HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

ewz

Chatbot für das Customer Care Center von ewz

Bachelorarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2018
Technische Dokumentation

Autoren:	Felix Varghese Enchiparamban, Christian Lindauer
Betreuer:	Prof. Stefan Richter
Projektpartner:	ewz
Experte:	Ettore Ferranti
Gegenleser:	Prof. Dr. Andreas Steffen
Arbeitsperiode:	17.0.8.2018 – 21.12.2018
Arbeitsumfang:	360 Stunden, 12 ETCS pro Studenten

Änderungsgeschichte

Datum	Version	Änderung	Autor
24.09.2018	1.0	Erstellen des Dokumentes	Felix Enchiparamban
24.09.2018	1.1	Einleitung und Übersicht hinzugefügt	Christian Lindauer
27.09.2018	1.2	Chatbot Evaluation hinzugefügt	Felix Enchiparamban
28.09.2018	1.3	NFA und Informelles zu Chatbot hinzugefügt	Christian Lindauer
01.10.2018	1.4	Chatbot Evaluation erweitert	Felix Enchiparamban
04.10.2018	1.5	Soll-Situation hinzugefügt	Christian Lindauer
04.10.2018	1.6	Chatbot Plattform Evaluation	Felix Enchiparamban
08.10.2018	1.7	User Experience hinzugefügt	Christian Lindauer
08.10.2018	1.8	Kapitel Umsetzung hinzugefügt	Felix Enchiparamban
12.10.2018	1.9	Korrekturlesen einzelner Kapitel	Christian Lindauer
18.10.2018	2.0	Kapitel Umsetzung hinzugefügt	Christian Lindauer & Felix Enchiparamban
19.10.2018	2.1	Kapitel Probleme bei der Umsetzung hinzugefügt	Christian Lindauer & Felix Enchiparamban
22.10.2018	2.2	Kapitel Systemarchitektur hinzugefügt	Christian Lindauer
25.10.2018	2.3	Kapitel Software Architektur hinzugefügt	Felix Enchiparamban
29.10.2018	2.4	Kapitel Deploymentdiagramm hinzugefügt	Felix Enchiparamban
30.10.2018	2.5	Systemsequenzdiagramm hinzugefügt	Felix Enchiparamban
30.10.2018	2.6	Zwischenstand Korrekturlesen	Christian Lindauer
31.10.2018	2.7	Kapitel Demonstrator 1 + 2 hinzugefügt	Christian Lindauer
05.11.2018	2.8	Kapitel Content Moderator hinzugefügt	Christian Lindauer
09.11.2018	2.9	Kapitel Usability Test fertiggestellt	Christian Lindauer
16.11.2018	3.0	Kapitel Demonstrator 3 fertiggestellt	Christian Lindauer & Felix Enchiparamban
19.11.2018	3.1	Kapitel Multi language Support hinzugefügt	Felix Enchiparamban
21.11.2018	3.2	Kapitel Business Case fertiggestellt	Christian Lindauer
10.12.2018	3.3	Kapitel Solution Architektur hinzugefügt	Felix Enchiparamban
11.12.2018	3.4	Abstract, Persönlicher Bericht hinzugefügt	Christian Lindauer & Felix Enchiparamban
12.12.2018	3.5	CUSOLL, Ergebnisse & Ausblick hinzugefügt	Christian Lindauer
13.12.2018	3.6	Kapitel Unit Test hinzugefügt	Felix Enchiparamban
14.12.2018	3.7	Kapitel Technologie und Architekturempfehlung für CUSOLL hinzugefügt	Felix Enchiparamban
19.12.2018	3.8	Finalisierung des Dokuments	Christian Lindauer

Danksagung

An dieser Stelle möchten wir uns bei Allen bedanken, die uns bei der Bachelorarbeit unterstützt und motiviert haben.

- **Prof. Stefan Richter** für die Betreuung und zahlreichen Ratschlägen in den Sitzungen
- **Otmar Lindauer** für die Korrekturen zur Dokumentation
- **Manfred Fäh** für die Feedbacks zu den jeweiligen Zwischenergebnissen
- **Roman Baldinger** für die wertvollen Gespräche und praktischen Anwendungsbeispielen
- **Simon Kuhn** für die technische Unterstützung in Bezug auf Azure

Natürlich wollen wir auch allen nicht persönlich erwähnten danken, wie unseren Partnern und Familien, welche uns motiviert haben.
Nicht zu vergessen sämtliche Entwickler und Entwicklerinnen die Tutorials und Communitybeiträge geschrieben haben.

Abstract

Chatbots sind textbasierte Dialogsysteme, die in natürlicher Sprache mit einem Benutzer kommunizieren können. Aktuell werden Chatbots häufig bei der Verkaufsberatung oder beim Helpdesk eingesetzt. Die vorliegende Bachelorarbeit soll sach- und fachgerecht beurteilen, ob ein Chatbot beim Energieversorger Unternehmen ewz im Telekommunikationsbereich Unterstützung bieten kann. Primär soll evaluiert werden, inwiefern und auf welcher Ebene ein Chatbot 1st Level Supportanfragen übernehmen kann.

In einem ersten Schritt haben wir die Thematik Chatbot untersucht und anschliessend die verschiedenen Frameworks miteinander verglichen. Dazu haben wir einen Kriterienkatalog erstellt, welcher nicht nur die technischen Details, sondern auch die effektiven Einsatzmöglichkeiten gegenübergestellt. Mithilfe eines Natural-Language-Processing Services wurde der Chatbot erweitert durch eine künstliche Intelligenz, welcher die Absichten der Nutzer erkennen soll. Aus den evaluierten Frameworks und Services sind drei Chatbot Prototypen entstanden, zugeschnitten auf das Szenario der Kunden Hilfestellung.

Schliesslich resultierte aus dieser Bachelorarbeit ein lauffähiger Prototyp, welcher Kundenanliegen im Bereich Telekommunikation beantworten kann. Erfahrungen und Erkenntnisse, welche wir gesammelt haben aus der Evaluation, sind in dieser Bachelorarbeit dokumentiert und können bei anderen Chatbot Projekten zur Hilfe genommen werden. Abschliessend können wir nur eine bedingte Empfehlung aussprechen für den Einsatz eines Chatbots im Bereich Telekommunikation. Die künstliche Intelligenz von Chatbots ist aktuell begrenzt, was wiederum als Konsequenz nachzieht, dass der Dialog teilweise strukturiert vorgegeben sein muss. Ungeachtet dessen, sind wir überzeugt das in Zukunft Chatbots vermehrt zum Einsatz kommen.

Management Summary

Ausgangslage

In der heutigen Generation von Google, WhatsApp und Co. findet häufig die Kommunikation in einem Frage- und Antwortdialog statt. Diese Art von Kommunikation hat sich über die Jahre etabliert und ist den Menschen dieser Generation bestens bekannt. Chatbots nutzen diese Kommunikationsart und bieten ein Benutzerinterface an, um eine Konversation zu führen. Mittels Wissensdatenbanken, Natural-Language-Processing (NLP) Services und Machine Learning wurde den Chatbots die nötigen Werkzeuge geboten, um einen effektiven Nutzen zu erbringen. Häufig werden Chatbots verwendet im Bereich Support und Verkaufsberatung. Mit dieser Bachelorarbeit möchten wir herausfinden, zu welchem Grad ein Chatbot den 1st Level Support in der Thematik Telekommunikation unterstützen kann. Wir möchten an dieser Stelle darauf hinweisen, dass es sich um eine Machbarkeitsstudie handelt.

Vorgehen und Technologien

Bei der Planung der Bachelorarbeit haben wir uns entschieden, uns erstmals gründlich über die Einsatzmöglichkeiten sowie Frameworks zu informieren. Dabei legten wir besondere Aufmerksamkeit auf die Anforderungen des Industriepartners ewz. Zugleich wurden wöchentlich die nächsten Schritte geplant mit dem Betreuer, Prof. Stefan Richter.

Nach der Evaluationsphase haben wir die Anforderungen auf drei Prototypen aufgeteilt, welche wiederum aufeinander aufbauen. Die Erkenntnisse aus der Evaluation haben dazu geführt, dass wir uns für das Microsoft Bot Framework entschieden haben, um den Chatbot zu entwickeln. Als Intelligente Wissensdatenbank kommt der QnA Maker Service zum Einsatz, der zuvor mit typischen Q&A-Fragen und Antworten gefüttert wird. Mit Microsofts NLP-Service LUIS (Language Understanding Intelligent Service) wird die Benutzerabsicht erkannt und ermöglicht so eine höhere Präzision der Antwortquote. Schliesslich wurde neben einer Architektur- und Technologieempfehlung für die ewz zusätzlich den Nutzungsgrad des Chatbots ermittelt.

Ergebnisse

Die vielversprechenden Möglichkeiten von Chatbots hören sich auf dem Papier umwerfend an. Jedoch mussten wir feststellen, dass aufgrund von komplexen Fragestellungen Chatbots oftmals Schwierigkeiten haben. Solche Systeme funktionieren am besten mit grossen Datensätzen und vielen Fragen- und Antworten. Hierbei wäre die Vorgehensweise Deep Learning interessant. So könnten Bots aus Fehlern lernen und sich so stetig weiterentwickeln. In dieser Bachelorarbeit haben wir uns auf LUIS sowie den Machine Learning Aspekt konzentriert. Der Chatbot ist nun in der Lage einfache Fragen im Bereich Telekommunikation zu beantworten und dient als unterstützender Service für den Helpdesk. Zudem sind wir überzeugt, dass häufig gestellte Fragen im 1st Level Support nun automatisiert und zuverlässig beantwortet werden können.

Ausblick

Für die Weiterführung des Chatbots ist vorgesehen die Rollout-Datenbank anzubinden. Zusätzlich soll evaluiert werden, welche der präsentierten Varianten am vielversprechendsten ist für das CUSOLL Programm. Zudem ist eine Ergänzung des Fragen- und Antwortenkatalogs sinnvoll.
Aus technologischer Sicht kann man die Azure Table Storage mit einer SQL Datenbank ersetzen, um die Wartbarkeit zu erhöhen. Eine detaillierte Beschreibung der möglichen Erweiterungen, wird im Kapitel 13. erläutert.

Aufgabenstellung

Bachelorarbeit «Chatbot für das Customer Care Center von ewz»

Einführung

Das CCC (Customer Care Center) von ewz Telecom ist die Anlaufstelle für Kunden und Hausbesitzer im Zusammenhang mit dem Ausbau des Glasfasernetzes und den entsprechenden Services. Pro Monat beantwortet das CCC über 400 Telefonische Anfragen.

Im Rahmen der Digitalisierungsinitiativen von ewz interessiert sich ewz für die technischen Möglichkeiten eines Chat Bot Service für die verschiedenen Kundenkanäle. Die Erkenntnisse aus dieser Bachelorarbeit sollen gemeinsam mit einer anschliessenden Testphase Grundlagen für die zukünftigen Entwicklungen im Rahmen des CUSOLL Programm von ewz liefern.

Aufgabe

Am Beispiel des CCC ewz Telecom soll ein Demonstrator eines Chat Bots basierend auf den Microsoft Azure Services implementiert und getestet werden. Der Demonstrator soll u.a. die Module der Kognitiven Services und des QnA Makers verwenden. Der QnA Maker soll dazu verwendet werden, neben dem ein QnA zu erstellen und zu pflegen.

Das System soll es zudem den Mitarbeitenden im CCC erlauben ihren eigenen QnA Katalog ohne Programmierkenntnisse zu pflegen. Dazu soll ein User Interface bereitgestellt werden, das von den Mitarbeitenden einfach zu verwenden ist.

Das QnA soll einerseits mit dem Chat-Bot-Service verknüpft werden, andererseits sollen aber auch andere Informationssysteme angebunden werden, so z.B. die Rollout-Datenbank von ewz. Das QnA muss die häufigsten Anfragen beinhalten.

Schliesslich sollen die Studierenden den Nutzungsgrad und die Einsatzfähigkeit des Systems beurteilen und dabei insbesondere auf ein mögliches Konzept für die Mandantenfähigkeit der Lösung (z.B. für andere Energieversorger) und den Businessnutzen Mini BC eingehen. Zusätzlich können sie Technologie- und Architektur-Empfehlungen für das Programm CUSOLL abgeben. Das umfasst auch sicherheitsrelevante Fragestellungen, wie den Datenschutz (Vermeidung des Speicherns von Personendaten in der Cloud) oder die Möglichkeit selbst-lernende Bots durch böswillige Nutzer fehl zu trainieren.

Termine

Die Bachelorarbeit beginnt am 18.9.2018. Abgabetermin ist der 21.12.2018.

Betreuung

Die Bachelorarbeit wird durch Prof. Stefan Richter betreut. Jeden Donnerstag von 13:00 bis 14:00 findet eine Besprechung statt, in der die Studierenden den Fortschritt der Arbeit präsentieren. Fragen und Angelegenheiten können ausserhalb dieses Termins auch per E-Mail erörtert werden.

Von ewz steht Manfrad Fäh als Ansprechpartner zur Verfügung. Experte für diese Arbeit ist Ettore Ferranti, ABB Schweiz.

Bewertung

Die Arbeit wird anhand der folgenden sechs gleichgewichteten Punkte bewertet:

1. Organisation, Durchführung (Projektplanung u. Nachführung Arbeit gemäss Projektplan, Selbständigkeit, Einsatz, Zusammenarbeit mit Auftraggeber, Betreuer)
2. Bericht (Inhalt des Projektschlussberichts, Gliederung, Darstellung, Sprache der gesamten Dokumentation)
3. Problemanalyse (Vorstudie, Literaturstudium, Anforderungsspezifikation, Anforderungsanalyse, Domainanalyse)
4. Lösungsentwurf (Lösungsvarianten und deren Beurteilung, Variantenentscheid, Konzept, Entwurf)
5. Realisierung und Test
6. Bachelor-Präsentation

Hinweise

Die folgende Seite bietet zahlreiche nützliche Informationen zum Schreiben einer wissenschaftlichen oder technischen Arbeit:

https://www.ifs.hsr.ch/index.php?id=13194&L=4metadata%2Foai_dc_1.dc

Unterschriften

Rapperswil, 18. September 2018

Stefan Richter

Felix Varghese Enchiparamban

Christian Lindauer

Inhaltsverzeichnis

1. Einleitung und Übersicht

Dieses Dokument behandelt die Bachelorarbeit von Felix Enchiparamban und Christian Lindauer, die an der Hochschule für Technik Rapperswil (HSR) im Herbstsemester 2018 entstand.
Prof. Stefan Richter (Professor, Partner des Instituts für Software der HSR) ist erneut Betreuer der Bachelorarbeit.

1.1 Vision

ewz plant bei der Digitalisierung sich an mehreren Fronten zu etablieren. Hierfür wurde extra das Programm CUSOLL[1], eine Anlauf- und Schnittstelle für Kunden, ins Leben gerufen. Ziel von CUSOLL ist es, den Kunden eine Plattform zu bieten, wo sie ihre Anliegen aussprechen können. Zusätzlich soll damit die Kundenzufriedenheit erhöht und die internen Kosten gesenkt werden.
Als Teil des CUSOLL Programms sollen die Kundenanfragen, welche üblicherweise telefonisch eintreffen durch einen Chatbot unterstützt werden. Der Chatbot soll dabei FAQ[2] und möglichst viele kundenspezifische Fragen beantworten können.

1.2 Gesetzliche Rahmenbedingungen

Beim Einsatz eines Chatbots sind oftmals persönliche Daten im Umlauf. Es beginnt mit der Identifikation und Authentifizierung des Benutzers. Der Datenschutz spielt demnach eine zentrale Rolle. Der Betreiber des Chatbots (in unserem Fall die ewz) ist verpflichtet, Kundendaten angemessen zu schützen. Technisch bedeutet dies, dass keine Kundendaten in der Cloud abgespeichert werden dürfen. Die einzelnen Gespräche dürfen ebenso nicht von einem anderen Kunden eingesehen werden.
Es empfiehlt sich, den Kunden vor der Interkation des Chatbots hinzuweisen, dass man nicht mit einem menschlichen Mitarbeiter in Kontakt tritt, sondern mit einer KI.

[1] Customer Solutions
[2] Frequently Asked Questions

2. Grundlegende Informationen zu Chatbots

In diesem Kapitel werden grundlegende Punkte erklärt rund um das Thema Chatbot, welche für die Einführung und Entwicklung von grosser Bedeutung sind. Einerseits wird erklärt wie Chatbots funktionieren und auf welche Arten sie sich differenzieren. Zusätzlich werden verschiedene Bot Frameworks miteinander verglichen und evaluiert, welche sich für den Anwendungsfall der ewz eignen.

2.1 Was sind Chatbots?

Chatbots sind einfach ausgedruckt ein Interface, das im Hintergrund mit einem Service kommuniziert, sowie es auch Websites oftmals tun.
Quasi ein Roboter mit dem Kunden über einen Chat kommunizieren können. Sie basieren auf Künstlicher Intelligenz (KI), die durch eine Reihe definierter Regeln und Parameter funktionieren. Man kann einen Chatbot auf unterschiedliche Art und Weise über Text und Sprache ansprechen.
Somit eignen sich Chatbots optimal für Gespräche, welche ein Mitarbeiter eines Kundendienstes führt. Die Kommunikation findet hierbei über einen Chat statt, ähnlich wie bei Whatsapp. Der Kunde hat die Möglichkeit frei formulierte Fragen im Kontext zur ewz zu stellen. Fragen werden dann geparsed (Stichwörter herausgefiltert) und an die Wissensdatenbank des Chatbots weitergeleitet. Das Chatbot Framework versucht nun aus der Frage des Kunden eine passende Antwort zu finden.

2.2 Lernfähigkeit durch maschinelles Lernen

Gerade im Bereich für Customer Service spielt maschinelles Lernen seine volle Stärke aus. Durch Training des Chatbots, ist es möglich den «Sinn» der Frage herauszulesen und eine korrekte Antwort zu liefern. Dies bedeutet, dass bei einer Abweichung der Frage diese nicht durcheinandergebracht werden. Dabei wird die künstliche Intelligenz des Chatbots grösser mit jedem Gespräch.
Im Endeffekt ist ein Chatbot nur so mächtig wie die Knowledge Base im Hintergrund. Grundsätzlich gilt: Je mehr und detaillierte Fragen vorhanden sind, desto bessere Antworten liefert ein Chatbot.

2.2.1 Wie trainiert man einen Chatbot? [1]

Bevor man einen Chatbot trainieren soll, muss man sich in die Lage des Kunden versetzen. Welche Frage habe ich als Kunde? Wie formulieren Kunden ihre Fragen mit wenig/kaum Fachwissen über die Thematik?
Oftmals schreiben Spezialisten und Fachleute Benutzerhandbücher. Diese sind zwar in den Augen von Fachkräften detailliert und gut beschrieben, aber für Personen, welche nicht aus diesem Umfeld kommen, zu kompliziert. Dieses Problem gilt auch beim Training von Chatbots. Die Fragen und dazugehörigen Antworten werden zu kompliziert abgelegt.

Wie geht man also vor, wenn nur ein begrenztes Kontingent an Fragen und Antworten zur Verfügung steht?

- Fragen und Antworten von einer weiteren Person reviewen lassen.
- Eine kleine Gruppe von Benutzern einladen und mit dem Chatbot chatten lassen. Die Benutzereingaben aus den Logfiles extrahieren und die Konversationen analysieren.
- Nach einigen Iterationen filtert man so die häufig gestellten Fragen (und Fragevariationen) aus.

Das Pareto Prinzip [2], bekannt auch als «80/20 Regel», gilt auch beim Training eines Chatbots. 80% aller Kundenfragen repräsentieren rund 20% der Knowledge Base. Dies hört sich zuerst erschreckend gering an, jedoch befinden sich unter diesen Fragen viele zu spezifische Fragen. Es empfiehlt sich 80% der FAQ[3] abzudecken. Die restlichen 20% sind eine Frage der Zeit und des Trainings, welche in ein bis drei Monaten ebenfalls gedeckt werden können.

[3] Frequently Asked Questions

2.3 Chatbot Arten

Chatbots lassen sich in ihrer Komplexität und Interaktionsmöglichkeiten in verschiedene Arten unterteilen.

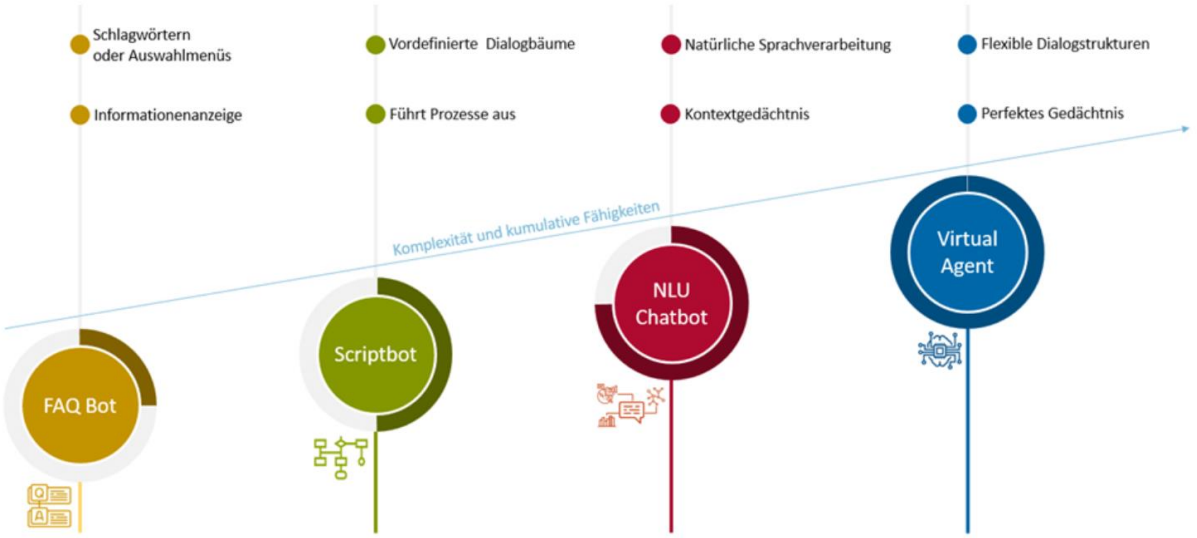


Abbildung 1: Einteilung von Chatbots in Kategorien [3]

FAQ Bot

FAQ Bots greifen auf eine Wissensdatenbank auf Fragen und Antworten zu. Aus den gestellten Fragen werden die Stichworte herausgefiltert und die passendste Antwort gesucht. Mit zunehmenden Fragen wird die Wissensdatenbank immer besser und genauer. Der Vorteil dieser Variante ist, dass mit zunehmenden Interaktionen immer mehr Daten einfliessen, welches das System nutzt um sich selbst zu trainieren. Zusätzlich soll das System mit dem Kundenfeedback die gegebenen Antworten überprüfen und so sich selbst kalibrieren und korrigieren.

Scriptbots

Diese Art von Bots wird typischerweise bei stark standardisierten Prozessen genutzt. Häufig anzutreffen bei Produktkonfigurationen, welche vorgegebene Optionen beinhalten. Der Kunde wählt aus vordefinierten Antworten aus, welche am passendsten ist. Diese Scriptbots entlasten den Customer Support bei stark standardisierten Geschäftsprozessen wie z.B. ein personalisiertes Zusammenstellen eines Mobil-Abonnements.

NLU (Natural Language Understanding) Chatbot

Diese Art von Chatbots bieten noch mehr Interaktionsmöglichkeiten an, da diese personalisierter daherkommen. Es wird versucht deren Anliegen und Stimmung zu erkennen und so auch die Absicht des Nutzers. Aufgebrachte Kunden werden somit als Massnahme eher dem menschlichen Call-Center weitergeleitet. Des Weiteren sind NLU Bots in der Lage den Kontext einer Nachricht zu entschlüsseln. Aus der Anfrage «Ich suche morgen einen Flug von Zürich nach Bangkok», liest der Chatbot automatisch heraus, dass der Nutzer einen Flug buchen möchte für morgen, mit dem Abflugort Zürich, Ankunftsort Bangkok.

Virtual Agents

Virtual Agents sind die vorerst letzte Stufe bei den Chatbots. Diese sind in der Lage mittels eines «Gedächtnisses» bereits geführte Konversationen und Entscheidungen zu merken. Bei der Frage «Wann fährt der letzte Zug nach Hause?», können Virtual Agents aus früheren Konversationen die Heimadresse ausmachen, sowie aus den GPS[4] Daten den aktuellen Aufenthaltsort verzeichnen. Mit einem Zugriff auf Googlemaps kann so eine Route berechnet und dem Nutzer angezeigt werden.
Virtual Agents sind noch nicht marktreif und daher erst als Prototypen erhältlich.

2.4 Bot-Development Frameworks

Chatbots basieren auf zahlreichen Komponenten. Zu diesen Komponenten zählt die Schnittstelle zu Messaging Platforms [5], Web-App des Chatbots sowie auch Natural-Language-Processing, welches die Absicht des Benutzers aus der geschriebenen Anfrage des Nutzers herausliest. Solche Komponenten werden von Bot-Development Frameworks zur Verfügung gestellt und erleichtern somit die Entwicklung.

2.5 Allgemeine Chatbot Architektur

Die meisten Chatbots funktionieren im Hintergrund ähnlich und unterscheiden sich nur marginal, abhängig von der gewählten Technologie. Zum allgemeinen Verständnis setzen wir bewusst auf ein vereinfachtes Diagramm, welches für die meisten Chatbots gültig ist.

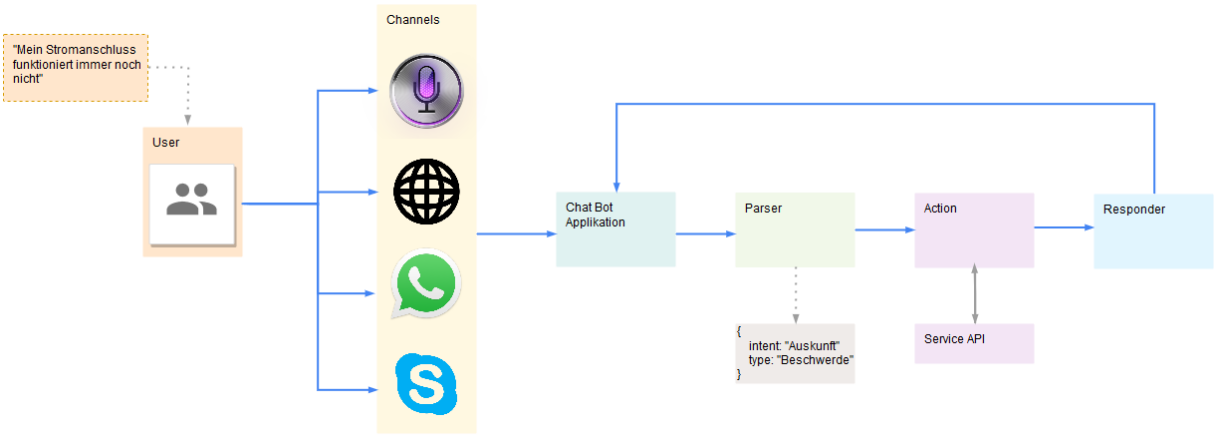


Abbildung 2: Generelle Chatbot Architektur [4]

Aus der Abbildung ist ersichtlich, dass der User eine Nachricht über einen der Channels verfasst. Der Chatbot leitet die Nachricht an den Parser weiter, welches aus der Nachricht extrahiert um welchen Typ Nachricht es handelt. Es fügt zusätzlich die Absicht der Nachricht hinzu und leitet es der Action weiter, welches entscheidet was mit der Nachricht vom Parser passieren soll. Der Responder generiert daraus schlussendlich eine benutzerleserliche Antwort, welche der Chatbot Applikation zur Verfügung gestellt wird. Die Chatbot Applikation sendet diese Antwort dann zurück an den User.

[4] Global Positioning System. Navigationssatellitensystem zur Positionsbestimmung
[5] Message Programme wie WhatsApp, Slack, Skype etc.

Bezeichnung	Beschreibung
User	Interagiert mit dem Chatbot und schreibt die Anfrage.
Channels	Verbindungsglied zwischen User und dem Chatbot. Kommunikationsart ist üblicherweise textuell, kann aber auch in gesprochener Form sein. Typische Channels in geschriebener Form sind WhatsApp, Slack oder Facebook. In gesprochener Form werden Häufig Siri, Alexa oder Cortana erwähnt. Chatbots sind in der Lage mehrere solcher Channels zu adressieren.
Chatbot Applikation	Die Chatbot Applikation kommuniziert über den gewählten Channel mit dem Benutzer. Zudem kümmert sie sich um das Session-Handling sowie die Weiterleitung der Nachricht an den Parser.
Parser [5]	Der Parser hat die Aufgabe, die Absicht des Users in eine maschinenverständliche Form zu übertragen. Dies führt dazu, dass diese in strukturierte Daten umgewandelt werden müssen. Die Umwandlung selbst findet mit NLP[6] statt. Aus einer Nachricht werden zwei Teilinformationen herausgefiltert. Zum einen den Intent (Absicht) und die Entity (Typ). Aus der textuellen Nachricht, wird versucht die Absicht des Benutzers herauszufinden. Die Entity gibt als zusätzliche Information darüber Auskunft, um was für einen Typ es sich handelt.
Action	In diesem Schritt werden aus den Intents zugehörige «Actions» bei der Service-API ausgelöst. Somit löst der Intent «Auskunft» eine andere Action aus als «Authentifizierung».
Responder	Im Prozess Responder, wird die Antwort der Service-API in ein benutzerverständliches Format umgewandelt. Konkret bedeutet dies, eine verständliche und kontextbezogene Antwort für den Benutzer zu generieren. Diese Antwort wird der Chatbot Applikation zur Verfügung gestellt, welche wiederum dem Benutzer angezeigt wird.

2.6 Konversation

Für den Chatbot wird bewusst als Konversationsmittel auf die geschriebene Sprache gesetzt. Es gibt Bots, welche auf die gesprochene Sprache ausgelegt sind wie Amazons Echo[7]. Eine Konversation mit einem Chatbot unterscheidet sich in gewissen Punkten mit einer rein menschlichen Konversation. Im nachfolgenden Kapitel werden die Bestandteile einer Konversation und der Konversationsfluss mit einem Chatbot erläutert.

2.6.1 Konversationsbestandteile [6]

Phase	Beschreibung	Beispiel
Einstieg	In der Startphase wird der Chatbot aktiv und stellt sich dem Benutzer vor und welche Funktion er hat. Nach der Begrüssung erklärt er wie man mit dem Chatbot interagieren kann. Es muss deutlich kommuniziert werden, dass kein menschlicher Mitarbeiter hinter dem Chatbot steckt.	- «Hallo! Ich bin der Support Bot der ewz Telecom, dein persönlicher Assistent und werde versuchen deine Fragen zu beantworten. Schreib einfach deine Frage in die nächste Textzeile.»

[6] Natural Language Processing

[7] Intelligenter Lautsprecher, welche Fragen akustisch beantworten kann

Fragen und Antworten	Nach der Begrüssung und Erklärung was ein Chatbot ist, kann die eigentliche Konversation beginnen. Anhand der Fragen des Benutzers wird die passendste Antwort aus der Knowledge Base herausgesucht.	- «Ist ein Glasfaseranschluss an der Bahnhofstrasse 9 verfügbar?» - «Ja, an der Bahnhofstrasse 9 ist der Glasfaseranschluss bereits verfügbar. Hast du noch eine weitere Frage?»
Proaktiver Chatbot	Damit ein Chatbot «lebendig» und engagiert wirkt, darf er nicht nur simpel auf Kundeneingaben warten. In dem Chatbots auf Eigeninitiative aktiv werden, wird die Akzeptanz von Chatbots erhöht.	- «Du hast nach der Verfügbarkeit an der Bahnhofstrasse gesucht. Möchtest du das Internet vorzeitig aufschalten?»
Abschluss	Wurden die Fragen des Benutzers geklärt bzw. wünscht er keine weitere Konversation mit dem Chatbot, beendet und verabschiedet sich der Bot. Sollte eine Frage nicht beantwortet werden können, bietet der Chatbot dem Benutzer ein persönliches Gespräch mit einem Supportmitarbeiter an.	- «Schön, dass ich dir weiterhelfen konnte. Bis bald!» - «Wünscht du ein persönliches Gespräch mit einem Supportmitarbeiter?»

2.6.2 Konversationsfluss

Zu den anspruchsvollsten Aufgaben zur Entwicklung eines Chatbots gehört der Konversationsfluss. Dieser entscheidet wie das Gespräch gelenkt wird und welche Antworten und Interaktionen er dem Benutzer liefert. Der Konversationsfluss ist deshalb eine Herausforderung, da praktisch bei jeder neuen Begegnung die Anzahl Möglichkeiten ändert. Es gibt daher keinen standardisierten und immer gültigen Weg für ein bestimmtes Ziel. Üblicherweise beginnt eine Konversation mit einer Willkommensnachricht des Chatbots und folgt mit der Aufforderung einer Frage. Idealerweise werden Hilfestellungen angeboten, wenn der Benutzer nicht mehr weiterkommt.
In diesem Projekt wird spezifisch ein Chatbot Konversationsfluss angestrebt, welcher sich auf die Fragen des 1st Level Support ausrichtet und allenfalls eine Verbindung zum Customer Support herstellt.

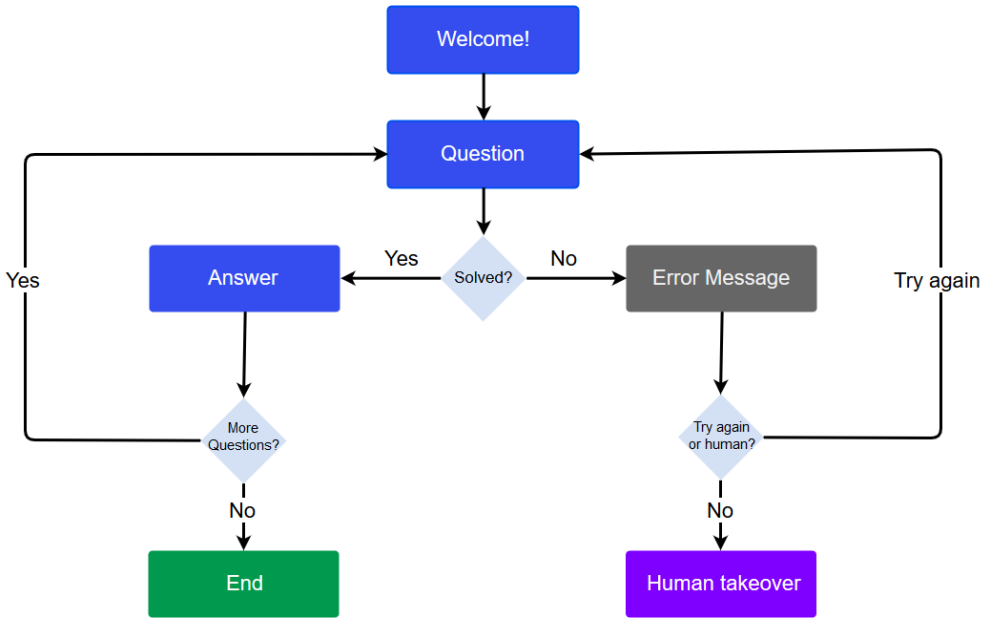


Abbildung 3: Beispiel Konversationsfluss [7]

Damit ein Konversationsfluss möglichst einwandfrei funktioniert, müssen einige Kriterien beachtet werden:

- Die Formulierung der Sätze sollte möglichst kurz und eindeutig sein. Die generelle Sprache ist einfach zu halten und sollte nicht unnötig verkompliziert werden (Keep it simple)
- Sollte dem Benutzer nicht klar sein, welche Interaktion von ihm erwartet wird, muss der Chatbot eine Hilfestellung bieten.
- Es kann passieren, dass der Chatbot keine passende Antwort auf die Frage des Benutzers hat. Ein Fallback Szenario muss jederzeit vorhanden sein, damit der Chatbot sich wieder in einen gültigen Status des Konversationsflusses setzen kann.
- Der Benutzer sollte jederzeit die Möglichkeit haben abzubrechen und den Chatbot bitten einen Supportmitarbeiter zu kontaktieren.

