch eigens dafür erstellte „APIs“, welche die nötigen Daten in Echtzeit zur Verfügung stellen. Ein Beispiel hierfür wäre die Bereitstellung von Informationen über aktuelle Veranstaltungen. Da diese Veranstaltungen zeitabhängig sind, fallen vergangene Veranstaltungen weg und neue kommen hinzu. Deshalb ist es hier sinnvoll, die Informationen aus einer Datenbank herauszulesen, anstatt diese immer wieder manuell zu aktualisieren. Voraussetzung dafür ist jedoch eine zentrale Datenverwaltung, die auch gewartet und mit aktuellen Daten versehen wird.

4.6.2 Abfahrtszeiten

Für die Abfahrtszeiten der Busse an den Haltestellen wurde mit dem Verkehrsverbund Vorarlberg GmbH (VVV) Kontakt aufgenommen. Da jedoch keine öffentliche Schnittstelle zur Verfügung steht, wurden die aktuellen Daten direkt von der Webseite <http://www.abfahrtszeiten.at> ausgelesen.

4.6.3 Veranstaltungen

Für Informationen zu aktuellen Veranstaltungen in Lustenau wurde mit der Massive Art Web Services GmbH Kontakt aufgenommen. Diese Firma wurde von der Marktgemeinde beauftragt, eine neue Webseite für die Marktgemeinde Lustenau zu erstellen und hatte deshalb eine Datenbank, unter anderem zur Verwaltung von Veranstaltungen, aufgesetzt. Nach einer Einführung und dem Erhalt eines Zugangs konnte auf die Datenbank zugegriffen werden, um folgende Informationen auszulesen:

- Veranstaltungsname
- Veranstaltungsort
- Veranstaltungsdatum
- Beginn der Veranstaltung
- Eintrittspreise

4.6.4 Wetterinformationen

Das Abrufen aktueller Wetterinformationen fällt ebenfalls unter die Thematik „dynamische Daten“. Die Wetterdaten können sich täglich oder auch stündlich ändern, was daher eine automatische *Implementation* erfordert. Ziel war es, Informationen zum Wetter am aktuellen Tag sowie am Tag darauf abrufen zu können. Die Daten wurden von der „YahooWeatherAPI“ abgerufen und anschließend aufbereitet. Es wurde diese „API“ ausgewählt, da bereits in der Dokumentation des Textverarbeitungs-„Frameworks“ „Dialogflow“, einige Beispiele mit der Verwendung dieser „API“ dokumentiert sind. Es stellte sich jedoch in späteren Tests heraus, dass der Zugriff auf die Daten bzw. der Wetterservice oftmals nicht erreichbar waren. Somit konnten keine Informationen abgerufen werden und der „Chatbot“ lieferte durch den „Fallback-Intent“ eine Fehlermeldung zurück. Da dieser Service den Zweck somit nicht oder nur sporadisch erfüllte, wurde der Wetterinformationsanbieter adaptiert. Es wurde zum Anbieter „OpenWeatherMap“ gewechselt. Die Verfügbarkeit der Informationen wurde dadurch deutlich erhöht und somit verbessert.

Ein Punkt, um den sich noch gekümmert werden musste, war die Übersetzung von Englisch auf Deutsch. Da es ein englischer Anbieter ist, der die Daten zur Verfügung stellt, werden die Wetterinformationen ausschließlich in Englisch bereitgestellt. Um dies zu lösen, mussten die einzelnen Phrasen, welche in einer Rückgabe stehen, übersetzt werden. Dabei wurde die Rückgabe von einem Ersetz-*Algorithmus* überprüft, der dann die englischen Phrasen durch die deutsche Übersetzung ersetzt hat.

4.6.5 Andere Datenquellen

Bei Informationen aus „anderen Quellen“ sind in diesem Bezug statische Daten gemeint. Die Informationen wurden meist von den entsprechenden Webseiten übertragen. Das heißt, die Daten wurden manuell von der Seite ausgearbeitet und in den „Bot“ durch das Erstellen eines neuen *Intent* (siehe Kapitel Einarbeitung → Dialogflow → 5.1.5 Erstellen eines *Intent*) *implementiert*. Ein Beispiel hierfür wären die Öffnungszeiten des Parkbads in Lustenau sowie Preisinformationen zu den Eintrittskarten. Auch z.B. die Informationen über die offene Jugendarbeit Lustenau, diverse Öffnungszeiten und andere Informationen verschiedener Angebote sind von den entsprechenden Webseiten oder von der „Homepage“ der Marktgemeinde Lustenau. Diese Art der Bereitstellung der Informationen erfordert keine Programmierung und ist daher die einfachste und schnellste Art der Vergrößerung des Informationsangebotes des „*Chatbot*“.

5 Einarbeitung

Das folgende Kapitel beschreibt die Phase der Einarbeitung, in welcher die grundlegende Funktion der einzelnen Technologien erlernt wurde. Weiters wurde in dieser Phase die Grundlage für den „Chatbot“ geschaffen.

5.1 Messenger API

Die „Messenger API“ ermöglicht es, Nachrichten und Medien, die vom Benutzer in das Chatfenster eingetragen werden, abzufangen und auszuwerten. Ist die Auswertung der Nachricht erfolgt, kann der „API“ dann auch die passende Antwort übergeben werden. Die Kommunikation mit der „API“ findet im Format der „JavaScript Object Notation“ („JSON“) statt. Dabei ist von Facebook eine gewisse Struktur dieses Formats vorgegeben. Die „Facebook Messenger API“ ist mit dem Textverarbeitungs-„Framework“ „Dialogflow“ verbunden und leitet automatisch jede Nachricht an dieses weiter. In den Funktionen des „WebHook“ wird grundsätzlich jedoch nie mit der „Messenger API“ direkt kommuniziert. Die Funktionen erhalten die Anfragen im „JSON“-Format von „Dialogflow“ und senden dann auch die entsprechende Antwort wieder an „Dialogflow“ im vorgegebenen „JSON“-Format zurück. Das „Framework“ leitet die Antwort dann im selben Format zurück an den „Messenger“, der dann schließlich aus der Formatierung den Antworttext herausliest und der Anwenderin präsentiert.

Die „Messenger API“ ist somit die zentrale Schnittstelle zwischen dem Benutzer und der ganzen Verarbeitung der Nachricht für eine passende Antwort. In Abbildung 13 ist die komplette Verarbeitung einer Anfrage nummeriert dargestellt.

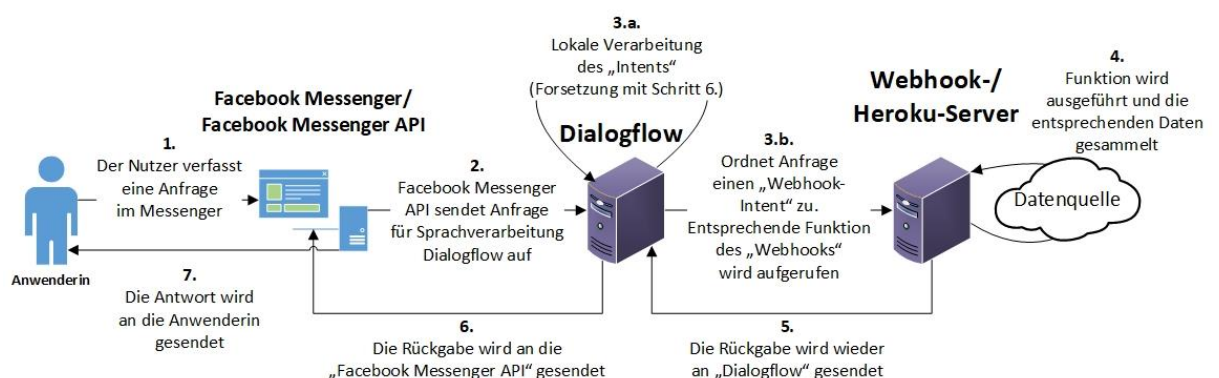


Abb. 13 Allgemeine Anfragenverarbeitung des „Chatbot“

5.2 Dialogflow

Im folgenden Teil wird die von Google entwickelte Plattform Dialogflow erklärt. Weiters wird die Funktionsweise des „*Chatbot*“ erläutert.

5.2.1 Verbinden von Dialogflow mit der Facebook

Messenger API

Um die mit der Entwicklung des „Facebook-*Chatbot*“ mit Dialogflow überhaupt erst beginnen zu können, musste „Dialogflow“ mit der „Messenger API“ von Facebook verbunden werden. Der bereits erhaltene „*Page Access Token*“ (siehe Kapitel 5.5.2 „Facebook Developer Console“) von der „Facebook Developer Console“ wird benötigt, um die Verbindung herstellen zu können. In der „Dialogflow“-Oberfläche gibt es die Option „Integrations“, bzw. die Option „Integrationen“. Hier kann ausgewählt werden, auf welchen Plattformen der „*Chatbot*“ operieren soll. In diesem Fall wird die Option „Facebook Messenger“ ausgewählt. Im aufkommenden Menü wird nun der besagte „*Page Access Token*“ in das entsprechende Feld kopiert und anschließend auf den Knopf „Start“ gedrückt (siehe Abb. 14). Somit ist eine Verbindung mit „*Framework*“ und der entsprechenden Facebook *Profilseite* erstellt worden. Nun kann der „*Chatbot*“ beliebig ausgebaut und Anfragen können verarbeitet werden. (Unbekannt, Dialogflow, 2017)

Facebook Messenger

Create and teach a conversational bot for Facebook Messenger.

After you design and test your Dialogflow agent, you can launch your Messenger bot

1. Get your Facebook Page Access Token and insert it in the field below.
2. Create your own Verify Token (can be any string).
3. Click 'START' below.
4. Use the Callback URL and Verify Token to create an event in the Facebook Messenger Webhook Setup.

[More in documentation.](#)

Callback URL

Verify Token

Page Access Token

STOP

Abb. 14 Eingabe des „*Page Access Token*“ in „Dialogflow“

5.2.2 Intents und Zuweisung einer Rückgabe

Ein *Intent* kann mit der Absicht des Benutzers bzw. mit dessen erwarteter Antwort übersetzt werden. Sie stellen die erwarteten Rückgaben für die Anwenderin da. Ein *Intent* enthält Beispielphrasen mit Schlagwörtern, die ein Nutzer senden könnte. Bei einer Anfrage wird dann die geschriebene Nachricht der Anwenderin mit den Beispielphrasen bzw. Schlagwörtern verglichen und einem zugehörigen *Intent* zugeordnet, welcher dann die passende Antwort enthält.

Dabei kann grundsätzlich von drei Arten/Konzepten der *Intents* differenziert werden:

1. *Intents*, welche direkt eine Rückgabe liefern
2. *Intents*, welche einen *WebHook* aufrufen, um die nötigen Informationen für die Rückgabe zu erhalten
3. „Fallback-Intent“, welcher eine Standardrückgabe im Falle eines Fehlers auslöst

Eine Anforderung des Benutzers kann immer einem dieser *Intents* zugeordnet werden. *Intents*, die der Anwenderin direkt eine Rückgabe liefern, sind solche, bei denen sich die Daten nicht bis sehr selten ändern (z.B. Öffnungszeiten, Rückgaben für Smalltalk und dergleichen). Ein sogenannter „Fallback-Intent“ ist eine Standardrückgabe, falls eine Anfrage keinem der anderen beiden „*Intents*“ zugeordnet werden kann. Dieser enthält eine Standardrückgabe, um den Benutzer zu informieren, dass der Anfrage keine Antwort zugeordnet werden konnte bzw. dass die angeforderte Information noch nicht in den „*Chatbot*“ implementiert wurde.

5.2.3 Intents mit Kontext

Beim Erstellen eines neuen *Intents* kann auch auf einen vorherigen Zustand verwiesen werden. Dabei kann dieser Zustand abhängig vom vorherigen Kontext aufgerufen werden und entsprechend reagieren. Ein Beispiel hierfür wäre der „Witze-Intent“. Dabei wird auf Anfragen der Anwenderin durch den ersten „*Intent*“ ein zufälliger Witz zugesendet. Anschließend wird die Anwenderin gefragt, ob sie gerne noch einen Witz lesen möchte. Antwortet die Anwenderin mit „Ja“, wird im Hintergrund vom ersten „*Intent*“ die Antwort „Ja“ mit dem entsprechenden Kontext an den zweiten „*Intent*“ weitergeleitet. Dieser wird nur dann aufgerufen, wenn er vom vorherigen „*Intent*“ einen Kontext übergeben bekommt.

5.2.4 Namenskonvention von Intents

Um die „*Intents*“ einheitlich und nachvollziehbar zu bezeichnen, wurde sich auf eine Namenskonvention geeinigt, nach der diese benannt werden mussten. Die Namensgebung setzte sich aus folgenden Teilen zusammen:

1. Das Thema/Themengebiet, beginnend mit einem kleinen Buchstaben
 - a. Das Thema kann aus mehreren Teilbereichen bestehen, die immer vertiefter werden.
 - b. Die einzelnen Teilbereiche sind jeweils durch einen Bindestrich („-“) getrennt
2. Trennung des Themas und Funktion durch einen (weiteren) Bindestrich („-“)
3. Die Funktion des „*Intents*“
 - a. Funktionen mit mehreren Wörtern werden mit einem Unterstrich („_“) getrennt

Allgemein waren Großbuchstaben sowie Umlaute und das scharfe S („ß“), nicht erlaubt.

5.2.4.1 Beispiele

freizeitangebot-sport

freizeitangebot-sport-fussball

smalltalk-about_me

5.2.5 Erstellen eines Intents

Es gibt drei wesentliche Punkte, aus denen ein „*Intent*“ besteht:

1. Name des „*Intents*“ – muss den Konventionen entsprechen
2. Beispiel Phrasen – um Schlagwörter herauszufiltern, um diese dann einer Anfrage zuzuordnen
3. Eine Rückgabe – dies kann ein einfacher Text sein, ein Bild, oder es können so genannte „*Quick-Replies*“ zurückgegeben werden. „*Quick-Replies*“ sind Rückgaben, die vom Nutzer als „*Button*“ oder Knopf verwendet werden können. Die Anwenderin kann so einen dieser „*Buttons*“ auswählen, der dann automatisch den Inhalt bzw. den Namen des Knopfes in den „Messenger“ schreibt und so eine Antwort hervorruft (siehe Abbildung 15 und Abbildung 16). Es können auch so genannte „*WebHooks*“ (siehe 1.1.5 „*WebHooks*“) als Rückgabe verwendet werden.

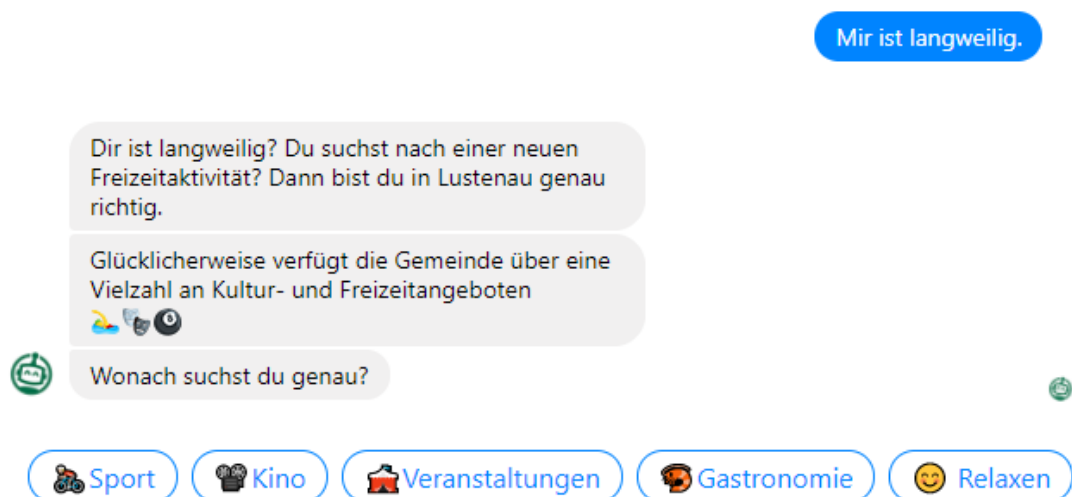


Abb. 15 Anfrage mit „*Quick-Replies*“ als Antwort

Mir ist langweilig

Dir ist langweilig? Du suchst nach einer neuen Freizeitaktivität? Dann bist du in Lustenau genau richtig.

Glücklicherweise verfügt die Gemeinde über eine Vielzahl an Kultur- und Freizeitangeboten



Wonach suchst du genau?

Sport

Sport in Lustenau: Jetzt gibt es keine Ausreden mehr!




 **Baden**

Parki, Bruggi, Alter Rhein



[Genauerer zu Baden](#)



 **Fußball**

Jugendplatz (Habedere), Rheinvorland, ...

[Genauerer zu Fußball](#)

Abb. 16 Anfrage mit ausgewähltem „Quick-Reply“ Knopf „Sport“

5.2.6 WebHook

WebHooks werden von „*Intents*“ aufgerufen, bei denen sich die Daten häufig und/oder regelmäßig ändern (z.B. Abfahrtszeiten von öffentlichen Verkehrsmitteln). Das Prinzip der *WebHooks* funktioniert so, dass eine Anfrage eine Funktion des „Sourcecodes“ auf einem (Online-) Server ausführt. Diese Funktion arbeitet dann die nötigen Sequenzen ab (z.B. Verbindung zu einer Datenbank und selektieren der benötigten Daten). Der *WebHook* sendet die Rückgabe dann an die „Facebook Messenger API“.

In der folgenden Grafik ist der allgemeine Nachrichtenfluss beim Aufruf eines *WebHook* beschrieben:

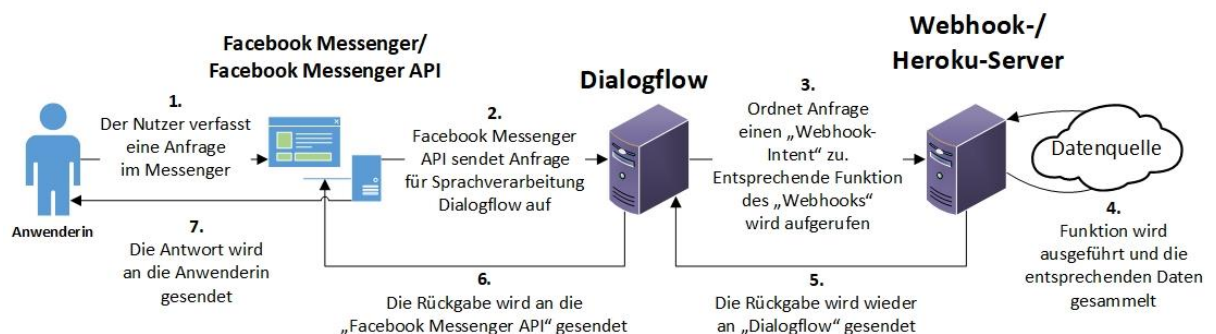


Abb. 17 Nachrichtenfluss bei WebHook Aufruf

Als ersten Schritt stellt der Nutzer eine Anfrage an den „*Chatbot*“, bei der ein *WebHook* benötigt wird. Ein Beispiel hierfür wäre eine Anfrage über das aktuelle Wetter oder Veranstaltungen. Diese Anfrage wird an den Facebook Messenger gesendet und dann von dort an die Textverarbeitung weitergeleitet. „*Dialogflow*“ erhält nun die von der Anwenderin gestellte Anfrage und ordnet diese dem entsprechenden „*Intent*“ zu. Aufgründessen wird die entsprechende Funktion des *WebHook* auf dem Heroku-Server aufgerufen und verarbeitet. Die Funktion ruft nun z.B. die entsprechende „*API*“, wie z.B. „*OpenWeatherMap*“ auf, oder stellt eine Datenbankverbindung her, um zum Beispiel Veranstaltungen auszulesen. Sind die Daten nun selektiert und gegebenenfalls aufbereitet, werden diese am Ende der Funktion wieder an das „*Framework*“ „*Dialogflow*“ zurückgesendet. Das „*Framework*“ leitet die Antwort dann wiederum an den „Facebook Messenger“ weiter, der die Antwort dann dem Nutzer präsentiert.