Übergabe an einen Customer Support

Egal wie ausgereift die künstliche Intelligenz des Chatbots ist, es wird immer Szenarien geben in welchen echte Supportmitarbeiter involviert sind und eingreifen müssen.
Hierbei gibt es drei Szenarien, welche man unterscheidet:

- **Triage**: Typische Help Desk Aufrufe beginnen mit einfachen Fragen über Kundenname, Beschreibung des Problems etc. Der Chatbot sammelt vorgängig triviale Informationen, sodass der Supportmitarbeiter keine Zeit vergeuden muss um solche Informationen abzufragen und kann sich direkt auf die Lösung des Problems konzentrieren.
- **Escalation**: Die Frage des Kunden ist zu komplex für den Chatbot und muss daher an einen Mitarbeiter weitergeleitet werden. Eine mögliche Lösung ist, dass LUIS die Anfrage «Ich möchte einen Supportmitarbeiter kontaktieren» korrekt interpretiert und gleichzeitig ein Event-Ticket im Kundencenter auslöst.
- **Superivision**: Bei diesem Szenario übernimmt ein Mitarbeiter die «Kontrolle» über den Chatbot. Während des Gesprächs mit dem Kunden, schlägt der Chatbot die möglichen Antworten vor, welche er dem Kunden gegeben hätte. In diesem Szenario prüft der Mitarbeiter die vorgeschlagene Antwort und kann eingreifen und die Antwort freigeben. Beispiel Szenarien sind hier technische Diagnose Probleme.

Aus der aktuellen Lage der ewz Telecom stellt sich heraus, dass Anfragen oftmals zu komplex sind für den 1st Level Support. Hierfür muss ein Supportmitarbeiter den aktuellen Fall übernehmen und mit dem Benutzer in Kontakt treten. Dies ist ein typischer Fall von «Escalation».
Für die Bachelorarbeit bedeutet dies, dass wir in den Prototypen eine Funktionalität implementieren, welche erlaubt bei einem bestimmten Stichwort «Ich möchte mit einem Supportmitarbeiter sprechen» oder «Support», wird ein Event-Ticket seitens des CCC ausgelöst.

3. Ist-Soll-Analyse

3.1 Ist-Situation

3.1.1 Interviews

- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]

4. Anforderungen

4.1 Benutzer und Personas

Persona-Beschreibungen

In der folgenden Tabelle werden typische Personas vorgestellt und differenziert, welche als Endnutzer klassifiziert werden können.

Rolle/Benutzer	Aufgaben	Affinität	Ziele/Motivation
Systemadministrator	• Azure Portal konfigurieren • Reports und Statistiken auslesen	• Verfügt über vertiefte IT-Kenntnisse • Ist vertraut mit Azure	• Chatbot Nutzung und Wartung
Chatbot Verantwortlicher	• Neue Fragen und Antworten in die Knowledge Base hinzufügen • Bestehende Fragen und Antworten in der Knowledge Base anpassen	• Gute Computerkenntisse • Bürotätigkeit gewohnt • Versteht den Aufbau der Knowledge Base	• Kundenzufriedenheit erhöhen • Korrektheit der Antworten überprüfen
Supportmitarbeiter	• Neue Fragen und Antworten formulieren, welche dem Teamleiter übergeben werden	• Bürotätigkeit gewohnt	• Sich nicht um 1st Level Supportanfragen kümmern müssen
Kunde	• Anfragen über den Chatbot stellen • Persönliche Beratung über einen Supportmitarbeiter erhalten	• Unbekannt	• Anliegen geklärt haben • Persönliche Beratung erhalten

Tabelle 1: Persona-Beschreibungen

4.2 Funktionale Anforderungen (User Stories)

Die Aufgabenstellung erfordert die Entwicklung einer spezifischen Software, womit sich diese Bachelorarbeit auf die Lösungserstellung fokussiert. Im Folgenden sind die Anforderungen als User Stories und in den Nicht-Funktionalen Anforderungen beschrieben und werden im darauffolgenden Kapitel erläutert.

4.2.1 Systemadministrator

- Als Systemadministrator möchte ich überprüfen können wie die aktuelle und vergangene Auslastung des Chatbots war, rückwirkend auf die letzten 30 Tage.
- Als Systemadministrator möchte ich Einsicht haben, wie viele Benutzer und Nachrichten eingegangen sind über die letzten 30 Tage.
- Als Systemadministrator möchte ich die Möglichkeit haben, flexibel zwischen Bot Service Modellen zu wechseln, um den Anfragen gerecht zu werden.
- Als Systemadministrator möchte ich die durchschnittliche Antwortzeit des Chatbots einsehen können.

4.2.2 Chatbot Verantwortlicher

- Als Verantwortlicher möchte ich neue Fragen und Antworten in der Knowledge Base hinzufügen, um den Chatbot zu erweitern.
- Als Verantwortlicher möchte ich bestehende Fragen und Antworten bearbeiten und korrigieren können, damit präzisere Antworten und besser verständliche Fragen vorhanden sind.
- Als Verantwortlicher möchte ich den Chatbot gezielt trainieren.
- Als Verantwortlicher möchte ich Einsicht haben in bereits abgehandelte Konversationen zwischen Chatbot und Kunde, um Rückschlüsse auf den Konversationsverlauf zu erhalten.

4.2.3 Supportmitarbeiter

- Als Supportmitarbeiter möchte ich dem Teamleiter vorschlagen und informieren können, einen oder mehrere neue Fragetypen in die Knowledge Base aufzunehmen.

4.2.4 Kunde

- Als Kunde möchte ich mein Anliegen rasch und kompetent beantwortet haben.
- Als Kunde möchte ich die Möglichkeit haben, mein Anliegen mit einem Chatbot oder einem Supportmitarbeiter behandelt zu haben.
- Als Kunde möchte ich jederzeit aus einem Gespräch mit dem Chatbot aussteigen und mich mit einem Supportmitarbeiter in Verbindung setzen.

4.3 Nicht funktionale Anforderungen (NFA)

4.3.1 Kompatibilität

Synopsis	Der Chatbot muss auf Microsoft Azure lauffähig sein
Messbarkeit	Installation und Konfiguration auf Microsoft Azure möglich für den Systemadministrator.

Tabelle 2: Kompatibilität (NFA)

4.3.2 Benutzbarkeit

Synopsis	Die Usability ist auf einem Stand, welche dem Benutzer einfach und zugänglich ermöglicht, seine Fragen zu stellen. Dies bedeutet, dass ein neuer Benutzer beim erstmaligen Benutzen, eine korrekte Antwort erhält oder aufgefordert wird die Frage zu präzisieren.
Messbarkeit	Benutzerfeedback ist positiv bei der Bedienung des Chatbots.

Tabelle 3: Benutzbarkeit (NFA)

4.3.3 Reaktionszeit

Synopsis	Chatbot soll dem Benutzer bei Eingaben nach max. 2 Sekunden eine Rückmeldung geben. Bei längeren Wartezeiten (z.B. Datenbankzugriff oder Datenbankabfrage), soll es eine Info geben (z.B. «Suche in der Datenbank, bitte warte einen kurzen Moment»)
Messbarkeit	Zeit messen zwischen den Benutzereingaben und der Response.

Tabelle 4: Reaktionszeit (NFA)

4.3.4 Mengenanforderung

Synopsis	Es sollen bis zu 50 Benutzer gleichzeitig auf den Chatbot zugreifen können.
Messbarkeit	Keine messbaren Performanceeinbrüche bei bis zu 50 Benutzern.

Tabelle 5: Mengenanforderungen (NFA)

4.4 Qualitätsmerkmale [9]

Die folgenden Qualitätsmerkmale richten sich nach dem ISO Standard 9241, eine standardisierte Richtlinie für «Mensch-Computer-Interkation». Über alle Merkmale hinweg kann man diese in drei Kategorien einteilen: Efficiency, Effectiveness und Satisfaction.

- «**Efficiency**» bezieht sich auf die Genauigkeit und Vollständigkeit, mit denen Benutzer bestimmte Ziele erreichen möchten.
- «**Effectiveness**» beschreibt wie gut Ressourcen eingesetzt werden, um diese Ziele zu erreichen.
- «**Satisfaction**» beschreibt die Zufriedenheit und Vertrauenswürdigkeit gegenüber der Software.

Die nachfolgende Tabelle unterteilt diese in verschiedene Kategorien.

EFFICIENCY	
Kategorie	**Qualitätsmerkmale**
Performanz	• Robustheit gegen Manipulation • Robustheit gegen unerwarteten Input • Verhindern von unangemessenen Äusserungen
EFFECTIVENESS	
Kategorie	**Qualitätsmerkmale**
Funktionalität	• Interpretiert die Frage korrekt • Antwort ist angemessen formuliert • Verarbeitet die angefragten Aufgaben
Menschlichkeit	• Chatbot legt offen, dass es sich um eine künstliche Intelligenz handelt • Fähig Antworten zu spezifischen Fragen zu liefern • Natürliche Interaktion
SATISFACTION	
Kategorie	**Qualitätsmerkmale**
Moral & Verhalten	• Schützt und respektiert die Privatsphäre des Benutzers • Wahrheitsgetreue Antworten
Zugänglichkeit	• Versteht die Frage im Kontext so gut wie möglich

Tabelle 6: Qualitätsmerkmale

4.5 User Experience [10]

Um eine möglichst positive User Experience zu erreichen, müssen messbare Merkmale definiert werden. In den folgenden Kapiteln, werde solche aufgelistet und erläutert.

4.5.1 Nachrichtengrösse

Abhängig vom Ausgangsgerät, sollte eine Nachricht vom Chatbot grundsätzlich nie mehr als drei Zeilen enthalten. Dies zieht als Konsequenz nach sich, dass Antworten kurz und prägnant zu halten sind. In Zahlen bedeutet dies, dass maximal 100 Zeichen verwendet werden. Bei grösseren und komplexeren Antworten empfiehlt es sich daher, die Antwort aufzuteilen sodass die maximale Nachrichtenlänge berücksichtigt wird. Eine gute User Experience zeichnet sich aus, indem der Benutzer nicht gezwungen ist im Chat zu scrollen, um die ganze Antwort zu lesen.

4.5.2 Hilfe anbieten

Es wird davon ausgegangen, dass ein grosser Teil der Benutzer wenig bis keine Interaktion mit einem Chatbot hatten. Daher ist es zwingend notwendig jederzeit Hilfestellung anzubieten. Durch eine Nachricht vom Benutzer wie «Hilfe» oder «Ich brauche Hilfe», wird eine Kurzanleitung zur Bedienung des Chatbots angeboten.

4.5.3 Begreiflichkeit

Dem Benutzer muss stets klar sein, welche Möglichkeiten er zurzeit hat und was von ihm erwartet wird als Interaktion. Der Chatbot fordert daher den Benutzer auf ihm sein Anliegen zu stellen. Nach der Beantwortung läuft die Konversation weiter und der Benutzer kann eine weitere Frage stellen. Sollte nach einer Weile kein Input stattfinden, weckt sich der Chatbot selbst auf und interagiert mit dem Benutzer.

4.6 Auswertung der Konversationsqualität

In der Bachelorarbeit werden drei verschiedene Prototypen entwickelt. Um diese in ihrer Qualität und Benutzerfreundlichkeit zu testen, werden einige Kennzahlen aus den abgehandelten Konversationen entnommen. Es werden grundsätzlich Usability- und Performance Tests durchgeführt.

Kriterium	Beschreibung
Konversationsrate	Anzahl der Benutzer, welche bereit waren ihre E-Mail-Adresse zu hinterlegen.
Konversationsdauer	Die durchschnittliche Dauer einer Konversation vom Aufruf des Chatbots bis zur Verabschiedung.
Anzahl Aktionen pro Konversation	Wie viele effektive Fragen und Antworten wurden ausgetauscht? Eine Aktion beschreibt einen Request, welcher an den Chatbot gesendet wird. Die dazugehörige Antwort ist inkludiert in den Request.
Abbruchrate	Anteil der Benutzer welche frühzeitig den Chatbot verlassen. Kriterium ist, dass ein Benutzer nicht die gewünschte Antwort erhielt und kurzerhand den Chatbot schliesst.
Reaktionszeit	Die Zeit, welche es braucht bis eine visuell sichtbare Aktion des Chatbots stattfindet. Sollte das Abrufen der Antwort länger als 2 Sekunden dauern, muss der Chatbot den Benutzer informieren, dass er noch beschäftigt ist und es einen kurzen Moment dauern kann

Tabelle 7: Konversationsqualität

5. Chatbot Evaluation [11]

5.1 Einleitung

In letzter Zeit haben die Chatbots von Twitter und Facebook für Schlagzeilen gesorgt, weil sie durch die Endbenutzer manipuliert wurden. Dadurch wurde der Begriff «Chatbot» auch in den Medien bekannt. Durch den Einsatz von Technologien wie Amazon Alexa, WhatsApp, WeChat usw. hat es dazu geführt, dass auch Chatbots in der Unterhaltungsindustrie weit verbreitet sind. Um Chatbots auf den aktuellen Stand zu bringen haben Ingenieure von grossen Technologiekonzernen wie Facebook, Google und Microsoft sowie auch etliche Startups neue Produkte entwickelt, damit die Entwicklung und Integration von Chatbots für Entwickler einfacher wird. Dabei unterscheidet man zwischen Bot Frameworks und Bot Plattform. Im nachfolgenden Kapitel werden diese Begriffe detailliert erklärt sowie deren Produkte. Am Ende wird entschieden welches Bot Framework sowie Bot Plattform für die Umsetzung der ewz Chatbot gewählt wird.

5.2 Bot Frameworks

Leider werden die Begriffe Bot Framework sowie Bot Plattform meistens verwechselt, obwohl sie zwei Unterschiedliche Bedeutungen haben. Bot Framework ist wie jedes andere Web Framework ein Framework, welches dem Entwickler hilft den Chatbot zu entwickeln und diese mit anderen Komponenten zu verknüpfen. Dabei wird häufig der Template Code vom Framework zur Verfügung gestellt. Die Bot Frameworks beinhalten viele vordefinierte Methoden und Klassen, welches die Entwickler unterstützt Grundlegende Funktionen nicht neu zu implementieren. Der Entwickler muss sich nur an den entsprechenden Hook einhängen und nur die nötigen Änderungen für die jeweiligen Applikation vornehmen. Ohne Bot Framework müsste jeder Entwickler für jeden neuen Chatbot vom Grund auf programmieren. Die bekanntesten Bot Frameworks sind Microsoft Bot Framework, «Botpress.io» sowie «Dialogflow» von Google. In den nachfolgenden Kapiteln werden diese Bot Frameworks detailliert analysiert.

5.2.1 Microsoft Bot Framework [12]

Microsoft gehört zu den ältesten Technologiekonzernen der Welt. Obwohl Microsoft zu den führenden Desktop Software Herstellern der Welt gehört, haben sie leider den Anschluss beim Mobile Software Bereich verpasst. Beim Chatbot Ökosystem sieht das ganz anderes aus, hier hat Microsoft die Vorreiterrolle eingenommen. Seit 2016 bietet Microsoft den Azure Bot Service mit anderen Cognitive Services wie LUIS oder auch bekannt als Language Understanding Service, QnA Maker usw. Microsoft stellt den Entwicklern eine einfache Plattform zur Verfügung, wo sie schnell einen Chatbot aufbauen und gleich in die Azure Cloud deployen können. Dabei können die Entwickler selbst entscheiden ob sie nodeJS oder C# einsetzen wollen, ob sie im «Built-in code editor» die Änderung vornehmen oder auf einem lokalen Editor und diese über die Github in die Azure Cloud mit Hilfe von Azure DevOps kontinuierlich deployen wollen. Namenhafte Unternehmen wie Daimler, ABB, Telefonica, UPS usw. setzen auf Microsoft Azure Bot Services um ihren eigenen Chatbot zu entwickeln [13]. Um den Chatbots noch mehr Reichweite zu geben kann es mit anderen Channels wie Facebook Messenger, Slack, Skype usw. relativ einfach verbunden werden. Die nachfolgende Abbildung gibt einen kurzen Überblick darüber welche Komponenten beim Aufbau eines Chatbots nötig sind.

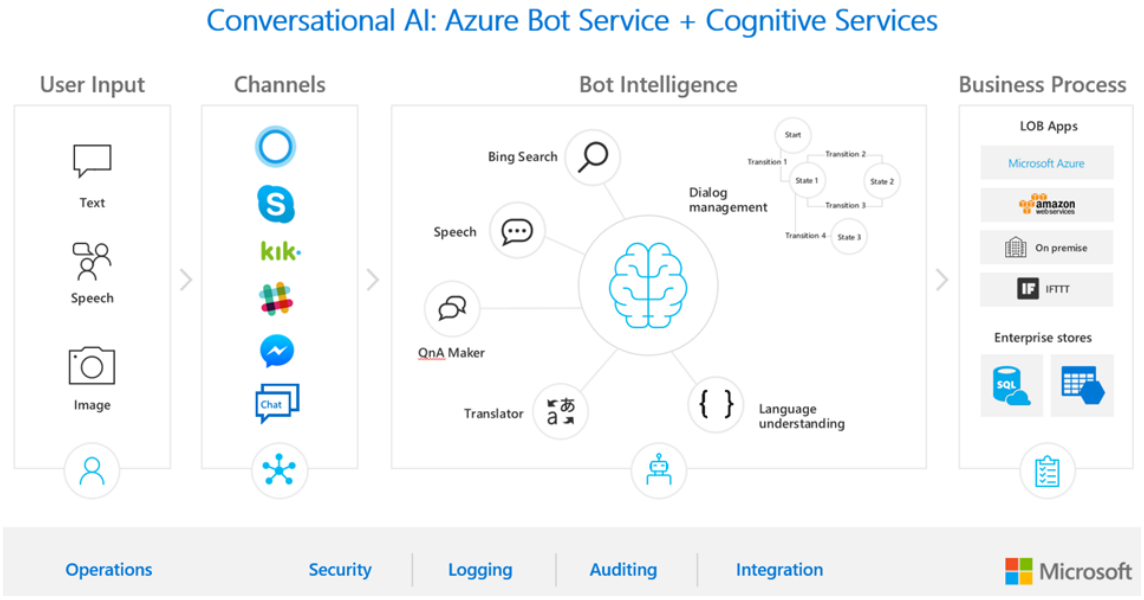


Abbildung 7: Azure Bot Service + Cognitive Service [14]

Ohne Künstliche Intelligenz kann der Chatbot nicht selbst lernen, aber dafür bietet Microsoft eine Palette an Funktionen unter den Namen «Cognitive Services» in den Bereichen Bildverarbeitung, Sprach Assistenten usw. an. Im nachfolgenden Bild sind alle Cognitive Services von Microsoft aufgelistet, welche momentan auf Microsoft Azure Cloud verfügbar sind.

Abbildung 8: Microsoft Cognitive Services Überblick[8] [15]

Das folgende Kapitel behandelt zwei dieser Services genauer welche bei dieser Bachelorarbeit zum Tragen kommen.

[8] Da Microsoft Cognitive Services immer mit neuen Produkten ergänzt werden, bitten wir sie die Microsoft Cognitive Services Webseite zu Besuchen um die aktuelle Liste zu erhalten.

5.2.1.1 QnA Maker [16]

Die meisten Unternehmen haben auf ihrer Webseite eine Liste von Häufigen Fragen und die dazugehörigen Antworten. Die ewz betreibt einerseits eine FAQ Seite bei der Telecom Abteilung sowie dem Kerngeschäft, der Stromversorgung. Leider wird dieses Angebot nicht immer genutzt und Kunden rufen direkt den 1st Level Support an, welche wertvolle Zeit von Mitarbeitern in Anspruch nimmt.

Abbildung 9: Häufige Fragen bei der ewz Telecom [17]

Microsoft bietet hier einen Lösungsansatz an mit dem QnA Maker Service. Mit diesem Service kann der Entwickler ein Modell von möglichen Fragen und dazu gehörigen Antworten aufbauen. Dieser Frage und Antwortkatalog dient als Wissensdatenbank für einen Chatbot, der darauf zugreifen kann. Die «Intelligenz» eines Chatbots, hängt also direkt ab wie gepflegt und ausgeklügelt diese Wissensdatenbank ist.

Für den Aufbau eines QnA Maker Services sind keine Programmierkenntnisse nötig, da Microsoft ein GUI anbietet um das Model aufzubauen, zu verwalten und trainieren. Nach dem Aufbau kann der QnA Maker Service auf Microsoft Azure als REST Schnittstelle zur Verfügung gestellt werden. Da es sich um eine REST Schnittstelle handelt, können auch andere Bot Frameworks für die Entwicklung des Chatbots eingesetzt werden, die den Entwicklern mehr Wahlfreiheit lässt. QnA Maker Services sind kostenpflichtig und können mehrere solcher Modelle oder auch bekannt als «Knowledge Base» bedienen. QnA Maker Service ist in unterschiedlichen Sprachen verfügbar unter anderem auch in Deutsch. Pro Sprache muss eine eigene QnA Maker Service Instanz aufgeschaltet werden [18]

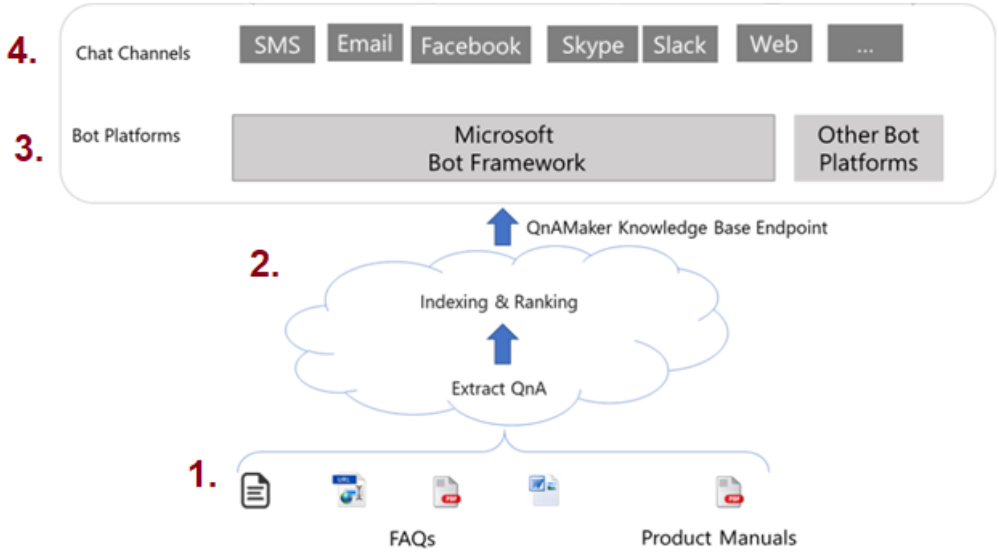


Abbildung 10: Logisches Diagramm QnA Maker Service [16]

Bemerkungen:

1. Häufige Fragen können direkt aus Web Link, PDF, Word Dokument usw. automatisch extrahiert werden.
2. QnA Maker Modelle können von Entwicklern angepasst und trainiert werden. Anschliessend kann das Modell als REST API End Point freigegeben werden. Das QnA Maker Modell wird im Azure Search Service gehostet.
3. Der Entwickler kann selbst entscheiden ob er mit Microsoft Bot Framework oder mit einem anderen Bot Framework arbeitet, da das QnA Maker Modell als REST API End Point für die Entwickler zugreifbar ist.
4. Anbindung an mehrere Channels ist möglich, um die Reichweite des Chatbots zu erweitern.

Die Kosten spielen eine wichtige Rolle bei der Entwicklung eines Chatbots. In den nachfolgenden Tabellen werden die Kosten für den Betrieb eines QnA Maker Services aufgelistet.

Instance	Transaktionen pro Sekunde	Zusätzliche Features	Kosten
Gratis	3	Jedes Dokument[9] darf max. 1 MB gross sein	Drei Dokumente können Gratis pro Monat verwaltet werden.
Standard	3	Keine	$ 10 für unlimitierte Anzahl Dokument

Tabelle 8: Kosten für QnA Maker Management Services (Portal und Management APIs)[10]

Zusätzlich zu den oben genannten Kosten fallen noch die Kosten für die Daten und die Runtime Umgebung an, wenn eine QnA Maker Ressource erstellt wird. Wie zum Beispiel: QnA Maker Modell wird im Azure Search Service gespeichert sowie die Runtime Umgebung wird im Azure App Service gehostet.

Beschreibung	BASIC [Dedizierte Umgebung für dev/test]	STANDARD [für Produktion Umgebung]	PREMIUM [gesteigerte Leistung & Skalierung]	ISOLATED [hohe Leistung, Sicherheit & Isolation]
Web, mobile oder API apps	Unbeschränkt			
Speicherkapazität	10 GB	50 GB	250 GB	1 TB
Max. Instanzen	Bis zu 3	Bis zu 10	Bis zu 20	Bis zu 100
Kundenspezifische Domain	Ja			
Automatische Skalierung	Nein	Ja		
Netzwerk Isolation	Nein			Ja
Kosten pro Stunde	$ 0.075	$ 0.10	$ 0.20	$ 0.30

Tabelle 9: Kosten für Azure App Service[11] [19]

[9] Der Frage und Antworten Katalog im PDF, CSV oder URL Format

[10] Diese Tabelle wurde am 01.10.2018 erstellt und kann noch Daten erhalten, die nicht aktuell sind. Aus diesem Grund bitten wir Sie die Website von Microsoft zu besuchen, um die aktuellen Kosten zu erhalten.

[11] Diese Tabelle wurde am 01.10.2018 erstellt und kann noch Daten enthalten, die nicht aktuell sind. Aus diesem Grund bitten wir Sie die Website von Microsoft zu besuchen, um die aktuellen Kosten zu erhalten.

	BASIC	STANDARD S1	STANDARD S2
Storage	2 GB pro Service	25 GB / Partition (max. 300 GB Dokumente pro Service)	100 GB / Partition (max. 1.2 TB Dokumente pro Service)
Max Index pro Service	5	50	200
Skalierung Limit	Bis zu 3 Einheiten pro Service (max. 1 Partition; max. 3 Replikation)	Bis zu 36 Einheiten pro Service (max. 12 Partition; max. 12 Replikation)	Bis zu 36 Einheiten pro Service (max. 12 Partition; max. 12 Replikation)
Daten Transfer	Ist abhängig vom Inbound und Outbound Traffic Tarif von Microsoft Azure. Achtung dies sind zusätzliche Kosten, welche von Nutzungsverhalten abhängen.		
Kosten pro Stunde	$ 0.101	$ 0.336	$1.344

Tabelle 10: Kosten für Azure Search Service[12] [20]

Zusätzlich hat der Entwickler die Möglichkeit die «Application Insights Service[13]» unter Azure Monitor Service für sein QnA Maker Service zu verwenden, die ebenfalls zusätzliche Kosten verursachen. Es gibt einen Einblick über die Instanz wie zum Beispiel über die Performance, Auslastung und Logs. Im Azure Portal kann es detailliert dargestellt und analysiert werden. Zusätzlich können automatisch abhängig von den Meldungen, bestimmte Aktionen ausgeführt werden. Die Kosten für die «Application Insights» sind abhängig vom Volumen an Daten, die von der Applikation gesendet und die Anzahl and Web Tests, die durchgeführt werden.

Feature	Gratis Paket	Kosten
Ingress Datenverkehr	5 GB pro Monat[14]	$ 2.99 pro GB pro Monat
Daten Speicherung	31 Tage[15]	$0.13 pro GB pro Monat
Multi schritt Web Tests	Keine	$10 pro Test pro Monat
Ping Web Tests	Keine	Gratis

Tabelle 11: Kosten für die «Application Insights» Service[16] [21]

Es gäbe noch weitere Analyse und Monitoring-Tools. Für den Chatbot der ewz reicht «Application Insight» aus, um an die nötigen Informationen zu gelangen. Daher wird an dieser Stelle auf die anderen Monitoring Tools nicht weiter im Detail darauf eingegangen.

5.2.1.2 LUIS [22]

Der Bereich Machine Learning und Künstliche Intelligenz kam erst in den letzten Jahren richtig auf und fand Verbreitung in der Software Entwicklung. Entwickler haben die Möglichkeit Chatbot Applikation mit Machine Learning zu verknüpfen. Dazu bietet Microsoft das LUIS Framework, Language Understanding, an. Beim LUIS handelt es sich um eine Cloud basierte API, welches Machine Learning

[12] Diese Tabelle wurde am 01.10.2018 erstellt und kann noch Daten enthalten, die nicht aktuell sind. Aus diesem Grund bitten wir Sie die Website von Microsoft zu besuchen, um die aktuellen Kosten zu erhalten

[13] Performance Monitoring Tool

[14] Die ersten 5 GB Ingress Datenverkehr sind gratis für jeden Monat.

[15] Alle Daten, die ins Azure Monitor Service kommen sind für die ersten 31 Tage gratis gespeichert.

[16] Diese Tabelle wurde am 02.10.2018 erstellt und kann noch Daten enthalten, welche nicht aktuell sind. Aus diesem Grund bitten wir Sie die Website von Microsoft zu besuchen, um die aktuellen Kosten zu erhalten.

Intelligenz auf die Benutzer Konversation anwendet um die relevanten Informationen wie Ziel oder auch die Stimmung der Benutzer zu extrahieren. Nach dem Aufbau des LUIS Model, wird es als REST API zur Verfügung gestellt. Dadurch können die Entwickler entscheiden mit welchem Framework sie den Chatbot oder die Applikation aufbauen wollen. Nachdem der Entwickler das LUIS Model auf Microsoft Azure gehostet hat, kann der Chatbot die Benutzer Eingabe an den End Point senden. Der Entwickler erhält das Resultat im JSON Format. Die nachfolgende Abbildung zeigt was im Hintergrund passiert, wenn eine Konversation zwischen einem Benutzer und dem Chatbot passiert.

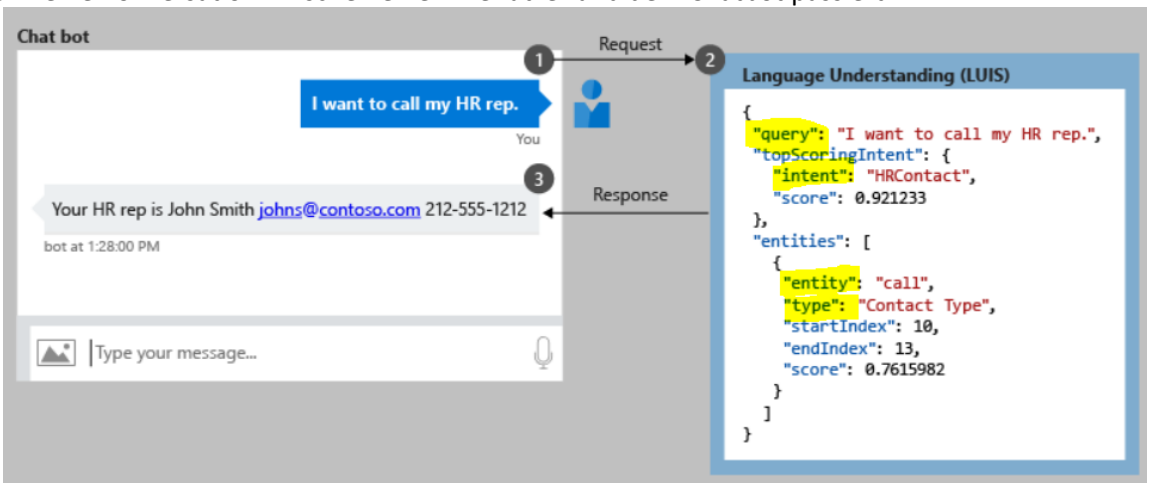


Abbildung 11: Konversationsfluss zwischen Chatbot und LUIS [23]

Bemerkung:

1. Der Benutzer sendet die folgende Anfrage über den Chatbot an der LUIS API.
2. Abhängig vom Modell, das bei der Entwicklung festgelegt wird, versucht LUIS das Ziel (Intent) des Benutzer zu extrahieren. Wie in der Abbildung ersichtlich ist, generierte die LUIS API ein JSON Resultat. Dabei erhält es folgende Informationen:
    a. «query» → Der Frage welche der Benutzer gestellt hat
    b. «intent» → Die Intention der Frage, in diesem Fall das HR kontaktieren.
    c. «entity» → In diesem Fall anrufen
    d. «type» → Abhängig vom Modell wurde «Contact Type» gewählt
3. Die Chatbot Applikation muss abhängig vom JSON Resultat die richtige Entscheidung treffen. Dies kann zum Beispiel ein Entscheidungsbaum sein der im Bot Framework Code vorhanden ist oder andere Services aufrufen, welche die Entscheidung treffen und die richtige Antwort zusenden. In diesem Fall ist die Antwort wie folgt: «Your HR rep is John Smith johns@contoso.com 212-555-1212».

Natürlich fallen auch für diesen Service wie auch beim QnA Maker Service zusätzliche Kosten an. Diese werden in der nachfolgenden Tabelle detailliert aufgelistet.

Instanz	Transaktionen pro Sekunde	Features	Kosten
Gratis	5	Textanfrage	10000 gratis Transaktionen pro Monat
Standard	50	Textanfrage	$1.50 pro 1000 Transaktionen[17]
		Sprachan-frage	$5.50 pro 1000 Transaktionen[18]

Tabelle 12: Kostengestaltung für die Azure LUIS Service [24]

[17] Eine Transkation heisst eine Frage mit einer Länge von 500 Buchstaben.
[18] Eine Transaktion heisst eine Frage von max. 15 Sekunden Dauer.

5.2.1.3 Monatliche Kosten

Dieses Kapitel behandelt die Kostenberechnung für QnA Maker und LUIS Service, sowie welche zusätzlichen Kosten anfallen durch andere Produkte wie Azure Web Service. In Kapitel 5.3.1 werden die Kosten vom Azure Web Service Hosting detailliert aufgelistet. In dieser Tabelle werden nur die Kosten für den Text basierten Chatbot berechnet, da dies die Anforderung von ewz deckt.