5.2.7 Entities

„Entities“ sind gewisse Schlüsselwörter, die für eine speziell abgestimmte Rückgabe nötig sind. Somit sind „Entities“ Synonyme für ein Wort, das später für die Antwort von Bedeutung ist. Ein Beispiel hierfür wäre die Abfrage von Abfahrtszeiten an einer Haltestelle. Um die richtige Busverbindung und die richtige Abfahrtszeit zu bestimmen, muss vorher die Haltestelle bekannt sein. Daher muss die Haltestelle aus der Anfrage herausgefiltert werden (siehe Abbildung 19). Um dies wiederum zu bestimmen, muss eine „Entity“, bzw. ein Objekt erstellt werden. In diesem Objekt werden die verschiedenen Haltestellen bzw. auch Synonyme für manche Haltestellen erfasst, um sie dann später der Funktion, die zur entsprechenden Haltestelle die passende Busverbindung sucht, zu übergeben. Ein „Intent“ kann mehrere „Entities“ haben um Schlüsselwörter aus der Anfrage herauszulesen. Die „Entities“ können auch von mehreren „Intents“ verwendet werden, sofern gewünscht ist, die gleichen Schlüsselwörter in einem anderen „Intent“ nochmals auszulesen. Wie bereits im vorherigen Beispiel erwähnt, können „Entities“ auch selber erstellt werden. Es können aber auch die bereits vorgefertigten Objekte verwendet werden. Ein Beispiel dafür wären Uhrzeiten, Datum, Orte, Zahlen und Währungen.

» abfahrtszeiten bei der haltestelle lustenau gasthaus austria

Abb. 18 Beispielsatz mit markierten "Entities"

REQUIRED	PARAMETER NAME	ENTITY	VALUE
<input type="checkbox"/>	geo-city	@sys.geo-city	\$geo-city
<input checked="" type="checkbox"/>	bushaltestellen	@bushaltestellen	\$bushaltestellen
<input type="checkbox"/>	Enter name	Enter entity	Enter value

Abb. 19 Darstellung der in der obigen Abb. 17 erkannten „Entities“

5.3 Git

Für die Versionskontrolle der Software wurde Git verwendet. Eine Versionskontrolle wie Git ermöglicht es, den Überblick über geänderten Code zu bewahren.

Während der Entwicklungsphase wurden drei *Branches* verwendet. Eine „*Branch*“ ist wie eine Kopie des Programmcodes, auf welcher Änderungen getestet und neue Funktionen programmiert werden können (siehe Abb. 20).

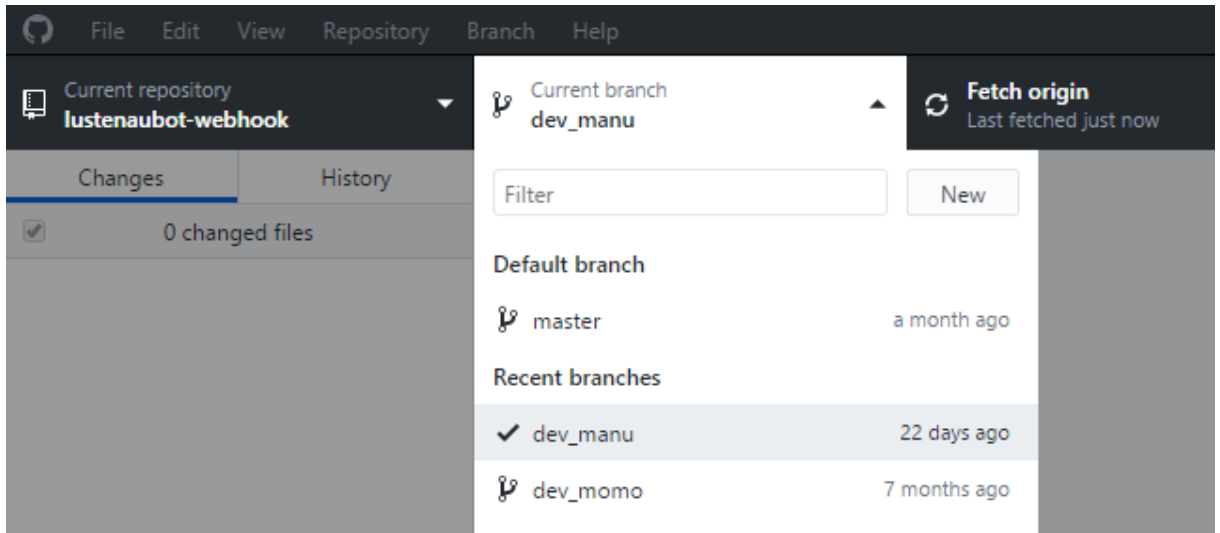


Abb. 20 Oberfläche von GitHub-Desktop inkl. Anzeige der einzelnen Branches

Der funktionierende Programmcode auf der „*master-Branch*“ wird dadurch nicht beeinflusst, sollten Fehler während der Entwicklung in den anderen *Branches* auftreten. Erst, wenn alle Fehler beseitigt sind, wird der neue Code zusammengeführt.

History for [lustenauBot-webhook](#) / `app.py`

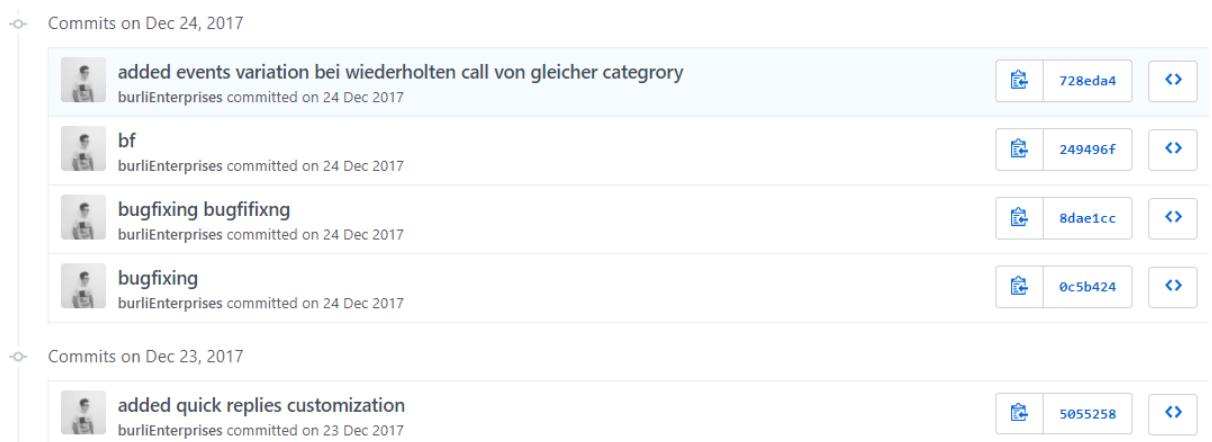


Abb. 21 Historien-Ansicht in GitHub

Auf zwei Entwicklungs-*Branches* wurde die Software weiterentwickelt, Tests gemacht und der Code verbessert. Waren diese Tests und Verbesserungen sinnvoll und fehlerfrei, wurden sie auf die „master“-*Branch* „merged“. „Mergen“ bedeutet, dass die funktionierenden Änderungen, welche der Entwickler auf seiner eigenen *Branch* gemacht hat, übernommen werden. Dies geschah jedoch nicht allzu oft, um eine Übersicht über den Fortschritt zu behalten. Alle Änderungen können in der sogenannten Historien-Ansicht betrachtet werden (siehe Abb. 21).

GitHub ist eine Webseite, welche das *Hosting* von Git-Projekten ermöglicht. Für das Updaten des *gehosteten* Codes (Git-Push) bzw. der lokalen Kopie (Git-Pull) wurde eine grafische Benutzeroberfläche verwendet. GitHub stellt dafür eine eigene Desktopanwendung bereit, GitHub Desktop. (Cooper, 2017)

5.4 Python

Im folgenden Abschnitt wird die Entwicklung des *WebHook* in Python betrachtet.

5.4.1 Struktur

Der *WebHook* wurde in Python 2.7 entwickelt. Der Grund hierfür war die bereits vorhandene Dokumentation auf der Entwickler-Plattform von Dialogflow. Grundsätzlich bedurfte es dreier Dateien zur Lauffähigkeit und *Implementierung* des Programmes:

- requirements.txt
 - In dieser Datei standen alle Bibliotheken, welche verwendet wurden. Die Bibliotheken sind hier untereinander nach dem Paketnamen angeordnet. Dies war nötig, da sie nur so vom *Paketmanager* „pip“ auf Heroku installiert werden konnten.

```
6 lines (5 sloc) | 88 Bytes
1 Flask==0.10.1
2 future==0.16.0
3 beautifulsoup4==4.6.0
4 lxml==3.2.5
5 mysqlclient==1.3.12
```

Abb. 22 Auflistung aller Pakete

- Procfile
 - Das „Procfile“ hat kein spezielles Dateiformat. In der Datei steht lediglich der Ort, an welchem das Python-Programm liegt und dass dieses auch ausgeführt werden soll.

```
1 lines (1 sloc) | 18 Bytes
1 web: python app.py
```

Abb. 23 Inhalt des Profils

(Myers, 2018)

- app.py
 - In dieser Datei steht der Programmcode (siehe Abb. 24). Alle Funktionen zum Abruf externer Daten sowie das Handling von Anfragen und Antworten sind hier gelistet.

1006 lines (921 sloc) | 45.4 KB

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  from __future__ import print_function
5  from future.standard_library import install_aliases
6  install_aliases()
7
8  from urllib.parse import urlparse, urlencode
9  from urllib.request import urlopen, Request
10 from urllib.error import HTTPError
11
12 import json
13 import os
14 import sys
15 import logging
16 import MySQLdb
17 import time
18
19 from bs4 import BeautifulSoup
20 from datetime import datetime
21
22 from flask import Flask, render_template
23 from flask import request
24 from flask import make_response
25
26 # Flask app should start in global layout
27 app = Flask(__name__)
28
29 app.logger.addHandler(logging.StreamHandler(sys.stdout))
30 app.logger.setLevel(logging.ERROR)
31 #des isch super
32
33 @app.route('/webhook', methods=['POST'])
34 def webhook():
35     req = request.get_json(silent=True, force=True) #req = anfrage json
36     print("Request:")
37     print(json.dumps(req, indent=4))
```

Abb. 24 Ausschnitt aus der app.py Datei

5.4.2 Python-Lauffähigkeit

Um den *WebHook* lauffähig zu machen, bedarf es lediglich des Quellcodes der Anwendung sowie eines Dependency-Files, in welchem alle Python-Bibliotheken angegeben sind, welche im Programmcode verwendet werden. Diese Datei muss „requirements.txt“ heißen.

1 lines (1 sloc) | 18 Bytes

```
1  web: python app.py
```

Abb. 25 Inhalt der Procfile-Datei

Ebenfalls bedarf es eines „Procfile“-Files. In diesem steht der Befehl, welcher von Heroku zu Beginn ausgeführt werden soll (Programm starten). (logicline, 2018)

5.4.3 Web-App-Konfiguration mit Flask

Das Python-Programm musste immer laufen und auf Anfragen warten. Diese Anfragen kamen an eine „URL“, die in Dialogflow definiert wurde (siehe Abb. 26).

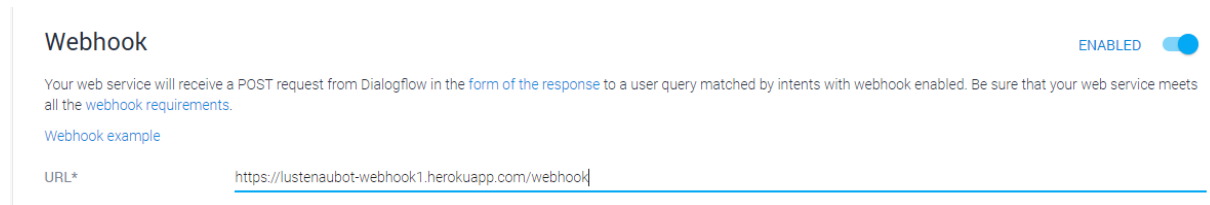


Abb. 26 WebHook-Konfiguration in Dialogflow

Damit in Python auf solche Anfragen reagiert werden kann, muss eine Web-App entwickelt werden. Aus diesem Grund wurde das Micro-Framework „Flask“ eingesetzt. Dieses bietet einige Vorteile:

- Einfache Nutzung
- Exzellente Dokumentation
- Einfaches Mapping von Routen

Die Installation der Bibliothek erfolgte durch das requirements.txt-File.

```
22 from flask import Flask, render_template
23 from flask import request
24 from flask import make_response
25
26 # Flask app should start in global layout
27 app = Flask(__name__)
28
29 app.logger.addHandler(logging.StreamHandler(sys.stdout))
30 app.logger.setLevel(logging.ERROR)
31 #des isch super
32
33 @app.route('/webhook', methods=['POST'])
34 def webhook():
```

Abb. 27 Implementierung von Flask

Nach dem Import der Bibliothek (siehe Bild) kann im Programmcode mittels „@app.route“ der Pfad (die Route) der Applikation angegeben werden. In der Fachsprache wird dies auch als „Mapping“ bezeichnet. Dieser Pfad ist wichtig, da genau dieser in Dialogflow angegeben wird. (Python Tutorials, 2018); (Flask, 2018)

Flask wird im Programmcode wie eine Klasse aufgerufen (Abb. 27, Zeile 27). Das Objekt, das hier initialisiert wird, kann dann in weiterer Folge Konfigurationen vornehmen.

5.4.4 Von der Anfrage zur Antwort

Nach der Konfiguration der Route und der Installation von Flask stellte sich die Frage, wie der Anfrage die richtige Funktion zugewiesen werden konnte. Dies geschieht zu Beginn des Flask-Programms. Die Anfrage der Anwenderin erfolgt mittels *JSON*. In dieser *JSON*-Datei steht u.a. der Name der sogenannten „Action“. Dieser Name wurde zuvor in Dialogflow in den einzelnen *Intents* definiert. Der Name wird in Zeile 35 der Abb. 28 ausgelesen und dann für jede mögliche Funktion verglichen. Die passende Funktion wird ausgeführt.

```
33 @app.route('/webhook', methods=['POST'])
34 def webhook():
35     req = request.get_json(silent=True, force=True)    #req = anfrage json
36     print("Request:")
37     print(json.dumps(req, indent=4))
38     action = req.get("result").get("action")    # auslesen der action = wird nac
39     query = req.get("result").get("resolvedQuery")    # auslesen der action = wi
40     if action == "yahooWeatherForecast":
41         res = processWeather(req)
42     elif action == "searchOnLustenauAT":
43         res = processSearchLustenauAT(req)
44     elif action == "parkbadLustenau":
45         res = processParkbadLustenau(req)
46     elif action == "getAbfahrtszeiten":
47         res = processAbfahrtszeiten(req)
48     elif action == "openCurrentWeatherForecast":
49         res = processCurrentOpenWeather(req)
50     elif action == "openVorhersageWeatherForecast":
51         res = processVorhersageOpenWeather(req)
52     elif action == "newEvents":
53         res = processNewEventsQuery(query, 256)
54     elif action == "newEventsMusic":
55         res = processNewEventsQuery(query, 255)
56     elif action == "newEventsSport":
57         res = processNewEventsQuery(query, 198)
58
59     res = json.dumps(res, indent=4)
60     # print(res)
61     r = make_response(res)
62     r.headers['Content-Type'] = 'application/json'
63     return r
64
```

Abb. 28 Zuteilung der korrekten Funktion bei der Anfrage

```

1 {
2   "id": "65c5162f-f73f-47e8-9e04-0440a10bb9d4",
3   "timestamp": "2018-01-27T22:08:00.534Z",
4   "lang": "de",
5   "result": {
6     "source": "agent",
7     "resolvedQuery": "wetter",
8     "action": "openVorhersageWeatherForecast",
9     "actionIncomplete": false,
10    "parameters": {
11      "geo-city": "Lustenau",
12      "date": "",
13      "wetter": "wetter"
14    },
15    "contexts": [],
16    "metadata": {
17      "intentId": "c5f3b9ab-f751-467e-ba9a-0903882cb715",
18      "webhookUsed": "true",
19      "webhookForSlotFillingUsed": "false",
20      "webhookResponseTime": 5000,
21      "intentName": "wetter-vorhersage"
22    },
23    "fulfillment": {
24      "speech": "Bitte sei mir jedoch nicht böse, ich bin noch ein Prototyp ... ^_^",
25      "messages": [
26        {
27          "type": 0,
28          "speech": "Tut mir leid, ich kann die Wetterfee derzeit nicht erreichen :(
29        },
30        {
31          "type": 0,
32          "speech": "Bitte sei mir jedoch nicht böse, ich bin noch ein Prototyp ... ^_^"
33        }
34      ]
35    },
36    "score": 1
37  },
38  "status": {
39    "code": 206,
40    "errorType": "partial_content",
41    "errorDetails": "Webhook call failed. Error: Request timeout.",
42    "webhookTimedOut": true
43  },
44  "sessionId": "dce6559b-1056-4ceb-8063-41ef19a0230f"
45 }

```

Abb. 29 JSON der Anfrage bei der Benutzereingabe „Wetter“

Jede Funktion verfügt über eine Rückgabe, die auch bereits im *JSON*-Format vorliegt (siehe Abb. 31). Diese Rückgabe wird dann an „res“ übergeben und in Zeile 63 an Dialogflow returniert (siehe Abb. 28).

⚡ Fulfillment

Webhook

ENABLED

Your web service will receive a POST request from Dialogflow in the form of the response to a user query matched by intents with webhook enabled. Be sure that your web service meets all the [webhook requirements](#) specific to the API version enabled in this agent.

[Webhook example](#)

URL* <https://lustenauBot-webhook1.herokuapp.com/webhook>

Abb. 30 Konfiguration des WebHook in „Dialogflow“

Der *WebHook* bzw. dessen *URL* kann in Dialogflow im Reiter „Fulfillment“ konfiguriert werden. Dialogflow wie auch *WebHook* sind SSL-geschützt, ein Dritter kann also nicht mitlesen. Die Anfrage selbst ist nicht durch eine Security-Maßnahme im Header geschützt, da dies in Dialogflow zum Zeitpunkt der *Implementiert* noch nicht möglich war.

```
196     return {
197         "speech": "",
198         "messages": [
199             {
200                 "type": 0,
201                 "speech": str(einleitung),
202                 "displayText": str(einleitung),
203             },
204             {
205                 "type": 0,
206                 "speech": str(heute),
207                 "displayText": str(heute)
208             },
209             {
210                 "type": 0,
211                 "speech": str(morgen),
212                 "displayText": str(morgen)
213             }
214         ],
215         "source": "lustenauBot"
216     }
```

Abb. 31 Rückgabe im JSON-Format

5.5 Facebook Messenger

5.5.1 Aufsetzen einer Facebook-Seite

Damit die Anwenderin überhaupt erst mit dem „*Chatbot*“ über den „Facebook Messenger“ kommunizieren kann, muss eine Facebook-Seite für den „Bot“ eingerichtet werden. Diese Seite ist im Grunde wie ein normales Facebook-*Profil* eines Nutzers. Es können „*Postings*“ verfasst, das *Profilbild* geändert, Bilder hochgeladen, oder eben auch mit dem jeweiligen *Profil* über den „Facebook Messenger“ kommuniziert, bzw. Nachrichten ausgetauscht werden.

Das *Profil* des „*Chatbot*“ wurde aufgrund der laufenden Entwicklung noch nicht veröffentlicht. Das heißt, dass nur ausgewählte, beim „Bot“ registrierte Entwickler und Tester den Dienst in Anspruch nehmen konnten. Als Entwickler war das Projektteam registriert. Die Tester waren die Betreuer, sowohl von Seiten der Markgemeinde Lustenau, als auch von der internen Betreuung der HTL Dornbirn.

5.5.2 Facebook Developer Console

Um die „Facebook Messenger API“ mit „Dialogflow“ zu verbinden, musste von der „Facebook Developer Console“ ein „Page Access Token“ generiert bzw. angefordert werden. Um dies zu erreichen, wird zum einen ein persönlicher Facebook-Account, sowie die bereits oben beschriebene Facebook-Seite des eigentlichen „*Chatbot*“.

Der erste Schritt war, sich bei der „Facebook *Developer Console*“ mit seinem Facebook *Profil* als Entwickler zu registrieren. Weiters wurde nun auf dieser Seite ein Projekt erstellt. Dieses Projekt wurde nun mit der *Profil-Seite* des „*Chatbot*“ verlinkt. Nachdem diese Verbindung erfolgreich war, wurde der „Messenger“-Teil aktiviert, damit man mit dem „Bot“ über den „Facebook Messenger“ kommunizieren kann. Nach dieser Aktivierung wurde besagter „*Page Access Token*“ für die Verbindung zwischen der *Profil-Seite* und „*Dialogflow*“ generiert (siehe Abb. 32). Dieser „*Access Token*“ wurde dann weiter für die Integration von „*Dialogflow*“ verwendet (siehe „5. Einarbeitung“ → „*Dialogflow*“ → „*Verbinden von Dialogflow und der Facebook Messenger API*“).



Abb. 32 „*Access-Token*“ in „*Facebook Developer Console*“ generieren

5.6 Heroku

5.6.1 Prinzip

Heroku ist eine Plattform, die Programmiererinnen das *Deployment* und *Hosting* von Python-Anwendungen erleichtert. Durch das automatisierte *Hosting* und *Deployment* ergeben sich einige Vorteile für die Entwicklerin: (Guder, 2018)

- Fokus: Die Programmiererinnen können sich auf Logik und Inhalt ihrer Anwendungen konzentrieren.
- Know-how: Die Anwenderin muss sich nicht in die *Implementierung* und Umsetzung von Serverkonfigurationen einlesen.
- Kosten: Heroku verringert den Aufwand für einen eigenen Server und senkt damit die Kosten, die sowohl für den Server selbst als auch für die *Implementierung* anfallen würden. (logicline, 2018)

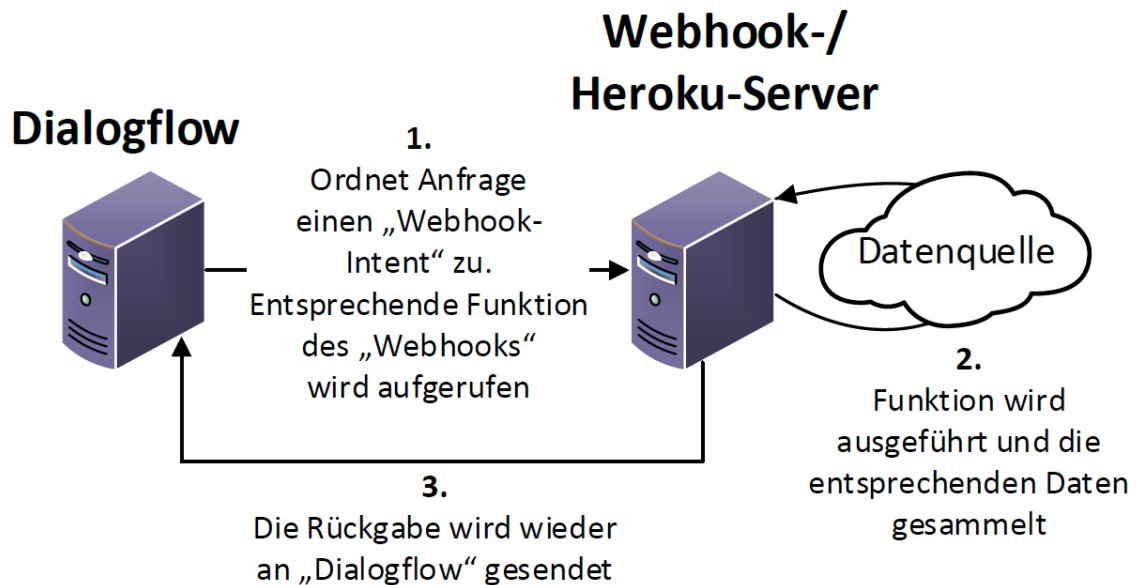


Abb. 33 Funktionalität von Heroku im Projekt

Der „Chatbot“ benötigte Heroku für die *Implementierung* des *WebHook*, der die Daten externer Dienste abrufen. Die Anfrage kommt dabei von Dialogflow, dem *Framework* zur Textverarbeitung. Die Rückgabe der Antwort erfolgt ebenfalls an Dialogflow (siehe Abb. 33).

5.6.2 Implementierung

5.6.2.1 Deployment

Bevor das Programm überhaupt laufen kann, muss es auf die Plattform übertragen werden. *Implementiert* wird dies mittels Git bzw. GitHub. Da sich der Programmcode des „Chatbot“ auf GitHub befand, konnte die Option „Automatic Deploy“ ausgewählt werden (siehe Abb. 34). Dies bedeutete, dass bei jeder Änderung der *master-Branch* auf GitHub der „Chatbot“ automatisch neu „deployed“ wurde. Dazu musste lediglich die „Branch“ angegeben werden. So konnte das *Deployment* automatisiert werden, da die Codeänderung lediglich auf GitHub geladen werden musste und Heroku selbstständig registrierte, wann dies der Fall war und dementsprechend selbstständig das *Deployment* startete.

Ist der Code einmal übertragen, erfolgt dessen „*Build*“. Dieser Prozess wird automatisch von Heroku initialisiert und erfolgt im Hintergrund. (logicline, 2018)

lustenauBot

Deployment method

Heroku Git Use Heroku CLI GitHub Connected Dropbox Connect to Dropbox Container Registry Use Heroku CLI

App connected to GitHub

Code diffs, manual and auto deploys are available for this app.

Connected to koflirm/lustenau-bot-webhook Disconnect...

- ✓ Releases in the [activity feed](#) link to GitHub to view commit diffs
- ✓ Automatically deploys from master

Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

Automatic deploys from master are enabled

Every push to `master` will deploy a new version of this app. **Deploys happen automatically**; be sure that this branch in GitHub is always in a deployable state and any tests have passed before you push. [Learn more.](#)

Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

Disable Automatic Deploys

Abb. 34 Auto Deployment Option auf Heroku

5.6.2.2 Logging

Heroku verfügt über „Logging“-Funktionen. Dies bedeutet, dass alle Aktionen der Anwenderin ausgegeben werden und vom Entwickler eingesehen werden können. Der große Vorteil, den Logging bietet, ist eine einfache Fehlersuche. Tritt ein Fehler auf, werden (nach dementsprechender Python-Programmierung) Fehlerort sowie Fehlergrund aufgezeigt (siehe Abb. 35).

Overview Resources Deploy Metrics Activity Access Settings

Application Logs ALL PROCESSES

```
2018-01-26T07:43:58.547401+00:00 app[web.1]: "code": 200,
2018-01-26T07:43:58.547402+00:00 app[web.1]: "webhookTimedOut": false
2018-01-26T07:43:58.547403+00:00 app[web.1]: }
2018-01-26T07:43:58.547403+00:00 app[web.1]: }
2018-01-26T07:43:58.547406+00:00 app[web.1]: Suchbegriff in processSearchLustenauAT: turr doch gschied
2018-01-26T07:43:58.547809+00:00 app[web.1]: 10.124.30.221 - - [26/Jan/2018 07:43:58] "POST /webhook HTTP/1.1" 200 -
2018-01-26T07:43:58.547564+00:00 heroku[router]: at=info method=POST path="/webhook" host=lustenau-bot-webhook1.herokuapp.com request_id=2065e136-5c44-4a27-ae3e-6b3db27c20b8 fwd="35.226.155.101" dyno=web.1 connect=0ms service=3ms status=200 bytes=954 protocol=https
2018-01-26T08:19:39.013992+00:00 heroku[web.1]: Idling
2018-01-26T08:19:39.014437+00:00 heroku[web.1]: State changed from up to down
2018-01-26T08:19:40.133051+00:00 heroku[web.1]: Stopping all processes with SIGTERM
2018-01-26T08:19:40.299754+00:00 heroku[web.1]: Process exited with status 143
```

Abb. 35 Logging-Funktion in Heroku

6 Design

Zu Beginn des Projektes wurde das Design des „Chatbot“ ausgearbeitet. Dieses ist die Grundlage für die Entwicklung, da es nicht nur die Gestaltung von Elementen, sondern auch den Stil des Kommunikationsflusses bestimmt.

6.1 Mockups

Zu Beginn erstellten wir ein Mockup, das einen einfachen Kommunikationsfluss darstellen sollte. Ein Mockup ist ein Modell, um den gewünschten Output zu Präsentationszwecken darzustellen (siehe Abb. 36). Zur Erstellung eines solchen verwendeten wir die Plattform „Botsociety“, da diese bereits über vordefinierte Werkzeuge für einen Messenger-Bot verfügt. (Unbekannt, 2018); (Banfi, 2018)

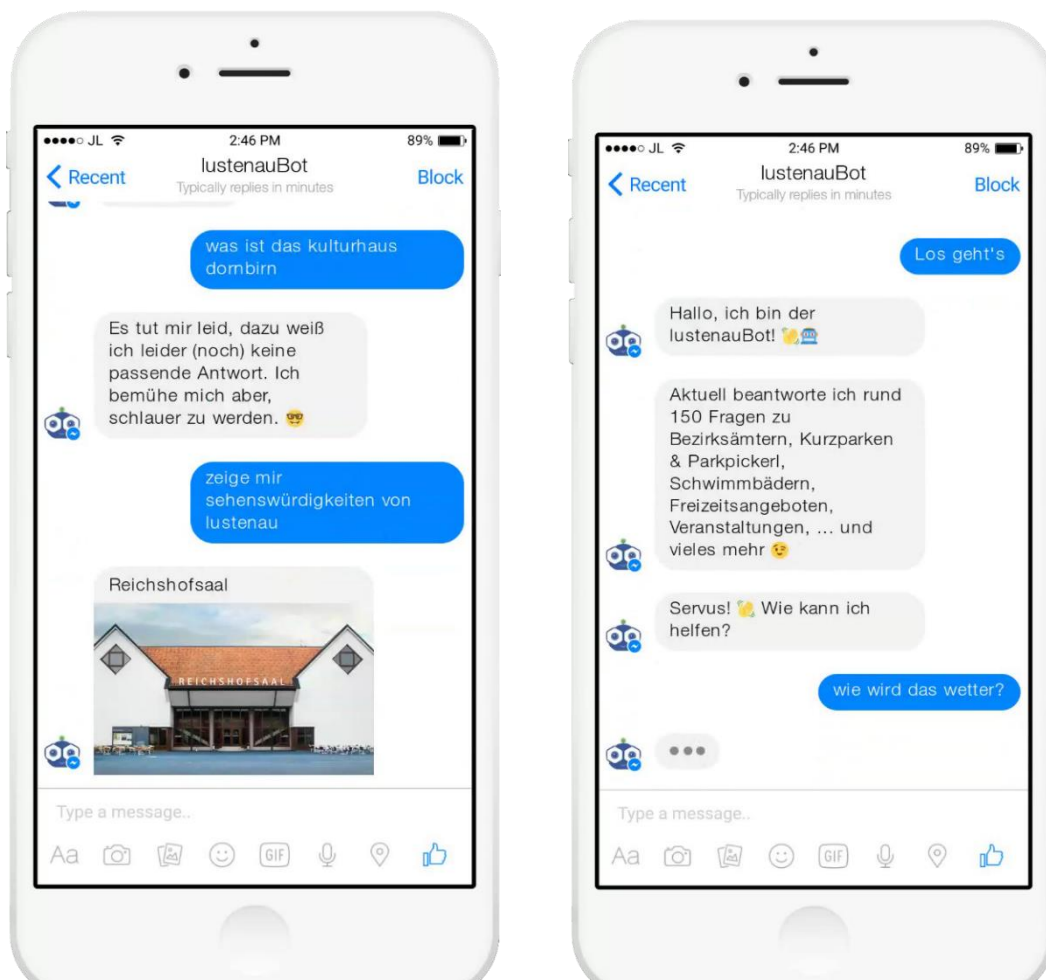


Abb. 36 Mockups Facebook Messenger „Chatbot“

6.2 User Interface

Der Facebook Messenger gibt das Design der *Buttons* und Textfelder vor, diese Elemente können nicht umgestaltet werden. Aus diesem Grund wurde der Fokus auf die Gestaltung der Facebook Seite und das Logo gelegt.

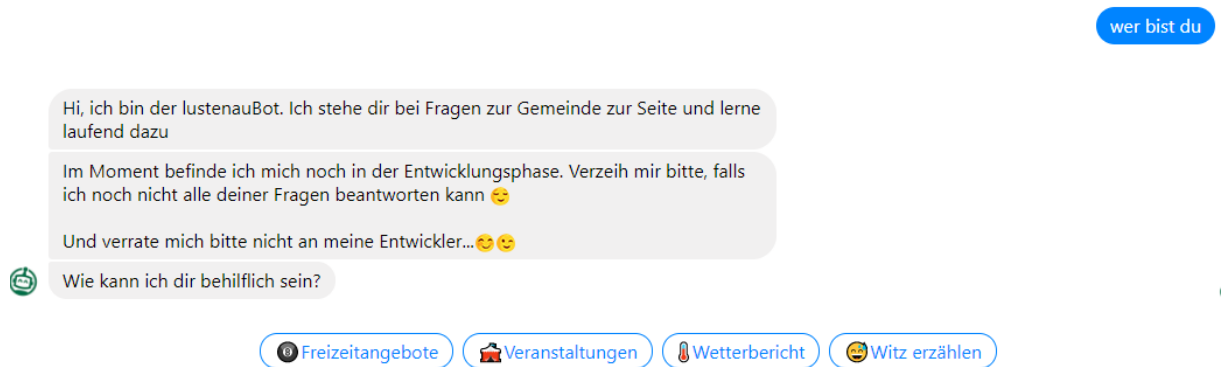


Abb. 37 Elemente der Messenger UI

6.3 Logo

Das Logo des „Chatbot“ wird der Anwenderin sowohl beim Aufruf der Facebook-Seite, als auch beim Interagieren im Chat angezeigt. Aus diesem Grund war es wichtig, ein eindeutiges und freundliches Logo zu erstellen (siehe Abb. 38).

In Absprache mit dem Auftraggeber wurde entschieden, das Logo in jenem Grünton zu gestalten, der auch von der Gemeinde verwendet wird. Grundsätzlich stellt das Logo den Kopf eines freundlichen Roboters mit Antenne dar.



Abb. 38 Logo des lustenauBot

6.4 Facebook Seite

„Chatbots“ werden immer in eine Facebook-Seite integriert. Diese gilt als Startpunkt und liefert dem Benutzer zudem Informationen zur Dienstleistung sowie Neuigkeiten über die Funktionalität (siehe Abb. 39).

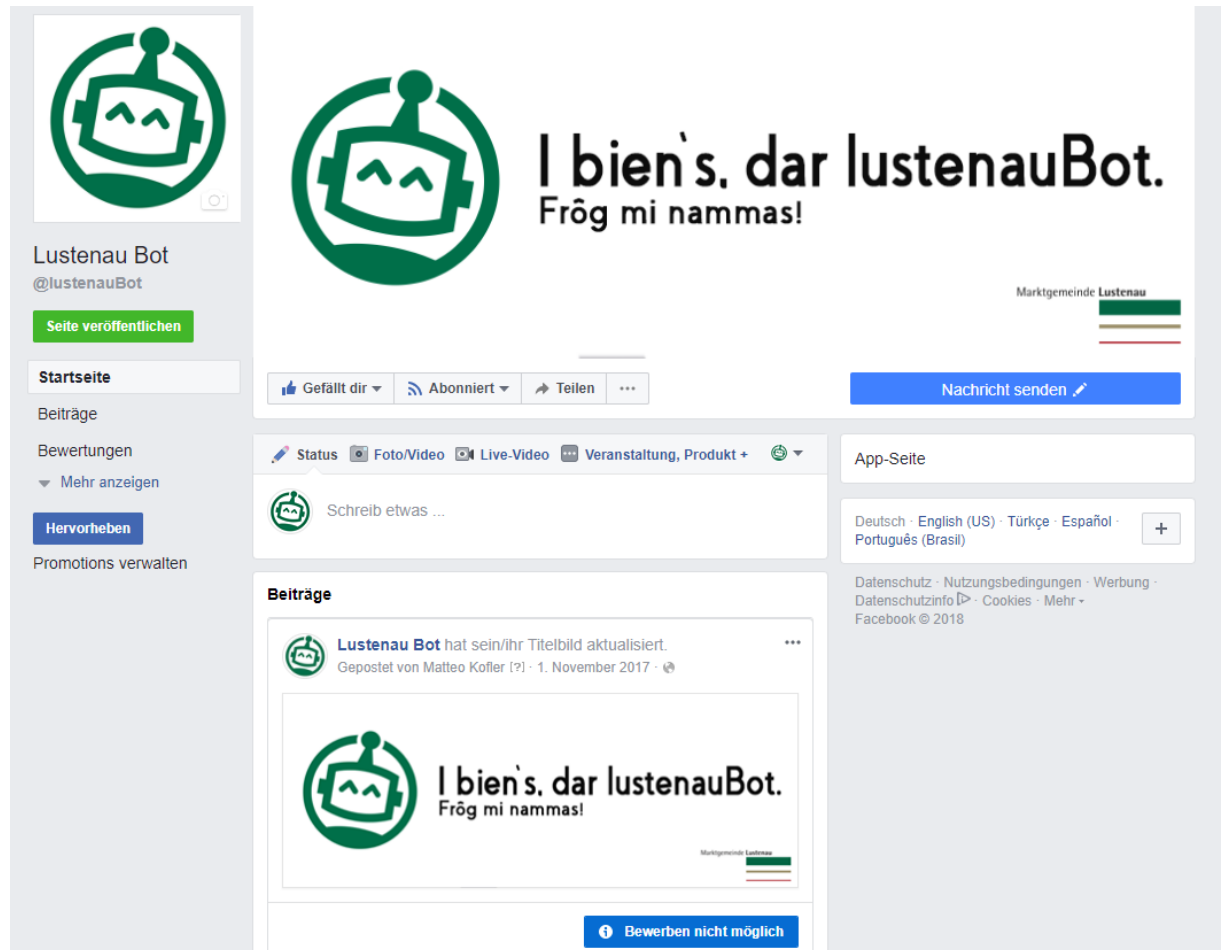


Abb. 39 Facebook Seite des lustenauBot

Beim Design achteten wir darauf, der Anwenderin eine einfache und gut strukturierte Oberfläche zu präsentieren. Das Titelbild (siehe Abb. 40) ist ebenso wie das Logo auf weißem Hintergrund dargestellt und beinhaltet Lustenauer Dialekt. Zudem werden die Logos des „Chatbot“ und der Marktgemeinde Lustenau präsentiert.

Ein weiterer wichtiger Aspekt ist der Benutzername und die damit verbundene *URL* der Seite. Diese kann nur einmal geändert werden, weshalb sich bereits frühzeitig mit dem Auftraggeber auf den Namen „lustenauBot“ geeinigt wurde. (Eishofer, 2018)



Abb. 40 Titelbild der Facebook Seite

6.5 User Experience

Buttons, Auswahlkästen, Textfelder, Bilder: Im Webbereich bestimmen diese Elemente die Benutzeroberfläche. „*Chatbots*“ verfügen jedoch nicht über die Möglichkeit, dieselben Elemente zu verwenden. Sie sind textorientiert, Platz für grafische Elemente ist kaum vorhanden. Da der Benutzer jedoch an die oben genannten Elemente gewöhnt ist, galt es, einen Kompromiss zwischen Text und Grafik zu finden.

6.5.1 Onboarding

Das Thema Onboarding behandelt den ersten Bildschirm (siehe Abb. 41), den die Anwenderin zu Gesicht bekommt. Dieses Begrüßungsfenster besteht grundsätzlich aus vier Elementen:

- Header-Bild der dazugehörigen Facebook-Seite
- *Profilbild* (Logo) der Facebook-Seite
- Willkommensnachricht (diese darf maximal aus 160 Zeichen bestehen)
- *Button*, mit dem die Anwenderin die *Interaktion* starten kann

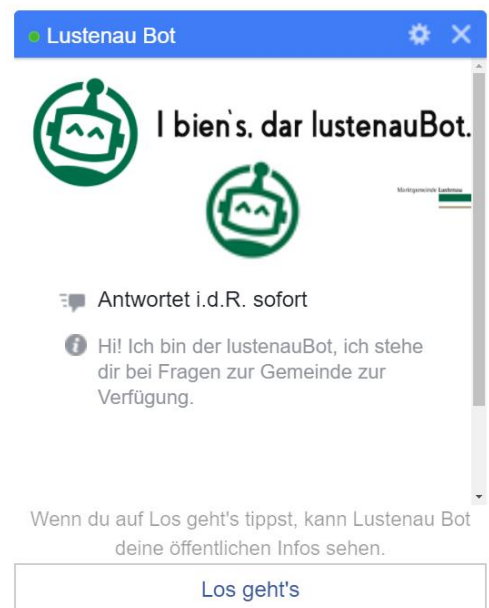


Abb. 41 Willkommensbildschirm des lustenauBot

6.5.2 Buttons

Obwohl ein „*Chatbot*“ textbasiert ist, stellt die Messenger UI von Facebook *Buttons* bereit, um den Fluss der Konversation zu erleichtern (siehe Abb. 42). Die *Implementierung* von *Buttons* anstelle textbasierter Antworten (in der Fachsprache „*Quick-Reply-Buttons*“ genannt) ermöglicht zudem eine strukturierte Konversation, wird durch vorgegebene Antworten das Risiko einer fehlerhaften Anfrage durch den Benutzer doch sehr reduziert.

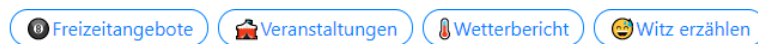


Abb. 42 Quick-Reply-Buttons im Facebook Messenger

6.5.3 Interaktionsmöglichkeiten

Der Facebook Messenger ermöglicht der Anwenderin eine Vielzahl an *Interaktionsmöglichkeiten*, von einfachem Schreiben über das Senden von Bildern bis hin zum Absenden einer Sprachnachricht. (Unbekannt, Facebook for developers, 2018)

Der „*Chatbot*“ reagiert auf all diese Eingabemöglichkeiten. Audio, *Emoticons* und Text werden verarbeitet und ausgewertet. Ebenso kann jeglicher Anhang (Bilder, Dateien) hochgeladen werden (siehe Abb. 43). Dies muss in der Entwicklung des „*Chatbot*“ beachtet werden, um den Benutzer immerzu eine geeignete Antwort bereitstellen zu können.

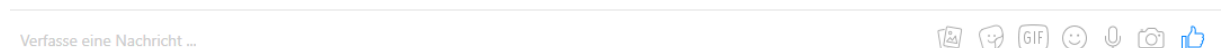


Abb. 43 Verschiedene Interaktionsmöglichkeiten für Anwenderin im FB-Messenger

Da der IustenauBot nicht auf Bilder oder andere Dateien reagieren kann, wurde eine entsprechende Standardantwort festgelegt (siehe Abb. 44).

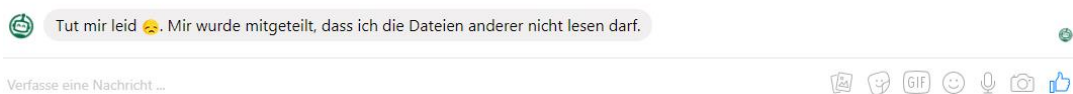


Abb. 44 Reaktion des Bots auf ein gesendetes Bild der Anwenderin

(Iurchenko, 2018)

6.6 Sprache

Ähnlich wie beim Thema *Buttons* gilt auch hier, weniger ist mehr. Lange Texte sind bei einem „Chatbot“ unangebracht, Menschen fühlen sich zu visuellen Elementen hingezogen. Aus diesem Grund wurde bei der Entwicklung des „Chatbot“ großer Wert auf *Emoticons* gelegt. (siehe Abb. 45). Das menschliche Gehirn assoziiert *Emoticons* mit menschlichen Gesichtern, wodurch die Anwenderin den „Chatbot“ vermenschlicht und personalisiert.

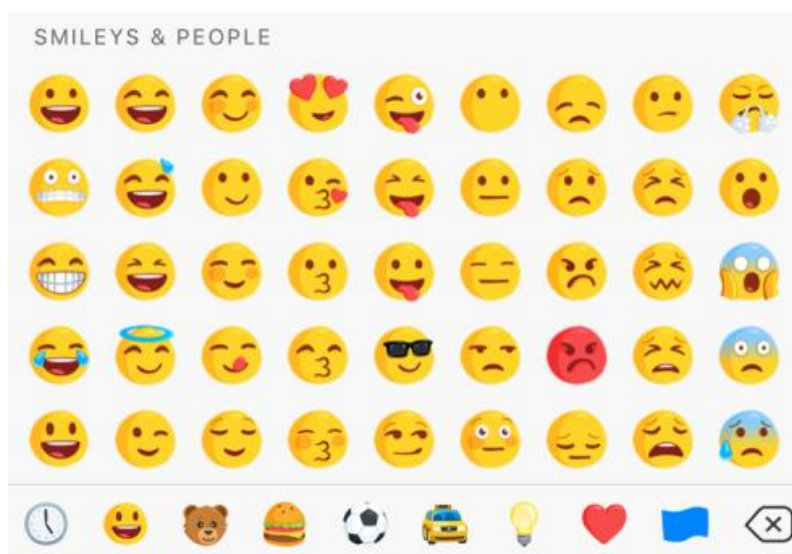


Abb. 45 Sammlung von Emoticons

Emoticons sind jedoch nicht gleich *Emoticons*, sie unterscheiden sich von Plattform zu Plattform. Aus diesem Grund wurde entschieden, dass nur die gängigsten *Emoticons* verwendet werden. Diese sind auf allen Plattformen ähnlich und der Benutzer reagiert auf allen Plattformen ähnlich darauf (siehe Abb. 46, wie dasselbe *Emoticon* durch eine andere Darstellung interpretiert wird). Zudem bieten sie den Vorteil, der Anwenderin bereits bekannt zu sein.

Same Emoji + Different Smartphone Platform = Different Emotion

For example, if you send the Apple emoji to a Google Nexus, they'll see the Google emoji, and vice versa!

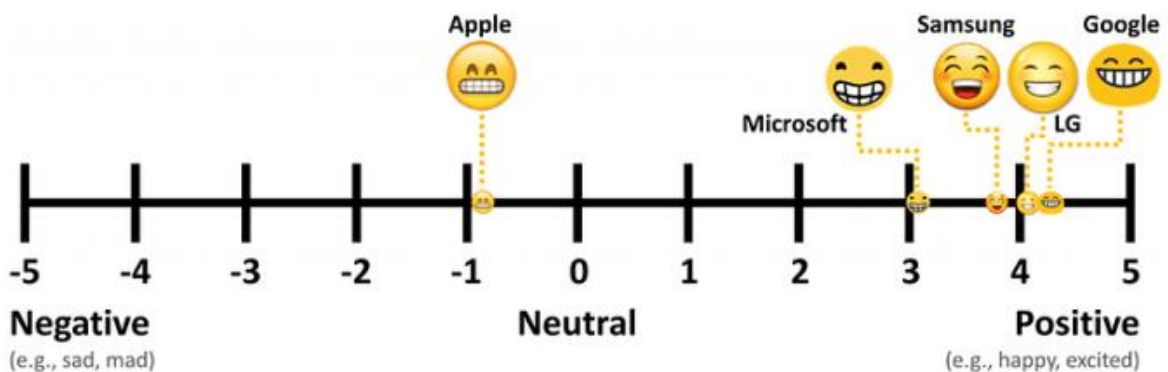


Abb. 46 Emoticons werden auf allen Plattformen unterschiedlich wahrgenommen

(Ondrisek, 2018); (Archer, 2018)

Emoticons werden überall verwendet, sei es beim Abruf der Abfahrtszeiten oder bei Veranstaltungen, da sie dem Benutzer ein Gefühl der Freundlichkeit vermitteln (siehe Abb. 47).

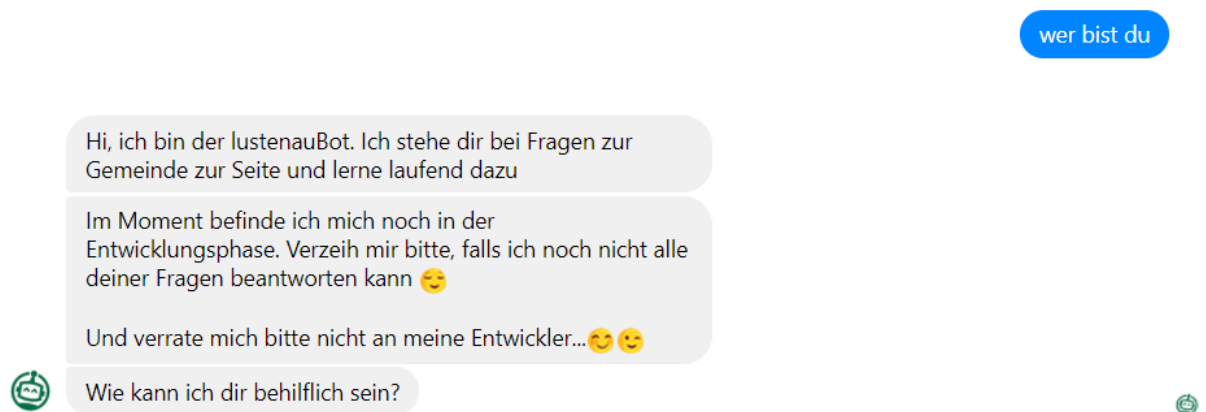


Abb. 47 Beispiel für die Verwendung von Emoticons im Smalltalk

7 Entwicklung und Umsetzung

Nach der Konzeption und der Analyse galt es, das Projekt auch umzusetzen. Die Programmierung und Entwicklung wurden dabei in mehrere Punkte aufgeteilt.

7.1 Wetterbericht

Bevor mit der Entwicklung des Wetterberichts begonnen werden konnte, musste ein geeigneter Dienst ausgewählt werden. Dazu wurden zwei Dienste miteinander verglichen, die beide kostenlos sind und Wetterinformation im *JSON*-Format zurückgeben:

OpenWeatherMap	Yahoo Weather
Vorteile: <ul style="list-style-type: none"> • Erfahrungswerte waren bereits vorhanden • Textrückgabe in deutscher Sprache 	Vorteile <ul style="list-style-type: none"> • Schritt für Schritt-Anleitung auf Dialogflow • eigene <i>Implementierung</i> von Übersetzungen nötig
Nachteil: <ul style="list-style-type: none"> • standardmäßig nur Abfragen von 5-Tages oder 16-Tages-Vorhersagen möglich 	Nachteil: <ul style="list-style-type: none"> • komplexe Abfragen in SQL möglich

Tab. 8 Vergleich: OpenWeatherMap vs. YahooWeather

(Kunal, 2018); (Wagner, 2018)

Zunächst fiel die Wahl auf den Wetterdienst von Yahoo, da bereits eine Anleitung, wie dieser Dienst zu *implementieren* ist, von Dialogflow bereitgestellt wurde.

7.1.1 YahooWeatherAPI

Die *Implementierung* der Wetterabfrage wurde in vier Funktionen aufgeteilt:

7.1.1.1 Hauptfunktion

Hier lief das Prozedere ab, die beiden anderen Funktionen wurden hier aufgerufen. So konnte eine bessere Übersichtlichkeit gewahrt werden.

7.1.1.2 URL-Funktion

In dieser Funktion wurde die passende *URL* zur Abfrage zusammengebaut. Zunächst wurde der Ort ausgelesen, welcher der Benutzer abfragt. Dieser wird dann in die *URL* eingebaut und an die Hauptfunktion zurückgegeben (siehe Abb. 48).