Produkt	Sub-Produkt	Kosten pro Monat in $	Kosten pro Monat in CHF
QnA Maker Management Service	Standard Instanz	$ 10	CHF 10
Azure App Service	Standard Instanz	$0.10 * 720 Stunden = $72	CHF 72
Azure Search Service	Basic Instanz	$0.101 * 720 Stunden = $72.72	CHF 72.72
Application Insights Service	Ingress Datenverkehr	$2.99 pro GB * 10 GB = $29.90	CHF 2.99
	Daten Speicherung	$0.13 pro GB * 20 GB = $2.6	CHF 2.60
	Multi schritt Web Tests	$10 pro Test	CHF 10
		Total Sub-Kosten pro Monat	170.31

Produkt	Sub-Produkt	Anzahl Transaktion[19] im Monat beim ewz Telecom	Kosten pro Anfrage im $	Kosten pro Anfrage im CHF[20]	Total Kosten pro Monat im CHF
Azure LUIS Service	Standard Instanz	12000	$0.0015	CHF 0.0015	CHF 18
			Total Kosten pro Monat		CHF 188.31

Tabelle 13: Monatliche Kosten für Azure basierte Chatbot beim Einsatz im ewz Telecom

5.2.2 botpress.io [25]

Beim “botpress.io” handelt es sich um ein Open Source Bot Framework. Der Vorteil von Open Source ist, dass keine monatliche Kosten für das Unternehmen anfallen. Auf der anderen Seite bietet auch «botpress.io» einen Enterprise Service für Unternehmen an welcher kostenpflichtig ist. Dabei begleitet «botpress.io», das Unternehmen bei der Entwicklung eines Chatbots. Leider gibt es keine detaillierten Angaben zum Enterprise Service von «botpress.io», da diese von Unternehmen zu Unternehmen verschieden sind. «botpress.io» bietet nicht nur die Möglichkeit, Web basierte Chatbots zu erstellen, sondern auch eine Anbindung an andere Chat Plattformen wie Slack, Facebook, Telegram usw. herzustellen. Ein Chatbot ohne NLU oder auch bekannt als «Natural Language Understanding» muss die Schlüsselwörter in einem Satz erkennen, um die richtige Antwort zu geben. Leider bietet «botpress.io» keine eigene NLU, stattdessen müssen die Entwickler auf die NLU anderer Hersteller zurückgreifen wie LUIS von Microsoft oder Watson von IBM. Dieser Dienst verursacht zusätzliche Kosten aufgrund des

[19] Eine Transkation heisst eine Frage mit einer Länge von 500 Buchstaben.
[20] Der Wechselkurs von USD auf CHF fand am 03.10.2018 statt [63]

Hostings. Mit dem Flow Editor wird ein mächtiges Tool für die Entwickler zur Verfügung gestellt, um Konversationsflüsse abzubilden. Wie auch jedes andere Bot Framework bietet auch «botpress.io» analytische Werkzeuge, um den Chatbot unter die Lupe zu nehmen. Da es sich um einen Open Source Bot Framework handelt, kann es auch auf bekannten Hosting Plattformen gehostet werden. Die Hosting Kosten werden im Kapitel Bot Plattform aufgezeigt.

Abbildung 12: botpress.io Logo [26]

5.2.3 Dialogflow [27]

«Dialogflow» ist wiederum eine anderes Webbasiertes Bot Framework. «Dialogflow», vorher auch bekannt als Api.ai und «Speakoit», wurde im September 2016 von Google aufgekauft [28]. Dialogflow erlaubt den Entwicklern nicht nur textbasierte, sondern auch sprachbasierte Chatbots zu entwickeln. Zusätzlich erlaubt es nicht nur eine direkte Anbindung an die Website, sondern auch an andere Dienste wie Google Assistant, Amazon Alexa, Telegram Messenger, WeChat usw. Dabei setzt Dialogflow für die «Natural Language Processing» (NLP) auf die Expertise von Google's «machine learning» und wird auf Google's Cloud Plattform gehostet. Grosskonzerne wie KLM Royal Dutch Airlines, Singapore Airlines usw. setzen auf Dialogflow, um ihre eigenen Chatbots zu entwickeln. Die Kosten spielen eine wichtige Rolle bei der Entwicklung von Chatbots, wobei eine gratis Version mit Einschränkungen zur Verfügung steht. Die Kosten richten sich monatlich nach von Version, Preisplan, Anzahl Abfragen und die Dauer von Konversationen. Die nachfolgende Tabelle zeigt eine detaillierte Auflistung der Kosten. Es gilt zu beachten, falls parallel zu Dialogflow andere Google Cloud Platform (GCP) Produkte wie Google App Engine Instanz im Einsatz sind, zusätzliche Kosten anfallen werden.

Abbildung 13: Dialogflow Logo

	Sprachen [29]	Standard Edition	Enterprise Edition	
			Essentials	Plus
Knowledge Connectors	Eng, Fra, Deu, Ita	Gratis	Die Kosten basieren auf Kanälen wie Text, Audio oder Telefon	
Text oder Google Assistant	Eng, Fra, Deu, Ita	Gratis	$0.002 pro Anfrage	$0.004 pro Anfrage
Audio	Eng	Gratis	$0.0065 pro 15 Sekunden	$0.0085 pro 15 Sekunden
Telefon Anrufe (Beta)	Eng	Gratis	$0.05 pro angerufene Minuten	$0.065 pro Minute

Gebührenfreie Telefon Anrufe (Beta)	Eng	Nicht verfügbar	$0.06 pro angerufene Minuten	$0.075 pro Minute

Tabelle 14: Preisgestaltung der Dialogflow Produkte[21]

Google bietet nicht nur das «Dialogflow» Framework, sondern auch die Künstliche Intelligenz Funktionalität, ähnlich wie bei Microsoft unter dem Namen «AI & Machine Learning Products» [30]. Die Kosten für die Produkte sind unterschiedlich, was einen direkten Vergleich zwischen diesen Produkten erheblich schwieriger macht. Dialogflow lässt sich ohne Probleme in QnA Maker sowie LUIS integrieren.

5.2.3.1 Monatliche Kosten

Dieses Kapitel enthält die Kostenberechnung für Dialogflow. Anfallende Kosten welche durch andere Produkte wie durch die Google App Engine Instanz, werden im Kapitel 5.3.3 detailliert aufgelistet. In dieser Tabelle werden nur die Kosten für den textbasierten Chatbot berechnet, da dies auch die Anforderung der ewz deckt.

Produkt	Anzahl Anfragen im Monat bei ewz Telecom	Kosten pro Anfrage in $	Kosten pro Anfrage in CHF	Total Kosten pro Monat in CHF
Text	6000	$0.002	CHF 0.002[22]	CHF 12

Tabelle 15: Monatliche kosten für Dialogflow beim Einsatz im ewz Telecom

5.2.4 Detaillierter Kriterien Katalog

Beschreibung (Gewichtung in %)	MS Botframework	Botpress	Dialogflow
Integration von Services (20%)	**6**	**1**	**4**
Lizenz (20%)	**5**	**6**	**3**
Sprachunterstützung (10%)	**6**	**3**	**3**
Programmiersprachen (10%)	**5**	**3**	**4**
Zukunftssicher (20%)	**5.5**	**4**	**5**
Channel-integration (15%)	**6**	**2**	**6**
Admin Usability (5%)	**5**	**1**	**4**
Total (Schulnote)	**5.15**	**3.15**	**4.2**

Tabelle 16: Kriterienkatalog Bot Frameworks

- **Integration von Services** Wie gut lassen sich weitere Services an das Bot Framework einbinden?
- **Lizenz** Wie sieht das Lizenzmodell aus? Mit welchem Kosten müssen einmalig und monatlich gerechnet werden?
- **Sprachunterstützung** Welche Sprachen, neben Englisch, werden unterstützt?
- **Programmiersprachen** Welche Programmiersprachen werden unterstützt?
- **Zukunftssicher** Inwiefern wird das Produkt vom Hersteller gepflegt und mit Updates versorgt?
- **Channel-integration** Welche 3rd Party Channels (Slack, Skype, Facebook etc.) können angesprochen werden?

[21] Diese Tabelle wurde am 27.09.2018 erstellt und kann noch Daten erhalten, die nicht aktuell sind. Aus diesem Grund bitten wir Sie die Website von Dialogflow zu besuchen, um die aktuellen Kosten zu erhalten.
[22] Der Wechselkurs von USD auf CHF fand am 27.09.2018 statt [63]

- **Admin Usability** Welche möglichen Einstellungen lassen sich auf dem Admin-Portal Vornehmen?
- **Total** Die erreichte Note, anhand der unterschiedlich gewichteten Kriterien

5.3 Bot Plattform

Im Gegensatz zum Bot Framework bietet die Bot Plattform die technische Grundlage, um einen Chatbot zu deployen und zu hosten. Anders gesagt, handelt es sich um ein Ökosystem wo der Chatbot läuft und mit dem Endbenutzer interagiert. Bot Plattformen gibt es von Startups wie Motion.ai und «Chatfuel» bis zu Bot Plattformen von grossen Technologiekonzernen wie Microsoft, Google und Amazon. Im nächsten Kapitel werden die Bot Plattformen von Microsoft, Google sowie Amazon unter die Lupe genommen.

5.3.1 AWS Amazon Web Service [31]

Amazon hat die Vorreiterrolle eingenommen bei der Entwicklung von Cloud Services. Dadurch kontrollieren sie auch mehr als 50 Prozent des Marktes im Jahr 2017 [32]. Amazon bietet auch ihre eigenen Produkte wie Amazon Lex um Chatbots zu entwickeln und diese auf AWS zu deployen. Amazon Lex kann mit LUIS und QnA Maker von Microsoft Azure vergleichen werden, welche sehr gut in AWS integriert werden können. Amazon Lex stellt Machine Learning Funktionalität in einer einfachen Form für die Entwickler zur Verfügung. Der grösste Vorteil von AWS ist, dass wir auch andere Chatbot Frameworks wie botpress.io nehmen können und diese auf die AWS Plattform deployen können. Dabei können wir diesen Chatbot in andere Produkte von AWS wie AWS Lambda, Amazon Lex usw. einfach integrieren. Bekannte Unternehmen, die Produkte von AWS verwenden um Chatbots zu entwickeln sind OhioHealth, CapitalOne, NASA usw.

5.3.1.1 Monatliche Kosten

Um die monatlichen Kosten zu berechnen nehmen wir die nachfolgende logische Architektur. Dabei setzen wir für die Entwicklung eines Chatbot, wenn es möglich ist immer auf ein AWS Produkt.

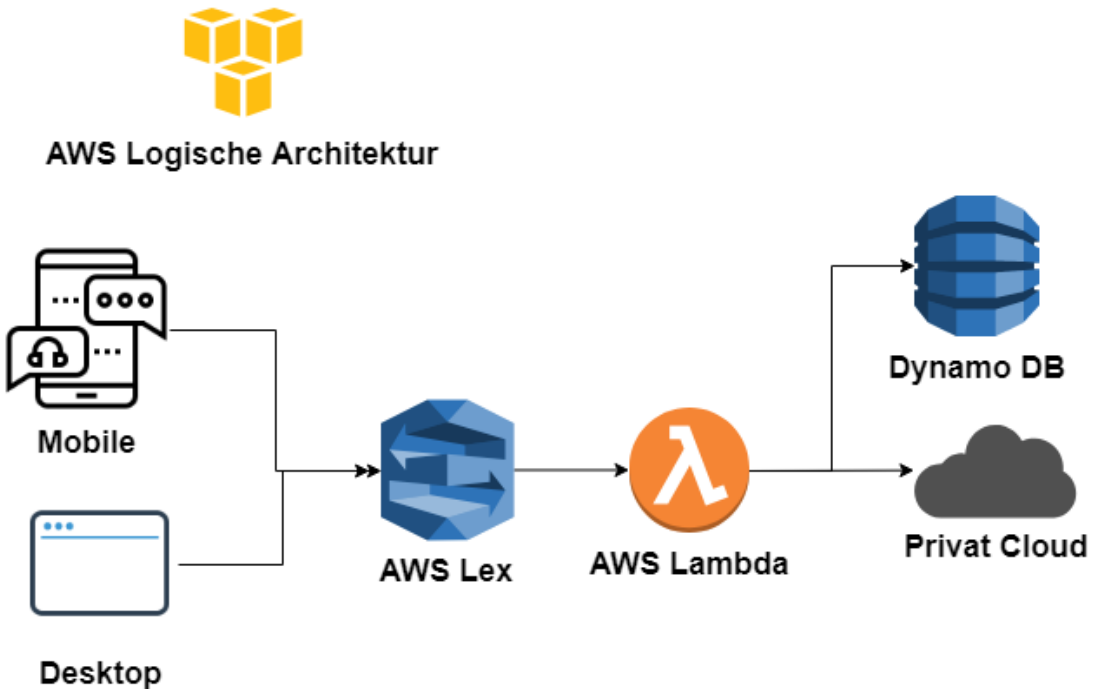


Abbildung 14: AWS Logische Architektur [33]

Produkt	Kosten pro Anfrage (USD)	Kosten pro Anfrage (CHF)[23]
Textanfrage	$ 0.004	CHF 0.004
Sprachanfrage	$0.00075	CHF 0.00075

Tabelle 17: Preisgestaltung für AWS Lex[24]

AWS Lambda oder auch bekannt als «Serverless function» ist ein Code, der ohne explizite Installation auf einem beliebigen Server ausgeführt werden kann. Dabei muss nur die Ausführungszeit des jeweiligen Lambda Code bezahlt werden. Dadurch fallen die Anschaffungskosten weg und es wird nur für die tatsächliche Nutzung bezahlt. Es enthält ein kostenloses Paket, das vom ausgewählten Arbeitsspeicher abhängt. Die nachfolgende Tabelle zeigt die unterschiedlichen Arbeitsspeichergrössen, die entsprechenden Sekunden und der Preis pro 100 ms.

Arbeitsspeicher (MB)	Sekunden pro Monat im kostenlosen Paket	Preis pro 100ms (USD)	Preis pro 100ms (CHF)[25]
128	3200000	0.000000208	0.000000208
256	1600000	0.000000417	0.000000417
512	800000	0.000000834	0.000000834
1024	400000	0.000001667	0.000001667
2048	200000	0.000003334	0.000003334

Tabelle 18: Preisgestaltung für AWS Lambda[26]

Produkt	Art des bereitgestellten Durchsatzes	Preis pro Stunde (USD)	Preis pro Stunde (CHF)	Performance
Bereitgestellter Durchsatz	Schreibkapazitätseinheit (WCU)	0,000793 pro WCU	0,000793 pro WCU	Eine WCU bietet bis zu 3600 Schreibvorgänge pro Stunde
	Lesekapazitätseinheit (RCU)	0,0001586 pro RCU	0,0001586 pro RCU	Eine RCU bietet bis zu 7200 Lesevorgänge pro Stunde

Produkt	Kapazität	Kosten pro GB (USD)	Kosten pro GB (CHF)
Datenspeicher	Erste 25 GB pro Monat	Gratis	Gratis
	Danach pro GB pro Monat	0.306	0.306

Produkt	Sub-Produkt	Übertragene Daten	Preis pro GB (USD)	Preis pro GB (CHF)

[23] Der Wechselkurs von USD auf CHF fand am 04.10.2018 statt
[24] Diese Tabelle wurde am 04.10.2018 erstellt und kann noch Daten erhalten, welche nicht aktuell sind. Aus diesem Grund bitten wir Sie die Website von AWS zu besuchen, um die aktuellen Kosten zu erhalten.
[25] Der Wechselkurs von USD auf CHF fand am 04.10.2018 statt
[26] Diese Tabelle wurde am 04.10.2018 erstellt und kann noch Daten erhalten, die nicht aktuell sind. Aus diesem Grund bitten wir Sie die Website von AWS zu besuchen, um die aktuellen Kosten zu erhalten.

Datenübertragung	Übertragung eingehender Daten	Alle eingehenden Datenübertragungen	Gratis	Gratis
	Übertragung ausgehender Daten	Bis zu 1 GB pro Monat	Gratis	Gratis
		Nächste 9.999 TB pro Monat	0.09	0.09
		Nächste 40 TB pro Monat	0.085	0.085

Tabelle 19: Preisgestaltung für Amazon DynamoDB [34]

Produkt	Sub-Produkt	Kosten pro Monat (USD)	Kosten pro Monat (CHF)
AWS Lex	Textanfrage[27]	$0.004 * 12000 Anfrage = $48	CHF 48
AWS Lambda	2048 MB (Arbeitsspeicher)	$0.0000000334 * 3600000ms [≈ 1h] = $ 0.120024	CHF 0.120024
AWS Dynamo DB	Schreibkapazitätseinheit (WCU)	$0.000793 * 100 = $0.0793	CHF 0.0793
	Lesekapazitätseinheit (RCU)	$0.0001586 * 100 = $0.01586	CHF 0.01586
	Datenspeicher	Total 100 GB - 25 GB pro Monat gratis - Rest 75 GB * $0.306 pro Monat = $22.95	CHF 22.95
	Übertragung ausgehender Daten	50 GB * $0.09 = $ 4.5	CHF 4.5
		Total Kosten pro Monat	CHF 75.66

Tabelle 20: Monatliche Kosten für AWS basierter Chatbot beim Einsatz der ewz Telecom[28]

5.3.2 Microsoft Azure

Mit etwa 30% Marktanteil an zweiter Stelle ist Microsoft Azure [35]. Microsoft ist auch sehr weit vorne dabei, wenn es um Chatbot Entwicklungs-Frameworks und Plattformen geht. Der Entwickler kann auch hier entscheiden ob er mit Microsoft Produkten wie Bot Framework, LUIS und QnA Maker einen Chatbot entwickelt und diese auf Azure aufschaltet. Allenfalls lässt sich auch ein anderes Bot Framework auf Azure deployen. Grundsätzlich empfiehlt es sich, alles aus der gleichen Produktfamilie zu wählen, da weniger Komplikationen auftreten und die Fehlerwahrscheinlichkeit verringert wird.

[27] Anfrage hier ist ein API Aufruf. Es gibt keine Einschränkung bezüglich der Grösse des Texts.
[28] Es beinhaltet keine Kosten für den Traffic von und zu einem Datacenter wie auch zwischen zwei Datacentern, die vom Nutzungsverhalten des jeweiligen Unternehmens abhängen.

5.3.2.1 Monatliche Kosten

Um die monatlichen Kosten zu berechnen nehmen wir die nachfolgende logische Architektur. Dabei setzen wir für die Entwicklung des Chatbots, wenn immer möglich, auf ein Azure Produkt.

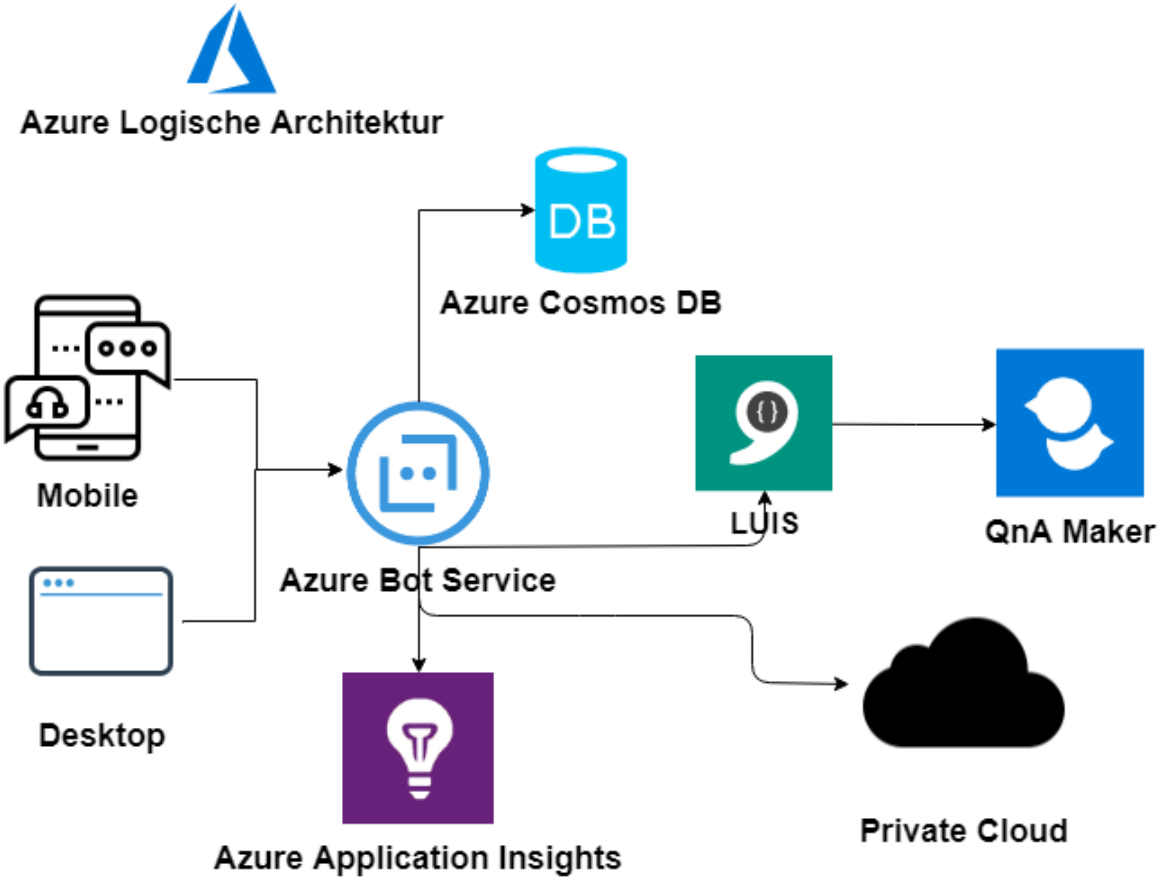


Abbildung 15: Azure Logische Architektur [36]

Die monatlichen Kosten für die Produkte wie Azure Bot Service, LUIS, QnA Maker und Azure Application Insights wurden schon in Kapitel 5.2.1 ausführlich beschrieben. In der unteren Tabelle werden die Kosten für das Produkt Azure Cosmos DB dargestellt.

Unit	Kosten (USD)	Kosten (CHF)
SSD Speicher (pro GB)	0.25 GB pro Monat	0.25 GB pro Monat
Reservierte RUs[29] pro Sekunde (pro 100 Rus, 400 Rus Minimum)	0.008 pro Stunde	0.008 pro Stunde
Bereitgestellter Durchsatz – 100 RU pro Sekunde	0.016 pro Stunde	0.016 pro Stunde

Tabelle 21: Preisgestaltung für Azure Cosmos DB [37]

[29] "In Azure Cosmos DB erfolgt die Abrechnung auf Grundlage des bereitgestellten Durchsatzes und dem Speicherverbrauch nach Stunden. Für die Angabe des Durchsatzes wird eine Währungseinheit für den normalisierten Durchsatz verwendet. Hierbei handelt es sich um Anforderungseinheiten pro Sekunde (RUs) oder im Englisch auch Request Units per Second".

Produkt		Kosten pro Monat (CHF)
QnA Maker Management Service, Azure App Service, Azure Search Service, Application Insights Service, LUIS[30]		CHF 188.31
Produkt	**Sub-Produkt**	**Kosten pro Monat (CHF)**
Azure Cosmos DB	SSD Speicher	50 GB * CHF 0.25 pro GB pro Monat = CHF 12.5
Total Kosten pro Monat in CHF		CHF 200.81

Tabelle 22: Monatliche Kosten für Azure basierten Chatbot beim Einsatz der ewz Telecom (inklusive Hosting)[31]

5.3.3 Google Cloud Plattform

Natürlich darf in diesem Fall Google nicht fehlen. Google bietet mit ihrer Cloud Plattform wiederum einen öffentlichen Cloud-Dienst wie die anderen beiden Technologiekonzerne. Auf der Google Cloud Plattform können auch Chatbots eingesetzt werden. Dabei bietet Google ihre eigenen Produkte wie Dialogflow und die Künstliche Intelligenz aus der eigenen Hand an. Auch hier kann sich der Entwickler selbstverständlich für andere Produkte entscheiden um den Chatbot zu entwickeln und letztendlich diese auf Google Cloud Plattform hosten. Falls man alles aus der Google Produktepalette nimmt ist das Zusammenspiel zwischen den einzelnen Komponenten vereinfacht, was wiederum der Entwicklung zugutekommt.

5.3.3.1 Monatliche Kosten

Um die monatlichen Kosten zu berechnen nehmen wir die nachfolgende logische Architektur. Dabei setzen wir für die Entwicklung des Chatbots, wenn möglich immer auf ein Google Cloud Plattform Produkt.

[30] Die Detaillierte Kostenberechnung dieser Produkte kann unter dem Kapitel 5.2.1 gefunden werden.
[31] Es beinhaltet keine Kosten für den Traffic von und zu einem Datacenter wie auch zwischen zwei Datacentern, der vom Nutzungsverhalten des jeweiligen Unternehmens abhängt.

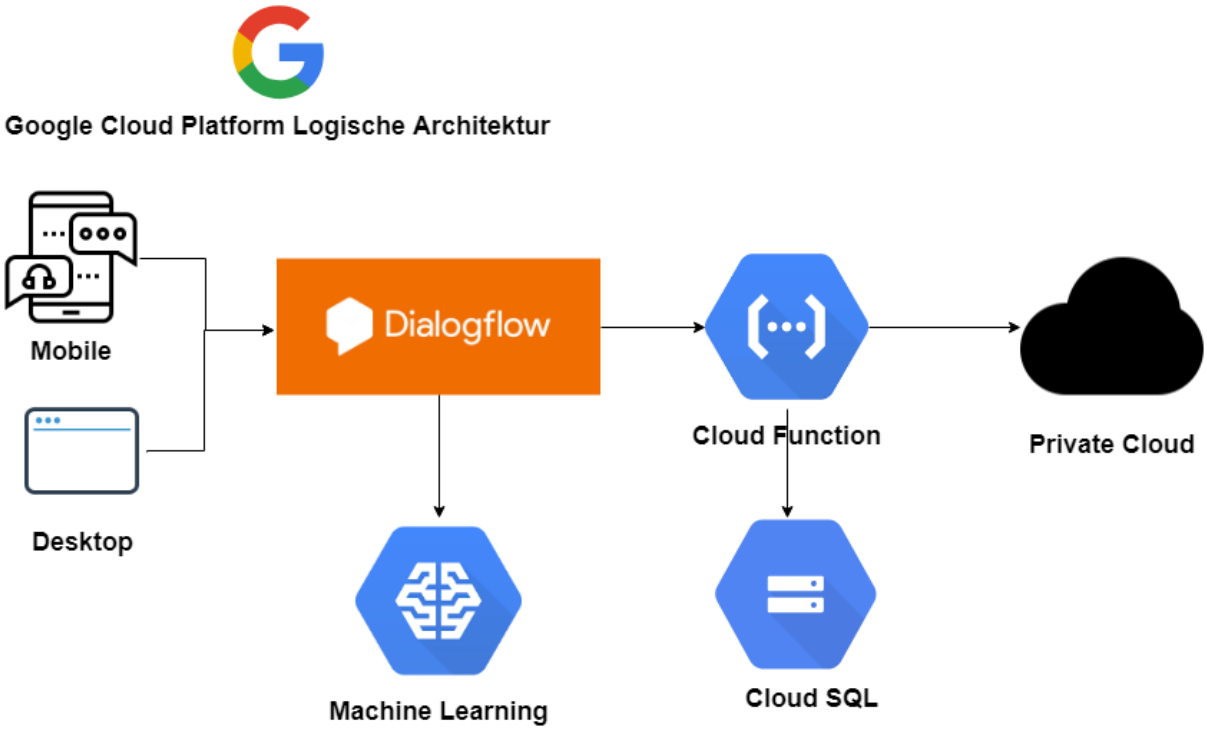


Abbildung 16: Google Cloud Platform Logische Architektur [38]

Die monatlichen Kosten für das Produkt Dialogflow wurde schon in Kapitel 5.2.3 ausführlich beschrieben. In der unteren Tabelle werden die Kosten für die Produkte «Machine Learning», «Cloud Storage» und «Cloud Function» dargestellt.

Option	0 bis 5000	5000+ bis 1 Million[32]
Entitätsanalyse	Gratis	CHF 1 pro 1000 Textdatensätze
Sentimentanalyse	Gratis	CHF 1 pro 1000 Textdatensätze
Syntaxanalyse	Gratis	CHF 0.5 pro 1000 Textdatensätze
Sentiment Analyse pro Entität	Gratis	CHF 2 pro 1000 Textdatensätze
Option	**0 bis 30000**	**30000+ bis 250000**
Inhaltsklassifizierung	Gratis	CHF 2 pro 1000 Textdatensätze

Tabelle 23: Preisgestaltung für Natural Language API (Produkt für KI und maschinelles Lernen) [39]

Die Kosten für «Google Cloud Function» wir nach den folgenden Kriterien berechnet:

- Wie lang war die Funktion aktiv?
- Wie viele Mal wurde es aufgerufen?
- Wie viele Ressourcen wurden für diese Funktion im Vorfeld reserviert?

[32] "Der Preis für die Nutzung der Natural Language API richtet sich danach, wie viele Textdatensätze Sie zur Analyse an die API senden. Ein Textdatensatz entspricht einem Dokument, das als Eingabe in eine Natural Language API-Anfrage bereitgestellt wird. Der Datensatz enthält bis zu 1000 Unicode-Zeichen." [39]

Produkt	Funktionsaufruf pro Monat	Kosten pro Million Aufruf	Kosten pro Aufruf
Funktionsaufruf	Erste 2 Millionen	Gratis	Gratis
	Nach 2 Millionen	CHF 0.40	CHF 0.0000004
Produkt	**Memory**	**CPU**	**Kosten pro 100ms**
Rechenzeit	128 MB	200 MHz	CHF 0.000000231
	256 MB	400 MHz	CHF 0.000000463
	512 MB	800 MHz	CHF 0.000000925
	1024 MB	1.4 GHz	CHF 0.000001650
	2048 MB	2.4 GHz	CHF 0.000002900
Produkt	**Type**		**Kosten pro GB**
Datenverkehr	Ausgehender Datenverkehr (Egress)		CHF 0.12
	Ausgehender Datenverkehr pro Monat		5 GB Gratis
	Eingehender Datenverkehr (Ingress)		Gratis
	Ausgehender Datenverkehr zur Google API im gleichen Datacenter		Gratis

Tabelle 24: Preisgestaltung für Google Cloud Function [40]

Produkt	Stufe	CPUs	RAM (GB)	Speicher (GB)	Kosten pro Tag (CHF)	Max. Anzahl gleichzeitiger Verbindungen§
Preise für MySQL der ersten Generation[33]	D0	0.125	0.125	0.5	0.36	250
	D1	0.25	0.5	1	1.46	250
	D2	0.5	1	2	2.93	250
	D4	1	2	5	4.40	500
	D8	2	4	10	8.78	1000
	D16	4	8	10	17.57	2000
Produkt		**Ressource**		**Kosten**		
Netzwerknutzung[34]		Ausgehender Datenverkehr		CHF 0.12 pro GB		

Tabelle 25: Preisgestaltung für Google Cloud SQL [41]

[33] «Google bietet für Instanzen der ersten Generation 2 Abrechnungstarife an nach Paketen und nach Nutzung. Welcher Abrechnungstarif für das Unternehmen am besten geeignet ist, hängt von der Nutzung der Datenbank ab. Grundsätzlich ist der Pakettarif preiswerter, wenn die Instanz länger als 450 Stunden im Monat genutzt wird.» Hier sind die Kosten nach paketbasiertem Abrechnungstarif dargestellt. [41]

[34] «Die Netzwerknutzung wird sowohl bei paket- als auch bei nutzungsbasierten Abrechnungstarifen in Rechnung gestellt.» [41]

Produkt	Sub-Produkt	Kosten pro Monat in CHF
Natural Language API (Produkt für KI und maschinelles Lernen)	Entitätsanalyse	Total Textdatensätze pro Monat = 12000 Gratis = 5000 Restliche Textdatensätze pro Monat = 12000 – 5000 = 7000 Kosten pro 1000 Textdatensätze = 1 CHF Kosten für 7000 Textdatensätze = 7 CHF
	Syntaxanalyse	Total Textdatensätze pro Monat = 12000 Gratis = 5000 Restliche Textdatensätze pro Monat = 12000 – 5000 = 7000 Kosten pro 1000 Textdatensätze = 0.5 CHF Kosten für 7000 Textdatensätze = 3.5 CHF
Sub-Total Kosten pro Monat		CHF 10.50
Produkt	**Sub-Produkt**	**Kosten pro Monat in CHF**
Google Cloud Function	Funktionsaufruf	Total Funktionsaufruf pro Monat = 4 Millionen Gratis Funktionsaufruf pro Monat = 2 Millionen Restliche Funktionsaufruf pro Monat = 2 Millionen Kosten pro Million Aufruf = CHF 0.40 Kosten für 2 Million Aufruf = CHF 0.80
	Rechenzeit	Paket 1 (Memory 256 MB, CPU 400 MHz, Kosten pro 100ms 0.000000463) Total Funktionsaufruf pro Monat = 4 Million Kosten für 4 Millionen Aufruf = 4 Million * 0.000000463 = CHF 1.852
	Datenverkehr (Ausgehender Datenverkehr)	50 GB * CHF 0.12 pro GB = CHF 6
Sub-Total Kosten pro Monat		CHF 8.652
Produkt	**Sub-Produkt**	**Kosten pro Monat in CHF**
Preis für MySQL der ersten Generation	Stufe D2	CHF 2.93 * 30 = CHF 87
Netzwerknutzung	Ausgehender Datenverkehr	CHF 0.12 pro GB * 50 GB = CHF 6
	Sub-Total Kosten pro Monat	CHF 93
	Total Kosten pro Monat	CHF 112.152

Tabelle 26: Monatliche Kosten für Google Cloud Plattform basierter Chatbot beim Einsatz der ewz Telecom

5.3.4 Detaillierter Kriterien Katalog

Beschreibung (Gewichtung in %)	Azure	AWS	Google Cloud
Preis-/Leistungsverhältnis (20%)	**4**	**6**	**5**
Schnittstellen Kompatibilität (20%)	**6**	**5**	**4**
Support (5%)	**6**	**4**	**4**
Entwicklungs- und Unterhaltungs-aufwand (10%)	**5**	**4.5**	**4.5**
Zukunftssicher (20%)	**6**	**3**	**4**
Datenschutz nach CH Gesetz (20%)	**4**	**1**	**4**
Admin Usability (5%)	**6**	**4**	**3**
Total (Schulnote)	**5.1**	**3.85**	**4.2**

Tabelle 27: Kriterienkatalog Bot Plattformen

- **Preis-/Leistungsverhältnis** Welche Leistungen erhält man für den geforderten Preis?
- **Schnittstellen Kompatibilität** Wie gut sind die verfügbaren Schnittstellen dokumentiert? Wie gut lassen sich weitere Services an die Schnittstellen anbinden?
- **Support** Wie schnell und in welcher Qualität erhält man Support?
- **Entwicklungs- und Unterhaltungsaufwand** Mit welchem Unterhaltungsaufwand muss man rechnen nach der Einführung?
- **Zukunftssicher** Inwiefern wird das Produkt vom Hersteller gepflegt und mit Updates versorgt?
- **Datenschutz nach CH Gesetz** Können die Schweizer Datenschutzgesetze eingehalten werden?
- **Admin Usability** Welche möglichen Einstellungen lassen sich auf dem Admin-Portal vornehmen?
- **Total** Die erreichte Note, anhand der unterschiedlich gewichteten Kriterien

5.4 Entscheidung

Vergleicht man die drei Kandidaten anhand der Kriterienkataloge, macht die Microsoft Produktpalette mit Vorsprung das Rennen. Ausschlaggebend für die Entscheidung war, dass wir möglichst viel aus der gleichen Produktefamilie nehmen. Das Zusammenspiel der einzelnen Komponenten funktioniert generell besser und einfacher als verschiedene Produkte von unterschiedlichen Herstellern zu verbinden. Microsoft hat mit Azure und dem Bot Framework grossartige Plattformen geschaffen, welche sehr gut die Anforderungen für einen Chatbot abdecken.

6. Lösungskonzept

Aus den Anforderungen und der Analyse kann nun ein Lösungskonzept erstellt werden. Eine Übersicht der funktionalen Anforderungen bietet das Use Case Diagramm, wobei die darin enthaltenen Use Cases im brief[35]- und fully dressed[36]-Format beschrieben sind. Das erste Lösungskonzept dient dazu, aus den aufgenommenen Anforderungen einen Vorschlag zu erstellen.

[35] Beschreibung in Textform. User, der das System nutzt um ein Ziel zu erreichen.
[36] Alle Tätigkeiten und Variationen im Detail beschrieben. [64]

6.1 Use Cases

6.1.1 Use Case Diagramm

Im nachfolgenden Use Case Diagramm sind alle Use Cases und Aktoren im Zusammenhang mit dem Chatbot aufgezeigt.