```
576 def makeYahooWeatherQuery(req):
577     result = req.get("result")
578     parameters = result.get("parameters")
579     city = parameters.get("geo-city")
580     if city is None:
581         return None
582
583     return "select * from weather.forecast where u='c' and woeid in (select woeid from geo.places(1) where text='" + city + "')"
584
```

Abb. 48 URL-Funktion der YahooWeather-Abfrage

7.1.1.3 Result-Funktion

In dieser Funktion wurden die Ergebnisse ausgelesen und die *Strings* für die Rückgabe an die Anwenderin generiert. Aufgrund der Tatsache, dass keine Abfrage der Daten in Deutsch möglich war, musste eine vierte Funktion *implementiert* werden, um die einzelnen Wörter zu übersetzen:

7.1.1.4 Übersetzungs-Funktion

Zur Übersetzung der einzelnen Fachwörter wurde ein bereits vorhandenes PHP-Skript verwendet und in Python umgeschrieben. Prinzipiell wurden hier alle englischen Wörter mit dem tatsächlich vorhandenen verglichen und dieses dementsprechend durch das deutsche Wort ersetzt (siehe Abb. 49).

```
627 def translateWeatherStrings(wetterText):
628     # im select kein deutsch verfuegbar
629     if wetterText == "AM Clouds/PM Sun":
630         wetterText = wetterText.replace('AM Clouds/PM Sun','vormittags bewölkt/nachmittags sonnig')
631     elif wetterText == "AM Drizzle/Wind":
632         wetterText = wetterText.replace('AM Drizzle/Wind','vorm. Nieselregen/Wind')
633     elif wetterText == "AM Drizzle":
```

Abb. 49 Ausschnitt aus der Übersetzungsfunktion der YahooWeather-Abfrage

7.1.2 OpenWeatherMap

Beim Testing der Funktion wurde ersichtlich, dass der Wetterdienst oftmals Probleme damit hat, die Daten abzurufen. Diese Probleme traten unregelmäßig auf, weshalb das Debugging des Fehlers nicht zielführend war. Aus diesem Grund wurde zu einem anderen Dienst gewechselt: OpenWeatherMap. Auf dem unten abgebildeten Screenshot (Abb. 50) ist die Rückgabe einer Abfrage von OpenWeatherMap dargestellt.



```
GET http://api.openweathermap.org/data/2.5/forecast/daily?q=LUSTENAU,AT&cnt=2&lang=DE&APPID=62b7126dccc0b9861f6b5487f62f2c71&format=json

Pretty Raw Preview JSON

1 {
2   "city": {
3     "id": 7872674,
4     "name": "Lustenau",
5     "coord": {
6       "lon": 9.6599,
7       "lat": 47.4277
8     },
9     "country": "AT",
10    "population": 20556
11  },
12  "cod": "200",
13  "message": 0.186545,
14  "cnt": 2,
15  "list": [
16    {
17      "dt": 1517137200,
18      "temp": {
19        "day": 280.62,
20        "min": 272.02,
21        "max": 280.62,
22        "night": 272.02,
23        "eve": 276.97,
24        "morn": 280.62
25      },
26      "pressure": 950.16,
27      "humidity": 91,
28      "weather": [
29        {
30          "id": 803,
31          "main": "Clouds",
32          "description": "Überwiegend bewölkt",
33          "icon": "04d"
34        }
35      ],
36      "speed": 1.2,
37      "deg": 194,
38      "clouds": 56
39    }
40  ]
41 }
```

Abb. 50 Rückgabe einer Abfrage von OpenWeatherMap

Durch den Umstieg auf OpenWeatherMap wurde auch die Übersetzungsfunktion eliminiert, da der Dienst in deutscher Sprache verfügbar ist. Dieser Wegfall ermöglichte es, den Fokus auf das Wetter selbst zu legen. Aus diesem Grund wurde das Wetter auf zwei verschiedene Antworten aufgeteilt:

- Aktuelles Wetter
- Wettervorhersage

” temperatur
” wie viel grad hat es
” wie warm ist es gerade
” aktuelle temperatur
” wetter aktuell
” wie ist das wetter gerade
” wie schaut das wetter aus
” aktuelles wetter
” Iustenau openweathercurrent

Abb. 51 Schlüsselwörter des aktuellen Wetters

” wetter
” wetter heute
” wetter morgen
” wie wird das wetter werden
” openVorhersageWeatherForecast
” wettervorhersage für dornbirn
” wetterbericht anzeigen
” zeige den wetterbericht an
” Wetterbericht
” wetterbericht

Abb. 52 Schlüsselwörter der Wettervorhersage



IustenauBot

Wichtig hierbei war die Unterscheidung bei den Schlüsselworten bzw. den vorgegebenen Benutzereingaben (siehe Abb. 51 und 52).

API keys Home

Setup API keys My Services My Payments Billing plans Map editor Block logs History bulk Logout

Activation of an API key for **Free** and **Startup accounts** takes **10 minutes**. For **other accounts** it takes from **10 to 60 minutes**. You can generate as many API keys as needed for your subscription. We accumulate the total load from all of them.

Key	Name	Create key
62b7126dccc0b9861f6b5487f62f2c71	Default	 

* Name

Abb. 53 Registrierung des Keys bei OpenWeatherAPI

Bevor OpenWeatherAPI genutzt werden konnte, musste ein Key generiert werden. Dieser kann nach Erstellung eines Accounts kostenlos angefordert werden (siehe Abb. 53). Dieser Key muss beim Abruf der Schnittstelle in der URL (siehe Abb. 54, Zeile 145 „&APPID=...“) im Programmcode mit angegeben werden. Durch dieses „&APPID=...“ wird der Schnittstelle mitgeteilt, welcher Benutzer bzw. welche App auf die Daten zugreifen will. Dies ist eine Sicherheitsmaßnahme, um Überlastungen und unerwünschte Zugriffe zu vermeiden.

7.1.2.1 Aktuelles Wetter

Der unten dargestellte Screenshot (Abb. 54), zeigt einen Codeausschnitt des aktuellen Wetterberichts:

```
144 def processCurrentOpenWeather(req):
145     url = "http://api.openweathermap.org/data/2.5/weather?q=Lustenau&lang=DE&APPID=62b7126dccc0b9861f6b5487f62f2c71"
146     result = req.get("result")
147     parameters = result.get("parameters")
148     city = parameters.get("geo-city")
149     result = urlopen(url).read()
150     data = json.loads(result)
151     res = makeCurrentOpenWeatherWebhookResult(data)
152     return res
153
154
155 def makeCurrentOpenWeatherWebhookResult(data):
156     main = data.get('main')
157     temperature = main.get('temp')
158     weather = data.get('weather')
159     #icon = weather_aktuell[0].get('icon')
160     description = weather[0].get('description')
161     final_temperature = int(temperature) - 273 # von kelvin in celsius
162     aktuell = "Aktuelle Wetterlage: " + str(description) + ", die 🌡️Temperatur liegt bei " + str(final_temperature) + " °C"
163     return {
164         "speech": str(aktuell),
165         "displayText": str(aktuell),
166         "source": "lustenaubot"
167     }
```

Abb. 54 Codeausschnitt des aktuellen Wetterberichts

Die Programmierung der Funktion im *WebHook* wurde in zwei Schritte aufgeteilt. In der ersten Funktion wurde der Dienst abgefragt, in der zweiten Funktion die Abfrage ausgewertet und eine Antwort generiert. Ein wichtiger Punkt hierbei war die Umwandlung der Temperatur von Kelvin in Celsius (siehe Abb. 54, Codezeile 161). Die Ausgabe im Facebook Messenger sah schlussendlich wie folgt aus:

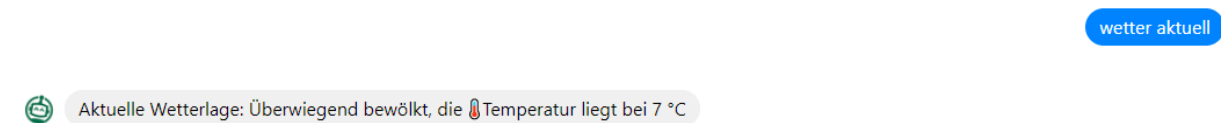


Abb. 55 Abfrage des aktuellen Wetters durch die Anwenderin im FB-Messenger

7.1.2.2 Wettervorhersage

Bei der Vorhersage galt es, nicht nur das aktuelle Wetter, sondern auch die heutige sowie morgige Vorhersage zurückzugeben. Auch die Wettervorhersage wurde in zwei Funktionen aufgeteilt. Die zweite Funktion unterschied sich allerdings darin, dass im *JSON* ein *Array* für die einzelnen Tage zurückgegeben wurde.

```
181 def makeVorhersageOpenWeatherWebhookResult(data):
182     vorhersage = data.get('list')
183     temperature_heute = vorhersage[0].get('temp')
184     weather_today = vorhersage[0].get('weather')
185     description_today = weather_today[0].get('description')
186     temperature_heute = temperature_heute.get('day')
187     final_temperature_heute = int(temperature_heute) - 273 # von kelvin in celsius
188     temperature_tomorrow = vorhersage[1].get('temp')
189     temperature_tomorrow = temperature_tomorrow.get('day')
190     final_temperature_tomorrow = int(temperature_tomorrow) - 273 # von kelvin in celsius
191     weather_tomorrow = vorhersage[1].get('weather')
192     description_tomorrow = weather_tomorrow[0].get('description')
193     einleitung = "☀️🌧️❄️ Glücklicherweise hat mir die Wetterfee die Wettervorhersage verraten: 😊"
194     heute = "Heute: " + str(description_today) + ", der 🌡️Tageshöchstwert liegt bei " + str(final_temperature_heute) + " °C"
195     morgen = "Morgen: " + str(description_tomorrow) + ", der 🌡️Tageshöchstwert liegt bei " + str(final_temperature_tomorrow) + " °C"
196     return {
197         "speech": "",
198         "messages": [
199             {
200                 "type": 0,
201                 "speech": str(einleitung),
202                 "displayText": str(einleitung)
203             },
204             {
205                 "type": 0,
206                 "speech": str(heute),
207                 "displayText": str(heute)
208             },
209             {
210                 "type": 0,
211                 "speech": str(morgen),
212                 "displayText": str(morgen)
213             }
214         ],
215         "source": "lustenauBot"
216     }
---
```

Abb. 56 Zweite Funktion der Wettervorhersage im WebHook

Die Rückgabe im Facebook Messenger sollte aus mehreren „Chatblasen“ bestehen. Aus diesem Grund wurde in dieser Funktion auch ein *Array* (im *JSON*-Format) returniert, jedes *Array*-Element stellte eine Chatblase dar (siehe Abb. 56, ab Zeile 196). Im Facebook Messenger durfte die Anwenderin Folgendes betrachten:

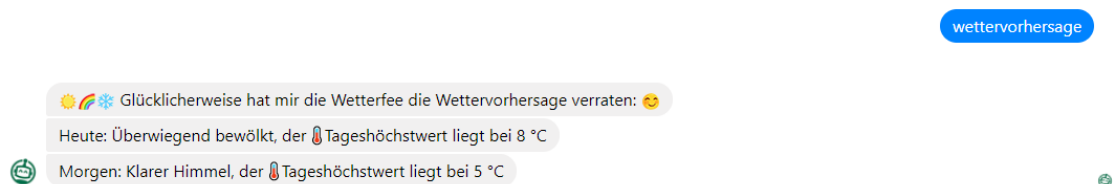


Abb. 57 Abfrage der Wettervorhersage durch den Benutzer im FB-Messenger

7.2 Jugendveranstaltungen

7.2.1 Probleme

Bei der Entwicklung einer Anbindung für den Abruf von Jugendveranstaltungen gab es grundsätzlich zwei Schwierigkeiten.

1. Der Zugriff auf die Datenbank
2. Schwierigkeiten mit der Codierung von Umlauten (siehe Kapitel 5.4 Python)

Ersteres, der Zugriff auf die Datenbank gestaltete sich anfangs sehr zeitintensiv. Geplant gewesen wäre, dass ein direkter Zugang zur Datenbank der Massive Art WebServices GmbH besteht. Dadurch hätte mit wirklichen Datensätzen gearbeitet werden können und es hätte gleich die echte Datenbank verwendet werden können. Es gab jedoch Probleme mit der „Firewall“ und somit mit der Einrichtung eines Zugangs auf der Datenbank. Später einigte sich das Projektteam darauf, ein Abbild des momentanen Standes der Datenbank zu machen und dieses Abbild auf einem privaten Server als Testumgebung zur Verfügung zu stellen. Dies wurde mit dem Projektauftraggeber vereinbart, da ansonsten ein hoher Kostenaufwand für die Einrichtung einer Verbindung auf die gegebene, gepflegte Datenbank entstanden wäre. Aufgrund dessen wurde auch kein besonderes Augenmerk auf die Sicherheit der Verbindung zur Datenbank gelegt, da es sich zum einen nur um ein Abbild, also eigentlich nur um eine Sicherung der wirklichen Datenbank handelte. Zum anderen war diese Lösung nur als Test-Umgebung gedacht. Würde der „Bot“ jedoch veröffentlicht werden, müsste die Datenbankverbindung auf die gegebene, gepflegte Datenbank geändert werden.

7.2.2 Übersicht der relevanten Tabellen

Jedoch hatten für den Zweck der Jugendveranstaltungen diese Tabellen die größte Relevanz:

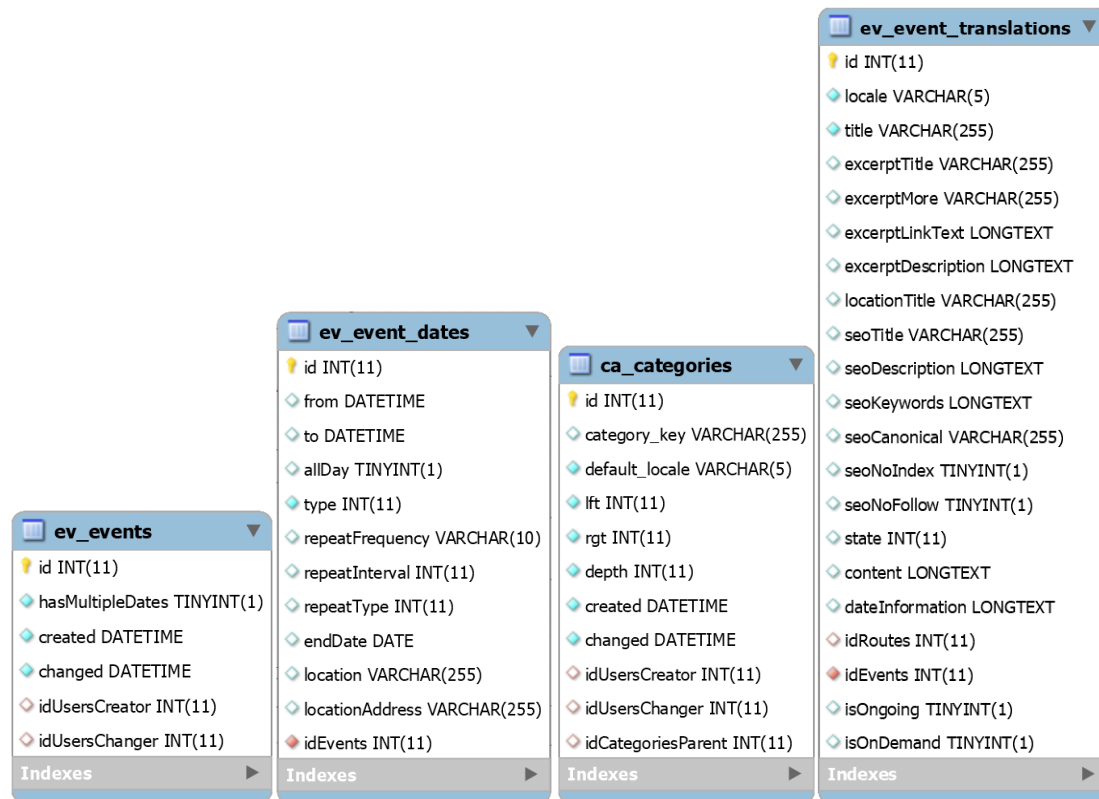


Abb. 58 Relevante Tabellen für Veranstaltungen

Für die Selektion der Daten wurden die folgenden Tabellen verwendet:

- ev_event_translations
 - Für den Titel des Events, den Veranstaltungsort, Eintrittspreise und andere Informationen
- ev_event_dates
 - Für Informationen zur Veranstaltung bezüglich des zeitlichen Rahmens (Beginn der Veranstaltung, Ende der Veranstaltung, ...)
- ev_events
 - Für Datenbank-interne Informationen der Veranstaltung
 - Diese Spalte ist für die Anwenderin nicht relevant, jedoch für den Entwickler, da diese Tabelle Aufschluss darüber gibt, wann das Event in der Datenbank eingetragen wurde, wann es zuletzt geändert wurde und welcher Nutzer den Datensatz eingefügt und verändert hat.

- ev_event_categories
 - Hier sind die verschiedenen Kategorien der Veranstaltungen aufgelistet. Diese Kategorien sind notwendig, um die Veranstaltungen nach jugendrelevanten Themen zu filtern.

Die Verwendung von Bildern der Veranstaltungen wurde anfangs auch besprochen, letztendlich aber nicht umgesetzt.

7.2.3 Select-Statement für erfolgreiche Selektion der Daten

7.2.3.1 Select-Statement mit Inner Joins

Anhand der Tabellen wurde dann ein entsprechendes „Select-Statement“, also eine Anweisung zum Selektieren der Daten, ausgearbeitet (siehe Abb. 59). Es mussten verschiedenen Tabellen verbunden werden um ein geeignetes Ergebnis zu erhalten. Wie bereits erwähnt, wurden folgende Tabellen mit den zugehörigen Spalten verwendet:

- ev_event_translations
- ev_event_dates
- ev_event
- ev_event_categories

Folgende Spalten wurden miteinander durch einen „inner join“ verbunden:

- ev_event_dates.idEvents → ev_events.id
- ev_event_translations.idEvents → ev_events.id
- ev_event_categories.idEvents → ev_events.id

Die Selektierungs-Anweisung sieht dann folgendermaßen aus:

```
select ev_event_translations.title, ev_event_dates.from, ev_event_dates.to,
       ev_event_translations.locationTitle, ev_event_translations.content
from ev_events
  inner join ev_event_dates
    on ev_event_dates.idEvents = ev_events.id
  inner join ev_event_translations
    on ev_event_translations.idEvents = ev_events.id
  inner join ev_event_categories
    on ev_event_categories.idEvents = ev_events.id
where idCategories = "" + str(cate_id) + "" and ev_event_dates.from >= '2016-06-24 00:00:00'
order by ev_event_dates.from asc
limit 6;
```

Abb. 59 „Select-Statement“ zum Selektieren der Veranstaltungsdaten

Das braun eingefärbte „str(cate_id)“ ist die jeweilige ID eines Events, das der User ausgewählt hat. Die ID des Events wird beim Funktionsaufruf der Funktion übergeben. Diese verschiedenen Kategorien können zum Beispiel folgende Nummern haben:

- 25: „leben-in-lustenau“
- 26: „Freizeit“
- 193: „Schule“
- 256: „Youth“
- 249: „events“
- 255: „concert“
- 198: „sport“
- 189: „gastronomie“ (jedoch sind im aktuellen Stand der Datenbank noch keine Einträge vorhanden)

In der „where“-Klausel wird zum einen nach der Kategorie-ID gefiltert und zum anderen auch noch nach dem Datum des Events. Es sollen Events angezeigt werden, die in naher Zukunft stattfinden und nicht bereits vergangen sind. Weil es sich bei dieser Datenbank jedoch um ein Test-Objekt handelt und nicht regelmäßig neue Datensätze dazu kommen, muss in diesem „Select-Statement“ von einem fixen Datum ausgegangen werden. Es wird daher überprüft, ob das Startdatum des Events später als der 24.06.2016 ist. Dieses Datum wird verwendet, da sich in diesem Zeitraum die meisten Datensätze in der Testdatenbank befinden. Es wurde somit testweise dieses Datum zur Präsentation der Daten gewählt. Am Ende der Anweisung werden die Events noch aufsteigend nach ihrem Datum sortiert und die ersten sechs Ergebnisse herausgeschrieben. Die Spalten des Ergebnisses lauten:

- Title
 - Titel des Events
- From
 - Datum und Uhrzeit, wenn das Event beginnt
- To
 - Datum und Uhrzeit, wenn das Events endet
- locationTitle
 - Ort an dem die Veranstaltung stattfindet.

- Content
 - Ausgabe verschiedener Informationen im *JSON*-Format (z.B. Titel der Veranstaltung, Artikel-Text, Straße, eventueller Verweis auf Bilder, ...)

Die einzelnen Spalten, die nun im Ergebnis präsentiert werden, werden dann in verschiedenen „*Arrays*“ für eine spätere Verwendung zwischengespeichert.

7.2.4 Aufbereitung der selektierten Daten

Die Inhalte der selektierten Spalten wurden in „*Arrays*“ übertragen. Um dem Nutzer einen ansprechenden Text präsentieren zu können, müssen die rohen Daten jetzt noch in eine Nachricht verpackt werden. In die Nachricht werden der Titel, das Datum bzw. die Uhrzeit und die Adresse geschrieben (siehe Abb. 60).



Abb. 60 Beispiel-Ausgabe eines Events

7.2.5 Beispiel für die Anfrage einer Veranstaltung

Im folgenden Beispiel wird veranschaulicht, wie die Ausgabe der Veranstaltungen für den Benutzer präsentiert wird. Als erstes wird eine Anfrage gestellt:

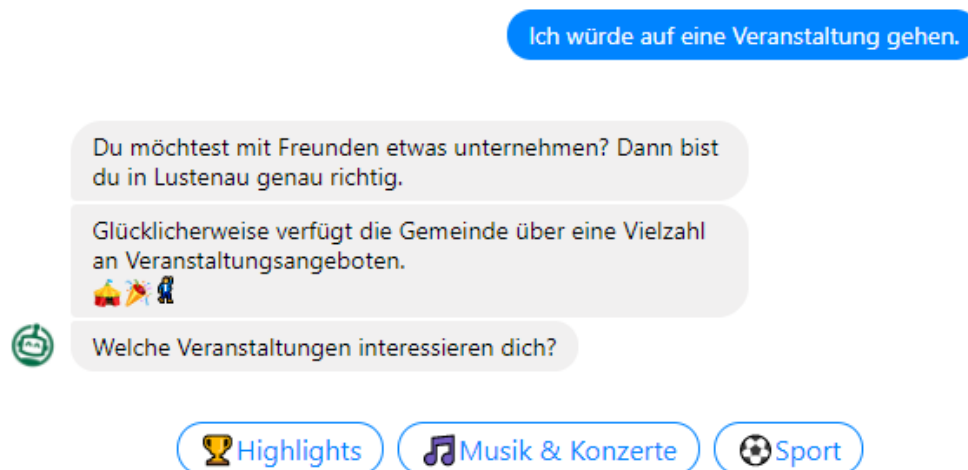


Abb. 61 Anfrage einer Anwenderin für eine Veranstaltung

lustenauBot

Anschließend wird eine der Kategorien ausgewählt. In diesem Beispiel wurde die Kategorie „Sport“ ausgewählt:

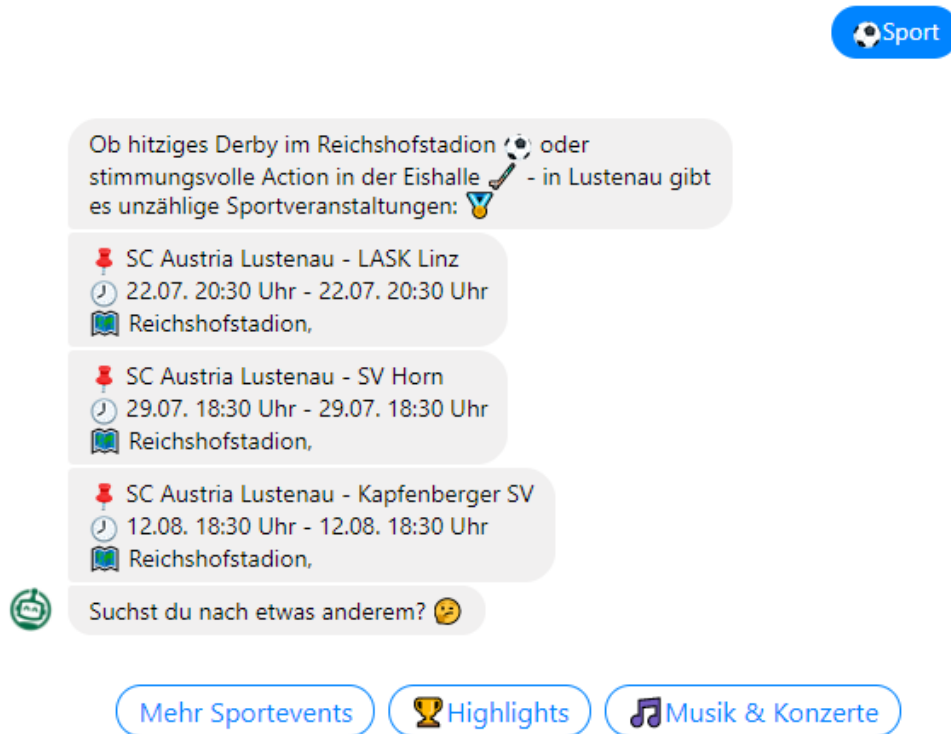


Abb. 62 Darstellung der Sportveranstaltungen

7.3 Freizeitangebote

Das Thema Freizeitangebote ist aus Sicht der Anwenderin das am breitesten gefächerte Gebiet des „Chatbot“. Mit Beispielphrasen wie „Mir ist langweilig“, „Was für Freizeitangebote gibt es in Lustenau“, oder „Was kann man in Lustenau unternehmen“ werden dem Nutzer eine Vielzahl an Auswahlmöglichkeiten geboten.

Der Anwenderin werden folgende Freizeitangebote vorgeschlagen:

- Sport
- Kino
- Veranstaltungen (für nähere Informationen siehe „Entwicklung und Umsetzung“ → „Jugendveranstaltungen“)
- Gastronomie
- Relaxen

Diese Optionen werden nun weiter beschrieben.

7.3.1 Sport

Wählt der Nutzer die Option „Sport“, wird wiederum zwischen verschiedenen Angeboten differenziert.

Diese sind:

- Baden
- Fußball
- Laufen
- Basketball
- Skating
- Sonstiges

Bei der Auswahl der Optionen wird ein entsprechender *Intent* aufgerufen. Bei diesem *Intent* sind statische Daten hinterlegt, d.h. die Rückgaben sind manuell verfasst und hinterlegt worden. Als Informationsquelle diente dabei meist die Homepage der Marktgemeinde Lustenau. Diese wurde dann bei den entsprechenden Angeboten auch immer verlinkt, falls die Anwenderin weitere Informationen wünscht, die der „Bot“ nicht präsentiert.

7.3.2 Kino

Für die Informationen zur Kinothek in Lustenau wurde die entsprechende Webseite verwendet. Auch hier wurden die wesentlichen Daten wie Öffnungszeiten und Kontaktdaten (Telefon, E-Mail, Verlinkung auf die Webseite und die Adresse), statisch *implementiert* (siehe Abb. 63).



Abb. 63 Statische Implementierung der Daten der Kinothek in „Dialogflow“

Das Projektteam hat sich aufgrund des Arbeitsaufwandes und der fehlenden Zielsetzung gegen die Bereitstellung des Kinoprogramms entschieden. Stattdessen wird ein Link bereitgestellt, der den Nutzer beim Anklicken auf die „Homepage“ bringt, auf der er das Kinoprogramm einsehen kann.

Nach Anfrage erhält die Anwenderin also folgende Daten:

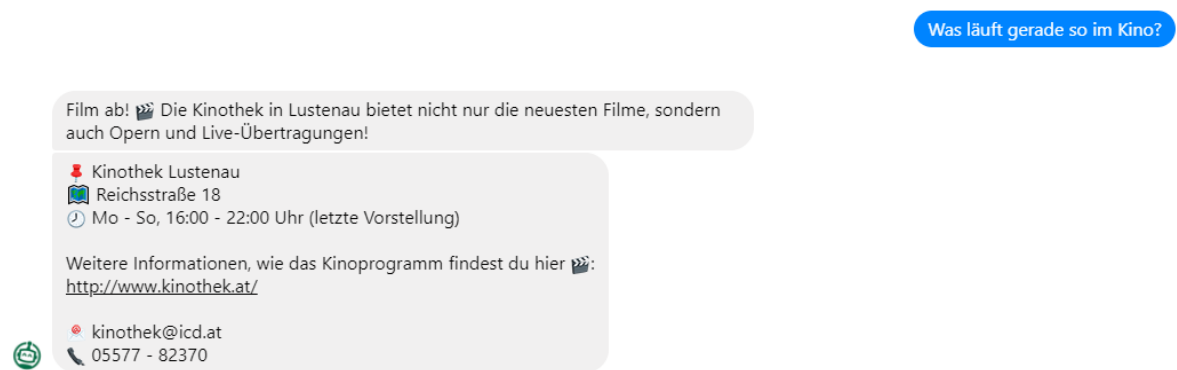


Abb. 64 Rückgabe auf Anfrage der Kinothek Lustenau

7.3.3 Gastronomie

Für eine Rückgabe im Bereich „Gastronomie“ wurde ebenfalls die Webseite der Marktgemeinde Lustenau als Referenz herangezogen. Auf dieser Seite befindet sich im Bereich Gastronomie eine Liste, in der die meisten Cafés, Restaurants und Bars aufgelistet sind.

Es wurde die Option „Gastronomie“ für den Nutzer in drei Teile geteilt, damit er seine Anfrage weiter spezifizieren kann:

- Café
- Restaurant
- Bar

In den drei Kategorien wurden jeweils drei bis vier Beispiele, aus der bereits erwähnten Liste manuell herausgeschrieben und in die Rückgabe des „*Chatbot*“ eingespeist. Die Liste wird der Anwenderin außerdem unter einem Link zur Verfügung gestellt (siehe Abbildung 65 auf Seite 73).



Die Lustenauer Eisdielen und Kaffeehäuser locken mit vielfältigen Kaffee-Variationen, ausgezeichneten Mehlspeis-Kreationen und einer unglaublichen Gastfreundlichkeit



Hier kannst du deinen Kaffee und Kuchen genießen:

Cafe König

Pontenstraße 25

Mo - So, 9:00 - 22:00 Uhr / Di geschlossen

Genauerer findest du auf der Homepage:

<http://www.cafekoenig.at/>

Eis Cafe Dolomiti

Kaiser-Franz-Josef-Straße 2

Täglich, 9:00–21:00 Uhr

Genauerer findest du auf der Facebook Seite:

<https://www.facebook.com/Eiscafe-Dolomiti-302449293264167/?f=143175935726586>

Bäckerei Mangold

Maria-Theresien-Straße 5

Mo - Sa, 6:00 - 18:15 Uhr / So, 7:00 - 18:00 Uhr

Genauerer findest du auf der Homepage:

<https://goo.gl/iaeKNr>



Eine Liste aller Bars und Clubs findest du hier:

<https://www.lustenau.at/de/freizeit/gastronomie/>

Abb. 65 Rückgabe von Cafés in Lustenau

7.3.4 Relaxen

Auch bei dieser Option gilt die Referenz der Homepage der Marktgemeinde Lustenau. Es wurden auch diese Daten statisch in den „Chatbot“ übertragen. Es wurde aber auch eigenes Wissen eingearbeitet. Bei der Rückgabe wird auf ein Gasthaus verwiesen, das sich in der Nähe des angegebenen Gebietes (Alter Rhein) befindet. Dabei wird auch wieder auf die Webseite des Gasthauses verlinkt. Auch ein Bild des besagten Erholungsgebietes wird der Anwenderin mitgesendet (siehe Abb. 66). Dieses ist im Bericht „Natur und Erholung“ auf der Homepage der Marktgemeinde Lustenau zu finden.


Text response

1 Einfach mal abschalten und die Natur genießen? 🌲🍁🍃
Dann ist das Natur und Erholungsgebiet perfekt für dich. 😊 Ob für
Spaziergänge, zum Baden 💧, Kaffee trinken ☕, oder einfach nur zum
Abschalten, im Naturpark ist für jeden was dabei. 😊

Und falls dann einmal der Hunger plagt, ist für Verpflegung im "Gasthaus Rohr"
gesorgt 🍽️🍷🍴:
<http://www.gasthausrohr.at/>

2 Enter a text response variant

Image



<https://www.lustenau.at/uploads/media/9>

Text response

1 Weitere Infos zum Naturpark findest du auf der Homepage 🌐:
<https://goo.gl/6hDrR1>

2 Enter a text response variant

Abb. 66 Implementierung des Natur- und Erholungsgebiets in „Dialogflow“

7.4 Abfahrtszeiten

Die Möglichkeit, Abfahrtszeiten vom Landbussen abzurufen, stellte eine komplexe Aufgabe dar, da der Verkehrsbund Vorarlberger keine öffentliche Schnittstelle zum Abruf der Daten bereitstellte. Nach gegenseitigem Austausch lehnten die Verantwortlichen auch einen eigens eingerichteten Zugang ab, jedoch verwiesen sie auf die Webseite „abfahrtszeiten.at“ (siehe Abb. 67), auf welcher die Daten bereitstehen.

Zur Lösung dieses Problems galt es, selbstständig eine passende *URL* für jede mögliche Haltestelle zusammenzubauen und den jeweiligen statischen *HTML-Body* auszulesen. Diese Struktur werde sich in den kommenden Jahren nicht ändern, versprachen die Verantwortlichen des Verkehrsbundes.



The screenshot shows the website 'abfahrtszeiten.at' with a navigation bar for 'LANDBUS UNTERLAND' and 'STADTBUS DORNBIRN'. A search bar contains 'Lustenau Kirchplatz' and a button 'anzeigen'. Below is a table of bus departures:

Plan	Linie	Ziel	Abfahrt in Minuten	
20:06	50	Dornbirn Bahnhof	7	Kirchplatz
20:13	53	Götzis Bahnhof	12	Kirchplatz
20:20	52	Lustenau Bhf-/Bundesstraße	20	Kirchplatz
20:36	52	Dornbirn Bahnhof		Kirchplatz
20:40	53	Lustenau Bahnhof		Kirchplatz
20:50	50	Gaißau Kesslerplatz		Kirchplatz

Abb. 67 Ausschnitt der Webseite „abfahrtszeiten.at“

7.4.1 Programmierung

Das Zusammenbauen der *URL* erfolgte im *WebHook*. Die *URL* bestand aus einem statischen Teil am Beginn und einer ID am Ende, welche die jeweilige Haltestelle repräsentierte. Jeder Haltestelle war genau eine ID zugeteilt, die sich jedoch nicht änderte.

7.4.1.1 Dialogflow

Action

REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST	PROMPTS
<input type="checkbox"/>	geo-city	@sys.geo-city	Sgeo-city	<input type="checkbox"/>	–
<input checked="" type="checkbox"/>	bushaltestellen	@bushaltestellen	Sbushaltestellen	<input type="checkbox"/>	Wo befindest du...
<input type="checkbox"/>	Enter name	Enter entity	Enter value	<input type="checkbox"/>	–

[+ New parameter](#)

Abb. 68 Ausschnitt des Abfahrtszeiten-Intent

Um die *URL* zu vervollständigen, musste die Anwenderin im Facebook Messenger eine Haltestelle angeben. Diese Haltestelle wurde dann in einem Parameter in der *JSON*-Anfrage mitübergeben. Zur Erkennung, was eine Haltestelle war und was nicht, wurde zusätzlich noch eine Entität *implementiert*. In dieser fand sich eine Auflistung aller Haltestellen jeweils mit und ohne Sonderzeichen:

bushaltestellen

Define synonyms Allow automated expansion

Search entries	
Am Schlatt	Am Schlatt
Augarten	Augarten
Badlochstraße	Badlochstraße
Bahngasse	Bahngasse
Bahnhof	Bahnhof
Bettleweg	Bettleweg
Bhf-/Bundesstraße	Bhf-/Bundesstraße
Binsfeld	Binsfeld
Brändlestraße	Brändlestraße
Carini Saal	Carini Saal

Abb. 69 Ausschnitt der Entität „Bushaltestellen“

7.4.1.2 WebHook

Die Abfahrtszeiten im *WebHook* wurden in zwei Funktionen aufgeteilt:

- Eine Funktion, die eine http-Anfrage sendete und eine passende Antwort generierte
- Eine Hilfsfunktion, die für die Generierung der passenden *URL* zuständig war (siehe Abb. 70)