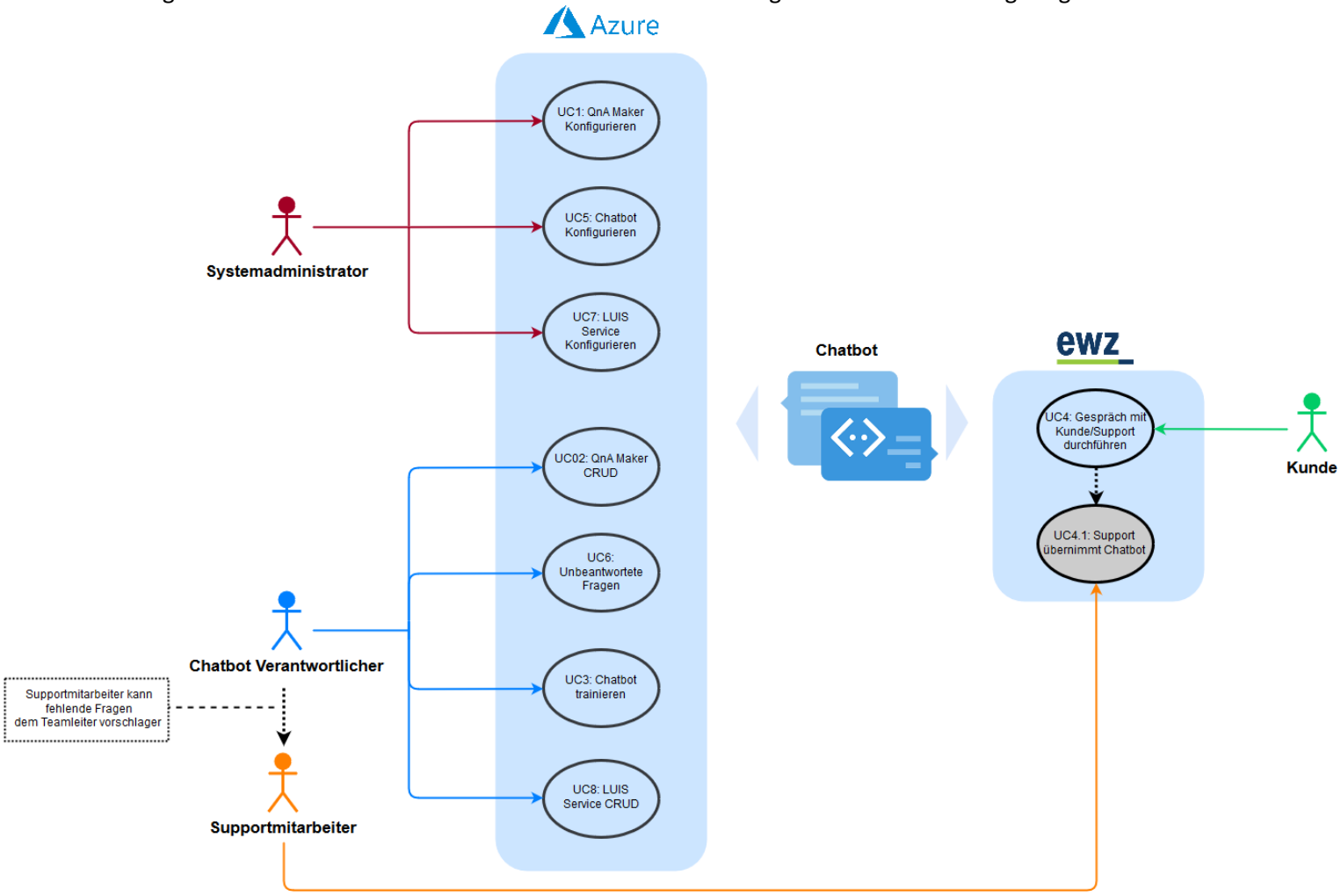


Abbildung 17: Use Case Diagramm

6.1.2 Use Cases (brief)

In den nachfolgenden Kapiteln werden alle Use Cases im brief-Format beschrieben.

UC1: QnA Maker Konfigurieren

Der Chatbot Verantwortliche erstellt und befüllt die Knowledge Base des QnA Makers mit Fragen und Antworten. Dazu verwendet er die verfügbaren Import-Tools des QnA Makers.

UC2: QnA Maker CRUD

Der Chatbot Verantwortliche bearbeitet und vervollständigt die Knowledge Base. Er kann neue Fragen und Antworten hinzufügen oder auch bestehende überschreiben.

UC3: Chatbot trainieren

Im Trainingsmodus trainiert der Chatbot Verantwortliche gezielt den Chatbot, indem er bei unklaren Fragen die korrekten Antworten definiert.

UC4: Gespräch mit Kunde/Support durchführen

Der Kunde führt selbstständig ein Gespräch mit dem Chatbot. Er formuliert schriftlich seine Anliegen in Form einer Frage und erhält die passendste Antwort vom Chatbot.

UC4.1 Support übernimmt Chatbot

Der Supportmitarbeiter kann auf Anfrage des Kunden in persönlichen Kontakt treten. Dazu übernimmt er via Ticketanfrage den Chatbot und interagiert direkt mit dem Kunden.

UC5: Chatbot konfigurieren

Der Systemadministrator editiert und stellt die nötige Konfiguration auf dem Azure Portal für den Chatbot vor. Darunter fallen Optionen wie das Preismodell, Statistiken auswerten oder mit zusätzlichen Kanälen verbinden.

UC6: Unbeantwortete Fragen auswerten

Unbeantwortete Fragen des Chatbots werden auf einer Datenbank zwischengespeichert und können vom Chatbot Verantwortlichen oder Systemadministrator ausgelesen werden für die Weiterverarbeitung.

UC7: LUIS Service konfigurieren

Der Systemadministrator erstellt einen LUIS Service auf dem LUIS Portal und nimmt die nötige Konfiguration vor. Darunter fallen Optionen wie Preismodell und URL Endpunkte festlegen.

UC8: LUIS Service CRUD

Absichten, Entitäten und Beispielsätze können hinzugefügt, editiert und gelöscht werden. Zusätzlich kann der Chatbot Verantwortliche den LUIS Service trainieren und publishen.

6.1.3 Use Case (fully dressed)

Nachfolgend werden die jeweils wichtigsten Use Cases im fully dressed Format aufgeführt.

UC2: QnA Maker CRUD

Goal	**Der Benutzer kann die Knowledge Base erweitern und anpassen**
Level	User Goal
Primary Actor	Benutzer
Trigger	Der Benutzer möchte die Knowledge Base erweitern
Stakeholders and Interests	**Benutzer**: einfache Erstellung/Bearbeitung einer Frage und Antwort **System**: Gewährleistung valider Daten
Preconditions	Der Benutzer ist bereits im QnA Maker angemeldet.
Postconditions	Die getätigten Änderungen sind erfolgreich übernommen worden.
Main Success Scenario	Der Benutzer möchte eine neue Frage inkl. Antwort in der Knowledge Base aufnehmen.
Extentions (or Alternative Flows)	Sollte bereits eine ähnliche Frage bzw. Antwort vorhanden sein, kann der Benutzer entscheiden welches die passendere Antwort ist. Wenn der Benutzer invalide Angaben speichern möchte, verhindert QnA Maker die Änderung und macht den Benutzer darauf aufmerksam.
Frequency of Occurence	Mehrmals im Monat

Tabelle 28: UC 7.2 Knowledge Base updaten

UC3 Chatbot trainieren

Goal	**Der Benutzer kann den Chatbot prüfen und trainieren**
Level	User Goal
Primary Actor	Benutzer
Trigger	Der Benutzer möchte den Chatbot verfeinern und präzisere Antworten liefern
Stakeholders and Interests	**Benutzer**: Chatbot prüfen und passende Antworten zuweisen **System**: Selbstständiger Lernprozess aus den gewählten Antworten
Preconditions	Der Benutzer ist bereits im QnA Maker angemeldet.
Postconditions	Die getätigten Änderungen sind erfolgreich übernommen worden.
Main Success Scenario	Der Benutzer möchte mittels korrekter Antworten den Chatbot gezielt trainieren. Dazu weist er gezielt die passendsten Antworten den Fragen zu.
Extentions (or Alternative Flows)	Bei Mehrfachantworten wählt der Benutzer die passendste Antwort aus oder formuliert eine alternative Antwort.
Frequency of Occurence	Mehrmals im Monat

Tabelle 29: Chatbot trainieren

UC4 Gespräch mit Kunde/Support durchführen

Goal	**Der Benutzer kann ein Gespräch mit dem Chatbot führen**
Level	User Goal
Primary Actor	Kunde
Trigger	Der Kunde hat ein oder mehrere Fragen
Stakeholders and Interests	**Benutzer**: Fragen stellen und beantwortet haben **System**: Retournierung der passendsten Antwort aus der Knowledge Base
Preconditions	Der Kunde hat den Chatdialog offen
Postconditions	Die Fragen wurden beantwortet oder der Kunde wird an einen Supportmitarbeiter weitergeleitet
Main Success Scenario	Der Kunde wird vom Chatbot begrüsst und wird nach dessen Namen gefragt. Danach fragt der Chatbot den Kunden, in welcher Kategorie (Installation, Reklamation, Verfügbarkeit, Preise & Angebote, Anderes) er eine Frage hat. Der Kunde wählt eines der verfügbaren Kategorien aus. Anhand der gewählten Kategorie präsentiert der Chatbot die Fünf meist gefragten Themen. Wählt der Kunde eines der vorgeschlagenen Fragen aus, gibt der Chatbot die zugehörige Antwort zurück. Befindet sich seine Frage nicht unter diesen Fünf, formuliert der Kunde seine Frage schriftlich und teilt dies dem Chatbot mit. Der Chatbot extrahiert die Frage und durchsucht die Knowledge Base nach der passendsten Antwort. Die Antwort mit dem höchsten «Confidence Score [22]» wird retourniert. Falls keine Antwort gefunden werden kann, wird der Kunde darauf aufmerksam gemacht und erhält als Antwort: a) «Bitte vereinfachen oder formulieren Sie die Frage neu» b) «Leider konnte keine passende Antwort gefunden werden. Sie können eine neue Frage formulieren oder einen Supportmitarbeiter kontaktieren»
Extentions (or Alternative Flows)	Es kann jederzeit aus dem aktuellen Dialog ausgestiegen werden und der persönliche Kontakt zu einem Supportmitarbeiter hergestellt werden. Mit einer Auswahlmöglichkeit kann der Kunde zwischen zwei Optionen wählen: a) Click-to-Call: Telefonanruf an den ServiceDesk b) FAQ Webseite der ewz Telecom aufrufen Wählt der Kunde keines der oben genannten Optionen aus, kann er jederzeit weitere Fragen stellen.
Frequency of Occurence	Mehrmals pro Tag

Tabelle 30: Gespräch mit Kunden/Support durchführen

6.1.4 Systemsequenzdiagramm

UC4: Gespräch mit Kunde/Support durchführen

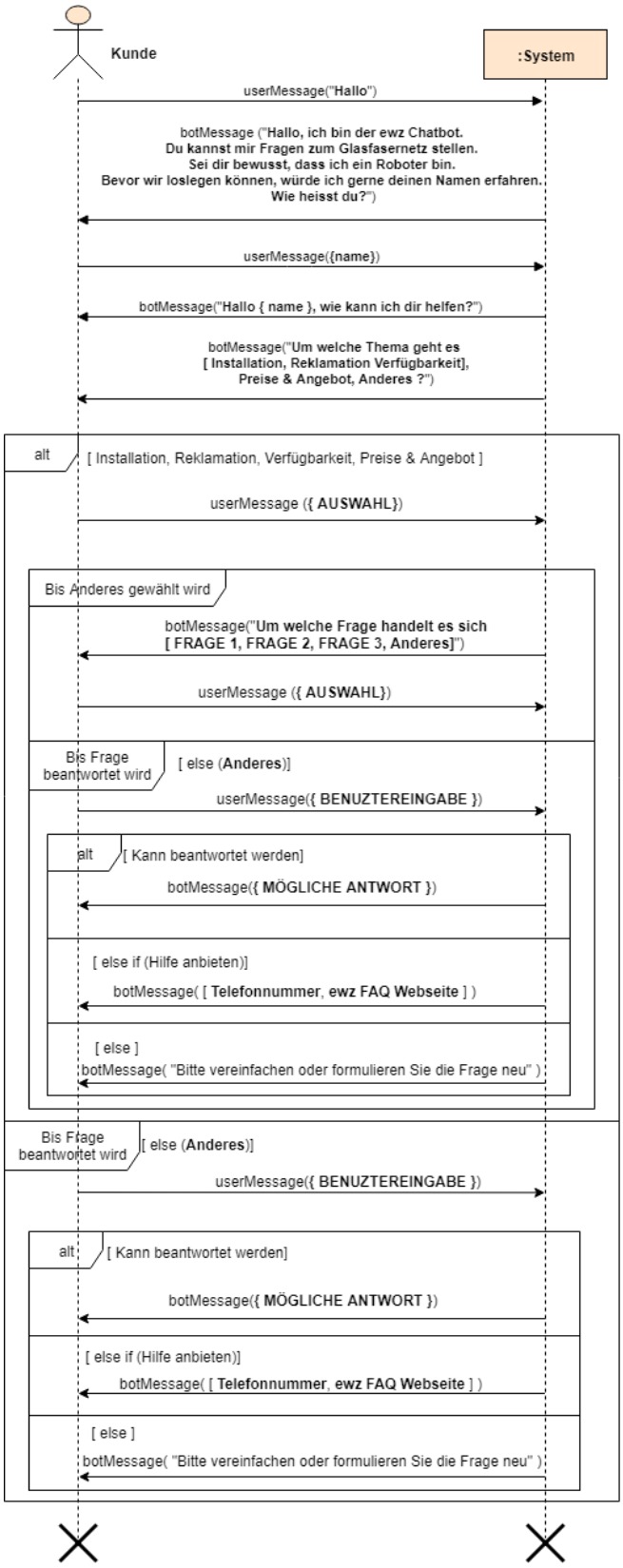


Abbildung 18: Systemsequenzdiagramm UC 4

6.2 Abuse Cases

In diesem Kapitel werden «Abuse Cases» aufgelistet, welches die Chatbot betreffen welche im Azure gehostet wird, wie auch die Daten betreffen können, welche im ewz private Cloud gespeichert sind.

6.2.1 Abuse Case Diagramm

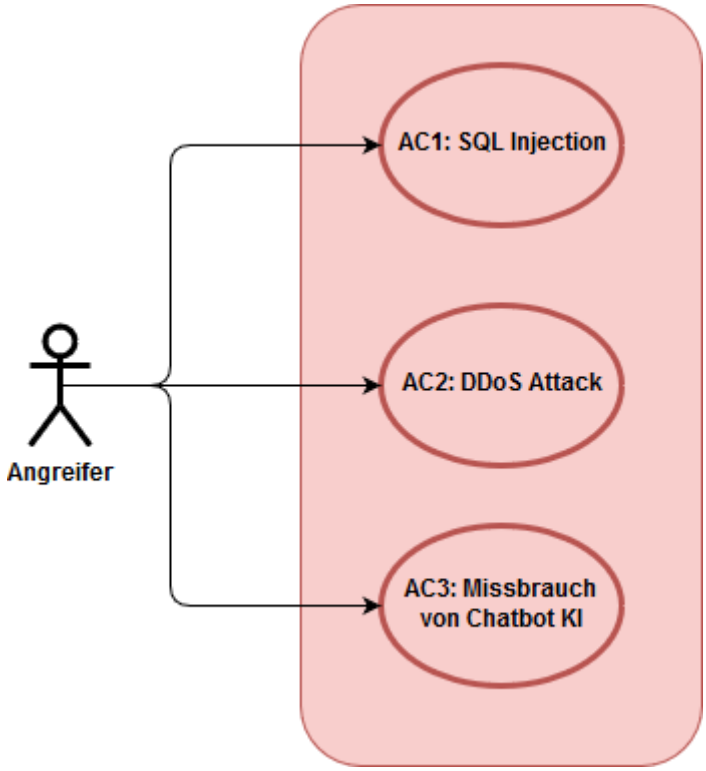


Abbildung 19: Übersicht der Abuse Cases

6.2.2 Abuse Case (Brief)

In den nachfolgenden Kapiteln werden alle Abuse Cases im brief-Format beschrieben.

AC1: SQL Injection

Der Angreifer kann sich als normaler Benutzer mit dem Chatbot eine Kommunikation anfangen. Anstatt eine normale Frage zu geben, kann der Angreifer auch Befehle geben wie zum Beispiel die Tabelle einer Datenbank zu löschen. Falls der Backend Implementation des Chatbot diese Eingabe ohne zu überprüfen an die Datenbank weiterleiten würde, könnte dieser Befehl durch die Datenbank ausgeführt werden, die erheblichen Schaden für das Unternehmen verursachen würde.

AC2: DDoS Attacke

Ein Angreifer, der dem Unternehmen einen erheblichen Schaden zufügen will, der den Chatbot betreibt, kann auf die klassische Methode der DDoS Attacke zurückgreifen. Dies führt dazu, dass Microsoft Azure automatisch mehr Ressource für den Chatbot zu Verfügung stellt, um die Nachfrage zu bestätigen, falls der Entwickler die Autoskalierung Funktion eingeschaltet hat. Zusätzliche Ressource heisst auch zusätzlich Kosten, dadurch hat der Angreifer sein Ziel, dem Unternehmen Schaden zufügen, erreicht. Falls eine Autoskalierungsfunktionalität nicht aktiviert ist kann der Chatbot die Nachfrage nicht mehr bedienen und wird irgendwann auch abstürzen. Dadurch ist der Chatbot für legitime Nutzer nicht mehr erreichbar, was wiederum Kunden Frustration und Umsatzeinbrüche bedeuten kann.

AC3: Missbrauch der Chatbot KI

Das Chatbot Framework von QnA Maker lernt aktiv aus den Benutzerinteraktionen. Dies ist eines der Stärken von Chatbots, da sie so kontinuierlich lernen und menschliches Verhalten lernen. Zugleich ist dies auch eine der Schwachstellen von Chatbots.
Bestes Beispiel ist der «Tay-Bot» von Microsoft, welcher 2016 für Gesprächsstoff sorgte. Ziel war es, dass der Chatbot gezielt von der menschlichen Kommunikation lernt und wie Künstliche Intelligenz im Alltag eingesetzt werden kann. Über Twitter gestattete der Tay Bot Konversationen mit den Benutzern. Leider gab es böswillige Benutzer, welcher den Tay Bot radikalisierten und absichtlich missbräuchliche Fragen und Antworten stellten. Nach nur 16 Stunden musste der Bot heruntergefahren werden, da er selbstständig unangemessene Nachrichten und Konversationen startete.

Beim QnA Maker Framework hat man gezielt eine Fraud Detection[37] implementiert, um dies zu verhindern. Das bedeutet, dass der Chatbot nur aktiv lernt, wenn mehr als 50 Benutzer die gleiche Antwort als richtig klassifizieren. So wird die Wahrscheinlichkeit drastisch reduziert, dass ein Benutzer absichtlich den Bot falsches Verhalten lehrt. Active Learning [47] nennt sich dieses Feature von QnA Maker.

6.3 Domain-Model

Die folgende Abbildung zeigt das Domain-Model, das serverseitig zum Einsatz kommt.

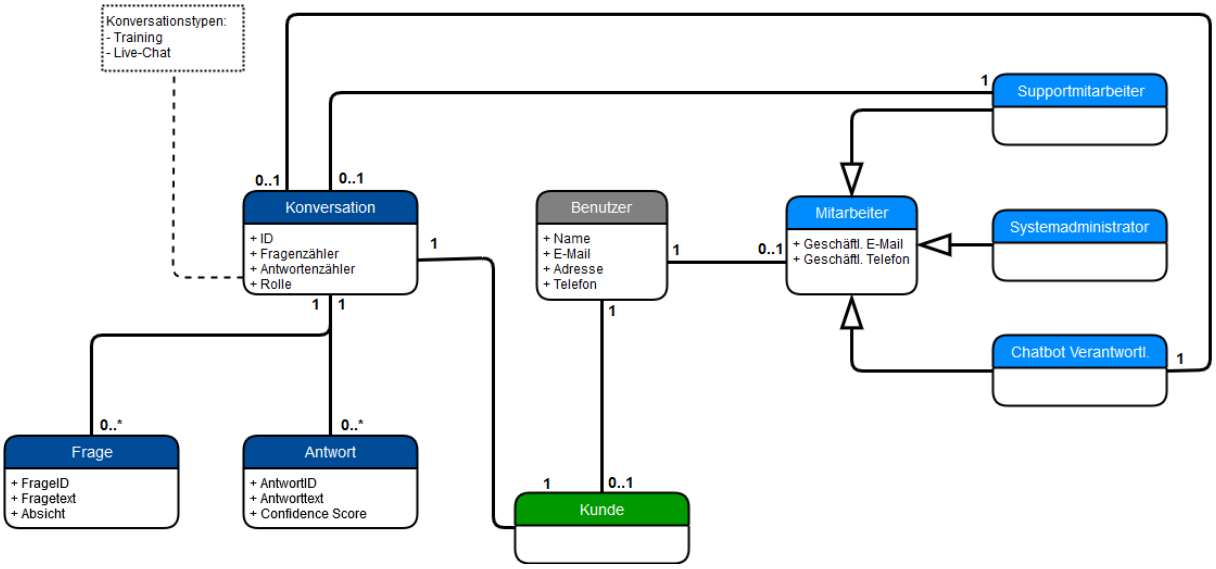


Abbildung 20: Domain-Model

- **Mitarbeiter** der ewz. Wird unterteilt in verschiedene Rollen wie Systemadministrator, Supportmitarbeiter oder Chatbot Verantwortlicher.
- **Kunde** ist die Person, welches ein Anliegen hat und aktiv mit dem Chatbot kommuniziert.
- **Konversation** besteht aus Fragen, die der Kunde stellt und den Antworten des Chatbots. Jede Frage und Antwort hat eine eindeutige ID, welche im Log-file nachverfolgt werden kann. Besonders schützenswerte Daten wie Personeninformationen werden anonymisiert.

[37] Vorbeugung und Entdeckung von Betrug, Missbrauch

7. Umsetzung

Der finale Prototyp des Chatbots wird in unterschiedliche Stufen unterteilt, wobei bei jeder Stufe die Funktionalität des Chatbots erweitert wird. Zusätzlich wird darauf Wert gelegt, dass der Chatbot bei jeder Stufe mehr an Intelligenz gewinnt. Damit haben wir als Entwickler immer eine Zwischenstufe, auf die wir zurückgreifen können, falls etwas bei der nächsten Stufe schiefläuft. Die Stufeneinteilung kann mit der unterschiedlichen Stufeneinteilung der autonomen Fahrzeuge verglichen werden, wobei bei jeder Stufe die Funktionalität des autonomen Fahrens erweitert wird.
In den folgenden Kapiteln wird jeweils von «Demonstrator» gesprochen. Ein Demonstrator verfügt über fix definierte Funktionen und baut auf dem Vorherigen auf. In der Bachelorarbeit werden bis zum finalen Prototyp vier Demonstratoren entwickelt. Die Software Architektur wird bewusst so konzipiert, dass keine grundlegenden Veränderungen vorgenommen werden müssen zwischen den einzelnen Demonstratoren. So können wir sicherstellen, dass beim Demonstrator 3 kein komplett neuer Prototyp erstellt werden muss.
Auf Design Aspekte verzichten wir grösstenteils. Einerseits geht es bei der Bachelorarbeit um eine Machbarkeitsstudie. Die Kernfrage, ob ein Chatbot bei der ewz Telecom eingesetzt werden kann, hängt hauptsächlich daran wie «intelligent» und clever der Chatbot mit dem Kunden interagiert.

7.1 Demonstrator 1

In der ersten Version werden lediglich Basisfunktionen implementiert. Es soll möglich sein, einfache Fragen zu stellen, ohne die Frage im Kontext zu ändern. Hierfür wird auf eine einzige QnA Maker Knowledge Base zugegriffen. Der Chatbot versucht die gestellte Frage in der Knowledge Base zu finden. Geringe Abweichungen in der Frage sind akzeptabel.
Diese Version wurde bewusst rudimentär gestaltet, um ein Gefühl zu bekommen wie das Zusammenspiel zwischen dem Bot Framework und dem QnA Maker Service ist. Zusätzlich soll eine generische Nachricht den Benutzer willkommen heissen.

7.1.1 Funktionalitäten

- Begrüssungsnachricht
- Alle FAQ von der ewz Homepage mit QnA Maker abbilden. Eine einzige Knowledge Base kommt hierfür zum Einsatz.
- Bei einer Eskalation (wenn der Benutzer nicht weiterkommt oder seine Frage zu spezifisch ist) wir dem Benutzer angeboten, den Support zu kontaktieren. Hierfür wird dem Benutzer die Supporttelefonnummer angezeigt.

7.1.2 Logische Systemarchitektur

Die Systemübersicht gibt einen guten Überblick über das System und ihre Komponenten und wie sie aufgebaut sind. Nachfolgend die Systemübersicht mit den einzelnen Komponenten sowie eine detaillierte Beschreibung.

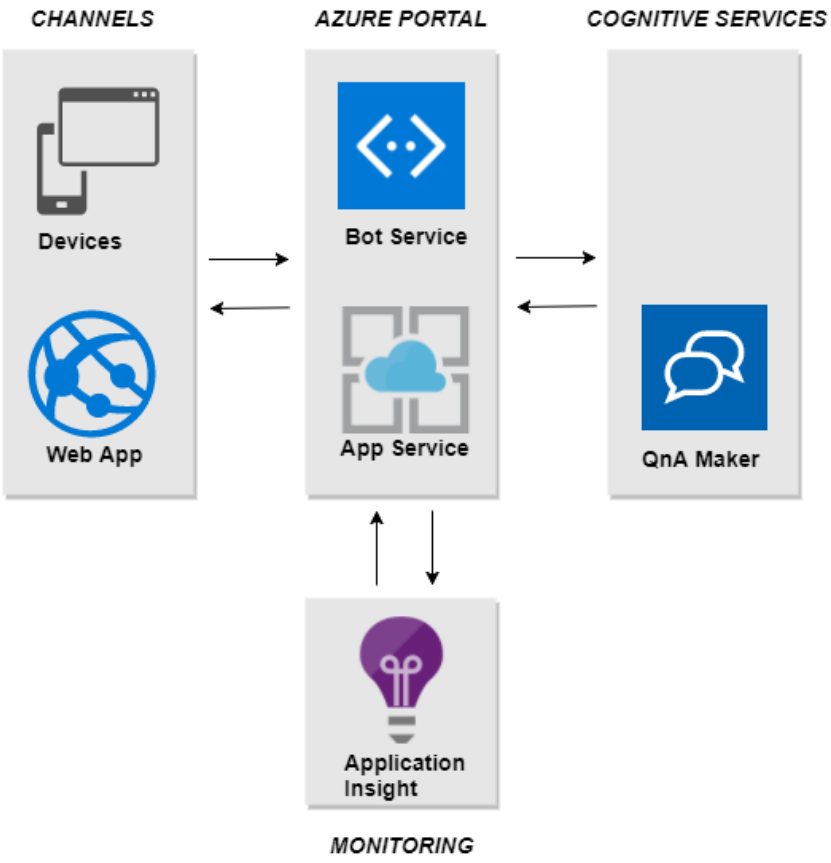


Abbildung 21: Systemarchitektur Demonstrator 1

7.1.3 Komponenten

- **Devices**, auf welchen es möglich ist, den Chatbot zu bedienen. Dies sind üblicherweise Smartphones oder Laptops mit Webbrowser.
- **Web App** ermöglicht das Erstellen von Webanwendungen in der gewählten Programmiersprache, ohne dass eine Infrastruktur verwaltet werden muss.
- **Bot Service** stellt die Kernfunktionalitäten des Chatbots zur Verfügung. Zudem ist es auch die Schnittstelle zu den Cognitive Services wie QnA Maker oder LUIS. Die Entwickler kümmern sich hauptsächlich um die Chatbot Logik.
- **App Service** übernimmt das Hosting der Chatbot Applikation. HTTP Anfragen und Skalierung der Chatbot Applikation können hier vorgenommen werden.
- **QnA Maker** bietet mit der Knowledge Base das Rückgrat des Chatbots. Hier werden die Fragen mit den dazugehörigen Antworten gespeichert und gepflegt.
- **Application Insight** ist ein Azure Dienst, welcher Live-Monitoring und Statistiken über den Chatbot bietet. Administratoren sehen hier ein, wie ausgelastet der Chatbot ist und ob Fehler aufgetreten sind.

7.1.4 Beschreibung des Datenflusses

Über den gewählten Channel ruft der Kunde die Chatbot Applikation auf. Nach der Begrüssung findet die eigentliche Konversation statt. Der Kunde sendet eine Nachricht über die Web App. Die Nachricht wird über eine HTTPS gesicherte REST-Schnittstelle an den App Service übertragen. Der App Service sendet sie an den QnA Maker Service, welcher versucht, die passende Antwort zurückzugeben. Bei einem Erfolg sendet der QnA Maker Service die Antwort zurück und wartet auf weitere Anfragen. Bei einem Misserfolg sendet der QnA Maker eine standardisierte Antwort «Tut mir leid, das habe ich leider nicht verstanden» zurück. Dem Kunden wird die Antwort präsentiert und er kann den Dialog

weiterführen. Im «Demonstrator 1» werden bewusst keine Konversationen in einer Datenbank gespeichert. Lediglich Meta Daten[38] werden dem Application Insight Service zur Verfügung gestellt.

7.1.5 Azure bot Service

Azure bot Service bietet mehr Möglichkeiten, um einen Chatbot aufzubauen, testen, deployen und an einem Ort zu verwalten. Darüber hinaus verwendet es das SDK, Templates und andere KI Services, welche die Entwicklung von Chatbots vereinfachen. Der grösste Vorteil von Azure bot Service ist, dass es relativ einfach ist Produkte wie Cortana, Skype, Facebook Messenger usw. verbinden zu können. Die Backend welches die «Business Logic» beinhaltet, kann für alle diese Frontend Produkte ein und dieselbe sein. Während der Durchführung der Bachelorarbeit waren zwei SDKs in Umlauf. Um unser Wissen in der Chatbot Umgebung zu vertiefen sowie die Neuerungen der Chatbot SDK zu verstehen, haben wir uns entschieden für den Demonstrator 1 in beiden SDK den gleichen Chatbot zu entwickeln. Im nachfolgenden Kapitel werden die Unterschiede zwischen den beiden Chatbots aufgeführt sowie die unterschiedliche Software Architektur der jeweiligen Implementation.

7.1.5.1 SDK v3

Azure bot Service war von Anfang an nur für die Sprache «C#» verfügbar, diese wurde später durch «node.js» erweitert. Es ist an dieser Stelle zu beachten, dass die SDK von «C#» viel mehr Funktionalität für die Entwickler bietet als «node.js». Konkret heisst das, es gibt viele Hilfemethoden in «C#» welche im «node.js» nicht vorhanden sind und die Entwickler müssen diese selbst programmieren. Im SDK v3 fallen ein paar Sachen wie Internationalisierung des Chatbots welche erst im SDK v4 zu Verfügung gestellt werden [48]. Bot Connector erlaubt es den Entwicklern den Chatbot über verschiedene Kanäle den Chatbot für die Endkunden zu Verfügung zur stellen. Dies ist auch ein Feature, das ebenfalls in der SDK v4 zur Verfügung steht.

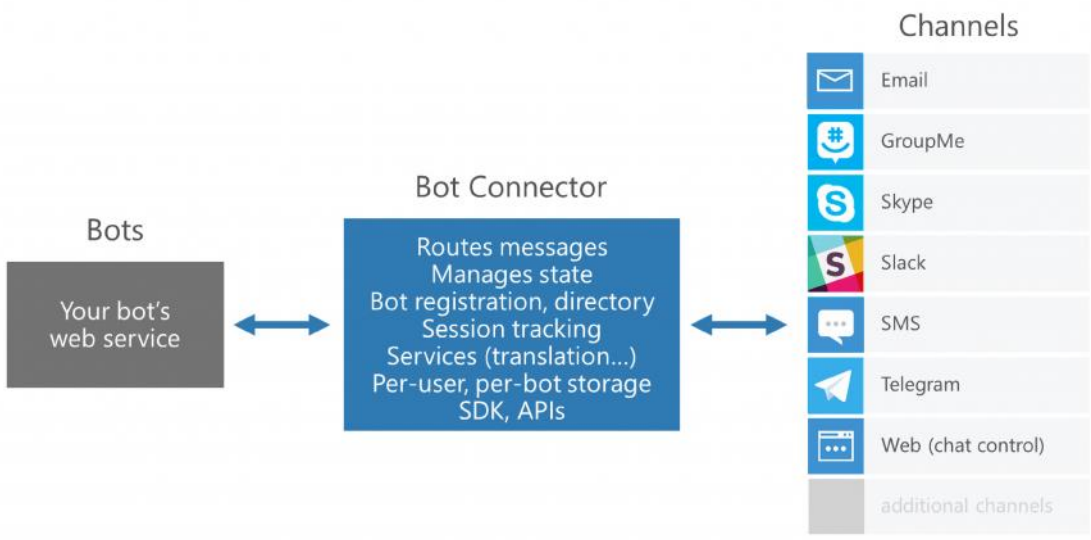


Abbildung 22: Microsoft Bot Connector [49]

Produkt	Beschreibung
Bots	Hier befindet sich die Bot Applikation also die Business Logik der Applikation. Dies ist von Chatbot zu Chatbot unterschiedlich, welche vom Entwickler für die jeweiligen Unternehmen programmiert werden.
Bot Connector	Bot Connector ist die Komponente, welche die Bots und «Channels» zusammenbringt. Bot Connector beinhaltet viele Services und Infrastruktur wie

[38] Informationen über eine Ressource. Beschreiben die Eigenschaften eines Objekts.

	State Manager, Session Trackings usw. welches das Zusammenspiel von Bots und «Channels» vereinfacht.
Channels	Die Kunden können selbst entscheiden über welchen Kanal sie die Chatbot Applikation ansprechen. Selbstverständlich müssen die Entwickler diese auch erlauben, ansonst ist es nicht möglich. Der Vorteil liegt darin, dass die Kunden nicht für jeden einzelnen Chatbot eine «app.» wie heutzutage üblich ist, auf das Smartphone herunterladen müssen. Der Entwickler muss sich keine Gedanken über das Frontend machen, da es vom Kanal abhängt.