```
273 def getIDfromHaltstelle(bushaltestelle):
274     halteID = "404"
275     if bushaltestelle == "Am Schlatt":
276         halteID = "211_16895"
277     elif bushaltestelle == "Augarten":
278         halteID = "211_17067"
279     elif bushaltestelle == "Badlochstraße"
280         halteID = "211_17069"
281     elif bushaltestelle == "Bahngasse":
282         halteID = "211_16893"
```

Abb. 70 Ausschnitt der Hilfsfunktion-Abfahrtszeiten im WebHook

Zunächst wurde der Parameter „Bushaltestelle“, der von Dialogflow mitgesendet wurde, ausgelesen. Diesem wurde dann in der Hilfsfunktion eine ID zugeteilt und die *URL* generiert.

```
225     haltenstelle_id = getIDfromHaltstelle(bushaltestelle)
226     createdUrl = "http://www.abfahrtszeiten.at/index.cfm?job=dsp&bk=1%2C2&htext=&HstId=" + haltenstelle_id
227     html = urlopen(createdUrl).read()
228     soup = BeautifulSoup(html)
229     tables = soup.findAll("table", { "class" : "cinfo" })
230     retVal = ""
231     count = 0
232     count_inside = 0
233     for table in tables:
234         for row in table.findAll("tr"):
235             if count > 2: #erste 2 zeilen nicht ausgeben (headings)
236                 for col in row.findAll("td"):
237                     if count_inside == 0:
238                         retVal += "Plan: " + col.getText() + "\n"
239                     if count_inside == 1:
240                         retVal += "Linie: " + col.getText() + "\n"
241                     if count_inside == 2:
242                         retVal += "Ziel: " + col.getText() + "\n\n"
243                     count_inside = count_inside + 1
244                 count = count + 1
245                 count_inside = 0
```

Abb. 71 Codeausschnitt der Abfahrtszeiten-Funktion im WebHook

Danach erfolgte der *http-Request* und das Auslesen der Antwort. Dazu wurde die Bibliothek „BeautifulSoup“ verwendet. Dieses ermöglicht es, Teilelemente aus dem gesamten HTML zu extrahieren und beliebig zusammenzufügen und auszulesen. Die Daten standen in einer Tabelle, in nächsten Schritt wurde daher die Daten jeder vorhandenen Zeile ausgelesen und zusammengefügt. Die Zeilenanzahl war hier variabel, daher erfolgte das Auslesen in einer Schleife (siehe Abb. 71).

7.4.1.3 Ergebnis

Schlussendlich wurden der Anwenderin nach Eingabe der Haltestelle die Abfahrtszeiten der nächsten (maximal) zehn Landbusse angezeigt (siehe Abb. 72). Eine größere Anzahl war nicht möglich, waren auf der Webseite doch nicht mehr Daten vorhanden.

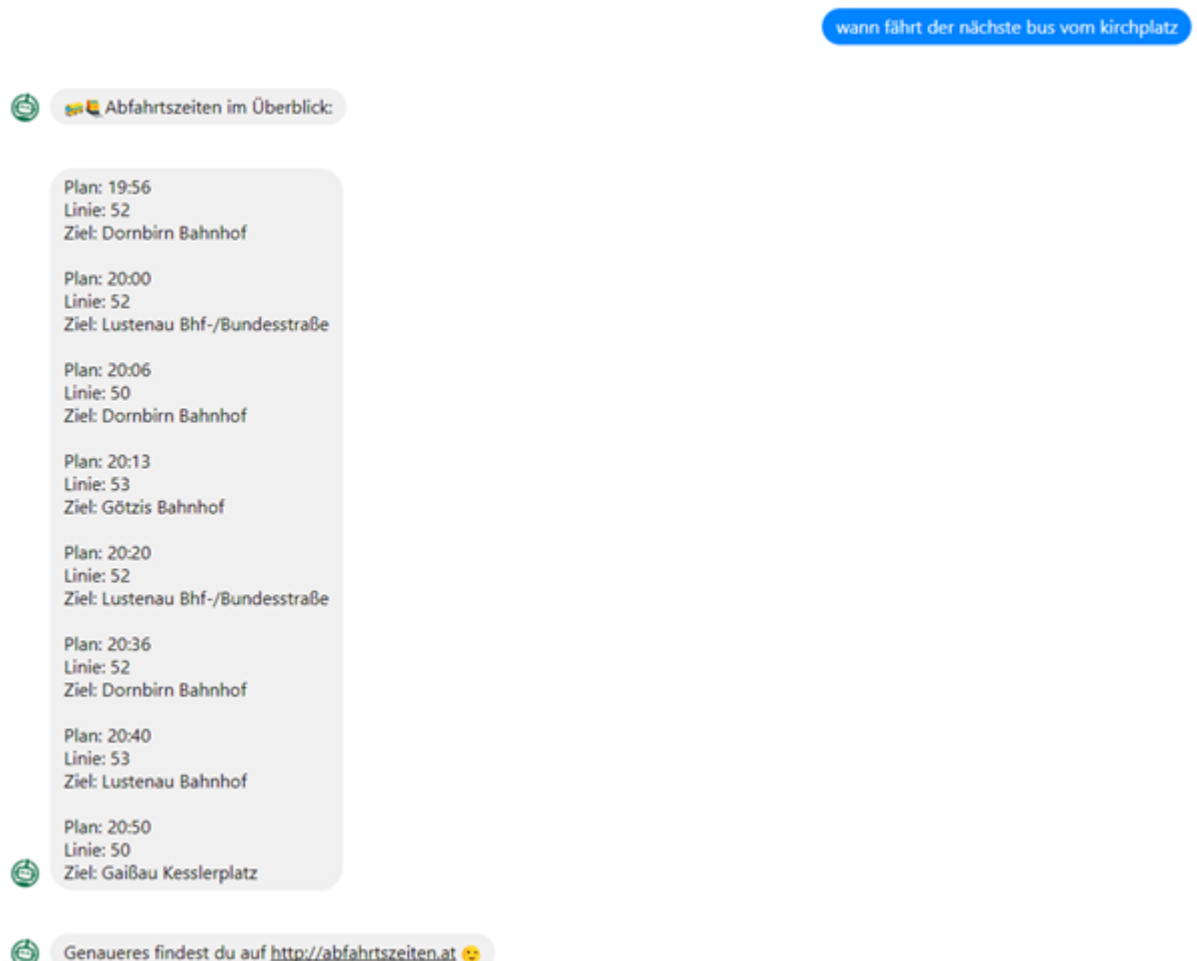


Abb. 72 Möglicher Dialog zum Thema „Abfahrtszeiten“

7.5 Kontakt zur Gemeinde

Die Kontakte zur Gemeinde der jeweiligen Einrichtungen sind von den entsprechenden Webseiten übernommen worden. Die zentrale Quelle war die Homepage der Marktgemeinde. Diese enthält z.B. Kontaktinformationen zum Rathaus, zum Parkstadion, zum Jugendplatz Habedere, zum Parkbad und vielen mehr. Dadurch, dass die Daten manuell bzw. statisch *implementiert* wurden, wurden sie durch einen normalen „*Intent*“, d.h. nicht durch einen „*WebHook*“ zur Verfügung gestellt.

7.6 Smalltalk mit „Chatbot“

Smalltalk wurde nicht im „*WebHook*“, sondern in Dialogflow umgesetzt. Der Sinn dahinter war, den Benutzer auf witzige und natürliche Weise mit dem „*Chatbot*“ kommunizieren zu lassen. Neben den konversationsähnlichen Antworten wurde dazu auch eigens einige *Intents implementiert*.

7.6.1 Erzählen von Witzen

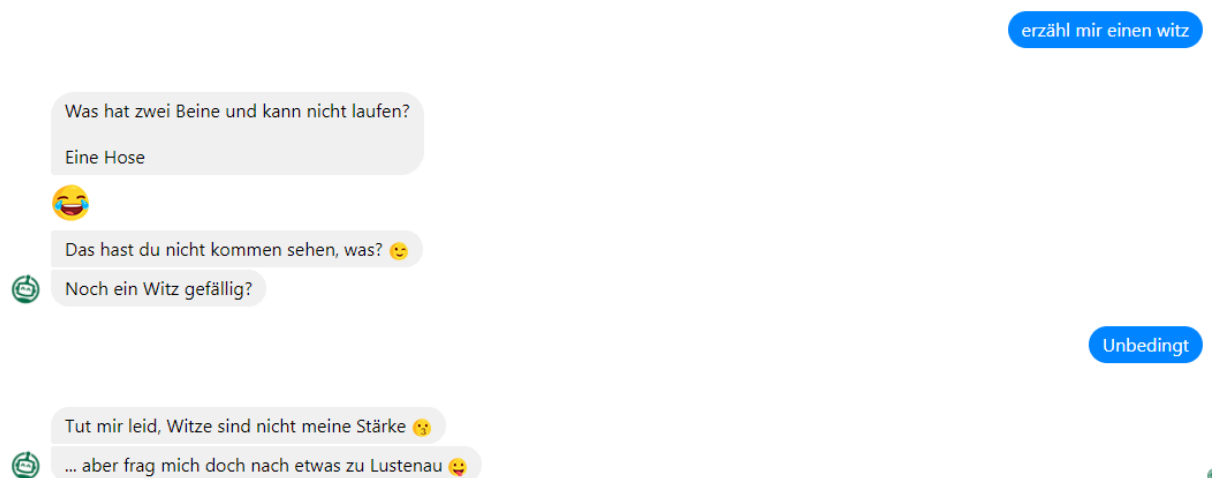


Abb. 73 Konversation im Witz-Kontext

Ein Punkt war die Möglichkeit, Witze abzurufen. Dazu wurden in Dialogflow rund zehn Witze definiert, welche dann zufällig ausgegeben werden. Nach dem ersten Witz erfolgt die Ausgabe zweier Quick-Reply-Buttons. Hier kann die Anwenderin wählen, ob sie noch einmal einen Witz lesen möchte. Ist dies der Fall, wird der Nachfolge-*Intent* aufgerufen und die Standardantwort „Tut mir leid, Witze sind nicht meine Stärke“ ausgegeben (siehe Abb. 73 und 74).

Der Nachfolge-*Intent* kann durch das Benützen von Kontexten aufgerufen werden (siehe Abb. 75). Dadurch wird auch vermieden, dass er durch eine sonstige Eingabe aufgerufen wird.

[35] (Liu, 2018)

Contexts

Add input context

1 zweiter-witz ⊗ Add output context

User says

” Add user expression

” Witz

” kennst du einen spaß

” erzähl einen witz

Abb. 74 Witz-Intent mit Output-Kontext

Contexts

zweiter-witz ⊗ Add input context

Add output context

User says

” Add user expression

” warum nicht

” ja

” unbedingt

” yes

Abb. 75 Nachfolge-Intent mit Input-Kontext

7.6.2 Grußformel

Die Anwenderin sollte den „*Chatbot*“ wie einen realen Menschen begrüßen und eine Antwort darauf erhalten können (siehe Abb. 76), weshalb eigens ein „*Intent*“ dafür *implementiert* wurde.



Abb. 76 Grußformel-Intent im Facebook Messenger dargestellt

Die Antworten variieren hierbei:



Abb. 77 Antwort auf eine Grußformel, dargestellt in Dialogflow

7.6.3 Vulgärsprache

Sollte ein Benutzer einen Fäkalausdruck benutzen, erfolgt hier eine Antwort darauf. Dazu wurde eigens eine Entität aller Fäkalausdrücke *implementiert*. Wird einer dieser Ausdrücke benutzt, erfolgt eine Antwort in diesem *Intent* (siehe Abb. 78). Auch hier wurden die Antworten variiert, von einem einfachen Zurechtstutzen bis hin zu einer frechen Antwort (siehe Abb. 79).

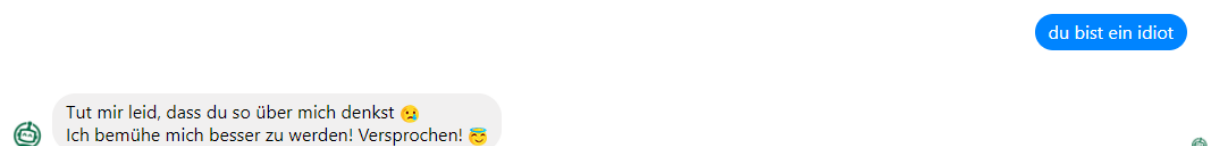


Abb. 78 Antwort auf das Benützen eines Fäkalausdrucks

lustenauBot

Enter action name

REQUIRED	PARAMETER NAME	ENTITY	VALUE
<input type="checkbox"/>	curse-words	@curse-words	Scurse-words
<input type="checkbox"/>	Enter name	Enter entity	Enter value

+ New parameter

Response

DEFAULT FACEBOOK MESSENGER +

Text response

- 1 Das ist aber nicht nett :(
- 2 "Scurse-words", ist das alles was du drauf hast 😞😞
Und ich dachte du hättest Klasse ...
- 3 Tut mir leid, dass du so über mich denkst :(
Ich bemühe mich besser zu werden! Versprochen! O:)
- 4 Enter a text response variant

Abb. 79 Die Entität wird im „Intent“ zugeordnet

7.6.4 Allgemeine Informationen

Speziell für neue Benutzer und Anwenderinnen war es wichtig, Informationen zu erhalten, was der lustenauBot eigentlich ist und welche Informationen er liefern kann. Aus diesem Grund wurde ein *Intent implementiert*, der diese Frage beantworten sollte (siehe Abb. 80).

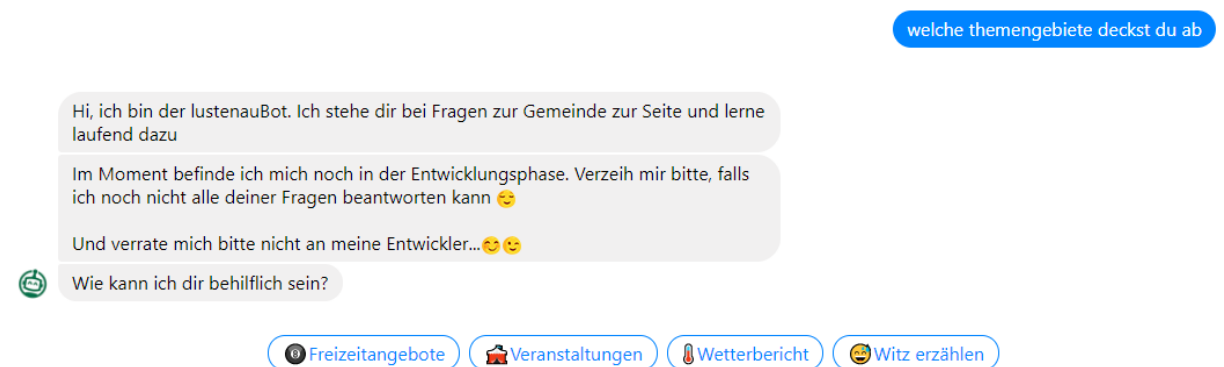


Abb. 80 Abrufen allgemeiner Informationen über den lustenauBot

lustenauBot

Hierfür wurde keine Funktion im „WebHook“ benötigt, sondern lediglich zwei Antworten im Textformat und eine im Quick-Reply-Format. Diese wurden im Reiter „Response“ in Dialogflow *implementiert*.

The image shows a screenshot of the Dialogflow interface. It consists of three main sections:

- Text response:** A light gray box containing two numbered entries:
 - 1 Hi, ich bin der lustenauBot. Ich stehe dir bei Fragen zur Gemeinde zur Seite und lerne laufend dazu
 - 2 Enter a text response variant
- Text response:** A second light gray box containing two numbered entries:
 - 1 Im Moment befinde ich mich noch in der Entwicklungsphase. Verzeih mir bitte, falls ich noch nicht alle deiner Fragen beantworten kann 😊
Und verrate mich bitte nicht an meine Entwickler...😁😁
 - 2 Enter a text response variant
- Quick replies:** A white box with a title "Quick replies" and two icons (a question mark and a trash can). Below the title is a list of five items:
 - Wie kann ich dir behilflich sein?
 - 🕒 Freizeitangebote
 - 📅 Veranstaltungen
 - 🌤️ Wetterbericht
 - 😄 Witz erzählen

Abb. 81 Antwort auf die Anfrage zu allgemeinen Informationen über den „Chatbot“

8 Testphase

Das Testen des „*Chatbot*“ bestand grundsätzlich aus manuellen Tests. Dabei wurden neue Funktionen laufend über das Userinterface des „*Chatbot*“ getestet. Durch diese Art des Testens, konnten mögliche Fehler rasch entdeckt und somit behoben werden. Des Weiteren wurden am Tag der offenen Tür der HTL Dornbirn Tests mit externen Usern durchgeführt.

8.1 Tag der offenen Tür der HTL Dornbirn

Am Tag der offenen Tür der HTL Dornbirn (TdoT) wurde die erste Testphase mit Usern durchgeführt. Dabei wurden die Besucher dazu aufgefordert, ihre eigenen Fragen zu den vorhandenen Themengebieten über das Chatfenster des Facebook Messengers einzugeben. Dadurch konnte in Erfahrung gebracht werden, wie sich die User in dieser Situation verhalten und wie sie die Fragen stellen. Anfallende Probleme oder Ergänzungen, welche aus den Aktivitäten der Tester in Erfahrung gebracht werden konnten, wurden zeitgleich behoben bzw. umgesetzt. Durch die Testphase am TdoT konnte der „*Chatbot*“ in einzelnen Details verbessert werden.

8.2 Laufende Tests

Während der gesamten Entwicklungsphase des „*Chatbot*“ wurden laufend manuelle Tests durchgeführt. Diese Tests bestanden aus Fragen, welche dem Bot gestellt wurden, um so Erkenntnisse darüber zu gewinnen, wie sich der „*Chatbot*“ auf die einzelnen Fragen verhält. Auf Grundlage dieser Ergebnisse konnten die Antwortmöglichkeiten des Bots verbessert werden. Weiters konnten dadurch neue Schlüsselwörter hinzugefügt werden.

IustenauBot

Auf dem unten abgebildeten Screenshot (Abb. 82) sind mögliche unterschiedliche Fragestellungen zum Thema Witze dargestellt. Durch die Erweiterung der Schlüsselwörter, welche in den Testphasen erfasst wurden, konnten die unterschiedlichen Fragestellungen erweitert werden.

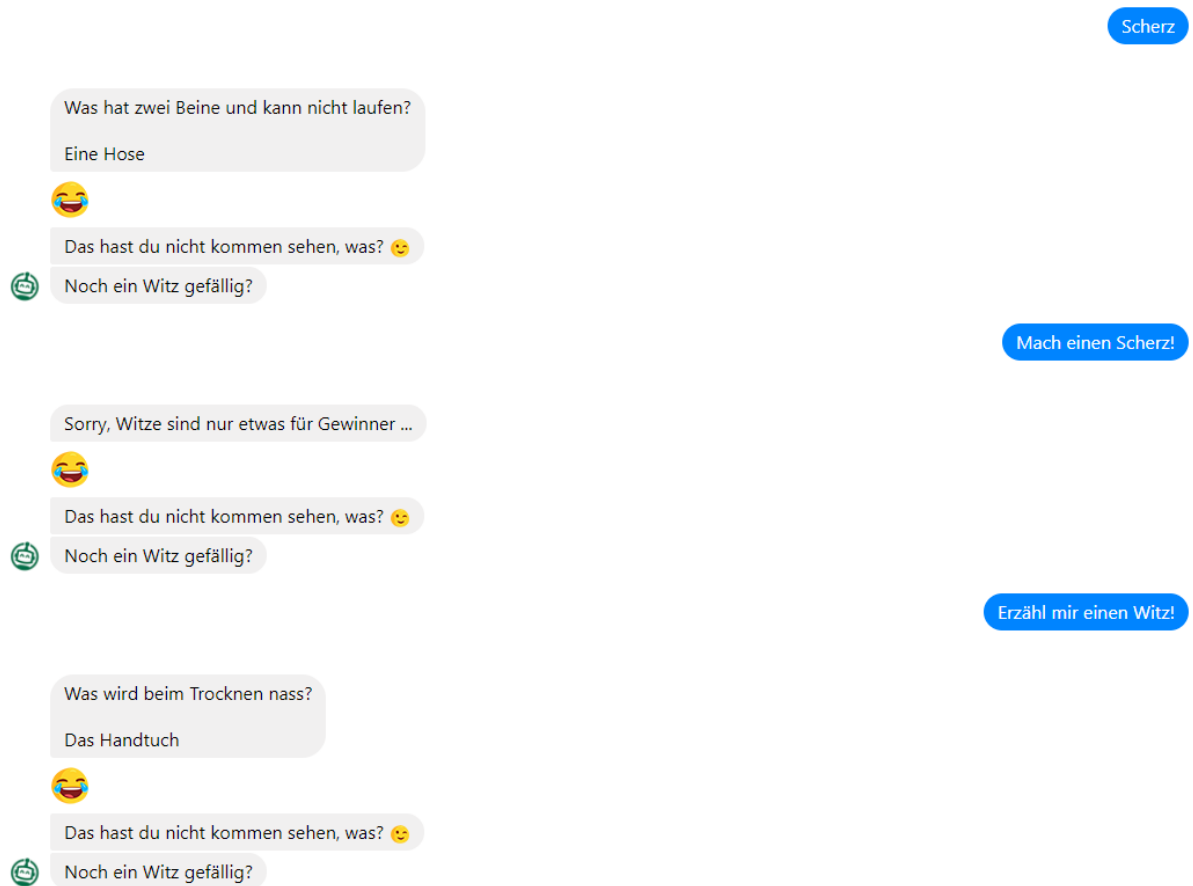


Abb. 82 Ausschnitt einer Witz-Konversation

9 Fazit

Die Aufgabestellung dieser Diplomarbeit war die Entwicklung eines Facebook Messenger „Chatbot“, welcher Informationen zu jugendrelevanten Themen in Lustenau bereitstellen kann. Dass der „Chatbot“ für den Facebook Messenger realisiert wurde, ist auf den Wunsch des Auftraggebers zurückzuführen.


Das Projekt verlief für das Projektteam sehr gut. Alle Projektteammitglieder konnten ihre Fähigkeiten im Bereich der Teamarbeit immens erweitern und dabei wertvolle Erfahrungen sammeln. Somit stellte das Projekt einen wichtigen Fortschritt für alle dar.


Eine Erkenntnis, welche das Team während der Projektlaufzeit gewinnen durfte, war das Einschätzen der Zeit, die man für das Entwickeln eines einzelnen Bestandteils einer Software benötigt. Dies ist auf Probleme, die während des Programmierens auftauchen, zurückzuführen, dessen Zeitaufwand im Vorhinein nicht abgeschätzt werden kann. Zusätzlich wurde erkannt, dass ein professionelles Projektmanagement ein entscheidender Erfolgsfaktor in der Softwareentwicklung ist.


Weiters konnte das Projektteam seine Programmierfähigkeiten, im speziellen im Bereich der Python-Programmierung verbessern bzw. erweitern. Diese erworbenen Fähigkeiten werden auch bei zukünftigen Projekten einen klaren Vorteil bieten.


Als Ergebnis des Projektes entstand ein funktionstüchtiger „Chatbot“, der als Alltagshelfer für Lustenauer Jugendliche dient. Es ist geplant, dass die Marktgemeinde Lustenau die Betreuung des lustenauBot übernehmen wird.

10 Hilfsmittel


<p>abfahrtszeiten.at https://abfahrtszeiten.at/</p>	 <p>Abb. 83 Abfahrtszeiten.at - Logo - 2018</p>
<p>Abfahrtszeiten.at ist eine Webseite, auf der Busabfahrtszeiten in Echtzeit abgerufen werden können.</p>	


<p>All-Inkl https://all-inkl.com/</p>	 <p>Abb. 84 All-inkl.com - Logo - 2018</p>
<p>All-Inkl ist einer der führenden <i>Webhoster</i> und <i>Provider</i> für <i>Domains</i>. Auf All-Inkl wurde die Datenbank <i>gehostet</i>.</p>	

<p>Atom https://atom.io/</p>	 <p>Abb. 85 Atom - Logo - 2018</p>
<p>Atom ist ein Open-Source-Texteditor mit GitHub-Anbindung. Atom wurde für die Python Programmierung verwendet.</p>	


<p>Beautiful Soup https://crummy.com/software/BeautifulSoup/</p>	 <p>Abb. 86 Beautiful Soup - Logo - 2018</p>
<p>Beautiful Soup ist eine freie Programmibibliothek, die zum Auslesen von Daten der Seite abfahrtszeiten.at verwendet wurde.</p>	


lustenauBot


<p>cURL https://curl.haxx.se/</p>	 <p>Abb. 87 cURL - Logo - 2018</p>
<p>cURL (Client for <i>URLs</i>) ist eine Programmbibliothek und ein Kommandozeilen-Programm zur Übertragung von Dateien in Rechnernetzen. Curl wurde für die Startmessage und den Startbutton des Messengers verwendet.</p>	


<p>Dialogflow https://dialogflow.com/</p>	 <p>Abb. 88 Dialogflow - Logo - 2018</p>
<p>Dialogflow (früher Api.ai) ist eine von Google entwickelte Plattform zur Textverarbeitung. Dialogflow wurde zur Textverarbeitung der Usereingaben verwendet.</p>	


<p>Facebook / Facebook Messenger https://facebook.com/</p>	 <p>Abb. 89 Facebook / FB-Messenger - Logo - 2018</p>
<p>Facebook ist ein soziales Netzwerk. Der Facebook Messenger ist eine Anwendung für Text und Audio Kommunikation. Der Facebook Messenger stellt das <i>Frontend</i> des lustenauBot dar.</p>	

<p>Flask http://flask.pocoo.org/</p>	 <p>Abb. 90 Flask - Logo - 2018</p>
<p>Flask ist ein in Python geschriebenes Webframework. Das <i>Micro-Framework</i> Flask wurde für die Web-App-Konfiguration verwendet.</p>	


<p>GitHub https://github.com/</p>	 <p>Abb. 91 GitHub - Logo - 2018</p>
<p>GitHub ist ein webbasierter Online-Dienst, der Software-Entwicklungsprojekte in Form von <i>Filehosting</i> auf seinen Servern bereitstellt. GitHub wurde zur Sourcecode-Verwaltung verwendet.</p>	


<p>Google Chrome https://google.com/chrome</p>	 <p>Abb. 92 Google Chrome - Logo - 2018</p>
<p>Google Chrome ist ein Webbrowser des Unternehmens Google. Er ist der derzeit am weitesten verbreitete Browser.</p>	

<p>Google Cloud Platform https://cloud.google.com/</p>	 <p>Abb. 93 Google Cloud Platform - Logo - 2018</p>
<p>Google Cloud Plattform ist ein Programmpaket von Cloud-Computing-Diensten. Dabei wird neben einer Reihe von Management-Tools, ein Cloud-Service angeboten, der die Datenspeicherung, Datenanalyse und maschinelles Lernen beinhaltet. Auf der Google Cloud Plattform wurde Dialogflow <i>gehostet</i>.</p>	


<p>Google Drive https://google.com/drive/</p>	 <p>Abb. 94 Google Drive - Logo - 2018</p>
<p>Google Drive ist ein <i>Filehosting</i>-Dienst, welcher Benutzern das Speichern von Dokumenten in der Cloud ermöglicht, außerdem ist das Teilen von gespeicherten Dateien und das gemeinsame Bearbeiten möglich. Google Drive wurde für das Speichern und Bearbeiten aller projektrelevanten Dokumente verwendet.</p>	


lustenauBot


<p>Heroku https://heroku.com/</p>	 <p>HEROKU <i>Abb. 95 Heroku - Logo - 2018</i></p>
<p>Heroku ist eine Cloud-Plattform, die als Bereitstellungsmodell für Webanwendungen verwendet wird, sie unterstützt unter anderem Python. Heroku wurde für das <i>Hosting</i> des Python Codes verwendet.</p>	


<p>Marktgemeinde Lustenau https://lustenau.at/</p>	<p>Marktgemeinde Lustenau  <i>Abb. 96 Marktgemeinde Lustenau - Logo - 2018</i></p>
<p>Die Homepage der Marktgemeinde Lustenau diente als Informationsgrundlage für den lustenauBot. Unter anderem werden die Veranstaltungsinfos aus der Datenbank der Marktgemeinde Lustenau abgerufen.</p>	


<p>Messenger API https://developers.facebook.com/docs/messenger-platform</p>	 <p>facebook for developers <i>Abb. 97 Messenger API - Logo - 2018</i></p>
<p>Die Messenger API (Messenger-Plattform) ist eine Toolbox zur Erstellung von „Chatbots“. Die Messenger API stellt die Entwickler Ebene des Facebook Messenger dar, sie wurde für die Entwicklung des Bots verwendet.</p>	


<p>Microsoft Excel https://products.office.com/excel</p>	 <p><i>Abb. 98 Microsoft Excel - Logo - 2018</i></p>
<p>Microsoft Excel ist das am weitesten verbreitete Tabellenkalkulationsprogramm. Es wurde zur Zeiterfassung und für verschiedenste Terminpläne verwendet.</p>	


<p>Microsoft Outlook https://products.office.com/outlook</p>	 <p><i>Abb. 99 Microsoft Outlook - Logo - 2018</i></p>
<p>Microsoft Outlook ist eine verbreitete Software zum Empfangen und Versenden von E-Mails. Outlook wurde zur Kommunikation mit dem Projektbetreuer verwendet.</p>	


<p>Microsoft PowerPoint https://products.office.com/powerpoint</p>	 <p><i>Abb. 100 Microsoft PowerPoint - Logo - 2018</i></p>
<p>Microsoft PowerPoint ist ein Präsentationsprogramm. Es wurde zur Erstellung der Zwischen- und der Abschlusspräsentation verwendet.</p>	


<p>Microsoft Visio https://products.office.com/visio</p>	 <p><i>Abb. 101 Microsoft Visio - Logo - 2018</i></p>
<p>Microsoft Visio ist ein Visualisierungsprogramm von Microsoft. Visio wurde zur Erstellung von Diagrammen und Übersichten verwendet. Unter anderem wurde der PSP mit Visio erstellt.</p>	


<p>Microsoft Word https://products.office.com/word</p>	 <p><i>Abb. 102 Microsoft Word - Logo - 2018</i></p>
<p>Microsoft Word bezeichnet ein Textverarbeitungsprogramm von Microsoft. Microsoft Word wurde für das Schreiben der Dokumentation, der Statusberichte und der Sitzungsprotokolle verwendet.</p>	


<p>MySQL https://mysql.com/</p>	 <p>Abb. 103 MySQL - Logo - 2018</p>
<p>MySQL ist eines der weltweit verbreitetsten relationalen Datenbankverwaltungssysteme. Die Veranstaltungsdatenbank ist als MySQL Datenbank ausgeführt.</p>	


<p>OpenWeatherMap https://openweathermap.org/</p>	 <p>Abb. 104 OpenWeatherMap - Logo - 2018</p>
<p>OpenWeatherMap ist ein Online-Dienst, der eine frei nutzbare Programmierschnittstelle <i>API</i> für Wetterdaten bzw. Wettervorhersagen bereitstellt. Diese <i>API</i> wurde für den Wetterbericht des „Chatbot“ verwendet.</p>	


<p>Adobe Photoshop https://adobe.com/photoshop</p>	 <p>Abb. 105 Adobe Photoshop - Logo - 2018</p>
<p>Adobe Photoshop ist ein Bildbearbeitungsprogramm des US-amerikanischen Softwarehersteller Adobe Systems. Adobe Photoshop wurde zur Erstellung des Logos verwendet.</p>	

<p>Postman https://getpostman.com/</p>	 <p>Abb. 106 Postman - Logo - 2018</p>
<p>Postman ist der meistgenutzte <i>REST</i>-Client weltweit. Postman bietet eine intuitive Benutzeroberfläche zum Senden von Anfragen an einen <i>REST</i>-Server und das Speichern seiner Antworten.</p>	

<p>Python https://python.org/</p>	 <p>Abb. 107 Python - Logo - 2018</p>
<p>Python ist eine universelle, meist interpretierte höhere Programmiersprache. Python zeichnet sich durch einen lesbaren und knappen Programmierstil aus. Python wurde unter anderem für die Übersetzung der englischen Fachwörter (Wetter) verwendet.</p>	

<p>Sourcetree https://sourcetreeapp.com/</p>	 <p>Abb. 108 Sourcetree - Logo - 2018</p>
<p>Sourcetree ist ein Produkt des Unternehmen Atlassian und vereinfacht die <i>Interaktion</i> mit einem Git-Repository. Es unterstützt die Verwaltung des Programmcodes durch die visualisierte Anzeige des Sourcetree.</p>	

<p>Trello https://trello.com/</p>	 <p>Abb. 109 Trello - Logo - 2018</p>
<p>Trello ist eine web-basierte Projektmanagementsoftware, dabei können auf sogenannten Boards mit Teammitgliedern Listen erstellt werden, welche dann nach dem Kanban-Prinzip abgearbeitet werden. Für die Diplomarbeit IustenauBot wurde ein Trello Board erstellt.</p>	

<p>WhatsApp https://whatsapp.com/</p>	 <p>Abb. 110 WhatsApp - Logo - 2018</p>
<p>WhatsApp ist ein Messenger, über welchen Benutzer Textnachrichten, Bilder, Videos, Sprachaufnahmen, Standortinformationen, Dokumente und Kontaktdaten zwischen zwei Personen oder Gruppen austauschen können. WhatsApp wurde als Kommunikationsinstrument verwendet, über eine WhatsApp-Gruppe wurde der größte Teil der projektinternen Kommunikation abgewickelt.</p>	

11 Glossar

Begriff	Beschreibung (Erklärung)
Algorithmus	Klare Abhandlungsvorschrift zur Problemlösung
API	Programmierschnittstelle
Array	listenähnliches Objekt
Branch	gleichzeitige Entwicklung mehrerer Softwareversionen
Build	Erstellungsprozess
Button	Taste
Chatbot	Textbasiertes Dialogsystem
Console	Kommandozeile
Deployment	Softwareverteilung
Developer	Entwickler
Developer Plattform	Facebook-Entwickler-Plattform
Domain	eindeutiger Name im Internet
Emoticons	Smileys, zeigen Gefühlszustände
Framework	Programmiergerüst
Frontend	Teil der Anwendung, die nahe am Benutzer ist
hosten	bereitstellen von Infrastruktur für Internetkunden
HTML-Body	HTML-Element, das den Inhalt des HTML-Dokuments enthält
http-Response/Request	Browser-Antwort/Anfrage
Implementierung	Umsetzung eines Entwurfs
Intent	abstrakte Beschreibung einer auszuführenden Operation
Interaktion	Wechselseitiges Handeln zwischen zwei Partnern
JSON	Dateiformat für Datenaustausch zwischen zwei Anwendungen

IustenauBot

Page Access Token	eindeutige Nummer für den Zugriff auf die Messenger <i>API</i>
Paketmanager	Software zur Verwaltung von Softwareteilen
posten	neuen Eintrag auf Facebook verfassen
Profile	Seite auf Facebook, die den IustenauBot darstellt
Provider	Anbieter, der eine Internet-Dienstleistung anbietet
reply	Antwort
request	Anfrage
REST	Web Service
String	Zeichenkette
URL	Link, der zu einer Webseite führt
User Interface	Benutzerschnittstelle
WebHook	Verfahren zur Kommunikation zwischen Servern
Webhost	Bereitsteller von Ressourcen für Webseiten

12 Autorenverzeichnis

Kapitel / Abschnitt	Autor	Seite
Eidesstaatliche Erklärung	Ott, Kofler, Waibel	I
Abstract	Ott	II
Vorwort und Danksagung	Ott	III
geschlechtsneutrale Formulierung	Ott	IV
Textformatierung	Ott	V
1 Impressum	Ott	1-3
2 Marktgemeinde Lustenau	Ott	4
3 Projektmanagement	Ott	5-19
4.1 Softwarearchitektur	Kofler	20-21
4.2 Facebook Messenger	Waibel	21
4.3 Textverarbeitung	Kofler	21-23
4.4 Konzeption	Waibel	24-25
4.5 Use-Case-Diagramm	Kofler	26
4.6 Datenquellen	Waibel	27-29
5.1 Messenger API	Waibel	30
5.2 Dialogflow	Waibel	31-37
5.3 Git	Kofler	38-39
5.4 Python	Kofler	40-45
5.5 Facebook Messenger	Waibel	45-46
5.6 Heroku	Kofler	46-48
6 Design	Kofler	49-55
7.1 Wetterbericht	Kofler	56-62
7.2 Jugendveranstaltungen	Waibel	63-69

7.3 Freizeitangebote	Waibel	70-74
7.4 Abfahrtszeiten	Kofler	75-78
7.5 Kontakt zur Gemeinde	Waibel	79
7.6 Smalltalk mit Chatbot	Kofler	79-83
8 Testphase	Ott	84-85
9 Fazit	Ott	86
10 Hilfsmittel	Ott	87-93
11 Glossar	Ott	94-95
12 Autorenverzeichnis	Ott	96-97
13 Abbildungsverzeichnis	Ott	98-102
14 Tabellenverzeichnis	Ott	103
15 Abkürzungsverzeichnis	Ott	104
16 Literaturverzeichnis	Ott	105-108
17 Anhang	Ott	109-111

13 Abbildungsverzeichnis

Abb. 1 Jakob Ott	1
Abb. 2 Matteo Kofler	1
Abb. 3 Manuel Waibel	2
Abb. 4 Diethard Kaufmann	3
Abb. 5 Benno Kofler	3
Abb. 6 Projektorganigramm.....	10
Abb. 7 Projektstrukturplan.....	11
Abb. 8 Risikoanalyse.....	14
Abb. 9 Projektumweltanalyse	16
Abb. 10 Projektabschlussbericht	19
Abb. 11 Typische Softwarearchitektur eines Chatbot [Pluut Interaction B.V.]	20
Abb. 12 Use-Case-Diagramm	26
Abb. 13 Allgemeine Anfragenverarbeitung des „Chatbot“	30
Abb. 14 Eingabe des „Page Access Token“ in „Dialogflow“	31
Abb. 15 Anfrage mit „Quick-Replies“ als Antwort	34
Abb. 16 Anfrage mit ausgewähltem „Quick-Reply“ Knopf „Sport“	35
Abb. 17 Nachrichtenfluss bei WebHook Aufruf	36
Abb. 18 Beispielsatz mit markierten "Entities"	37
Abb. 19 Darstellung der in der obigen Abb. 17 erkannten „Entities“	37
Abb. 20 Oberfläche von GitHub-Desktop inkl. Anzeige der einzelnen Branches	38
Abb. 21 Historien-Ansicht in GitHub.....	38
Abb. 22 Auflistung aller Pakete	40
Abb. 23 Inhalt des Profils	40
Abb. 24 Ausschnitt aus der app.py Datei	41

lustenauBot

Abb. 25 Inhalt der Procfile-Datei	41
Abb. 26 WebHook-Konfiguration in Dialogflow.....	42
Abb. 27 Implementierung von Flask.....	42
Abb. 28 Zuteilung der korrekten Funktion bei der Anfrage	43
Abb. 29 JSON der Anfrage bei der Benutzereingabe „Wetter“	44
Abb. 30 Konfiguration des WebHook in „Dialogflow“.....	44
Abb. 31 Rückgabe im JSON-Format.....	45
Abb. 32 „Access-Token“ in „Facebook Developer Console“ generieren	46
Abb. 33 Funktionalität von Heroku im Projekt	47
Abb. 34 Auto Deployment Option auf Heroku	48
Abb. 35 Logging-Funktion in Heroku.....	48
Abb. 36 Mockups Facebook Messenger „Chatbot“	49
Abb. 37 Elemente der Messenger UI	50
Abb. 38 Logo des lustenauBot	50
Abb. 39 Facebook Seite des lustenauBot	51
Abb. 40 Titelbild der Facebook Seite	52
Abb. 41 Willkommensbildschirm des lustenauBot.....	52
Abb. 42 Quick-Reply-Buttons im Facebook Messenger.....	53
Abb. 43 Verschiedene Interaktionsmöglichkeiten für Anwenderin im FB-Messenger	53
Abb. 44 Reaktion des Bots auf ein gesendetes Bild der Anwenderin.....	54
Abb. 45 Sammlung von Emoticons	54
Abb. 46 Emoticons werden auf allen Plattformen unterschiedlich wahrgenommen ..	55
Abb. 47 Beispiel für die Verwendung von Emoticons im Smalltalk.....	55
Abb. 48 URL-Funktion der YahooWeather-Abfrage	57