Tabelle 31: Microsoft Bot Connector Diagramm Beschreibung

7.1.5.2 Software Architektur (Klassenstruktur)

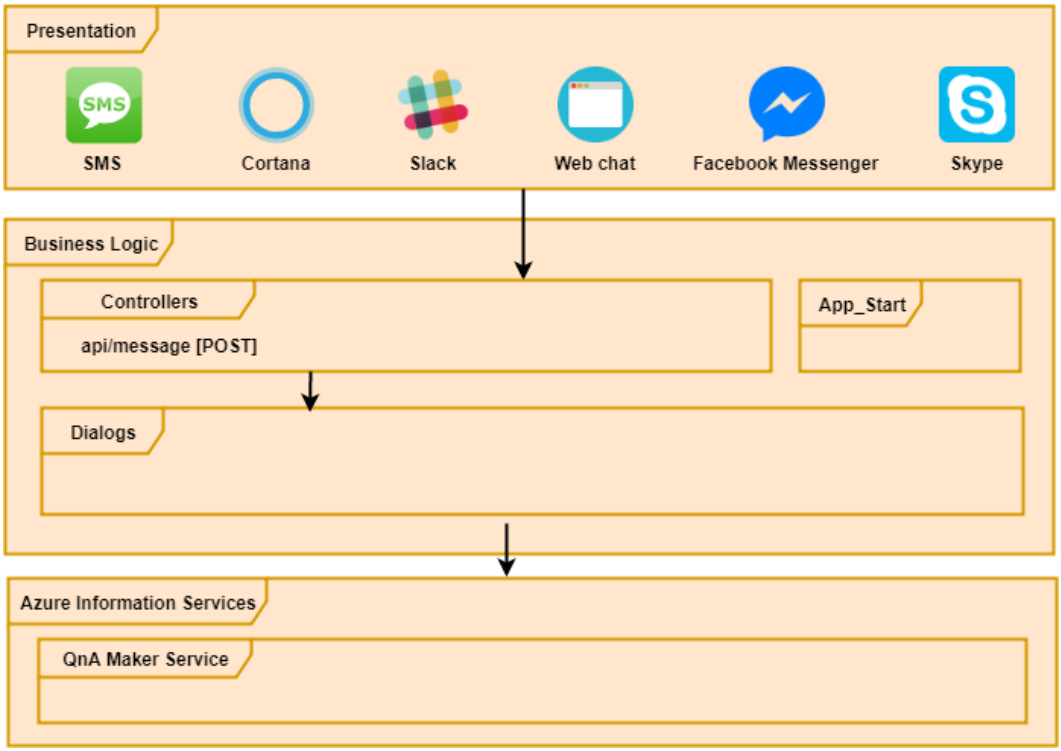


Abbildung 23: Software Architektur Demonstrator 1 SDK v3

Klasse/Ordner		Beschreibung
Presentation		Im Presentation Layer sind alle möglichen Kanäle verfügbar. Die Software Entwickler müssen sich nur noch Gedanken machen auf welchem Kanal sie den Chatbot publizieren wollen. Die Grundimplementation ist überall gleich.
Business Logic	Controllers	In diesem Package werden alle Routen definiert, welche den Chatbot unterstützen. In unserem Fall gibt es nur eine Route und eine Methode und zwar die «POST» Methode für die Route «api/messages». Da wir ja nur die Nachrichten vom Benutzer über den jeweiligen Kanal entgegennehmen. Selbstverständlich können die Entwickler weitere Routen je nach Bedarf definieren.
	App_Start	In diesem Package befindet sich die Web API Konfiguration und Services. Ebenfalls werden die Web API Routen hier registriert. Dieses

		Package beinhaltet weitere wichtige Konfigurationen für den Chatbot sowie die Zugangsdaten für QnA Maker und Azure Chatbot.
	Dialogs	In diesem Package wird die Business Logic der Chatbot Applikation definiert. Aus diesem Package finden auch die Abfragen auf den QnA Maker Service statt.
Azure Information Services	QnA Maker Service	QnA Maker Service, welcher durch das QnA Maker Portal vom Entwickler erstellt werden kann. Diese kann dann als eine REST API veröffentlicht und angesprochen werden.

7.1.5.3 SDK v4 [50]

Azure Bot Services SDK v4 ist eine neue Version und bietet zusätzliche Features an. Das Microsoft Entwicklungsteam versucht die Unterschiede zwischen den verschiedenen SDK (Sprachen) zu verringern. Zum Beispiel gibt es im «C#» ein Feature namens «FormFlow» welches aber im «Node.js» nicht vorhanden ist. Dies war der Fall beim SDK v3. Dies ist ein Nachteil für die Entwickler falls er über mehrere Sprachen hinweg einen Chatbot entwickelt. Nicht nur «C#» und «Node.js» SDKs werden entwickelt, sondern in der Zukunft wird auch die SDK in Sprachen «Python» und «Java» verfügbar sein. Dadurch können mehr Entwickler die neue Plattform für sich entdecken. In der SDK v4 kommt ein neues Konzept namens «Middleware», welches den Entwicklern erlaubt, Plugins für einen Chatbot zu entwickeln und diese dann später auch bei einem anderen Chatbot wiedereinzusetzen. Dadurch wird die Wiederverwendbarkeit von Codes erhöht was eines der wichtigsten Software Engineering Prinzipien ist. Sämtlicher Verkehr, der zu einem Chatbot fliesst, wird zuerst durch diese Middleware Komponente geleitet. Der Entwickler kann den Verkehr vorher abfangen und bestimmte Sachen wie Authentifizierung, Autorisierung, Logging, Fehlerbehandlung oder LUIS Abfragen machen bevor der Verkehr zum Chatbot weitergeleitet wird. Einerseits kann ein Teil der Businesslogic vom Bot auf die Middleware übertragen werden, was den Chatbot vereinfacht da kein duplizierter Code entsteht. Es gibt bereits Middlewares für die Anbindung von LUIS und QnA Maker Cognitive Services, welches bis SDK v3 im Chatbot selbst abgehandelt wurde. In der nachfolgenden Abbildung ist das Zusammenspiel von Middleware und Chatbot dargestellt.

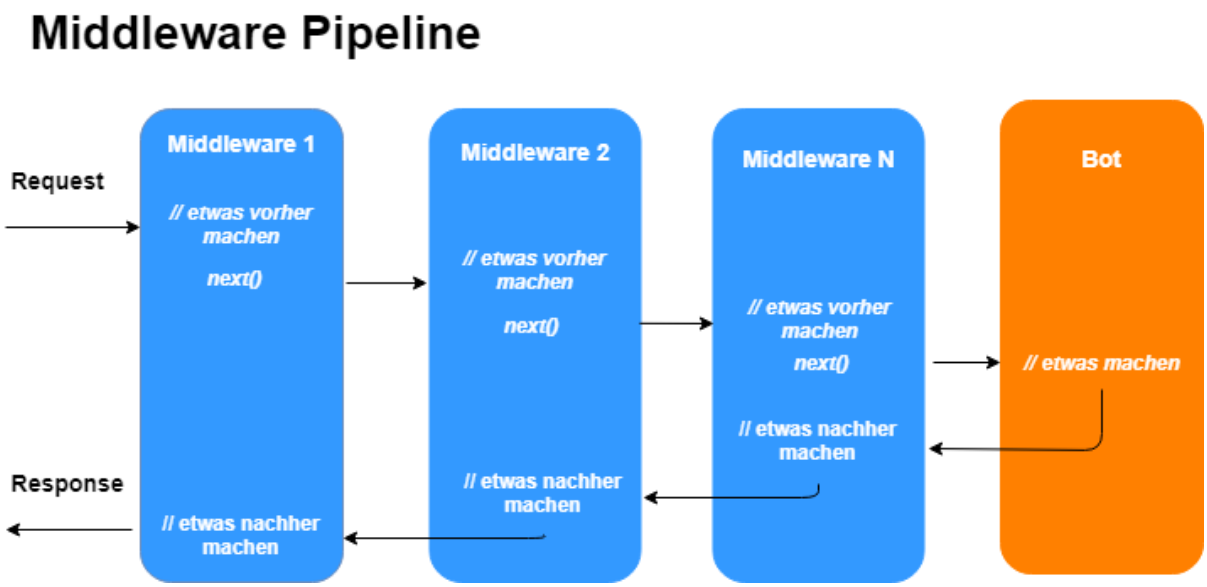


Abbildung 24: Microsoft Bot Services Middleware Pipeline [51]

7.1.5.4 Software Architektur (Klassenstruktur)

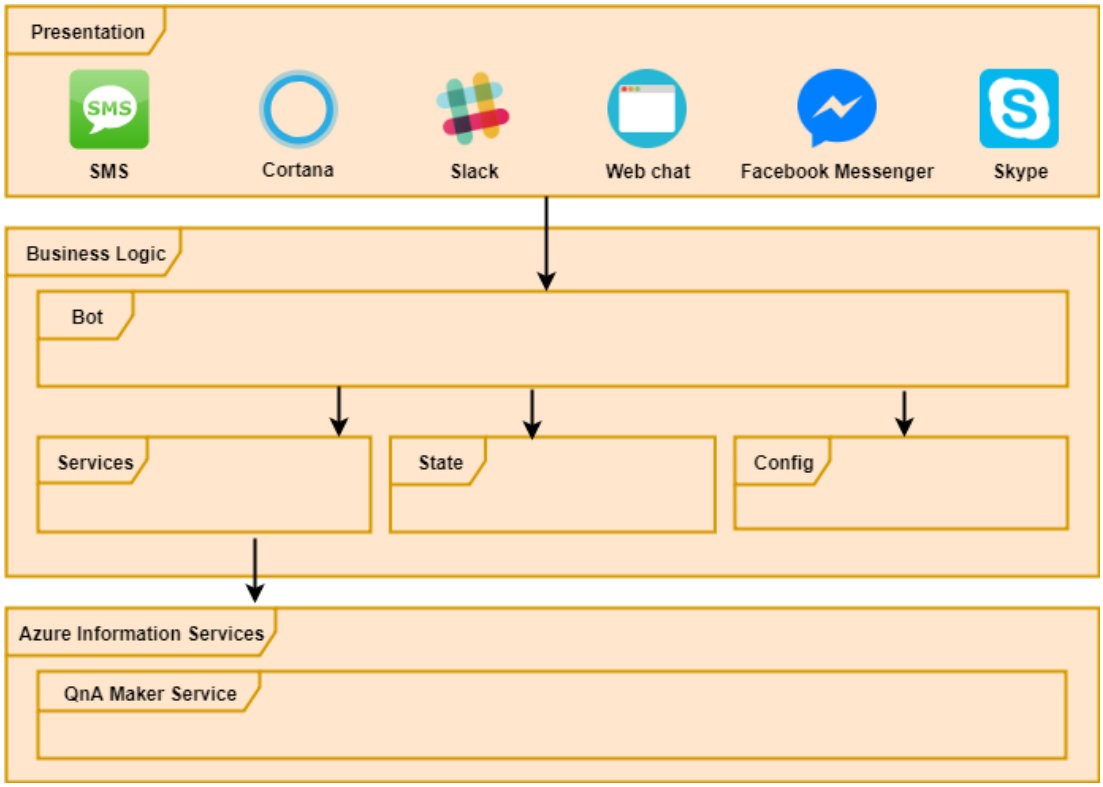


Abbildung 25: Software Architektur Demonstrator 1 SDK v4

Klasse/Ordner		Beschreibung
Presentation		Im Presentation Layer sind alle möglichen Kanäle verfügbar. Die Software Entwickler müssen sich nur noch Gedanken machen auf welchem Kanal sie den Chatbot publizieren wollen. Die Grundimplementation ist überall gleich.
Business Logic	Bot	Dieses Package beinhaltet die Business Logik von Chatbot. Darüber hinaus werden hier ebenfalls die Middleware sowie andere Services registriert die bei Laufzeit über «Dependency Injection» zum jeweiligen Package hinzugefügt wird.
	Services	In diesem Package wird die Abfrage für die QnA Maker Services vom Bot abstrahiert. Dieser Service ist über «Dependency Injection» für das Bot Package erreichbar. Selbstverständlich können die Entwickler hier weitere Services definieren welche über «Dependency Injection» für die jeweiligen Packages zur Verfügung gestellt werden.
	State	State Package beinhaltet die State der Chatbot sowie «State Accessor» welches den Zugriff auf den Zustand regelt. Der «State Accessor» wird ebenfalls über die «Dependeny Injection» für die jeweiligen Package zu Verfügung gestellt. In diesem Fall das Bot Package.
	Config	Der «Config» Package beinhaltet wichtige Informationen für den Zugriff auf die QnA Maker Services. Ebenfalls werden hier auch Informationen für das deployment über Powershell definiert. Dadurch kann der Bot direkt vom lokalen Rechner auf das Azure Portal deployt werden ohne es vorher in einem «git repository» einzuchecken.

Azure Information Services	QnA Maker Service	QnA Maker Service, der durch das QnA Maker Portal vom Entwickler erstellt werden kann. Diese kann dann als eine REST API veröffentlicht und angesprochen werden.

7.2 Demonstrator 2

Die zweite Version erweitert den Chatbot um Funktionen, welche die Kundenzufriedenheit und Personalisierung erhöhen. Es wird aktiv nach dem Namen gefragt und während der Session verwendet. Das Gespräch fühlt sich natürlicher an, da die Konversation in drei Phasen aufgeteilt ist.

- **Begrüssungsphase:** Der Chatbot stellt sich vor und erklärt welche Funktion er einnimmt. Ein kurzer Small-Talk Dialog wird geführt und anschliessend nach dem Namen des Kunden gefragt.
- **Frage & Antwortphase**: Der Hauptbestandteil besteht aus der Beantwortung der Kundenanfragen. Der Kunde kann beliebig viele Fragen stellen, während im Hintergrund der Chatbot jeweils die Knowledge Base nach den geeigneten Antworten durchsucht.
- **Eskalationsphase**: Der Kunde kann jederzeit seinen Wunsch äussern mit einem Service Mitarbeiter zu sprechen. Der Chatbot bietet an, eine Telefonverbindung zum Customer Support aufzubauen. Andererseits wird dem Kunden die Möglichkeit geboten, die offizielle FAQ-Webseite zu besuchen.
- **Abschlussphase**: Diese Phase wird eingeleitet, sobald die Absicht des Kunden besteht, das Gespräch zu beenden. Der Chatbot bedankt und verabschiedet sich vom Kunden.

7.2.1 Funktionalitäten

- Persönliche Begrüssungsnachricht. Der Chatbot fragt aktiv nach dem Namen des Benutzers und grüsst ihn/sie.
- FAQ wird in mehrere Knowledge Bases aufgeteilt. Die Wartbarkeit wird einerseits erhöht und zudem ist es für das «mapping» zwischen QnA und LUIS zwingend notwendig.
- LUIS kommt zum Einsatz in Verbindung mit mehreren QnA Knowledge Bases. Fragen können nun auf unterschiedliche Weise gestellt werden. Synonyme zu Wörtern werden teilweise unterstützt.
- Der Benutzer kann nach einem Service Mitarbeiter fragen. In einem Zwischenschritt wird auf das FAQ der Webseite verwiesen. Zusätzlich hat er die Möglichkeit direkt einen Telefonanruf zu starten, um den Service Desk zu kontaktieren.

7.2.2 Logische Systemarchitektur

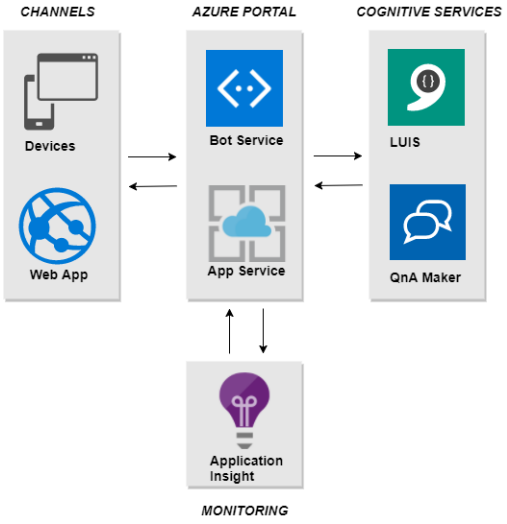


Abbildung 26: Systemarchitektur Demonstrator 2

7.2.3 Komponenten

Aus Wiederholungsgründen verzichten wir auf die Komponenten, die bereits im Kapitel 7.1.3 erwähnt wurden. Zusätzlich zum Demonstrator 2 wird das LUIS hinzugefügt.

- **LUIS** kümmert sich um die Intelligenz des Chatbots und filtert aus den Nachrichten die Absichten des Kunden. Die Frage des Kunden wird nach dem «Intent» gefiltert, damit in der korrekten QnA Knowledge Base gesucht werden kann.

7.2.4 Beschreibung des Datenflusses

In dieser Version wird zusätzlich eine Verbindung zum LUIS Service aufgebaut. Der Unterschied besteht darin, dass die Nachricht des Kunden ungefiltert an den LUIS Service gesendet wird. LUIS versucht nun die Absicht der Nachricht zu entschlüsseln, damit die korrekte QnA Maker Knowledge Base angesprochen werden kann. Es werden mehrere QnA Maker Knowledge Bases (z.B. Verfügbarkeitsfragen) erstellt, damit die Wartbarkeit verbessert wird.

7.2.4.1 Software Architektur (Klassenstruktur)

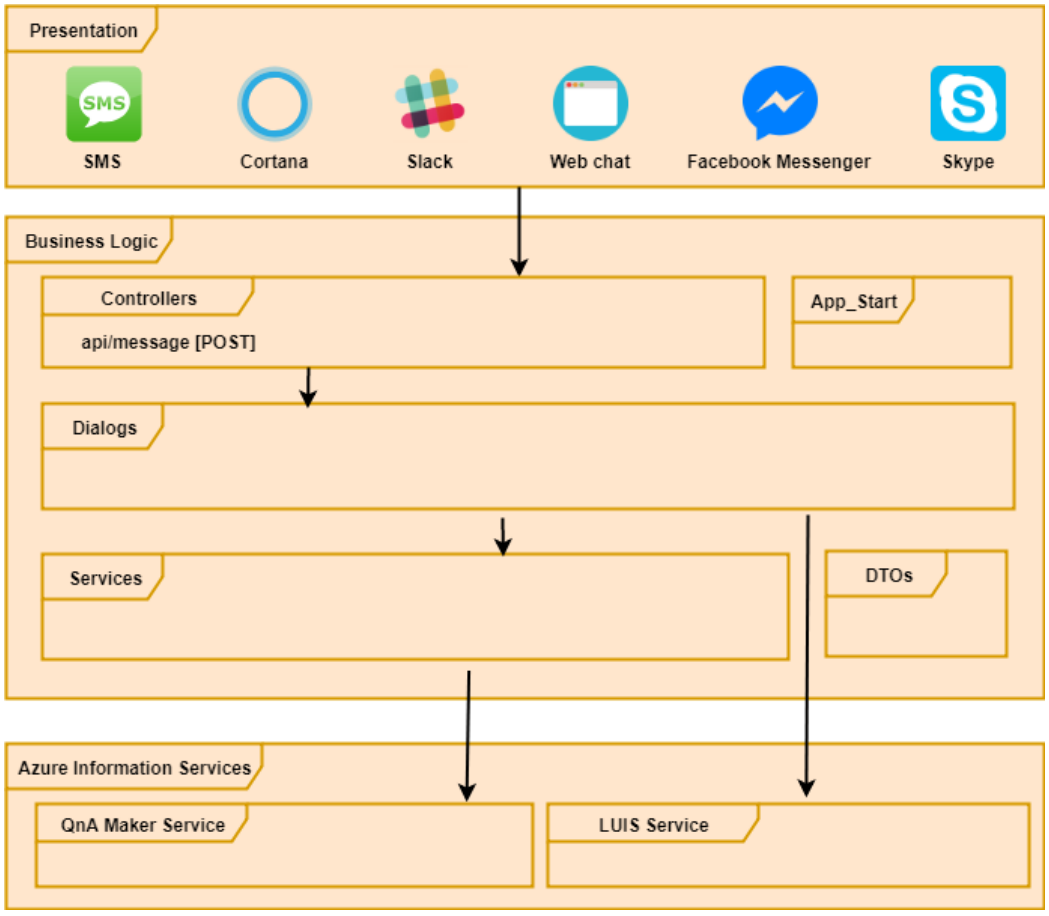


Abbildung 27: Software Architektur Demonstrator 2

Klasse/Ordner	Beschreibung
Presentation	Im Presentation Layer sind alle möglichen Kanäle verfügbar. Die Software Entwickler müssen sich nur noch Gedanken machen auf welchem Kanal sie den Chatbot publizieren wollen. Die Grundimplementation ist überall gleich.

Business Logic	Controllers	In diesem Package werden alle Routen definiert, welche den Chatbot unterstützen. In unserem Fall gibt esnur eine Route und eine Methode und zwar die «POST» Methode für die Route «api/messages». Da wir ja nur die Nachrichten vom Benutzer über den jeweiligen Kanal entgegennehmen. Selbstverständlich können die Entwickler weitere Routen je nach Bedarf definieren.
	App_Start	In diesem Package befindet sich die Web API Konfiguration und Services. Ebenfalls werden die Web API Routen hier registriert. Dieses Package beinhaltet weitere wichtige Konfigurationen für den Chatbot sowie die Zugangsdaten für QnA Maker und Azure Chatbot.
	Dialogs	Dialogs sind einzelne Komponenten, welche in sich isoliert sind. Um den Konversationsfluss steuern zu können, werden einzelne Dialogs nacheinander aufgerufen.
	Services	Stellt die REST API Anfrage auf den QnA Maker Service. Serialisiert und Deserialisiert die Antwort in das JSON Format für die Weiterverarbeitung.
	DTOs	Einzelne DTO[39]
Azure Information Services	QnA Maker Service	QnA Maker Service, welcher durch das QnA Maker Portal vom Entwickler erstellt werden kann. Diese kann dann als eine REST API veröffentlicht und angesprochen werden.
	LUIS Service	LUIS Service, der durch das LUIS Portal vom Entwickler erstellt werden kann. Diese kann dann als eine REST API veröffentlicht und angesprochen werden.

7.3 Zwischenergebnisse

Da wir uns nun in der Halbzeit der Bachelorarbeit befinden, ziehen wir ein Fazit aus den gewonnenen Erkenntnissen. Microsoft ist optimal vorbereitet für die Entwicklung professioneller Chatbots. Die ganze Entwicklungsumgebung ist bestens integriert mit den jeweiligen kognitiven Services. Zu diesem Zeitpunkt sprechen wir eine klare Empfehlung aus für Microsoft Azure und dessen Services.
Der QnA Maker Service sowie LUIS sind zwar mächtige Tools, aber ohne entsprechendes Training leider nutzlos. Während der Bachelorarbeit haben wir die Lehre gezogen, dass der Benutzer oftmals vollkommen andere Fragen stellt als wir es uns vorgestellt haben. Als Konsequenz mussten wir mehr Zeit investieren in die Knowledge Base vom QnA Maker sowie das gezielte Trainieren von LUIS.
Wir haben diese Schwachstelle bemerkt, als wir uns entschieden haben, dass der Chatbot erst eine Antwort zurückgibt, wenn er sich zu 70% sicher ist. Dies hatte zur Folge, dass schon bei einer kleinen Abweichung einer Frage (z.B. anstatt Glasfaseranschluss wurde Glasfaserkabel geschrieben) der Confidencescore[40] rapide sinkt. So konnte der Chatbot praktisch keine Antwort aus der QnA Knowledge Base rausziehen.
Auf der technischen Seite hatten wir, dank der sauberen Dokumentation von Microsoft, kaum Probleme bei der Implementation. Einzig die Frage auf welche Bot Framework Version wir setzen wollen, gab Diskussionsstoff. Eine Entscheidung musste gefällt werden zwischen dem bereits etablierten, aber «veralteten» SDK V3 oder dem bald erscheinenden SDK V4. Da die Zukunft von SDK V4 noch ungewiss ist, haben wir uns für die sichere Variante entschieden, SDK V3. Einer der grossen Vorteile ist die Sicherheit, dass es genügend Beispiele gibt für Chatbotfunktionen.

[39] Data Transfer Object
[40] Numerischer Wert, wie sicher der Chatbot ist die korrekte Antwort gefunden zu haben

Wir sind zufrieden und zuversichtlich für die zweite Halbzeit der Bachelorarbeit. Aufgrund der gewonnenen Erkenntnisse können wir die grössten Risiken und Chancen bereits aufdecken und in die Arbeit einfliessen lassen.

7.3.1 Feature Priorisierung

Nach der Zwischenpräsentation hatten wir die Möglichkeit eine Diskussionsrunde zu führen mit den Stakeholdern der ewz. Es war uns ein Anliegen die zukünftigen Features priorisieren zu lassen. Aufgrund der Gewichtung der einzelnen Features, werden wir uns für den zweiten Teil der Bachelorarbeit auf die «Must have» Features konzentrieren. Aus der Diskussionsrunde ging folgendes Ergebnis hervor:

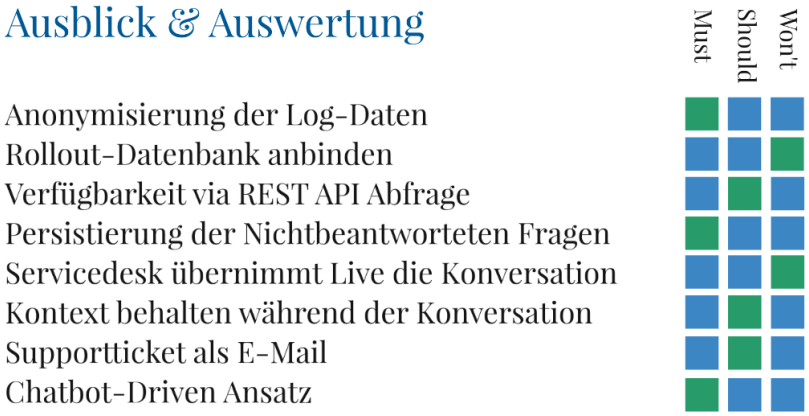


Abbildung 28: Priorisierung nach MoSCoW[41] Methode

7.3.2 GUI-Entwurf für Demonstrator 3

Als weiteres Feature wir der «Chatbot Driven» Ansatz umgesetzt. Hierfür wird eine Vorselektionierung vorgenommen. Der Benutzer wird nach der Begrüssung gefragt, in welcher Kategorie sein Anliegen passt. Der folgende Entwurf zeigt eine mögliche Umsetzung dieses Features.

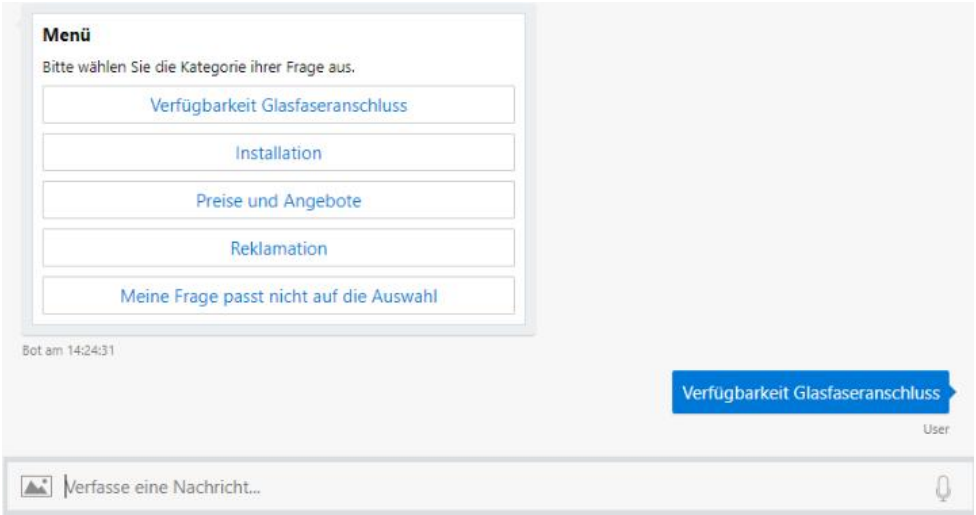


Abbildung 29: Chatbot Driven Entwurf

[41] Priorisierungsmethode die beim Projektmanagement häufig anzutreffen ist.

7.4 Demonstrator 3

7.4.1 Funktionalitäten

- Anonymisieren der Log-Dateien. E-Mailadressen, Telefonnummern sowie Namen werden aus Datenschutzgründen unkenntlich gemacht.
- Persistieren der Konversation zwischen den Benutzer und dem Chatbot auf einer Datenbank.
- Nichtbeantwortete Fragen werden separat auf einer Datenbank gespeichert werden.
- Chatbot-Driven Ansatz. Benutzer werden «geführt» durch die Konversation.
- {*Optional*} Bei einer Verfügbarkeitsabfrage wird direkt ein REST Aufruf gestartet. Die JSON Antwort wird in ein menschlich lesbares Format umgewandelt und präsentiert.
- {*Optional*} Der Chatbot kann sich an gewisse «Nachrichten» erinnern. Aus diesem Kontext gibt er angepasste Antworten dem Benutzer.
- {*Optional*} Ein Supportticket kann ausgelöst werden an den Servicedesk, wenn der Benutzer dies wünscht. Der Servicedesk erhält ein E-Mail mit der bereits geführten Konversation.
- {*Optional*} Der Eingabetext des Benutzers wird auf Rechtschreibefehler geprüft. Die Bing Spell Check API prüft die Eingabe bevor es an LUIS weitergereicht wird.

7.4.2 Logische Systemarchitektur

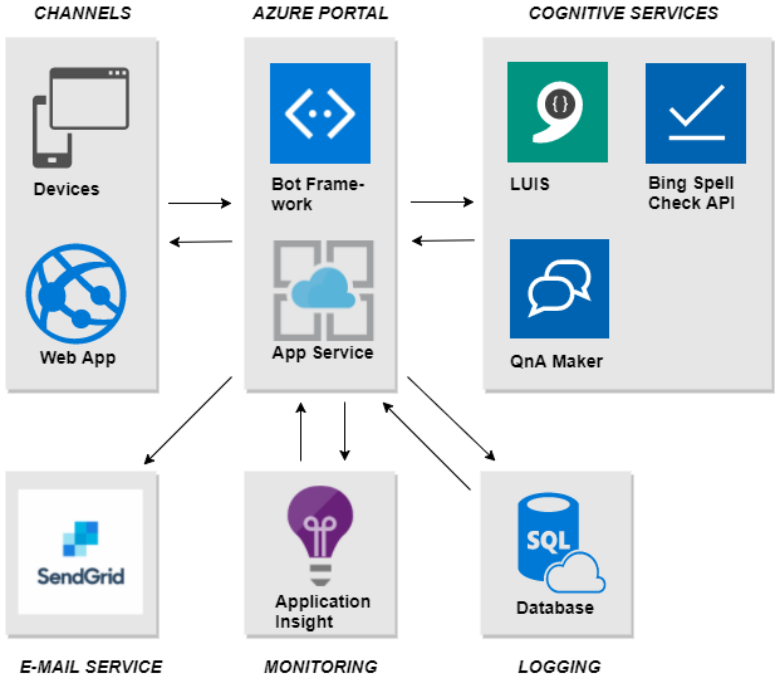


Abbildung 30: Systemarchitektur Demonstrator 3

7.4.3 Komponenten

- **Database** persistiert die Konversationen zwischen dem Benutzer und des Chatbots. Hier können die einzelnen Sessions abgespeichert und auch ausgewertet werden. Als Datenbank wird eine Azure Table Storage gewählt, da dies vollkommen ausreicht für unsere Anforderungen.
- **E-Mail Service** übernimmt die Rolle eines Mail Delivery Services. Sendgrid [52] sendet bei einer Supportanfrage des Kunden den ganzen Konversationsverlauf im E-Mail mit. Das Customer Care Center Team hat somit die Möglichkeit, die bereits abgehandelte Konversation nachzuvollziehen.
- **Bing Spell Check API** überprüft die Eingaben des Benutzers auf Rechtschreibefehler. Bei Bedarf korrigiert es die Eingaben, bevor die Nachricht ans LUIS weitergeleitet wird.

7.4.4 Beschreibung des Datenflusses

Die Anbindung der Datenbank hat zur Folge, dass sämtliche Konversationen zwischen dem Benutzer und dem Chatbot persistiert werden. Jede Nachricht wird zuvor geprüft ob es sich um persönliche Daten handelt. Falls ja, werden diese anonymisiert und unkenntlich gemacht. Zudem werden bei jeder Nachricht folgende Informationen zusätzlich in die [conversationContainer] Tabelle der Datenbank geschrieben:

- **PartitionKey** hält die SessionID der Konversation fest.
- **RowKey** hält fest ob es sich um einen bekannten oder unbekannten Intent handelt
- **Timestamp** hält den Zeitpunkt fest, an welchem die Nachricht übermittelt wurde.
- **Question** hält die Frage des Benutzers fest
- **Answer** hält die Antwort des Chatbots fest, welche es geliefert hat.

Unbeantwortete Fragen werden in der [newQuestionTable] persitiert. Der Chatbot Verantwortliche hat somit Einsicht, welche Fragen dem QnA Maker fehlen. Mit der zusätzlichen Information des «IntentScore» lässt sich herauslesen, wie sicher der QnA Maker war mit der Beantwortung der Frage.

Der Chatbot ist in der Lage nun Kategorien vorzuschlagen. Die Kategorien dienen dazu, die Frage des Benutzers herauszufiltern. Mit jeder Auswahl werden die Antwortmöglichkeiten des Benutzers eingeschränkt. Am Ende dieser Iteration, werden dem Benutzer spezifische Beispielfragen präsentiert. Es besteht jedoch die Möglichkeit jederzeit selbst aktiv zu werden und die Frage selbst zu schreiben.

7.4.5 Software Architektur (Klassenstruktur)

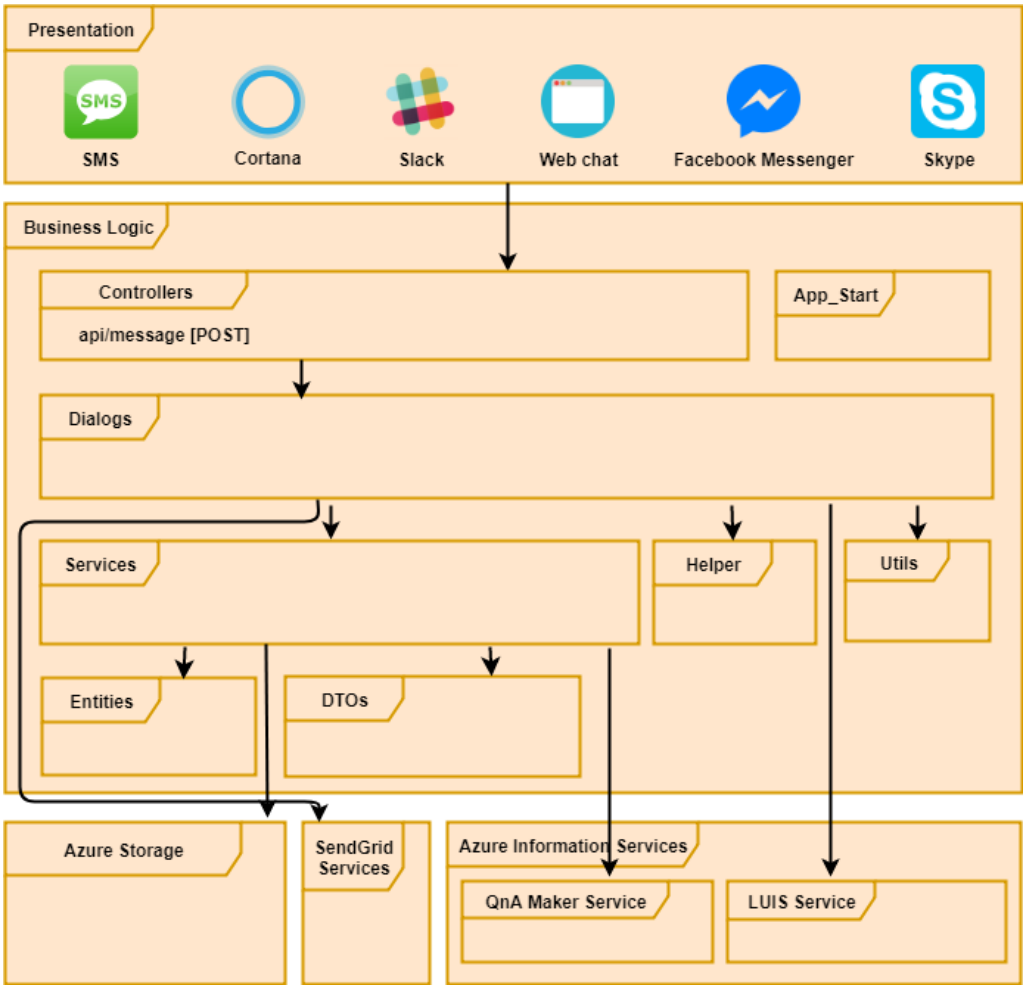


Abbildung 31: Software Architektur Demonstrator 3

Klasse/Ordner		Beschreibung
Presentation		Im Presentation Layer sind alle möglichen Kanäle verfügbar. Die Software Entwickler müssen sich nur noch Gedanken machen auf welchem Kanal sie den Chatbot publizieren wollen. Die Grundimplementation ist überall gleich.
Business Logic	Controllers	In diesem Package werden alle Routen definiert, welche den Chatbot unterstützen. In unserem Fall gibt es nur eine Route und eine Methode und zwar die «POST» Methode für die Route «api/messages». Da wir ja nur die Nachrichten vom Benutzer über den jeweiligen Kanal entgegennehmen. Selbstverständlich können die Entwickler weitere Routen je nach Bedarf definieren.
	App_Start	In diesem Package befindet sich die Web API Konfiguration und Services. Ebenfalls werden die Web API Routen hier registriert. Dieses Package beinhaltet weitere wichtige Konfigurationen für den Chatbot sowie die Zugangsdaten für QnA Maker und Azure Chatbot.
	Dialogs	Dialogs sind einzelne Komponenten, die in sich isoliert sind. Um den Konversationsfluss steuern zu können, werden einzelne Dialogs nacheinander aufgerufen.

		Die Logik des Chatbot-Driven Ansatzes findet sich in diesem Package.
	Services	Stellt die REST API Anfrage auf den QnA Maker Service. Serialisiert und deserialisiert die Antwort in das JSON Format für die Weiterverarbeitung. Dieses Package kümmert sich auch um den Zugriff auf die Azure Table Storage. Schreib- und Lesevorgänge auf die Datenbank finden hier statt.
	DTOs	Einzelne DTO[42]
	Helper	Hilfsmethoden für das Dialog Package
	Entities	Datenbankschema für die Azure Storage Table
	Utils	Enthält das Caching der Konversation sowie das persistieren in die Datenbank
Azure Information Services	QnA Maker Service	QnA Maker Service, welcher durch das QnA Maker Portal vom Entwickler erstellt werden kann. Diese kann dann als eine REST API veröffentlicht und angesprochen werden.
	LUIS Service	LUIS Service, der durch das LUIS Portal vom Entwickler erstellt werden kann. Diese kann dann als eine REST API veröffentlicht und angesprochen werden.
Azure Storage		Persistierung der Daten in der Azure Table Storage
SendGrid Services		Unabhängiger E-Mail Service zum Versenden der geführten Konversation.

7.4.5.1 Solution Architektur

Software Architektur spielt eine wichtige Rolle beim Aufbau einer Applikation. Es ist vor allem wichtig, sollte die Software zu einem späteren Zeitpunkt erweitert werden. Unser Chatbot basiert auf dem Microsoft Bot Framework und ist in der Programmiersprache «C#» geschrieben. In «C#» kann der Code in unterschiedliche Solutions oder auch Projekte unterteilt werden. Ein Vorteil ist die Verteilung der Aufgaben unter den unterschiedlichen Teams. Dadurch können unterschiedliche Teams unabhängig voneinander entwickeln. Zum aktuellen Zeitpunkt haben wir alle Packages in der gleichen Solution. Dies ist bei einer Teamgrösse von zwei Personen nicht problematisch. Sobald mehr Personen im Projekt mitentwickeln, kann es zu einem Mehraufwand kommen. In der unteren Abbildung haben wir eine mögliche Solution Architektur für den Microsoft Azure Chatbot abgebildet.

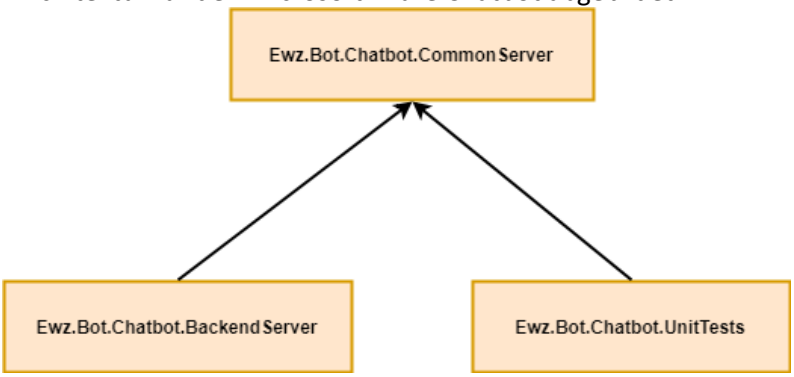


Abbildung 32: Mögliche Solution Architektur

[42] Data Transfer Object

Von Unternehmen zu Unternehmen ist der Aufbau von Teams unterschiedlich. Es gibt oftmals ein Team, das die Entwicklung macht und eine anderes für das Testen der Applikation zuständig ist. Das Entwicklungsteam kann auf «Ewz.Bot.Chatbot.CommonServer» Solution die Implementation zur Verfügung stellen, das gleichzeitig vom Testingteam verwendet werden kann. Diese Kapselung bringt die nötige Unabhängigkeit zwischen den einzelnen Teams. In der nächsten Abbildung haben wir eine weitere mögliche Solution Architektur dargestellt.

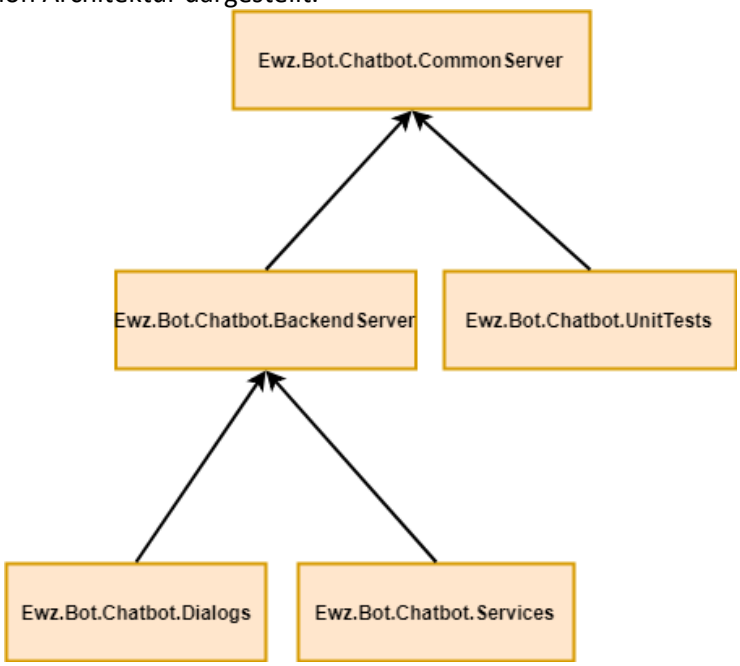


Abbildung 33: Mögliche Solution Architektur für ewz Chatbot

Die Solution oder auch Projekte können wie eine Art Microservice verwendet werden, das unabhängig untereinander entwickelt werden kann.

7.4.6 Azure Storage (Datenbank)

Während der Entwicklung sind wir vor die Entscheidung gestanden, wie wir die Daten speichern möchten. Damit sind sämtliche Konversationen sowie die unbeantworteten Fragen gemeint. Azure bietet zwei Datenbankservices an, Cosmos DB und Azure Table Storage. Wir haben die beiden Typen gegenübergestellt anhand einer Tabelle [53]:

Name	Azure Cosmos DB	Azure Table Storage
Description	Globally distributed, horizontally scalable, multi-model database service	A Wide Column Store for rapid development using massive semi-structured datasets
Initial Release	2014	2012
SQL	SQL-like query language	no
APIs	DocumentDB API Graph API (Gremlin) MongoDB API RESTful HTTP API Table API	RESTful HTTP API
Programming languages	.Net, C#, Java, JavaScript, Python	.Net, C#, C++, Java, Ruby, Python, PHP

Abbildung 34: Cosmo DB vs. Table Storage

Die Gegenüberstellung gleicht einem Vergleich von Birnen mit Äpfeln. Die beiden Datenbanktypen sind schlichtweg für unterschiedliche Anforderungen konzipiert. Table Storage lässt sich einsetzen als Datenablage für Typen wie Excel, Bilder, Filme, Audio etc. Cosmos DB dagegen ist eine klassische Datenbank mit Tabellen.
Wir haben uns für Azure Table Storage entschieden, ausfolgenden Gründen.

1. Cosmos DB wäre ein «Overkill». Ein Bruchteil der Funktionen würde genutzt werden. Der Lizenzierungspreis ist schlichtweg zu hoch für unseren Bedarf. Table Storage ist imVergleich um einiges günstiger.
2. Die Subscription des Azure Accounts erlaubt es nicht eine Cosmos DB aufzusetzen. Um möglichst schnell zu testen ob und wie wir auf eine Datenbank zugreifen können, haben wir uns für das kostengünstigere Table Storage entschieden.

7.4.7 Datenbankschema

Das folgende Diagramm zeigt die Datenbankstruktur auf mit den Tabellen, die einen Dialogcharakter besitzen. Die jeweiligen Dialoge beinhalten die vorgefertigten Dialogauswahloptionen.

7.4.7.1 Tabellen mit Dialogcharakter

Die Beziehung innerhalb der Tabelle wird im Kapitel 7.4.8 erläutert. Die Spalte *TagValue* aus der Tabelle *MainMenuDialogTable* verweist auf die Zugehörigkeit der jeweiligen Subdialoge. Das Gegenstück auf den Subdialogen, bilden die *PartitionKeys*. Diese müssen zwingend Paarweise gleich benannt werden.

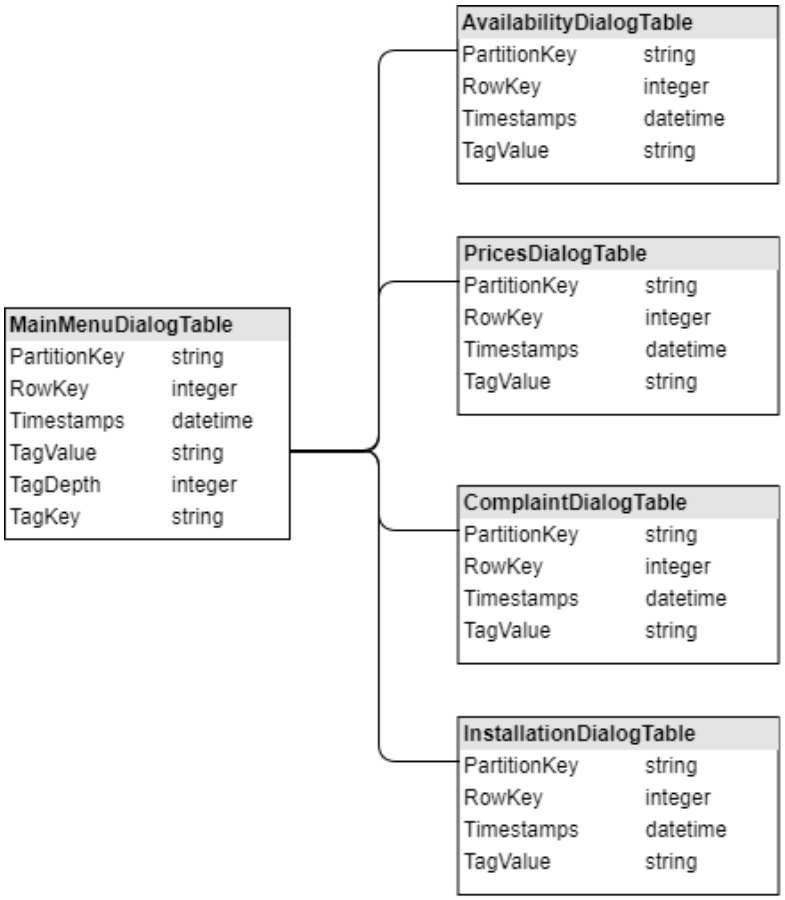


Abbildung 35: Teilausschnitt Datenbankschema

- ***MainMenuDialogTable***: Enthält Informationen über das Hauptmenü. Diese Tabelle legt auch fest, wie viele Hierarchien die einzelnen SubDialoge enthalten
- ***InstallationDialogTable***: Enthält die Auswahloptionen der Kategorie «Installation»

- ***ComplaintDialogTable***: Enthält die Auswahloptionen der Kategorie «Reklamation»
- ***PricesDialogTable***: Enthält die Auswahloptionen der Kategorie «Preise und Angebote»
- ***AvailabilityDialogTable***: Enthält die Auswahloptionen der Kategorie «Verfügbarkeit»

Erklärung und Beziehung zu den einzelnen Spalten:

- ***PartitionKey***: Dient zur Identifikation der Dialogzugehörigkeit
- ***RowKey***: Indexnummer, welche zum jeweiligen PartitionKey gehört. Ein PartitionKey kann mehrere RowKeys besitzen
- ***Timestamps***: Datum und Zeit, an welchem der Eintrag zuletzt verändert wurde
- ***TagValue***: Enthält den sichtbaren Text, welcher bei der Dialogauswahl angezeigt wird
- ***TagDepth***: Gibt an, wie viele Hierarchiestufen das jeweilige Dialogmenü besitzt

7.4.7.2 Tabellen für die Analyse und Zusatzinformationen

conversationContainer	
PartitionKey	string
RowKey	string
Timestamps	datetime
Question	string
Answer	string

newQuestionTable	
PartitionKey	string
RowKey	string
Timestamps	datetime
Question	string
IntentScore	double

InfoMessageTable	
PartitionKey	string
RowKey	integer
Timestamps	datetime
TagValue	string

Abbildung 36: Hilfstabellen mit Zusatzinformationen

- ***InfoMessageTable***: Enthält die Standardnachricht, wenn keine Antwort beim QnA Maker gefunden werden kann sowie die Willkommensnachricht.
- ***conversationContainer***: Enthält Nachrichten, welche zwischen Benutzer und Chatbot ausgetauscht wurden
- ***newQuestionTable***: Enthält sämtliche Fragen, welche der Chatbot nicht beantworten konnte

Erklärung und Beziehung zu den einzelnen Spalten:

- ***PartitionKey***: Identifikation, zur Session Zugehörigkeit
- ***RowKey***: Indexnummer, die zum jeweiligen PartitionKey gehört. Ein PartitionKey kann mehrere RowKeys besitzen
- ***Timestamps***: Datum und Zeit, an welchem der Eintrag zuletzt verändert wurde
- ***Question***: Die geschriebene Frage des Benutzers
- ***Answer***: Die gefundene Antwort des Chatbots
- ***IntentScore***: Gibt an, wie sicher der Chatbot war mit der Antwort. Der Wert entspricht einer Zahl zwischen 0 bis 1. Eine 1 bedeutet: Der Chatbot war sich sehr sicher, den richtigen Intent gewählt zu haben

7.4.8 Dynamische Hierarchiestufe der Dialogmenüs

Wie aus dem Diagramm [Abbildung 39: Konversationsfluss Chatbot Driven] zu entnehmen ist, gibt es mehrere Hierarchiestufen bei den Dialogoptionen. Die Dialogoptionen können daher beliebig tief verschachtelt werden. Man legt zuerst in der Datenbank fest, wie viele Hierarchiestufen die jeweilige Hauptmenüoption beinhaltet soll.

PartitionKey	RowKey	TagDepth	TagValue	TagKey
1	1	5	Verfügbarkeit Glasfaseranschluss	MainMenuDialogTag
1	2	5	Preise und Angebote	MainMenuDialogTag
1	3	3	Installation	MainMenuDialogTag
1	4	3	Reklamation	MainMenuDialogTag

Abbildung 37: Festlegung Hierarchiestufe

In der jeweiligen Tabelle müssen dann noch die Dialogoptionen hinterlegt werden. Dabei verfolgt es ein einfaches Prinzip. Der *TagValue* repräsentiert den visuellen Text, der beim Chatbot als Dialogoption angezeigt wird. Jeder *TagValue* kann als Unteroption einen oder mehrere *PartitionKey* enthalten. Diese wiederum können wieder Unteroptionen enthalten. So kann beliebig tief verschachtelt werden.

PartitionKey	RowKey	TagValue
X1.1	2	X1.1.2
X1.1	1	X.1.1.1
X1	2	X1.2
X1	1	X1.1
Service Provider	4	X1
Service Provider	3	Wie kann ich den Internetprovider wechseln?
Service Provider	2	Welche Service Provider Angebote via ewz.zuerinet gibt es?
Service Provider	1	Welche Service Provider sind verfügbar?
Preise und Angebote	4	Info zu OTO-ID oder OTO Dose
Preise und Angebote	3	Info zu Try und Buy
Preise und Angebote	2	Glasfaseranschluss Kosten
Preise und Angebote	1	Service Provider

Abbildung 38: Hierarchiestufe festlegen

7.4.9 Konversationsfluss im Chatbot Driven Design

Das folgende Diagramm zeigt den Konversationsfluss auf, wie er beim Demonstrator 3 zum Einsatz kommt. Die Hierarchiestufe der einzelnen Dialogoptionen kann dynamisch nach Belieben variieren. Zu Gunsten der Übersicht bildet das Diagramm Hierarchiestufen bis zur 3. Stufe ab. Mit Hierarchiestufen sind die Menüoptionen gemeint, welche wiederum einzelne Auswahlmöglichkeiten haben.

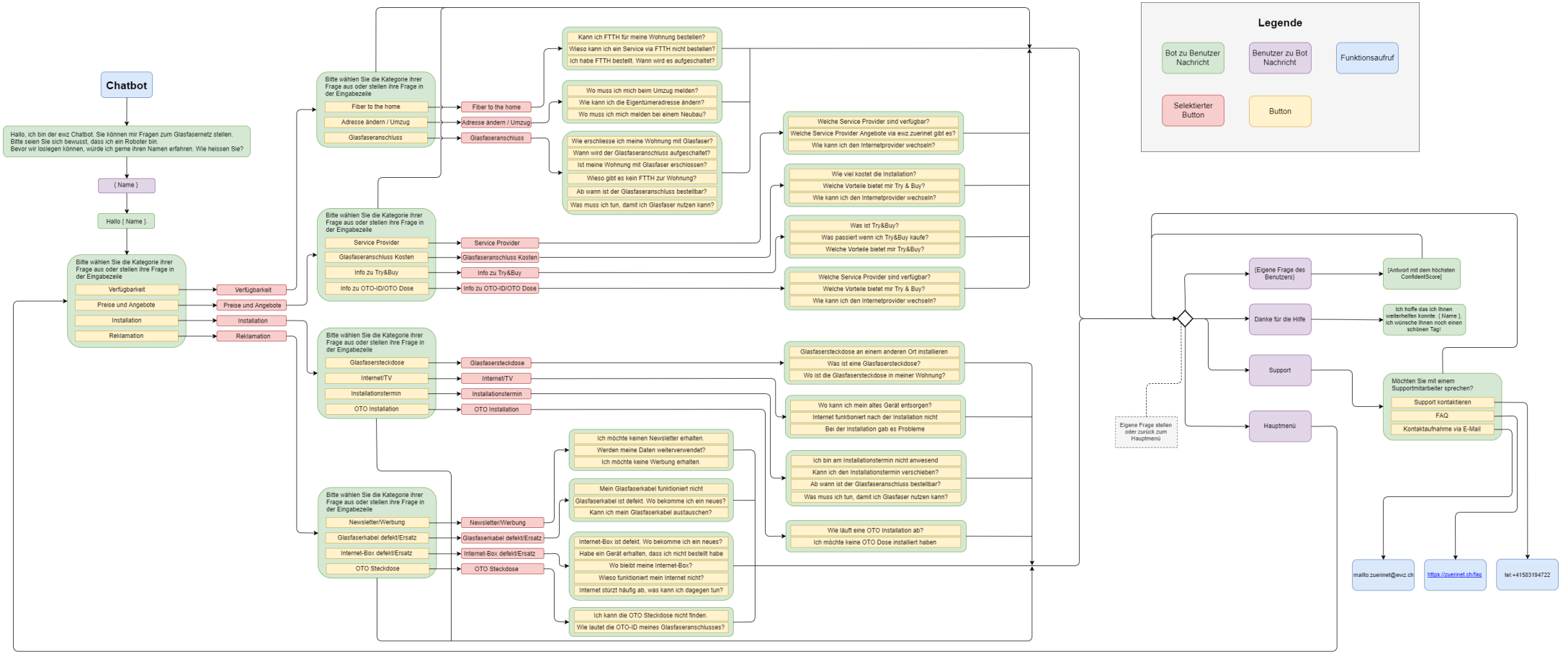


Abbildung 39: Konversationsfluss Chatbot Driven

7.4.10 Personendaten schützen

Das Cloud Portal Azure bietet zum heutigen Zeitpunkt keine Möglichkeit an den Server in der Schweiz zu hosten. Damit gibt es ein Problem mit den ewz Sicherheitsrichtlinien. Aus der Aufgabenstellung der Bachelorarbeit geht hervor, dass keine personenbezogenen Daten in der öffentlichen Cloud wie z.B. Azure gespeichert werden dürfen. Auch wenn der Chatbot nie explizit nach der E-Mailadresse oder den Namen fragt, kann der Benutzer dies preisgeben. Der Chatverlauf einer Konversation wird stets aufgezeichnet und in der Cloud Datenbank abgelegt.
Damit keine Personendaten in der Azure Datenbank gespeichert werden, braucht es einen Content Moderator [54]. Dieser Service prüft Text-, Bild- und Videoeingaben auf potentiell schädlichen, unangebrachten oder anderweitig anstössigen Inhalt.
Für den Chatbot würden wir generell diesen Service empfehlen, damit persönlich identifizierbare Informationen entdeckt werden. Damit wird sichergestellt, dass zumindest E-Mailadressen erkannt werden. Dieser Service ist jedoch beschränkt auf die englische Sprache. Damit werden US & UK Telefonnummern gefiltert, jedoch CH nicht. Durch diese Einschränkung wird nur ein Teil dieses Services effektiv genutzt.
Namen werden nicht vom Content Manager behandelt. Um trotzdem keine Namen in die Log-Dateien zu schreiben, entschieden wir uns dies im Programmcode abzufangen.
Bei der Willkommensnachricht wird aktiv nach dem Namen des Benutzers gefragt. Der Name wird dann beim Speichern in die Azure Datenbank unkenntlich gemacht.

7.5 Allgemeine Architektur

7.5.1 Deploymentdiagramm

Der Chatbot wird auf dem Azure Portal bereitgestellt. Der Web Chat wird auf der Homepage der ewz Telecom Webseite embedded. Über den Web Chat kann der Benutzer den Chatbot verwenden. Selbstverständlich kann der Chatbot über andere Kanäle verwendet werden. Die Anforderung für die Bachelorarbeit lautet, dass der Chatbot über die ewz Webseite verfügbar ist. Aus diesem Grund werden die anderen Kanäle nicht berücksichtigt. Für den letzten Prototypen soll eine Anbindung an die Datenbank der ewz Telecom sichergestellt werden.

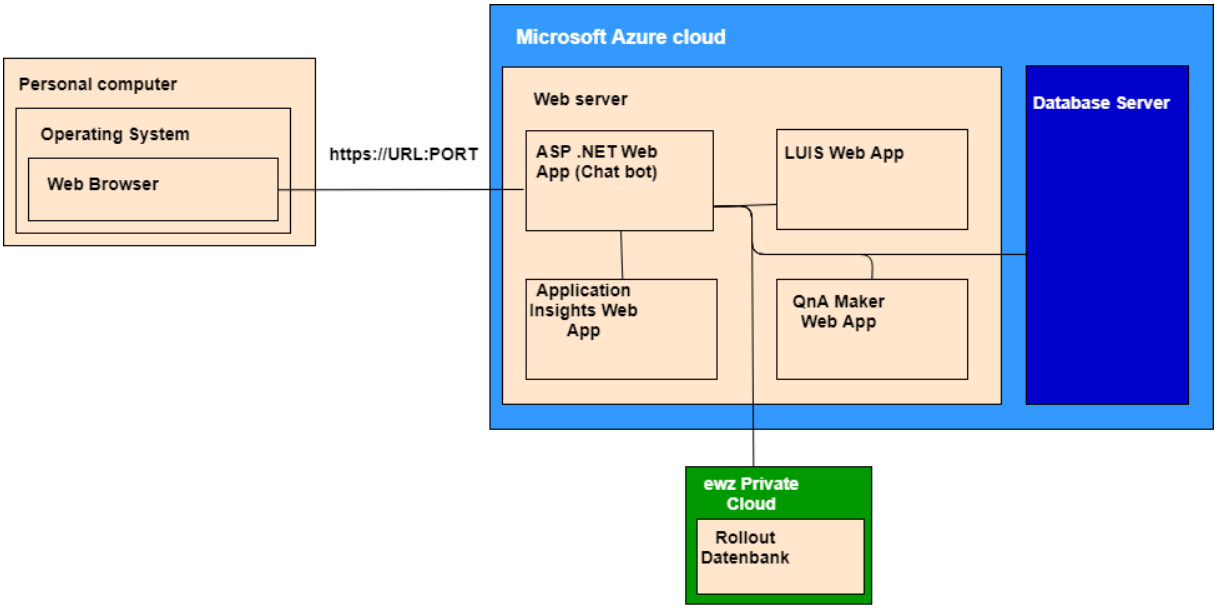


Abbildung 40: Deploymentdiagramm ewz Chatbot

7.6 Zusätzliche Features

7.6.1 Multi language Support

Die aktuelle Chatbot Version erkennt nur Deutsch als Sprache die restlichen Sprachen werden von LUIS nicht erkannt. Bei der Zwischenpräsentation kam die Frage auf wie man mehrere Sprachen unterstützen kann. Es gibt zwei mögliche Lösungen. Die erste Variante benötigt für jede Zielsprache einen eigenen QnA Maker sowie LUIS Service. Im Fall der ewz bedeutet dies, dass sie fünf QnA Maker sowie LUIS Services aufbauen müssten. Der Nachteil liegt darin, dass mit höheren Betriebskosten zu rechnen ist. Die folgende Abbildung verdeutlicht es.

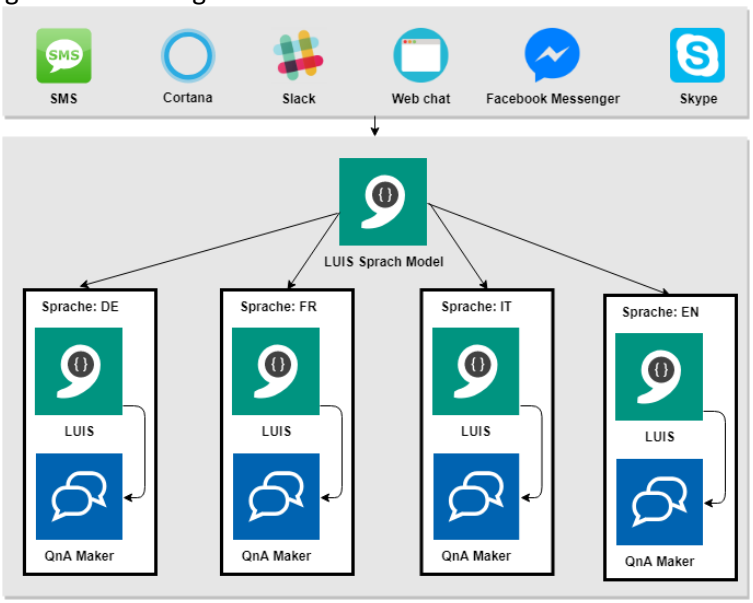


Abbildung 41: Multilanguage Support Variante 1

Die zweite Variante ist etwas günstiger und eleganter. Hier werden die Texteingaben der jeweiligen Sprache des Benutzers ins Deutsche übersetzt und erst danach zu LUIS für die Verarbeitung gesendet. Das Unternehmen «DeepL» bietet eine fortgeschrittene Schnittstelle an, um diverse Sprachen zu übersetzen. Die REST API nimmt die Ursprungssprache entgegen und liefert die gewünschte Zielsprache zurück.

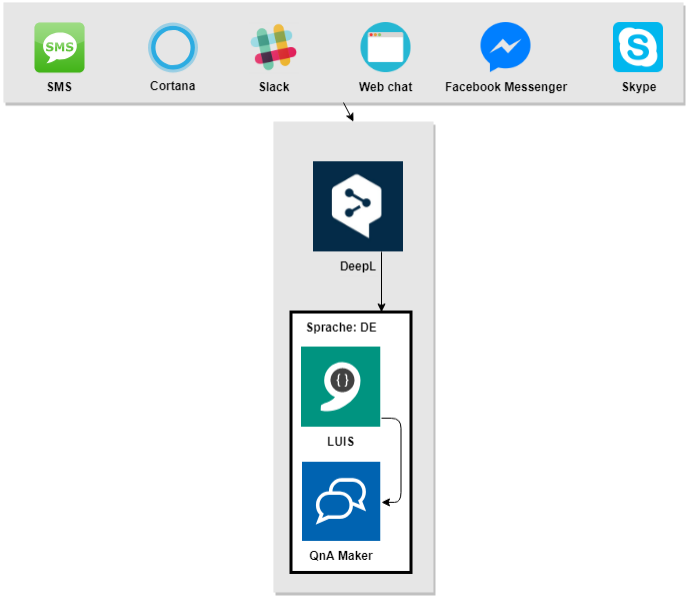


Abbildung 42: Variante 2 mit DeepL

7.6.2 Bing Spell Check API [55]

Grammatik- und Tippfehler passieren häufiger als man es sich wünscht. Azure bietet einen Service an, welches es erlaubt Texteingaben auf Grammatikfehler zu prüfen, bevor diese an den LUIS Service weitergereicht werden. Die Basis bildet die Bing Suchmaschine, welche bereits Benutzereingaben zur Laufzeit prüft. Wir sprechen an dieser Stelle eine klare Empfehlung aus für einen Spell Checker. Leider besteht Microsoft, dass sämtliche Benutzereingaben gespeichert und weiterverwendet werden dürfen.
Bei einer Benutzereingabe wird der «query» des Benutzers zuerst überprüft und allenfalls im «alteredQuery» die korrigierte Version zurückgegeben. Der Benutzer sieht die korrigierte Version im Chatbot nicht. Diese wird im Hintergrund an LUIS weitergeleitet und von dort aus an den QnA Maker. Eine Beispielkorrektur sieht dann dabei so aus:

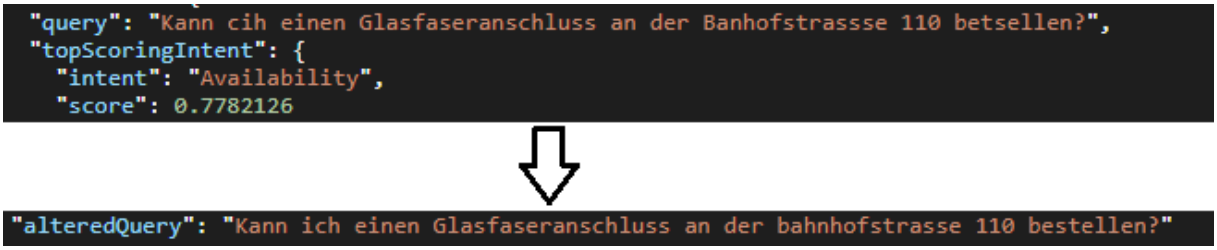


Abbildung 43: Originalquery zur korrigierten Version

7.6.3 Support E-Mail senden

Zur Unterstützung der Kundenzufriedenheit bietet der Chatbot die Möglichkeit ein Supportticket auszulösen. Dabei wird die Konversation zwischen dem Bot und dem Benutzer aufgezeichnet und in chronologischer Reihenfolge im Supportticket wiedergegeben. Das Supportticket entspricht dann einem E-Mail, welches an das Customer Care Center gesendet wird. Der Benutzer muss lediglich seine E-Mail-Adresse hinterlegen, damit der Kundendienst eine Kontaktadresse hat. Der Kundendienstmitarbeiter erhält kurzdarauf eine E-Mail mit der Konversation.


[Support] - neue Kontaktanfrage

Chatbot <no-reply@mail.com>
Mi 12.12, 10:44
Lindauer Christian

Antworten

Dies ist eine automatisch generierte E-Mail.

Bot: Hallo, ich bin der ewz Chatbot. Sie können mir Fragen zum Glasfasernetz stellen. Bitte seien Sie sich bewusst, dass ich ein Roboter bin. Bevor wir loslegen können, würde ich gerne ihren Namen erfahren.
Wie heissen Sie?
User: Chris
Bot: Hallo Chris, wie kann ich Ihnen helfen?
User: Was kostet die Erschliessung für Grundeigentümer?
Bot: Die Erschliessung für den Grundeigentümer ist kostenlos im Zuge der erstmaligen Erschliessung der bis 31. Dezember 2019 bereits bestehenden und bezugsbereiten Gebäude
User: Menü
User: Support
User: Kontaktaufnahme via E-Mail
Bot: Damit wir Sie kontaktieren können brauchen wir ihre E-Mail Adresse. Wie lautet diese?
User: marcolindauer@yahoo.de
Diese Anfrage wurde von marcolindauer@yahoo.de gesendet

Abbildung 44: Beispiel Support E-Mail

7.7 Unit Testing [56]

Das Testen einer Applikation, vor allem Unit Tests, ist ein wichtiger Bestandteil der Qualitätssicherung einer Software. Unit Tests sind vorteilhaft, um Funktionalitäten auf Fehler zu überprüfen. So kann sichergestellt werden, dass auch mit zusätzlichen Features, Funktionalitäten einwandfrei laufen. Nach einem Refactoring müssen Unit Tests fehlerfrei sein. So kann sichergestellt werden, dass das Refactoring keinen negativen Einfluss auf die Funktionalitäten hat. Dabei müssen Themen wie Kopplung zwischen Komponenten, «Dependency Injection», Mocking, Unit Test Framework usw. beachtet werden.

7.7.1 Mocking

Beim Unit Testing geht es darum Komponenten zu Isolieren und diese jeweils unabhängig voneinander zu testen. Dabei geht es um die Logik der jeweiligen Komponenten zu Testen und die Logik von anderen Komponenten sollen keinen Einfluss auf die Unit Testing haben. Falls mehrere Komponenten bei der Unit Testing beteiligt sind, dann ist es kein Unit Test mehr, sondern es ist ein Integrationtest. Je grösser das Projekt, desto mehr Abhängigkeiten zwischen Software Komponenten wird es geben. In der .Net Umgebung gibt es einen gratis Framework namens «Moq», die über die «NuGet» Package Manager installiert werden kann. Dieses Framework erlaubt den Entwicklern einfach die Abhängigkeiten zu Mocken und so die jeweiligen Komponenten einzeln zu Testen. Das gemockte Objekt kann beim Aufruf der Methode oder beim Aufbau des Objekts mitgegeben werden. Aufrufe auf andere Komponenten finden dann über das gemockte Objekt statt und wird Zugriff haben auf das gemockte Objekt. Dadurch können wir zum Beispiel überprüfen ob eine bestimmte Methode aufgerufen wurde oder nicht.

7.7.2 Kopplung zwischen Komponenten

Die Software Komponente oder Service, das von einer Klasse verwendet wird, soll wenn möglich vom Aufrufer mitgegeben werden. Falls die Komponenten in derselben Klasse instanziert wird, kann es zu einer erhöhten Kopplung führen. Eine zu starke Kopplung zwischen Komponenten führt dazu, dass diese später nicht einfach separiert werden können. Eines der wichtigsten Software Engineering Prinzipien ist es mit Interfaces zu arbeiten. Dadurch können die spezifischen Implementationen ohne grossen Aufwand ausgetauscht werden. Dies führt gleichzeitig auch zu einer geringeren Kopplung.

Die Logik des Chatbots ist in Dialogklassen abgebildet. Die Kopplung zwischen Dialogklassen und dem *IDialogContext* Interface ist stark. Dies führt dazu, dass die Logik des Chatbots nicht einfach getestet werden kann. Die starke Kopplung zwischen den Komponenten führt dazu, dass diese gar nicht gemockt werden können. Vor allem beim Unit Testing muss es «gemockt» werden, damit diese Tests unabhängig von anderen Software Komponenten getestet werden können. Das untere Klassendiagramm zeigt die Kopplung zwischen *WelcomeDialog* und dem *IDialogContext*.