Abb. 49 Ausschnitt aus der Übersetzungsfunktion der YahooWeather-Abfrage	58
Abb. 50 Rückgabe einer Abfrage von OpenWeatherMap	58

Abb. 51 Schlüsselwörter des aktuellen Wetters	59
Abb. 52 Schlüsselwörter der Wettervorhersage	59
Abb. 53 Registrierung des Keys bei OpenWeatherAPI	60
Abb. 54 Codeausschnitt des aktuellen Wetterberichts	61
Abb. 55 Abfrage des aktuellen Wetters durch die Anwenderin im FB-Messenger	61
Abb. 56 Zweite Funktion der Wettervorhersage im WebHook	62
Abb. 57 Abfrage der Wettervorhersage durch den Benutzer im FB-Messenger	62
Abb. 58 Relevante Tabellen für Veranstaltungen.....	64
Abb. 59 „Select-Statement“ zum Selektieren der Veranstaltungsdaten	66
Abb. 60 Beispiel-Ausgabe eines Events	68
Abb. 61 Anfrage einer Anwenderin für eine Veranstaltung	68
Abb. 62 Darstellung der Sportveranstaltungen.....	69
Abb. 63 Statische Implementierung der Daten der Kinothek in „Dialogflow“	71
Abb. 64 Rückgabe auf Anfrage der Kinothek Lustenau	72
Abb. 65 Rückgabe von Cafés in Lustenau	73
Abb. 66 Implementierung des Natur- und Erholungsgebiets in „Dialogflow“	74
Abb. 67 Ausschnitt der Webseite „abfahrtszeiten.at“	75
Abb. 68 Ausschnitt des Abfahrtszeiten-Intent	76
Abb. 69 Ausschnitt der Entität „Bushaltestellen“	76
Abb. 70 Ausschnitt der Hilfsfunktion-Abfahrtszeiten im WebHook	77
Abb. 71 Codeausschnitt der Abfahrtszeiten-Funktion im WebHook.....	77
Abb. 72 Möglicher Dialog zum Thema „Abfahrtszeiten“	78
Abb. 73 Konversation im Witz-Kontext.....	79
Abb. 74 Witz-Intent mit Output-Kontext.....	80
Abb. 75 Nachfolge-Intent mit Input-Kontext	80
Abb. 76 Grußformel-Intent im Facebook Messenger dargestellt	81

Abb. 77 Antwort auf eine Grußformel, dargestellt in Dialogflow	81
Abb. 78 Antwort auf das Benützen eines Fäkalausdrucks	81
Abb. 79 Die Entität wird im „Intent“ zugeordnet.....	82
Abb. 80 Abrufen allgemeiner Informationen über den lustenauBot.....	82
Abb. 81 Antwort auf die Anfrage zu allgemeinen Informationen über den „Chatbot“ .	83
Abb. 82 Ausschnitt einer Witz-Konversation	85
Abb. 83 Abfahrtszeiten.at - Logo - 2018.....	87
Abb. 84 All-inkl.com - Logo - 2018	87
Abb. 85 Atom - Logo - 2018	87
Abb. 86 Beautiful Soup - Logo - 2018	87
Abb. 87 cURL - Logo - 2018.....	88
Abb. 88 Dialogflow - Logo - 2018	88
Abb. 89 Facebook / FB-Messenger - Logo - 2018	88
Abb. 90 Flask - Logo - 2018.....	88
Abb. 91 GitHub - Logo - 2018	89
Abb. 92 Google Chrome - Logo - 2018	89
Abb. 93 Google Cloud Platform - Logo - 2018	89
Abb. 94 Google Drive - Logo - 2018.....	89
Abb. 95 Heroku - Logo - 2018.....	90
Abb. 96 Marktgemeinde Lustenau - Logo - 2018	90
Abb. 97 Messenger API - Logo - 2018	90
Abb. 98 Microsoft Excel - Logo - 2018	90
Abb. 99 Microsoft Outlook - Logo - 2018.....	91
Abb. 100 Microsoft PowerPoint - Logo - 2018.....	91
Abb. 101 Microsoft Visio - Logo - 2018	91
Abb. 102 Microsoft Word - Logo - 2018	91

Abb. 103 MySQL - Logo - 2018.....	92
Abb. 104 OpenWeatherMap - Logo - 2018	92
Abb. 105 Adobe Photoshop - Logo - 2018	92
Abb. 106 Postman - Logo - 2018	92
Abb. 107 Python - Logo - 2018	93
Abb. 108 Sourcetree - Logo - 2018	93
Abb. 109 Trello - Logo - 2018.....	93
Abb. 110 WhatsApp - Logo - 2018	93
Abb. 111 Projektstatusbericht 04 Seite 1	110
Abb. 112 Projektstatusbericht 04 Seite 2	111

14 Tabellenverzeichnis

Tab. 1 Projektauftrag.....	6
Tab. 2 Projektorganisation	9
Tab. 3 Projektterminplan	12
Tab. 4 Meilensteinplan	13
Tab. 5 Risikoanalyse	15
Tab. 6 Projektumweltbeziehungen	17
Tab. 7 Plattformenvergleich	23
Tab. 8 Vergleich: OpenWeatherMap vs. YahooWeather	56

15 Abkürzungsverzeichnis

Abkürzung	Bezeichnung
AG	Auftraggeber
API	application programming interface
DA	Diplomarbeit
FB	Facebook
HTL	Höhere technische Lehranstalt
HTML	Hypertext Markup Language
ID	Identifikator
JSON	Java Script Object Notation
PHP	Hypertext Preprocessor
PL	Projektleiter
PM	Projektmanagement
PSP	Projektstrukturplan
PTM	Projektteammitglied
PTP	Projektterminplan
REST	Representational State Transfer
SQL	Structured Query Language
SWP	Softwareentwicklung und Projektmanagement
TdoT	Tag der offenen Tür
UI	User interface
URL	Uniform Resource Locator
VVV	Vorarlberger Verkehrsverbund
XML	Extensible Markup Language

16 Literaturverzeichnis

- Archer, A. (05. 02 2018). *Medium - Chatbots Magazine*. Von <https://chatbotsmagazine.com/how-to-design-a-great-messenger-chatbot-dc33c1b2dfb6> abgerufen
- Banfi, V. (01. 02 2018). *Botsociety Blog*. Von <https://botsociety.io/blog/2017/11/messenger-mockup-design/> abgerufen
- BBK Deutschland. (10. 01 2018). *BBK Deutschland*. Von https://bbk.bund.de/SharedDocs/Downloads/BBK/DE/Publikationen/Praxis_Bevoelkerungsschutz/Band_16_Risikoanalyse_im_BS.pdf abgerufen
- Cooper, K. (17. 09 2017). *Hackernoon*. Von <https://hackernoon.com/understanding-git-fcffd87c15a3> abgerufen
- Debecker, A. (04. 01 2018). *Medium-Chatbotlife*. Von <https://chatbotlife.com/releasing-a-chatbot-facebook-above-all-ddf10685a94d> abgerufen
- Eishofer, A. (04. 02 2018). *Eishofer*. Von <https://eishofer.com/facebook-vanity-url-und-andere-tipps-zur-seitenerstellung/> abgerufen
- Flask. (29. 01 2018). *Flask*. Von <http://flask.pocoo.org/> abgerufen
- Franco, A. G. (28. 12 2017). *codementor*. Von <https://codementor.io/aarongfranco/chatbots-how-to-make-a-bot-for-messenger-from-scratch-6e73wchwr> abgerufen
- Franco, A. G. (28. 12 2017). *codementor*. Von <https://codementor.io/aarongfranco/chatbots-how-to-make-a-bot-for-messenger-from-scratch-6e73wchwr#accepting-webhooks> abgerufen
- Guder, M. (30. 01 2018). *Linkedin*. Von <https://de.linkedin.com/pulse/dezentralisierung-von-webanwendungen-durch-heroku-markus-guder> abgerufen
- Heroku. (27. 01 2018). *Heroku*. Von <https://www.heroku.com/pricing> abgerufen
- Imura, T. (28. 12 2017). *GIRLIEMAC.COM*. Von <https://giriemac.com/blog/2017/01/06/facebook-apiai-bot-nodejs/> abgerufen

- Kang, A. (05. 01 2018). *linkedin*. Von <https://linkedin.com/pulse/understanding-differences-between-different-ai-platforms-abraham-kang/> abgerufen
- Kunal, K. (05. 02 2018). *Superdevresources*. Von <https://superdevresources.com/weather-forecast-api-for-developing-apps/> abgerufen
- Liu, C. L. (08. 02 2018). *Google Cloud Platform Blog*. Von <https://cloudplatform.googleblog.com/2017/07/how-to-build-a-conversational-app-that-sees-listens-talks-and-translates-using-Cloud-Machine-Learning-APIs-part-1.html> abgerufen
- logicline. (30. 01 2018). *logicline*. Von https://logicline.de/blog/2015/11/heroku_technical_view/ abgerufen
- logicline. (30. 01 2018). *logicline*. Von https://logicline.de/blog/2015/11/heroku_technical_view/ abgerufen
- logicline. (30. 01 2018). *logicline*. Von https://logicline.de/blog/2015/11/heroku_technical_view/ abgerufen
- lurchenko, A. (05. 02 2018). *Medium - Chatbots Magazine*. Von <https://chatbotsmagazine.com/cheat-sheet-all-facebook-chatbot-interactions-4b14e4e00178> abgerufen
- Mahanoor, A. (27. 01 2018). *Miningbusinessdata*. Von <https://miningbusinessdata.com/api-ai-tutorial-using-webhooks-built-chuck-norris-joke-bot/> abgerufen
- Maruti Techlabs. (05. 01 2018). *Medium - Chatbotlife*. Von <https://chatbotlife.com/which-are-the-best-on-site-chatbot-frameworks-3dbf5157fb57> abgerufen
- Myers, K. (28. 01 2018). *Kenmyers*. Von <https://kenmyers.github.io/tutorial/2016/03/11/getting-flask-on-heroku.html> abgerufen
- Nikita. (05. 01 2018). *Medium - Botsfloor*. Von <https://tutorials.botsfloor.com/the-best-nlp-language-understanding-tools-to-make-your-chatbot-smarter-d6db34fbe90d> abgerufen

- Ondrisek, B. (05. 02 2018). *Medium - Chatbots Magazine*. Von <https://chatbotsmagazine.com/why-emoji-fit-perfectly-for-chatbots-e9c3e8d433ad> abgerufen
- Python Tutorials. (29. 01 2018). *Python Tutorials*. Von <https://pythonspot.com/flask-web-app-with-python/> abgerufen
- Ring, F. (05. 01 2018). *Blog Frederik Ring*. Von <http://blog.frederikring.com/articles/building-an-api-ai-webhook/> abgerufen
- Roth, P. (05. 01 2018). *allfacebook.de*. Von <https://allfacebook.de/mobile-2/facebook-messenger-nutzerzahlen> abgerufen
- Statista. (04. 01 2018). *Statista*. Von <https://de.statista.com/statistik/daten/studie/419453/umfrage/anzahl-der-monatlich-aktiven-nutzer-des-facebook-messengers-weltweit/> abgerufen
- Unbekannt. (17. 09 2017). *Dialogflow*. Von <https://dialogflow.com/docs/integrations/facebook> abgerufen
- Unbekannt. (27. 01 2017). *Sparx Systems Central Europe*. Von <https://sparxsystems.de/ressourcen/literatur/leseprobe-zu-projektentwicklung-mit-uml-und-enterprise-architect/anwendungsfalldiagramm-use-case-diagram/> abgerufen
- Unbekannt. (05. 02 2018). *Facebook for developers*. Von <https://developers.facebook.com/docs/messenger-platform/send-messages> abgerufen
- Unbekannt. (05. 01 2018). *Google Docs*. Von <https://docs.google.com/spreadsheets/d/1RgG-dRS42EHIG7QdJOTg2ZO587KutTTPeUfyxVKoln8/edit#gid=0> abgerufen
- Unbekannt. (01. 02 2018). *Wikipedia*. Von <https://de.wikipedia.org/wiki/Mock-up> abgerufen
- Wagner, J. (05. 02 2018). *ProgrammableWeb*. Von <https://programmableweb.com/news/top-10-weather-apis/analysis/2014/11/13> abgerufen

xVir. (27. 01 2018). *GitHub*. Von <https://github.com/dialogflow/fulfillment-webhook-weather-python> abgerufen

17 Anhang

17.1 Projektstatusbericht

Zur Darstellung des aktuellen Projektstatus wurde zu Beginn jedes Monats ein Projektstatusbericht durch den PL erstellt. Dieser diente dem Projektbetreuer zur Orientierung über den aktuellen Stand des Projektes und diente weiters als Projektcontrolling-Instrument.

17.1.1 Momentaufnahme des Projekts

Dieser Teil des Berichts enthielt eine Tabelle, die alle aktuell laufenden Aktivitäten darstellte. Des Weiteren wurde der Fortschritt der Aktivität bewertet, wenn vorhanden wurden Probleme aufgezeigt. Außerdem wurde das geplante Fertigstellungsdatum jeder Aktivität angegeben.

17.1.2 Statuszusammenfassung

In einem kurzen Absatz wurde der aktuelle Projektstatus kurz und knapp zusammengefasst, um so einen raschen Überblick über den aktuellen Stand des Projektes zu ermöglichen.

17.1.3 Zeiterfassung Diplomarbeit

Zur Erfassung der für die Diplomarbeit geleistete Arbeitszeit wurde eine Excel-Tabelle angelegt, in der jedes Projektmitglied seine Arbeitsstunden selbst eintragen konnte. Der jeweils monatsaktuelle Stand der Zeiterfassung wurde ebenfalls im Projektstatusbericht angeführt.

In den nachfolgenden Abbildungen ist der vierte Projektstatusbericht dargestellt:

09.01.2018

Projektstatusbericht 04

Projektname
lustenauBot

Auftraggeber
Marktgemeinde Lustenau

Projektleiter
Jakob Ott

**Projektteam-
mitglieder**
Matteo Kofler
Manuel Waibel

Projektbetreuer
Diethard Kaufmann

Erstellt von
Jakob Ott

Momentaufnahme des Projekts

Aktivität	% abgeschl ossen	Probleme	Fertig bis	Bearbeite r
Implementierung Abfahrtszeiten	100%	Keine offizielle API bereitgestellt, UTF-8 Fehler → gelöst	31.01.2018	Matteo, Manuel
Implementierung Veranstaltungen	100%	UTF-8 Codierungsproblem-Umlaute werden nicht dargestellt → gelöst	07.01.2018	Matteo, Manuel
Implementierung Fakten über Lustenau	100%	Keine	31.12.2017	Manuel
Testphase	35%	Relativ Zeitaufwändig	28.02.2018	Jakob, Matteo, Manuel
Projektkoordination	40%	Keine	05.04.2018	Jakob
Projektdokumentation – erste Fassung	50%	Keine	31.01.2018	Jakob, Matteo, Manuel
Projektterminplan	100%	Keine	07.01.2018	Jakob
Risikoanalyse	5%	Keine (erst begonnen)	31.01.2018	Jakob
Projektumweltanalyse	100%	Keine	07.01.2018	Jakob

Erklärung zum Farbcode: grün = erledigt; gelb = in Arbeit; rot = fällig

Statuszusammenfassung

Die Entwicklung des Chatbots ist abgeschlossen. Die erste Fassung der Dokumentation haben wir über die Weihnachtsferien etwa zur Hälfte geschrieben. Aus heutiger Sicht werden wir das Projekt innerhalb des Zeitplanes erfolgreich abschließen können.

lustenauBot

Diplomarbeit von Jakob Ott,
Matteo Kofler und Manuel Waibel

HTL Dornbirn 2017/18

Projektbetreuer:
Diethard Kaufmann

Zeiterfassung Diplomarbeit:

Aktueller Stand der Zeiterfassung:

- Jakob: 72 Stunden
- Matteo: 131 Stunden
- Manuel: 91 Stunden

Details sind in folgender Datei einsehbar:

https://drive.google.com/open?id=0B7YKyMNDZ_D4N09XaU85MWx4Rkk