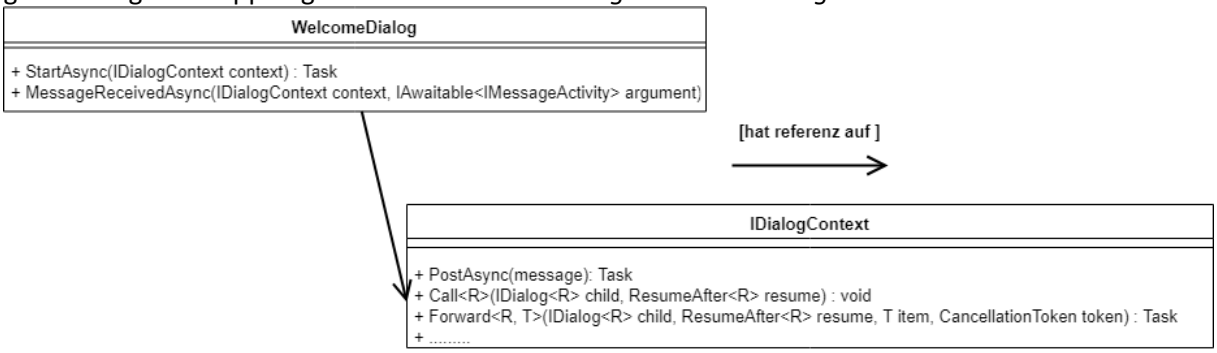


Abbildung 45: Klassendiagramm - Kopplung zwischen WelcomeDialog und IDialogContext

Um die Kopplung zu verringern muss ein zusätzliches Helperinterface aufgebaut werden, das den Zugriff auf die *IDialogContext* Implementation abstrahiert. Dadurch können wir beim Testen ein gemocktes *IDialogContext* sowie *ChatHelper* Objekt bei jedem Aufruf mitgeben. Zur Laufzeit erhalten wir dann ein Objekt vom Microsoft Bot Framework, welches vom *IDialogContext* ableitet. Das untere Klassendiagramm zeigt wie eine Entkopplung der Klassen aussehen könnte.

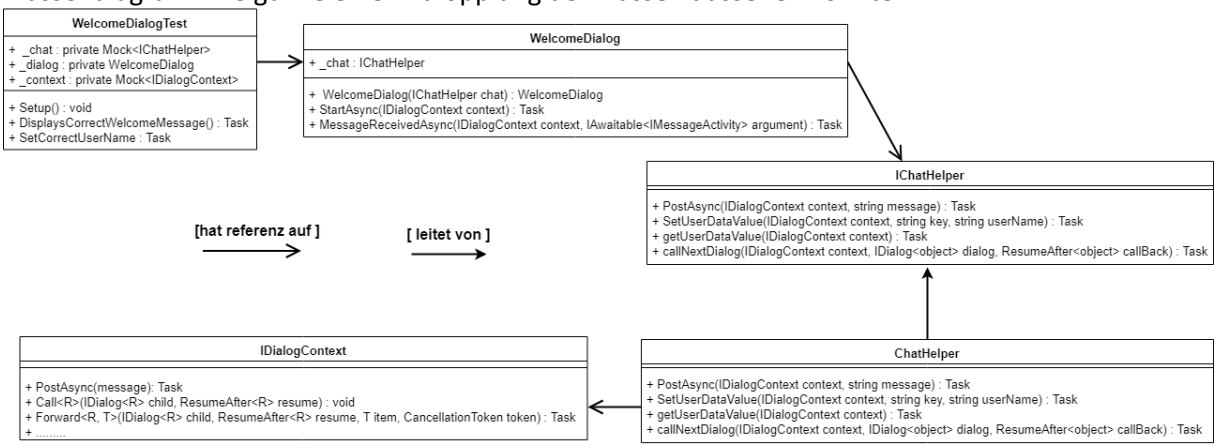


Abbildung 46: Klassendiagramm - Entkopplung zwischen WelcomeDialog und IDialogContext

7.7.3 Dependency Injection

Die Software Komponenten oder Services, die von einer Klasse verwendet werden, müssen nicht unbedingt vom Aufrufer instanziert und mitgegeben werden. Das eigentliche Ziel der Entkopplung wird dabei verletzt. Um dieses Problem zu beheben gibt es ein Konzept namens «Dependency Injection» oder auch bekannt als «Inversion of Control». Der Einsatz von «Dependency Injection» erhöht die Modularität von Programmen. Dadurch sind diese einfacher zu erweitern. Dabei werden alle Software Komponenten und Services in einem zentralen Container registriert, die diese auch zur Verfügung stellen. Die Klassen werden zur Laufzeit vom jeweiligen Framework der Software Komponenten zur Verfügung gestellt, welche benötigt werden. Dies hat zur Folge, dass die Services einfach ausgetauscht werden können.

Auch in der «.Net» Umgebung gibt es verschiedenen Frameworks die solche Prinzipien einsetzen. Das Framework names «Autofac» ist ein «IoC Container» für die Microsoft .Net Umgebung, das über den «NuGet» Package Manager, gratis heruntergeladen werden kann. Dieses Framework erlaubt es den Entwicklern, die Services in der «*Global.asax.cs*» Datei zentral zu registrieren. Die Services, die im «IoC Container» registriert sind werden beim Aufruf der jeweiligen Klasse mitgegeben. Dadurch entfällt die Notwendigkeit diese Klassen lokal zu instanziieren.

7.7.4 Unit Testing Framework

Unit Testing Framework ist einer der wichtigsten Bestandteile um ein Software Produkt erfolgreich und einfach zu testen. Abhängig von der Programmiersprache und Framework kann es vorkommen, dass kein built-in Unit Testing Framework vorhanden ist, was auch der Fall beim Microsoft Bot Framework ist. Eines der bekanntesten Unit Testing Framework im Microsoft «.Net» ist der «NUnit» Unit Testing Framework, welche über die «NuGet» Package Manager gratis heruntergeladen werden kann. Dabei handelt es sich um eine Open-source Unit Testing Framework, das die gleiche Funktionalität anbietet wie JUnit in der Java Welt. Da wir gute Kenntnisse in der JUnit haben und dadurch die Einarbeitung in das NUnit Framework etwas leichter wird, haben wir uns entschieden die NUnit Unit Testing Framework für unser Unit Testing einzusetzen.

7.8 Probleme bei der Umsetzung

Während der Entwicklung sind wir auf zahlreiche Stolpersteine gestossen. Die nachfolgenden Kapitel beschreiben kurz das Problem und welche Gedanken wir uns gemacht haben, um es bestmöglich zu lösen.

7.8.1 Problem 1: Willkommensnachricht erscheint zweimal [57] [58]

Beschreibung

Bei der Entwicklung haben wir gemerkt, dass es bei manchen Sachen unterschiedliches Verhalten zwischen Web Chat und dem Bot Emulator gibt. Ein Beispiel dafür ist, dass die gleiche Willkommensnachricht zweimal bei Bot Emulator erscheint wobei bei Web Chat es nur einmal erscheint. Der Grund wieso, dass zweimal die Willkommen Nachrichten erscheint ist simpel. Es werden zwei «ConversationUpdate» Events erstellt, eine beim Eintritt des Bots und der andere beim normalen Benutzer. Unser Code soll nur die Willkommensnachricht senden, wenn der Benutzer in die Konversation. Bisher haben wir bei jeder einkommenden Nachricht angeschaut ob es sich um eine «ActivtyType» von «ConversationUpdate» handelt, denn es wird nur ausgelöst, wenn jemand in die Konversation eintritt. Falls der Benutzer eine Nachricht sendet, dann handelt der «ActivityType» um «Message». Dadurch können die Entwickler das Verhalten auseinanderhalten.

```
private Activity HandleSystemMessage(Activity message)
{
    if (message.Type == ActivityTypes.ConversationUpdate)
    {
        // Add introduction here:
        ConnectorClient connector = new ConnectorClient(new Uri(message.ServiceUrl));
        Activity reply = message.CreateReply("Hallo, Willkommen bei der ewz Chat bot");
        connector.Conversations.ReplyToActivityAsync(reply);
    }
}
```

Abbildung 47: Nachricht senden falls es sich um ein «ConversationUpdate» handelt

Lösung

Der Entwickler muss bei jedem «ConversationUpdate» überprüfen ob es sich um ein Bot oder einen normalen Benutzer handelt und nur die Willkommensnachricht senden, falls es sich um einen Benutzer handelt, der in die Konversation beigetreten ist. Dabei handelt es sich nicht um einen Bug, sondern es handelt sich um ein Update für die Gruppen Konversation, bei welcher sich mehrere Benutzer zu beliebigen Zeitpunkten in die Konversation eintreten und wieder gehen können.

```
private Activity HandleSystemMessage(Activity message)
{
    if (message.Type == ActivityTypes.ConversationUpdate)
    {
        // Add introduction here:
        IConversationUpdateActivity update = message;
        var client = new ConnectorClient(new Uri(message.ServiceUrl),
                                         new MicrosoftAppCredentials());
        if (update.MembersAdded != null && update.MembersAdded.Any())
        {
            foreach (var newMember in update.MembersAdded)
            {
                if (newMember.Id != message.Recipient.Id)
                {
                    var reply = message.CreateReply();
                    reply.Text = "Hallo, Willkommen bei der ewz Chat bot";
                    client.Conversations.ReplyToActivityAsync(reply);
                }
            }
        }
    }
}
```

Abbildung 48: Nachricht senden falls es sich um ein "ConversationUpdate" und einen Benutzer handelt

7.8.2 Problem 2: Unterschiedliches Verhalten Bot Emulator und Web Chat

Beschreibung

Wie bei Problem 1 geschildert haben wir bei der Entwicklung des Chatbots bemerkt, dass es sich zwischen dem Bot Emulator und Web Chat unterschiedlich verhält. Beim Bot Emulator haben wir die gleichen Willkommensnachrichte zweimal erhalten wobei beim Web Chat diese nur einmal angezeigt wurde.

Lösung

Sollten merkwürdige Asynchrone Probleme während der Entwicklung auftreten, lohnt sich die Applikation auf Azure zu deployen. Da letztendlich der Chatbot auf Azure läuft gilt dies als Referenz. Wir konnten leider bis zu diesem Zeitpunkt nicht herausfinden, weshalb sich das Verhalten lokal ändert.

7.8.3 Problem 3: QnA Maker Key kann nicht der «.bot» Datei hinzugefügt werden

Beschreibung

Bei der Entwicklung vom Demonstrator 1 mit SDK v4 ist es uns nicht gelungen die Anmeldedaten für den QnA Maker, wie von Microsoft Azure vorgeschlagen, auf der «.bot» Datei zu definieren. Die «.bot» Datei wurde jedes Mal verändert, wenn man den Chatbot lokal testen wollte.

Lösung

Um die Entwicklung des Chatbots voranzubringen haben wir uns entschieden als Zwischenlösung die Anmeldedaten vom QnA Maker in der *Startup.cs* Klasse als Konstante zu definieren. Dadurch können wir bei der Ausführung des Chatbots den Service erzeugen welcher den Zugriff auf den QnA Maker

sicherstellt. Da es sich um einen Prototyp handelt, haben wir diesen Weg gewählt. Für die produktive Umgebung sollten solche Zugangsdaten unbedingt separat in einer Datei abgelegt werden.

7.8.4 Problem 4: Azure Bot Service Version SDK v3 vs. SDK v4

Beschreibung

Während der Entwicklung des Demonstrator 1 wurden wir mit der Entscheidung konfrontiert, auf welche SDK Version wir setzen möchten. Microsoft bietet mit dem Azure Bot Service SDK v3 ein etabliertes Framework um Chatbots zu entwickeln, deployen und testen. Mit der zukünftigen Version SDK v4 wird jedoch langsam aber sicher SDK v3 abgelöst. Wann diese Umstellung stattfinden wird, konnten wir während der Bachelorarbeit nicht sagen. Offiziell ist SDK v4 nur als Preview Version Verfügbar. Anhand der Roadmap von SDK v4 dauert die Umstellung sicher bis Dezember 2018 an. Der Abgabetermin der Bachelorarbeit ist jedoch bereits Ende Dezember.
Der Unterschied der beiden Versionen beruht hauptsächlich darauf, dass unterschiedliche Komponenten zur Entwicklung von Chatbots genutzt werden. Chatbots welche auf Version 3 basieren, werden auch in Zukunft durch ein Upgrade auf V4. [59] lauffähig bleiben.

Lösung

Anhand der stetigen Änderungen an SDK v4 war es uns schlichtweg zu unsicher auf die neuere Version zu setzen. Für die Entwicklung der einzelnen Demonstratoren wollen wir kein zusätzliches Risiko eingehen. Darum haben wir uns entschieden für alle Demonstratoren auf SDK v3 zu setzen.

7.8.5 Problem 5: QnA Maker unzureichend bei abweichenden Fragestellungen

Beschreibung

Ein Chatbot ist nur so intelligent wie die Knowledge Base im Hintergrund. Ist diese Knowledge Base dürftig oder mit qualitativ schlechten Fragen und Antworten gefüllt, kann der Chatbot kein guter Dialog mit dem Benutzer führen. Während des Testens ist uns aufgefallen, dass bei ähnlichen Fragen der Chatbot Mühe hatte die korrekte Antwort zu finden.

Ein Beispiel bei dem der Chatbot Probleme hat:

Abbildung 49: Auszug eines Dialogs

Lösung

Im Idealfall besteht eine Knowledge Base aus möglichst vielen Synonymfragen zu einer Antwort. Um solchen Fehlern vorzubeugen, müssen zu einer Antwort möglichst viele Synonymfragen im QnA Maker hinzugefügt werden.

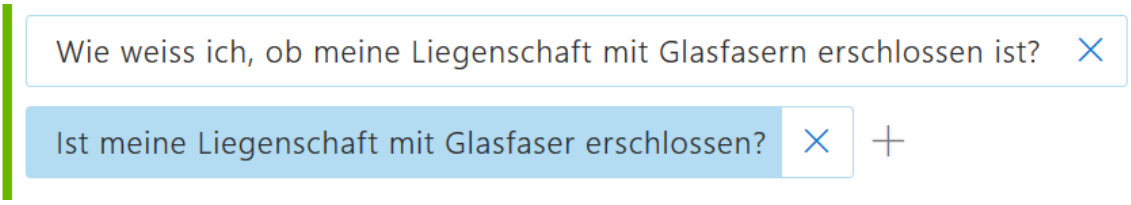


Abbildung 50: QnA Maker Knowledge Base

7.8.6 Problem 6: Speichern von LUIS und QnA Maker Schlüssel

Beschreibung

Um den Entwicklungsprozess von Demonstrator 2 SDKv3 etwas zu vereinfachen haben wir uns entschieden zuerst die Chatbot auf die Azure Portal zu erstellen und diese dann auf die lokale Maschine zu herunterladen. Dadurch haben wir ein Template mit welchem wir die Entwicklung auf die lokale Maschine fortsetzen können. Leider sind die Anmeldedaten für die LUIS und QnA Maker nicht mit dem Code heruntergeladen worden. Dies hat dazu geführt, dass der Chatbot nicht lokal gestartet werden kann.

Lösung

Die beste Methode ist die Anmeldedaten in der «Web.config» Datei zu definieren. Dies hat den Vorteil, dass wir auch die Anmeldedaten über das Azure Portal ändern können die automatisch auch die Anmeldedaten im «Web.config» Datei ändert.

```
<appSettings>
  <!-- update these with your Microsoft App Id and your Microsoft App Password-->
  <add key="MicrosoftAppId" value="[redacted]" />
  <add key="MicrosoftAppPassword" value="[redacted]" />

  <!-- LUIS Settings -->
  <add key="LuisAppId" value="[redacted]"/>
  <add key="LuisAPIKey" value="[redacted]"/>
  <add key="LuisAPIHostName" value="[redacted]"/>
```

Abbildung 51: Definition von Anmeldedaten für LUIS und Azure Chatbot im *Web.config*

```
public BasicLuisDialog() : base(new LuisService(new LuisModelAttribute(
        ConfigurationManager.AppSettings["LuisAppId"],
        ConfigurationManager.AppSettings["LuisAPIKey"],
```

Abbildung 52: Zugriff auf die Anmeldedaten von LUIS via *BasicLuisDialog.cs*

7.8.7 Problem 7: LUIS unzureichend bei abweichenden Fragestellungen

Beschreibung

Nach dem ersten Usability Test haben wir bemerkt, dass die beiden Testpersonen teilweise komplett andere Fragen stellten. LUIS war schlichtweg nicht in der Lage den Intent des Benutzers zu erkennen. Dies hat zur Folge, dass LUIS keinen Intent zur Verfügung hat, um die entsprechende QnA Maker Knowledge Base abzufragen. Durch die abweichenden Fragestellungen wird der Confidencescore drastisch gesenkt. Letztendlich resultiert daraus eine schlechtere Usability gegenüber dem Benutzer.

Lösung

Damit eine höhere Akzeptanz erreicht werden kann muss das LUIS Model sowie die QnA Maker Knowledge Base trainiert und aufbereitet werden. Anhand von Synonymwörtern, unterschiedlich gestellten Fragesätzen sowie Alternativfragen, kann der Chatbot den Confidencescore erhöhen. Dieses Training muss von einem Chatbot Verantwortlichen manuell getätigt werden.

Abbildung 53: Confidencescore vor und nach dem Training

7.8.8 Problem 8: Sonderzeichen Dekodierung

Beschreibung

Nachdem LUIS den Intent aus der Fragestellung herausfiltert gibt es eine Anfrage auf der jeweiligen QnA Maker Knowledge Base. Die Frage der Benutzer wird als JSON im http anfrage mitgesendet. Der QnA Maker Knowledge Base gibt die folgende Fehlermeldung «http 400 Bad Request» zurück. Jedoch

gibt es diese Fehlermeldung nur bei bestimmten Zeichen wie «'». Eine Beispielfrage für welches der QnA Maker Knowledge Base eine «http 400 Bad Request» gibt ist «Welche Vorteile bietet mir 'try & buy'?». Der Grund dafür ist, dass diese Fragestellung wiederum dieses Sonderzeichen enthält. Dieses Fehlverhalten ist auf ein schlechten Codedesign zurück zu führen. Um genau zu sein, das JSON Objekt, welches in der http Anfrage mitgegeben wird, ist als «String» aufgebaut. Siehe untere Abbildung.

```
string questionJSON = @"{'question': '" + question + "'}";
```

Abbildung 54: Aufbau von JSON Objekt mit String (falsch)

Da die Frage des Benutzers dieses Sonderzeichen «'» enthält führt es dazu, dass es für den «key» «question», zwei «values» gibt, welche vom QnA Maker nicht aufgelöst werden kann. Aus diesem Grund gibt es die «http 400 Bad Request» Fehlermeldung zurück.

Lösung

Eine Möglichkeit wäre den «String» so aufzubauen damit es keinen Fehler wegen Sonderzeichen gibt. Also «escapen» von Character, damit die QnA Maker Knowledge Base kein Problem hat, die Frage zu lesen.

```
string questionJSON = $"{ {\"question\": \"{ question }\" } }";
```

Abbildung 55: Aufbau von JSON Objekt mit String (richtig)

Aber natürlich die Best Methode ist eine «ExtensionMethod» zu verwenden, welche den Aufbau von JSON Objekt vereinfacht. [60]

7.8.9 Problem 9: Dialog Optionen deaktivieren

Beschreibung

Während einer Konversation wählt der Benutzer verschiedene Dialogoptionen aus. Rückwirkend sieht der Benutzer jeweils welche Fragemöglichkeiten ihm angeboten wurden. Der Chatverlauf ist somit für den Benutzer einsehbar. Das Botframework sieht es vor, die einzelnen Dialoge als Stack auf- und abzubauen. Dies hat den Nachteil, dass man ohne weiteres nicht von einem Stackdialog zu einem anderen «springen» kann. Sollte nun der Benutzer eine vergangene Menüoption nochmals anklicken wollen, kann dies der aktuelle Dialogstack nicht korrekt behandeln. Dieses Verhalten tritt zumindest auf, wenn man einen Web App Chatbot entwickelt.

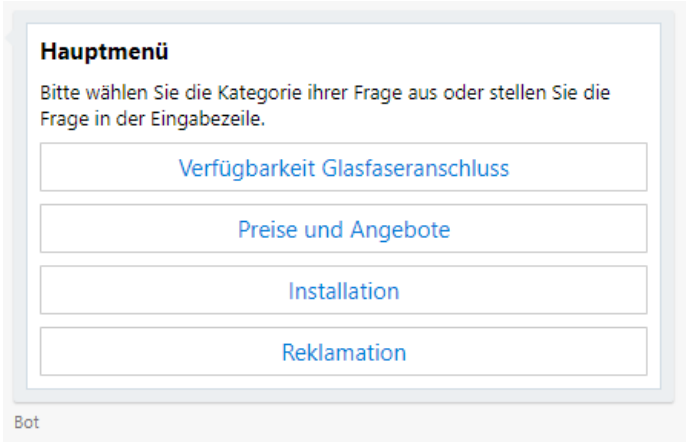


Abbildung 56: Dialog Optionen

Lösung

Zu diesem Problem haben wir keine befriedigende Lösung gefunden. Der Grund liegt darin, dass das Bot Framework keine Implementation erlaubt, um aus dem aktuellen Kontext herauszuspringen. Laut der Microsoft Dokumentation hat man sich für diese Implementation entschieden, da man keine Abhängigkeit gegenüber den verschiedenen Channel-interpretationen (z.B. Skype, Facebook Messenger, Slack etc.) haben möchte. Wir versuchten eine Lösung zu finden, wie man vergangene Dialog Optionen deaktivieren kann. Dies wird leider vom Bot Framework ebenfalls nicht unterstützt. Diese Feature wurde mehrfach gefordert und auch diskutiert auf Stackoverflow. Nachzulesen unter: [61]

7.8.10 Problem 10: Keine Referenzen bei den Dialogoptionen

Beschreibung

Es ist zwar möglich beliebig tiefe Hierarchiestufen abzubilden in der Datenbank. Jedoch gibt eine keine automatische Überprüfung, ob der gegebene *PartitionKey* zum *TagValue* existiert (mehr dazu im Kapitel Zusätzliche Features auf Seite 71). Der Datenbankadministrator muss also selbst kontrollieren ob die zugehörigen Paare identisch geschrieben sind. Sollte im Worst-Case Fall der *Partitionkey* und *TagValue* nicht identisch sein, kann der Chatbot diese Dialogoption nicht darstellen.

Lösung

Für dieses Problem haben wir leider keine Lösung gefunden. Einerseits liegt es daran, dass wir die Design Konsequenzen zu spät bemerkt haben. Andererseits haben wir diesen Design Fehler erst in der zweitletzten Woche bemerkt und der Codefreeze bereits stattgefunden hat. Dieses Problem lässt sich lösen, indem man Referenzen verwenden würde bei der Kopplung der Spalten. Somit müsste man nur in einer Spalte den Wert ändern und nicht in allen.

7.8.11 Problem 11: Funktionale Tests können nicht umgesetzt werden [62]

Beschreibung

Funktionale Tests sind eine Art von Software Testing, bei welchem das System nach gewissen Spezifikationen getestet wird. Es ist eine Art «Black Box Test», also der Tester hat keine Informationen über die Implementation. Funktionale Tests sind vor allem für Endkunden interessant, da sie sofort sehen ob die Anforderungen im Produkt umgesetzt wurden.
Da es zum aktuellen Zeitpunkt kein Framework gibt, welches uns erlaubt effektiv Funktionale Tests zu schreiben, haben wir uns entschieden nach einem Tutorial das Grundgerüst selbst aufzubauen. Der Entwickler, welcher dieses Tutorial geschrieben hat, benutzte jedoch eine eigene Library. Leider ist dieser Code bis zum aktuellen Zeitpunkt nicht zugreifbar.

Lösung

Da der Aufwand sehr gross wäre, diesen Code selbst zu schreiben, haben wir uns entschieden die Funktionalen Tests nicht zu implementieren.

7.9 Usability Tests

In diesem Kapitel werden die beiden Demonstratoren 2 und 3 (wird abgekürzt durch «D2» und «D3») einem Usability Test unterzogen.

7.9.1 Testkonzept

Der Usability Test der Applikation dient zur Ermittlung ob es einer Person aus den definierten Zielgruppen möglich, ist den Chatbot zu bedienen und sich zurecht zu finden, sowie ob sie die möglichen Szenarien durchspielen kann.

Der Testablauf erfolgt in diesen Schritten, zuerst werden die Probanden kurz über den Nutzen des Chatbots informiert. Danach werden ihnen die Szenarien präsentiert und sie werden diese auch gleich durchspielen. Abschliessend folgt eine kurze allgemeine Befragung über ihre Erfahrungen und Eindrücke des Chatbots.

Die allgemeine Befragung beinhaltet diese Fragen:

- Wie gut ist die User Experience insgesamt?
- Wie natürlich verhält sich der Chatbot?
- Ist der Ablauf einer Konversation nachvollziehbar?

7.9.2 Zielgruppen

Als Zielgruppe haben wir Benutzer aus der HSR ausgewählt, welche kein Fachwissen aus dieser Domäne besitzen. Um ein möglich repräsentatives Ergebnis zu erhalten, wählen wir bewusst Probanden mit unterschiedlichem Skill-Level.

7.9.3 Zu testende Szenarien D2 & D3

- **Szenario 1: Verfügbarkeit**
  Sie sind kein Kunde der ewz Telecom. Sie möchten die Verfügbarkeit eines Glasfaseranschlusses an Ihrer Wohnadresse prüfen. Ihre Wohnadresse lautet «8048 Zürich, Bahnhofstrasse 12».
- **Szenario 2: Glasfasersteckdose auffinden**
  Sie finden die installierte Glasfasersteckdose nicht in ihrer Wohnung. Sie möchten wissen, wo sich diese in ihrer Wohnung befindet.
- **Szenario 3: Glasfaserkabel defekt**
  Ihr Glasfaserkabel ist defekt. Sie möchten ein neues bestellen.
- **Szenario 4: Liegenschaft erschliessen**
  Sie möchten ihre Liegenschaft mit Glasfaser erschliessen, wissen aber nicht wie das funktioniert.
- **Szenario 5: Reklamation I**
  Sie haben eine Internet-Box erhalten, welche sie nicht bestellt haben. Sie möchten nun diese Internet-Box retournieren.
- **Szenario 6: Reklamation II**
  Bei der Installation haben Sie bemerkt, dass die Internet-Box defekt ist. Sie möchten ein Ersatzgerät erhalten.
- **Szenario 7: Reklamation III**
  Seit der Installation Ihres Internets stürzt es häufig unerklärlich ab. Sie möchten sich erkundigen was Sie dagegen unternehmen können.

- **Szenario 8: OTO-ID herausfinden**
  Der Glasfaseranschluss wurde bereits in Ihrer Wohnung installiert. Damit Sie diesen bei der Swisscom aufschalten können, müssen Sie die OTO-ID auf der Glasfasersteckdose kennen.
- **Szenario 9: Glasfaseranschluss Kosten**
  Sie möchten sich informieren über die Kosten für einen Glasfaseranschluss. Konkret möchten Sie wissen welche Kosten die ewz übernimmt.
- **Szenario 10: Glasfaseranschluss vorgängig bestellen**
  Sie möchten gleich an Ihrem ersten Tag in ihrer neuen Wohnung Glasfaseranschluss haben. Daher beschliessen Sie, den Glasfaseranschluss vorgängig zu bestellen.

7.9.4 Durchführung D2

In diesem Unterkapitel werden sämtliche Notizen aufgelistet, welche wir aufgeschrieben haben während des Usabilitytests für D2.

- **Szenario 1: Verfügbarkeit**
  - Die Anfrage konnte nicht direkt vom Chatbot beantwortet werden, da der Intent nicht herausgelesen werden konnte. Der Benutzer fragte den Chatbot direkt nach der Verfügbarkeit an der Strasse.
- **Szenario 2: Glasfasersteckdose auffinden**
  - Konnte ohne Probleme beantwortet werden.
- **Szenario 3: Glasfaserkabel defekt**
  - Konnte ohne Probleme beantwortet werden.
- **Szenario 4: Liegenschaft erschliessen**
  - Das Wort «Liegenschaft» wurde durch «Wohnung» ersetzt. Dadurch konnte LUIS keinen Match finden. Die Anfrage wurde damit nicht beantwortet.
- **Szenario 5: Reklamation I**
  - LUIS konnte auf Anhieb kein Match finden, da die Testpersonen «Internetbox» anstatt «Internet-Box» geschrieben haben. Die Anfrage wurde damit nicht beantwortet.
- **Szenario 6: Reklamation II**
  - Auch hier wurde das Wort «Internet-Box» unterschiedlich geschrieben und konnte nicht erkannt werden.
- **Szenario 7: Reklamation III**
  - Die Frage konnte im zweiten Anlauf erfolgreich beantwortet werden.
- **Szenario 8: OTO-ID herausfinden**
  - Konnte ohne Probleme beantwortet werden.
- **Szenario 9: Glasfaseranschluss Kosten**
  - Die Präzisierung zwischen «Kosten für Glasfaseranschluss» und «Kostenübernahme der ewz» konnte nicht unterschieden werden.
- **Szenario 10: Glasfaseranschluss vorgängig bestellen**
  - Konnte ohne Probleme beantwortet werden.

7.9.5 Nachinterview & Verbesserungen D2

Ein kurzes Nachinterview von D2, wobei folgende Punkte besprochen wurden:

Probleme während des Tests, Hürden:

- Die Probanden hatten oftmals sehr kurze und unvollständige Fragesätze verwendet. LUIS konnte zwar den Intent herauslesen, jedoch konnte der QnA Maker keine passende Antwort finden.

- Als Frage wurden teilweise mehrere Sätze geschrieben. Die Erfassung des Fragesatzes wurde damit erheblich schwieriger für LUIS.
- Synonymwörter werden nicht in jedem Fall unterstützt. Dadurch kann LUIS den Intent nicht herausfinden.
- Die Satzstellung ist wichtig für den QnA Maker. Sätze bei denen man den Haupt- mit dem Nebensatz «vertauscht», führen zu Problemen bei der Erkennung.

Nützlichkeit Beurteilung:

- Die Probanden hatten im Vorfeld wenig Erfahrung gesammelt mit Chatbots. Daher waren sie weder vorbelastet noch voreingenommen.
- Die Probanden waren nach dem Test skeptisch bezüglich den Einsatzmöglichkeiten. Bei komplexen Fragen ist der Chatbot überfordert.
- Der Konversationsfluss ist klar und verständlich. Die Probanden wussten jederzeit welche Möglichkeiten sie hatten.

Möglichkeiten zur Verbesserung:

- **Synonymwörter akzeptieren** Wörter mit der gleichen Bedeutung sollten unterstützt werden damit LUIS den Intent herausfinden kann.
- **Grammatikfehler tolerieren** Geringe Grammatikfehler sollten toleriert werden. Ein Spell-Checker wäre erwünscht.
- **Satzstellungen flexibel halten** Der Fragesatz sollte auch bei unterschiedlichen Satzstellungen erkannt werden.
- **Mehrzahl Unterscheidung** Singular und Plural Unterscheidung wäre eine sinnvolle Verbesserung.

7.9.6 Durchführung D3

In diesem Unterkapitel werden sämtliche Notizen aufgelistet, welche wir aufgeschrieben haben während des Usabilitytests für D3.

- **Szenario 1: Verfügbarkeit**
  - Konnte mittels der Auswahlmöglichkeit beantwortet werden. Benutzer war jedoch enttäuscht, dass er auf die Webseite verwiesen wurde.
- **Szenario 2: Glasfasersteckdose auffinden**
  - Konnte mittels der Auswahlmöglichkeit beantwortet werden.
- **Szenario 3: Glasfaserkabel defekt**
  - Konnte mittels der Auswahlmöglichkeit beantwortet werden.
- **Szenario 4: Liegenschaft erschliessen**
  - Der Benutzer konnte erst im zweiten Anlauf die Frage beantworten. Zuerst wählte er die falsche Dialogoption.
- **Szenario 5: Reklamation I**
  - Konnte mittels der Auswahlmöglichkeit beantwortet werden.
- **Szenario 6: Reklamation II**
  - Konnte mittels der Auswahlmöglichkeit beantwortet werden.
- **Szenario 7: Reklamation III**
  - Konnte mittels der Auswahlmöglichkeit beantwortet werden.
- **Szenario 8: OTO-ID herausfinden**
  - Konnte mittels der Auswahlmöglichkeit beantwortet werden.
- **Szenario 9: Glasfaseranschluss Kosten**
  - Benutzer wählte die falsche Dialogoption. Konnte erst beim zweiten Anlauf beantwortet werden.

- **Szenario 10: Glasfaseranschluss vorgängig bestellen**
    - Benutzer hat die Frage manuell gestellt. Als Eingabe hat er lediglich «Glasfaseranschluss bestellen» geschrieben. Die Anfrage an den QnA Maker war dadurch zu unpräzise.

7.9.7 Nachinterview & Verbesserungen D3

Ein kurzes Nachinterview von D3, wobei folgende Punkte besprochen werden:

Probleme während des Tests, Hürden:

- Es hat sich herausgestellt, dass die Probanden ebenfalls kurze Sätze schrieben bis hin zu Stichwörtern. Wir konnten damit feststellen, dass häufig Benutzer zu «faul» sind, um ganze Sätze zu schreiben, wenn sie mit einem Chatbot im Kontakt sind.
- Probanden suchten vergeblich nach einer Möglichkeit für einen Assistenten. Es war ihnen nicht klar, dass mittels «Support» oder «Menü» der entsprechende Dialog geöffnet wird.
- Die Probanden nutzten weitgehend den Chatbot-Driven Ansatz. Selbst die Frage zu formulieren wurde zum Ausnahmefall.
- Aufgrund der Kategorisierung der Fragen war nicht immer klar in welchem Bereich man suchen musste.

Nützlichkeit Beurteilung:

- Chatbot-Driven kommt dem Benutzer entgegen. Vor allem tippfaule Benutzer finden dieses System besser als die Frage selbst zu schreiben.
- Es freute die Benutzer, dass bei einer Abfrage an die Glasfaserverfügbarkeit eine direkte Antwort des Chatbots zurückkam. Der zusätzliche Schritt auf https://zuerinet.ch/availability-check fällt damit weg.

Möglichkeiten zur Verbesserung:

- **Stichwortartige Sätze** sollten besser unterstützt werden.
- **Hilfestellung anbieten**, wenn man nicht mehr weiterweiss.

8. Abschätzung Trainingsaufwand

Ein Chatbot ist nur so intelligent und gut wie dessen Wissensdatenbank. Daher gilt es diese zu trainieren und erweitern. Nur wenn die Wissensdatenbank kontinuierlich trainiert wird, kann der Chatbot mit einer hohen Trefferwahrscheinlichkeit Antworten liefern.
Für die Bachelorarbeit haben wir uns entschieden eine der vier Wissensdatenbanken (Availability, Complaint, Installation, Prices) iterativ zu trainieren. Mit jeder Trainingsiteration wird gemessen, ob abgewandelte Fragestellungen erkannt werden.
Als Trainingsdatenbank haben wir die «Availability» gewählt. Diese soll gezielt trainiert werden. Das bedeutet, dass das QnA Maker sowie LUIS Model nach jeder Iteration angepasst wird.

8.1 Trainings Life-Cycle

Jede Trainingsiteration orientiert sich nach dem Deming Zyklus. Dieser Zyklus beschreibt eine Vorgehensweise des Kontinuierlichen Verbesserungsprozesses. Der vierstufige Regelkreis lässt sich optimal adaptieren auf das Training.

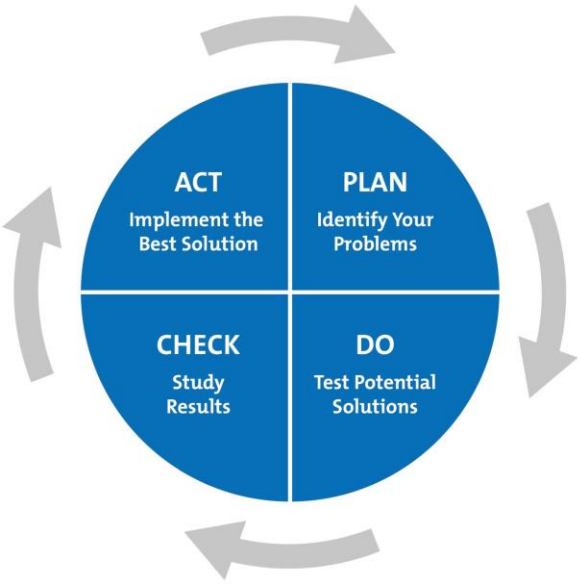


Abbildung 57: PDCA Zyklus [42]

Jede Phase verfolgt unterschiedliche Ziele. Abgestimmt für den Trainingsaufwand der Wissensdatenbank, lässt sich folgendes Modell für alle anderen Wissensdatenbanken anwenden.

- **Plan**: Identifizierung der aktuellen Probleme. Welche Fragesätze müssen erweitert und verbessert werden? Zur Planung gehört auch dazu, dass man sich auf ein Thema fokussiert und dieses ausgiebig testet.
- **Do**: Hinzufügen, erweitern der Lösung. Neue Fragesätze beim QnA Maker implementieren. Beim LUIS Modell werden «Entities» hinzugefügt. «Entities» dienen dazu, Stichwörter zu identifizieren in einer Satzstellung.
- **Check**: Nach den Änderungen müssen diese getestet und ausgewertet werden. LUIS sowie QnA Maker bieten eine umfangreiche Testumgebung an. Mittels den Feedback Wertungen sieht man rasch ob sich das jeweilige Modell verbessert hat.
- **Act**: Im letzten Schritt werden die ausgewerteten Messwerte analysiert. Verhält sich das QnA Maker sowie LUIS Modell wie gewünscht? Falls nicht, eine neue Trainingsiteration starten und neue Ziele im Plan definieren.

Die folgende Grafik zeigt ein Testergebnis von LUIS auf, beim Fragesatz «Wie wird meine Wohnung mit Glasfaser erschlossen?»

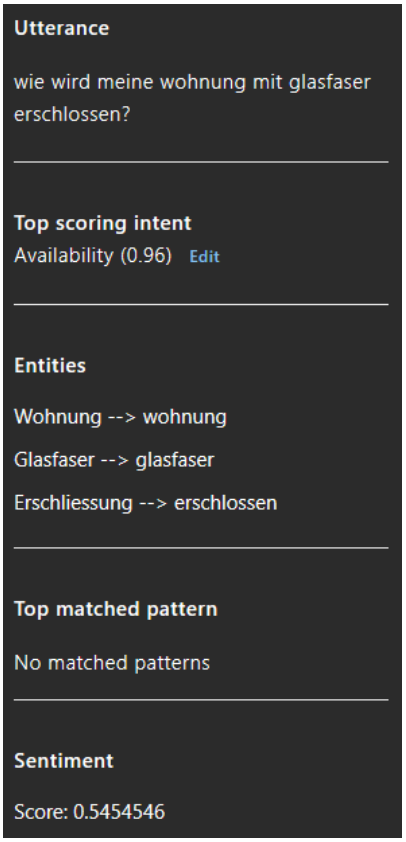


Abbildung 58: LUIS Testergebnis

8.2 Zeitaufwand

Je nach Grösse der Wissensdatenbank und Komplexität variiert der zeitliche Aufwand für das Training. Für das Training der «Availability» Wissensdatenbank brauchten wir mehrere Iterationen sowie Usability Tests. Es wurde uns bewusst, dass man nicht alle möglichen Fragestellungen schon bei den ersten Paar Iterationen herausfinden konnte. Dazu müsste man definitiv eine umfangreiche Testphase aufgleisen, bei der die unbeantworteten Fragen analysiert werden.
Für eine Wissensdatenbank mit rund 20 Frage- und Antwortsätzen hatten wir rund 13 Trainingsstunden. In diesen 13 Stunden wurde das QnA Maker sowie LUIS Modell erweitert und verfeinert. Hinzu kommt noch der Initialaufwand der Kategorisierung. Hierbei wurde der gesamte Fragenpool aufgeteilt nach Themen.

Die folgende Tabelle zeigt die aufgewendete Zeit für diese einzelne Wissensdatenbank

Tätigkeit	Zeitaufwand
Fragen ermitteln aus Quellen (FAQ, CCC Tool, etc.)	32h
Kategorisierung der Gesamtfragen	8h
Einzelner Trainingszyklus	N *2h

In unserem Fall wird «N» ersetzt durch «4». Der komplette Trainingsaufwand resultiert damit in 48h.

9. Mandantenfähigkeit und Nutzungsgrad

Dieses Kapitel widmet sich der Frage, ob diese Chatbot Implementation mandantenfähig ist. Das bedeutet, ob es möglich ist auf der Azure Instanz mehrere Chatbot Services unabhängig zu verwalten. Es stellt sich die Frage

Der Begriff ist wie folgt beschrieben: «*Die Mandantenfähigkeit beschreibt eine IT-Infrastruktur, die unterschiedliche Kunden beziehungsweise Auftraggeber auf demselben Server- oder Softwaresystem bedienen kann, ohne dass die Beteiligten die Daten und Informationen der anderen einsehen können.*» [45]

Damit man von einer Mandantenfähigkeit sprechen kann müssen folgende Punkte erfüllt sein:

- **Datenschutz** gegenüber den anderen Mandanten. Es darf keine Einsicht auf die Daten der anderen Teilnehmer haben.
- **Zentrale Verwaltung** und Installation der Software. Die allgemeine Wartung und mandantenunabhängige Konfiguration müssen möglich sein.
- **Customizing** der jeweiligen Chatbots muss möglich sein. Unterschiedliche Wissensdatenbanken im Hintergrund.

9.1 Mehrere Chatbots hosten

Mit der derzeitigen Lösung ist es möglich mehrere unterschiedliche Chatbots auf Azure zu hosten. Der Code wurde so konzipiert, dass lediglich die Wissensdatenbank sowie die Azure Table Storage ausgetauscht werden muss. In Azure erstellt man in einem ersten Schritt eine neue Ressourcen Gruppe. Jede Ressourcen Gruppe verwaltet die einzelnen Chatbot App Services.
Die Wissensdatenbank vom QnA Maker muss natürlich ersetzt werden mit den eigenen Fragen des Energieversorgers. Das LUIS Modell kann jedoch beibehalten respektive erweitert werden mit eigenen Intents. Lediglich ein eigener Service muss erstellt werden.
Damit ist sichergestellt, dass die jeweiligen Energieversorger keinen Einfluss auf die anderen Teilnehmer haben. Damit ein weiterer Energieversorger hinzugefügt werden kann, müssen folgende Schritte durchgeführt werden:

- Ressourcen Gruppe auf Azure bereitstellen
- In dieser neuen Gruppe die Azure Services (App Service, Web App Bot, App Service Plan, Search Service und Storage Account) bereitstellen
- Bei der Storage Account Ressource eine Azure Table Storage hinzufügen.
- QnA Maker Wissensdatenbanken registrieren und erstellen.
- Den vorhandenen LUIS Service kopieren oder exportieren und dem neuen Energieversorger zur Verfügung stellen.
- Im «web.config»-File die neuen QnA Maker und LUIS Endpunkte hinterlegen.

9.2 Nutzungsgrad

Der Nutzungsgrad eines Chatbots lässt sich zurzeit nur schwierig beziffern. Laut der «D21-Digital-Index [46]» Studie hatten erst rund 50% der Befragten mit einem Chatbot kontakt. Das allgemeine Interesse hat sich in den letzten Jahren gesteigert und stagniert seit Anfang 2018.
Die nachfolgende Grafik zeigt das Suchverhalten auf Google beim Begriff «Chatbot».

Abbildung 59: Google Trend "Chatbot"

Die Werte geben das Suchinteresse relativ zum höchsten Punkt im Diagramm für die ausgewählte Region im festgelegten Zeitraum an. Der Wert 100 steht für die höchste Beliebtheit dieses Suchbegriffs. Der Wert 50 bedeutet, dass der Begriff halb so beliebt ist und der Wert 0 bedeutet, dass für diesen Begriff nicht genügend Daten vorlagen.
Spannend ist das Interesse nach Kantonen zu analysieren. Keine grosse Überraschung ist, dass die städtischen Kantone ein grösseres Suchinteresse aufzeigen anstatt die ländlichen.

Abbildung 60: Suchinteresse nach Kantonen

9.3 Schlussfolgerung

Auf die Frage ob der Chatbot nun Mandantenfähig ist oder nicht, geben wir grünes Licht. Azure bietet genug Möglichkeiten an, um die Ressourcengruppen sauber zu trennen. Die Konfiguration ist weder Chatbot noch Channel abhängig. Die QnA Maker Knowledge Base sowie das LUIS müssen teilweise angepasst werden, je nach Anwendungsfall des Energieversorgers.
Die Dialogoptionen lassen sich bequem austauschen in der Azure Table Storage Datenbank. Damit können spezifische Menüoptionen erstellt werden, ohne Programmierkenntnisse. Schliesslich ist auch das Erstellen und pflegen der Knowledge Base unabhängig vom Chatbot und kann im Hintergrund ausgetauscht werden.

10. Business Case

Als Geschäftsmodell durchleuchten wir welchen Impact Chatbots auf Unternehmen haben. Dabei wird der effektive Nutzen, anfallende Kosten und mögliche Risiken bewertet. Zusätzlich möchten wir die Profitabilität eines Chatbots abschätzen.

10.1 Geschäftsnutzen

Der ewz Chatbot soll den Zugang zur Supportdienstleistung erleichtern. Eine intuitive und selbsterklärende Bedienung ersetzt den Suchvorgang auf der Homepage. Dabei coacht sich der Benutzer selbst während des Vorgangs.
Als weiteres Merkmal ist die verkürzte Durchlaufzeit. Chatbots sind immer verfügbar und antworten innert Sekunden. Die Verzögerung oder Warteschleife bei einem Telefonanruf entfällt.
Zudem punkten Chatbots in Bezug auf die soziale Sicherheit. Der Benutzer ist anonym unterwegs und muss sich nicht einloggen. Viele Benutzer schätzen diese Anonymität im Zeitalter der Datensammlung.
Erwähnenswert ist, dass Benutzer häufiger Apps nutzen mit Messaging Funktionen. Somit sind sie routiniert und ein Chatbot ist nicht vollkommen fremd. Die Nutzungshürde ist damit geringer, die Chatbots fügen sich wunderbar in die Alltagsroutinen der Benutzer ein.
Abschliessend lassen sich durch die unbeantworteten Fragen ein bisher unbekanntes Geschäftsfeld entdecken: Bedürfnisse der Kunden. Diese Daten sind höchst wertvoll, um den Fragekatalog zu erweitern oder aktuelle Problemstellungen zu erkennen.

10.2 Wirtschaftlichkeit und Nutzen

Die meisten Chatbots zielen auf den Anwendungsfall die Kosten im Service zu senken. Die Telefonkosten im Kundenservice lassen sich – Studien zufolge – um bis zu 43% senken [43].

Die Folgefrage ist damit: «Wie gross ist die Akzeptanz von Chatbots?». Statistiken zeigen, dass eine Rückmeldung einer Anfrage im Durchschnitt bis zu zehn Stunden dauern kann. Oftmals eine Wartezeit, die Kunden ungern hinnehmen. Chatbots bieten hier den grossen Vorteil der 24/7 Verfügbarkeit. Unsere Generation ist aufgewachsen mit Google und somit bestens vertraut mit dem «Frage-Antwort» Mechanismus. Das Nutzungserlebnis ist somit intuitiv und die Abbruchquote deutlich geringer. Oftmals hat man keine Möglichkeit ein Telefongespräch zu führen oder ein vollständiges E-Mail zu schreiben. Für solche Anwendungsfälle ist der Chatbot optimal, da es kein spezielles Setup braucht.

10.3 Risiken von Chatbots

Chatbots sind nur so gut wie deren Entwickler und die Knowledge Base im Hintergrund. Daher möchten wir an dieser Stelle einige Punkte erwähnen, welche nicht vernachlässigt werden dürfen.

- Chatbots können die natürliche Sprache nur zu einem gewissen Grad verstehen. Schlagwörter oder Tags sind zwingend notwendig.
- Uncanny Valley beschreibt eine Akzeptanzlücke, wenn ein System versucht mittels künstlicher Intelligenz einen Menschen zu verkörpern.
- Der Themenbereich eines Chatbots ist limitiert. Fragen aus einem vollkommen anderen Kontext können nicht beantwortet werden.

Chatbotsmagazine.com hat eine umfassende Umfrage im März 2017 lanciert rund um das Thema Chatbot.
Auf die Frage weshalb ein Benutzer keinen Chatbot nutzen möchte kam folgendes Ergebnis zustande:

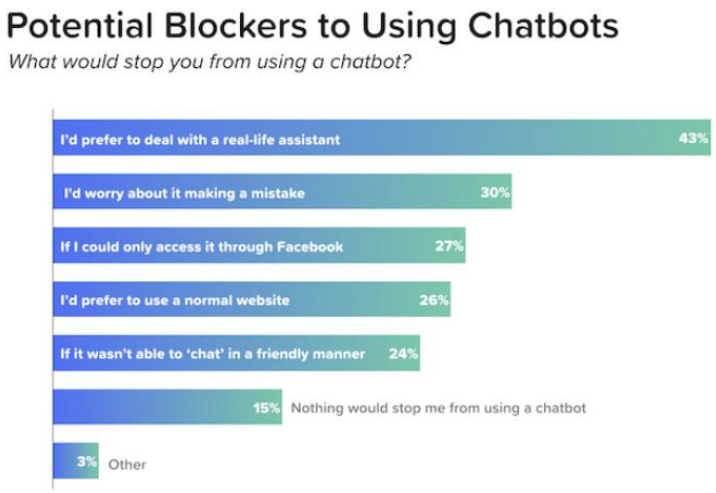


Abbildung 61: Umfrageergebnis von chatbotsmagazine.com

Aus dem Diagramm lässt sich herauslesen, dass knapp die Hälfte der Befragten einen echten Service Desk Mitarbeiter bevorzugen. Dieses Bild zeigt auf, dass ein grosser Anteil der Benutzer noch kritisch den Chatbots gegenübersteht.

10.4 Return-on-Investment (ROI)

Knapp mehr als die Hälfte der Unternehmen, die Chatbots im Einsatz haben, sehen einen positiven Effekt auf den ROI. Mit einer Beispielrechnung möchten wir zeigen, welche Kosteneinsparungen im Bereich Kundensupport möglich sind.

Wir nehmen an, dass drei Supportmitarbeiter zu je acht Stunden arbeiten. Der Kundenkontakt schätzen wir auf fünf Minuten mit einer Nachbearbeitungszeit von zehn Minuten. Aus den uns bekannten Zahlen, landen rund 4700 Anfragen pro Jahr beim 1st Level Support. Weitere 2700 Anfragen werden vom 2nd Level Support bearbeitet.
Somit werden im Durchschnitt täglich 29 Anfragen beantwortet. Angenommen jeder Anruf, E-Mail-Kontakt, Brief etc. kostet für das Unternehmen 16.50 CHF (Kosten für Infrastruktur, Schulung, Lohn etc.).
Somit entstehen tägliche Kosten von 478.50 CHF für die Beantwortung der täglichen Anfragen. Auf das ganze Jahr hinausgerechnet gibt das einen Betrag von 478.50 CHF * 252 Tage (Feiertage, Ferien, Wochenenden bereits abgezogen) = 120’582 CHF.
Der Chatbot soll nun 50% der Anfragen im vorab klären können. Die Kosten für den Support würden somit halbiert werden auf 60'291 CHF. Dem gegenüber stehen die Kosten für die Implementierung und Betrieb eines Chatbots.
Aus den vorherigen Kapiteln wurden die monatlichen Kosten für Azure und die Produktpalette berechnet. Rund 188 CHF pro Monat kostet der Betrieb der Azure Infrastruktur. Das sind im Jahr 2’256 CHF. Dazu kommen noch die Wartung, Instanthaltung und Aktualisierung der Wissensdatenbank. Wir rechnen mit vier Arbeitstagen pro Monat (1'400 CHF) für den administrativen Aufwand. Hinzu kommen noch drei einmalige Schulungstage für die involvierten Personen.
Je nach Komplexionsgrad variieren die Implementierungskosten. Mit den verfügbaren Prototypen dieser Bachelorarbeit ist das Fundament bereits gelegt und kann wiederverwendet werden.
Der ROI (Gewinn/Eingesetztes Kapital = ROI) Faktor wäre demnach 0.92.

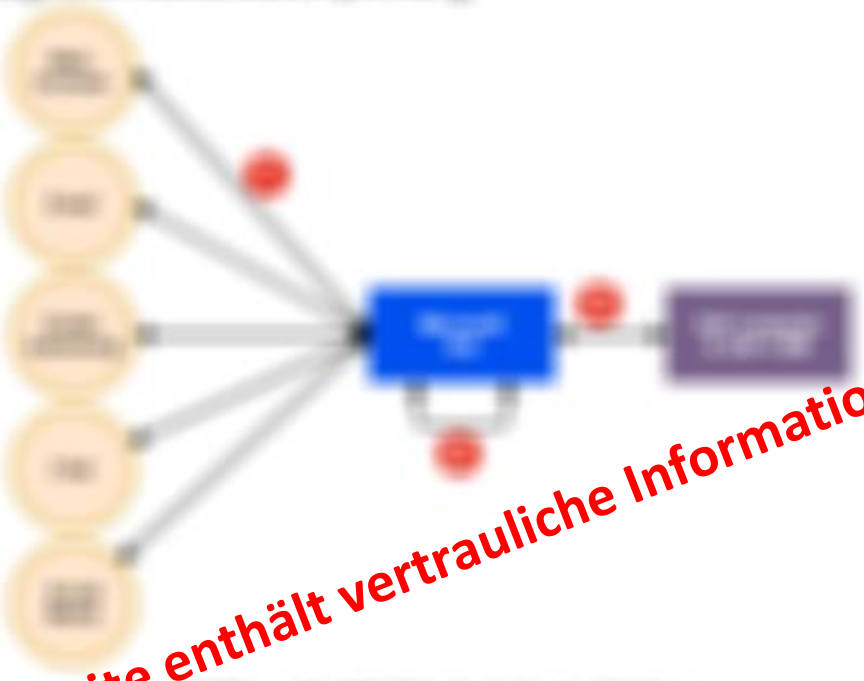
$$ROI = \frac{Einsparung}{Eingesetztes\ Kapital} = \frac{60'291}{65'106} = 0.92$$

Beschreibung	Ohne Chatbot	Mit Chatbot
Anzahl Anfragen pro Tag	29	29
Bearbeitungskosten pro Anfrage	16.50.-	Sind gedeckt in den Betriebskosten von Azure
Betriebskosten pro Jahr	(29 * 66.-) * 252 Tage = 120’582.-	188.- * 12 Monate = 2’256.-
Implementationskosten	0.-	2 Entwickler * 7'500 = 15’000.- Aufwandschätzung: 3 Monate Total: 45’000.-
Einmalige Schulungskosten Chatbot	0.-	3 * 350 = 1050.-
Wartungskosten Chatbot pro Jahr	0.-	1400.- * 12 = 16'800.-
Totale Kosten	**120'582.-**	**65'106.-**

Tabelle 32: Beispielrechnung ROI

Die Berechnung zeigt, dass der Einsatz eines Chatbots finanziell interessant ist. Es hängt jedoch stark von der Pflege sowie der Implementierung ab. Bereits ab dem zweiten Jahr rentiert sich der Einsatz eines Chatbots.


Diese Seite enthält vertrauliche Informationen



12. Ergebnis

In diesem Kapitel erläutern wir die Erkenntnisse, die wir in dieser Arbeit gewonnen haben. Zudem blicken wir zurück auf den Projektverlauf und reflektieren, was gut funktioniert hatte und wo es Schwierigkeiten gab. Schliesslich listen wir alle Anforderungen auf, welche umgesetzt wurden und welche noch offen sind.

12.1 Schlussfolgerung

Aus dieser Bachelorarbeit schliessen wir darauf, dass Chatbots sehr wohl geeignet sind, um Kundenanfragen automatisch zu beantworten. Jedoch mussten wir auch feststellen, dass dies nur zu einem gewissen Grad möglich ist. Sobald es um kundenspezifische Anfragen handelt, wird es schwierig. Dies liegt daran, dass Chatbots noch nicht ausgereift sind, um Abhängigkeiten zwischen einzelnen Nachrichten zu erkennen. Die komplexen Vorgänge, die ein menschlicher Mitarbeiter macht bei einer Konversation, kann ein Chatbot bisher nicht erschliessen. Wir wagen jedoch zu behaupten, dass in den kommenden Jahren, mittels Deep und Machine Learning, diese Baustellen grösstenteils behoben sein werden.

Rückblickend auf die Bachelorarbeit, sind wir der Meinung, dass wir eine sorgfältige Herangehensweise gewählt haben, um die Anforderungen umzusetzen. Mittels einer ausführlichen Evaluation der Bot Frameworks und dem ständigen Kontakt zur ewz, konnten wir die funktionalen sowie Nicht funktionalen Anforderungen erfassen. Diese konnten auch vollumfänglich umgesetzt werden. Eine der grössten Herausforderungen war, die Umstellung des Konversationsflusses. Durch den Entscheid einen «Chatbot-Driven» Ansatz zu wählen, hat sich auch die Software Architektur geändert. Glücklicherweise haben wir von Anfang an mehrere Prototypen geplant, so hat die Umstellung unsere Projektplanung nicht komplett über den Haufen geworfen.

Schlussendlich sind wir zufrieden mit dem Verlauf der Bachelorarbeit. Die gewonnenen Erkenntnisse aus der Studienarbeit haben sicherlich dazu beigetragen. Der Projektplan konnte mehrheitlich eingehalten und die Kernfunktionalitäten umgesetzt werden.

12.2 Was erreicht wurde

Folgende Punkte und Funktionen konnten wir in der Bachelorarbeit umsetzen bzw. vorbereiten

- Business Case
- Architektur- und Technologieempfehlung für das Programm CUSOLL
- Beurteilung des Nutzungsgrades auf Basis der ewz Telecom
- Funktionaler Chatbot, basierend auf den FAQ der ewz Telecom
- Chatbot-Driven gesteuerter Konversationsfluss
- Dynamisch anpassbare Hierarchieebene der Dialogmenüs
- Absichten- und Stimmungserkennung der Kundennachricht
- Datenschutz; Namen sowie E-Mailadressen werden anonymisiert in der Azure Table Storage persistiert
- Nichtbeantwortete Fragen können nachträglich in der Datenbank herausgelesen werden
- Direkter Telefonanruf zur ewz Zentrale bei Supportanfrage
- Kompletter Chatverlauf kann dem Support via E-Mail zugeschickt werden
- Verfügbarkeitsanfragen werden direkt im Chat beantwortet
- Bequemes anpassen und erweitern der QnA Maker Wissensdatenbanken
- Leicht zugängliches anpassen und erweitern der LUIS Logik
- Persönliche Willkommens- und Abschiedsnachricht

12.3 Was nicht erreicht wurde

Folgende Punkte und Funktionen konnten wir in der Bachelorarbeit nicht umsetzen

- Kontext behalten während der Konversation
- Deaktivieren von vergangenen Dialogmenüs
- Training der QnA Maker Knowledge Bases auf produktivem Niveau

13. Ausblick

In diesem Kapitel möchten wir mögliche Zusatzerweiterungen auflisten sowie potentielle Chancen erläutern.

13.1 Mögliche Erweiterungen

13.1.1 Kontext behalten während einer Konversation

Wie schon im Kapitel 7.4.1 erläutert, wäre es eine sinnvolle Ergänzung um den Chatbot noch «menschlicher» erscheinen zu lassen. Dabei kann sich der Bot an vergangene Nachrichten erinnern und sich darauf beziehen. Zu einem gewissen Grad kann der derzeitige Chatbot dies bereits, z.B. sich den persönlichen Namen merken. Dieses Feature geht aber darüber hinaus und könnte aus dem Kontext präzisere Antworten geben.

13.1.2 Rollout-Datenbank anbinden

Dieses Feature wurde bei der Zwischenpräsentation als «Won't» priorisiert, aus dem Grund da den anderen Features eine höhere Priorität gegeben wurde. Trotzdem wäre es sinnvoll in Zukunft, die Rollout-Datenbank der ewz einzubinden. Damit könnte der Chatbot auf kundenspezifische Daten zugreifen, die im generischen QnA Maker nicht vorhanden sindund damit genauere Antworten liefern oder bisher nicht beantwortbare Fragen wie «Wie hoch war meine Monatsrechung im September 2018?» erledigen.

13.1.3 Zusätzliche Sprachen mit DeepL

In der bisherigen Lösung sind sämtliche Frage- und Antwortsätze auf Deutsch. Das bedeutet, andere Sprachen werden nicht unterstützt. Damit trotzdem Eingaben auf Englisch, Französisch, Italienisch oder Spanisch verstanden werden, könnte man die Translator API von DeepL nutzen. Diese übersetzt zuverlässig gängige Sprachen in die Zielsprache. Damit müsste man nur eine QnA Maker Datenbank pflegen anstatt für jede Sprache eine separate.

13.1.4 Konversation an einen Supportmitarbeiter übergeben

Während einer Konversation kann es vorkommen, dass es sinnvoll ist den Chatbot durch einen menschlichen Supportmitarbeiter zu ersetzen. Der Mitarbeiter tritt an die Stelle des Chatbots und kann direkt mit dem Kunden chatten. Bei einer Anfrage des Kunden erhält das Supportteam live ein Ticket, welches jemand akzeptieren oder ablehnen kann. Wenn das Ticket akzeptiert wird, übernimmt der Mitarbeiter den Chat und es wird dem Kunden eine Meldung angezeigt, dass er sich nun mit einem Supportmitarbeiter in Kontakt befindet.

14. Anhänge

Anhang A: Persönliche Berichte

Persönlicher Bericht von Felix

Als ich vom Thema Chatbot von Stefan Richter erfahren habe, hatte ich etwas Bedenken gehabt dieses als Thema meiner Bachelorarbeit zu wählen, da ich zu wenig über dieses Gebiet weiss. Aus verschiedenen Medien habe ich oft über die Chatbot im Zusammenhang mit Künstliche Intelligenz und Maschinelles Lernen gehört. Um dieses Themengebiet besser zu verstehen und meine Befürchtungen zu überwinden habe ich mich entschlossen dieses Thema für meine Bachelorarbeit zu nehmen. Rückblickend denke ich, es war eine richtige Entscheidung.
Bei der Studienarbeit haben wir auf einem existierenden Produkt weiterentwickelt und bei der Bachelorarbeit haben wir die Möglichkeit gehabt, ein Produkt von Null auf zu entwickeln. Dadurch haben wir gesehen, dass als Entwickler mehr Spielraum besteht als wenn man auf einem bestehenden Produkt basiert.
Oftmals haben die Software Entwickler lieber, wenn sie ihre Produkte entwickeln können ohne über die Infrastruktur viel nachzudenken. Mit der Entwicklung von Public Cloud-Diensten werden diese Wünsche von Entwicklern erfüllt. Wir haben auch unseren Chatbot über die Microsoft Azure Plattform eingesetzt und gesehen wie leicht es ist, so ein System zu verwenden. Diese Erfahrung wird sicherlich vorteilhaft für unser Berufsleben sein, da viele Unternehmen ihre Applikationen in eine Public Cloud bringen.
Diese Bachelorarbeit hat mir einen guten Einblick in die Welt von Microsoft sowie ihren Produkten wie Chatbot, Public Cloud, Maschinelles Lernen gegeben. Der Chatbot ist ein sehr aktuelles Thema und viele Unternehmen versuchen Ihren eigenen Teil dieses Kuchens zu bekommen. Ich bin sicher, falls wir dieses Thema noch in einem Jahr durchführen, würden die Ergebnisse anderes aussehen, da das Thema Chatbot sehr dynamisch ist und viele neue Entwicklungen zurzeit noch stattfinden, welches die Chatbot Landschaft massiv verändern wird. Der Chatbot ist hier gekommen um zu bleiben, er wird in Zukunft eine sehr wichtige Rolle einnehmen, da viel repetitive Arbeit automatisiert werden kann.
Die Zusammenarbeit mit Christian, Stefan Richter und ewz hat sehr gut funktioniert. Ich bin Stefan Richter sehr dankbar, da er stets den richtigen Weg gezeigt hat. Eine wichtige Lehre, die ich aus dieser Bachelorarbeit gezogen habe, ist das es keine schnelle Implementation gibt. Sondern es muss sehr gut geplant vorgegangen werden. Denn Qualität ist wichtiger als die Quantität.

Persönlicher Bericht von Christian

Die Partnersuche für die Bachelorarbeit war für mich kein Thema, da ich mich mit Felix seit der Studienarbeit hervorragend verstehe. Mir persönlich hat die Bachelorarbeit grosse Freude bereitet, da ich mich in ein Thema einarbeiten durfte, welches mich ohnehin interessiert hat. In der Zwischenzeit konnte ich vieles über das Thema Machine Learning und Chatbot dazulernen.

Aus den gesammelten Erfahrungen aus der Studienarbeit, funktionierte die Zeiteinteilung besser. Ich bin der Meinung, dass wir die Arbeitspakete besser einschätzen konnten und so auch das Zeitmanagement besser im Griff hatten. Die gewählte Vorgehensweise, sich zuerst über das Thema und deren Frameworks zu informieren, war ein notwendiger Schritt. Nur so konnten wir die Prototypen erfolgreich implementieren.

Die Zusammenarbeit mit der ewz sowie den Beteiligten der HSR, hat aus meiner Sicht sehr gut funktioniert. Von Stefan Richter erhielten wir stets nützliche Ratschläge für Verbesserungen. Dieses Feedback schätzte ich sehr, da ich mich als Entwickler oftmals in den Details verloren habe. Auch die Zusammenarbeit mit den Beteiligten der ewz, hat wunderbar funktioniert. Schlussendlich hatte ich das Gefühl, dass unsere Arbeit wertgeschätzt wird und wir ein brauchbares Produkt abliefern konnten.

Auf die Frage, ob ich das Projekt nochmals durchführen würde, wäre meine Antwort ein klares «Ja». Dies hat auch damit zu tun, dass ich mehr und mehr von meinem erworbenen Wissen in der Praxis umsetzen konnte. Die Kombination aus Programmieren, Planen, neue Anforderungen evaluieren und schliesslich die beste Lösung zu implementieren, gefiel mir sehr gut. Abschliessend bin ich froh jede einzelne Erfahrung in dieser Bachelorarbeit gemacht zu haben, auch wenn nicht alle zu meinen Lieblingstätigkeiten zählen.

Anhang B: Projektgrösse

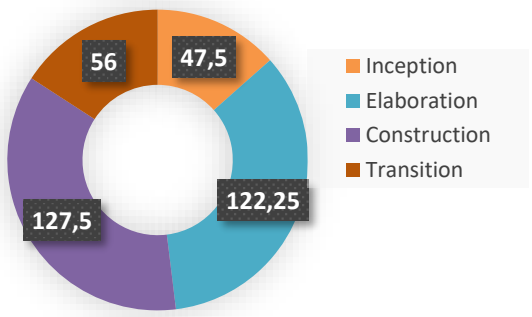
B.1 Codestatistiken

Beschreibung	Anzahl
Codezeilen	1744
Codezeilen (ohne Kommentare und Unit Tests)	1322
Klassen	25
Packages	8
Maintainability Index[43] (max 100)	81
Commits	141

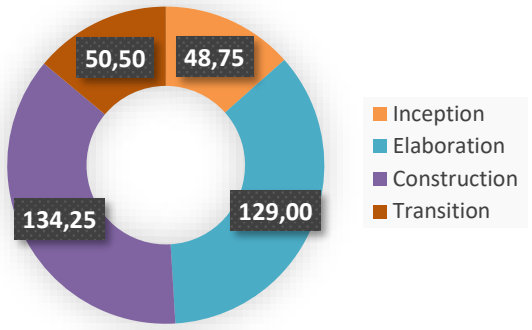
[43] Index der beschreibt wie Wartbar die Software ist. Ein Wert gegen 100 sollte angestrebt werden.

Anhang C: Zeitmanagement

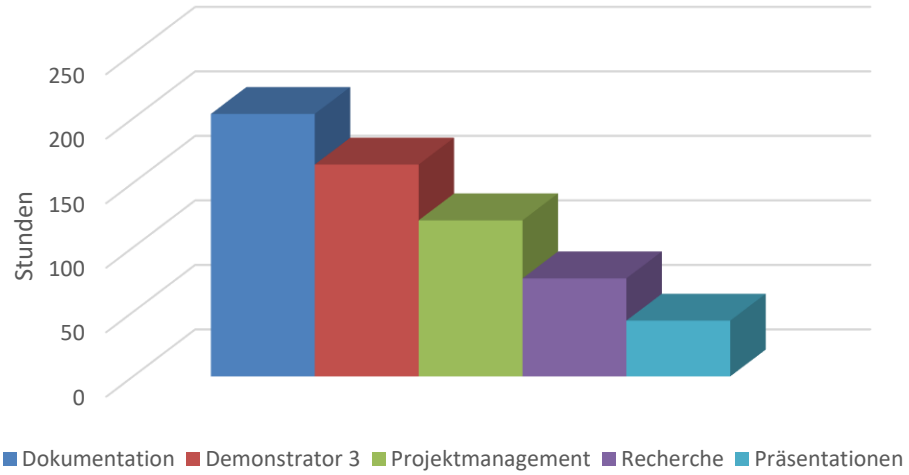
C.1 Aufgewendete Zeit (h) nach Meilensteinen – Christian Lindauer



C.2 Aufgewendete Zeit (h) nach Meilensteinen – Felix Enchiparamban



C.3 Top 5 Issues



Anhang D: Glossar

#	Abkürzung	Begriff	Beschreibung
A	API	Application Programming Interface	Softwaresystem über welches ein System mit anderen kommunizieren können
	AWS	Amazon Web Services	Cloud Computing Anbieter
C	CI	Continuous Integration	Fortlaufendes Zusammenfügen von Softwarekomponenten
	CUSOLL	Customer Solutions	Programm zur Förderung der Kundenzufriedenheit
	CWC	Customer Welcome Center	Anlaufstelle für Kunden
	CCC	Customer Care Center	Anlaufstelle für Kunden mit Anfragen
F	FAQ	Frequently Asked Questions	Häufig gestellte Fragen
	FTTH	Fiber to the Home	Fernmeldenetz, das über Glasfaser geführt wird
G	GUI	Graphical User Interface	Grafische Benutzeroberfläche
	GCP	Google Cloud Platform	Google's eigener Cloud Computing Service
	GPS	Global Positioning System	Globales Navigationssatellitensystem zur Positionsbestimmung
I	ISP	Internet Service Provider	Internetdienstanbieter
	LoC	Lines of Code	Codezeilen in einem Software Programm
	IoC	Inversion of Control	Steuerung und Ausführung von Unterprogrammen wird an das Framework abgegeben.
L	LUIS	Language Understanding Intelligent Service	Machine Learning basierter Service um natürliche Sprachen zu erkennen
N	NFA	Nichtfunktionale Anforderungen	
	NLP	Natural Language Processing	Methode zur maschinellen Verarbeitung der natürlichen Sprache
	NLU	Natual Language Understanding	Strukturiert natürliche Sprache in maschinenlesbaren Syntax
R	ROI	Return-of-Investment	Betriebswirtschaftliche Kennzahl zur Messung der Rendite
S	SQL	Structured Query Language	Datenbanksprache zur Definition von Datenstrukturen
X	XTRO	-	Abteilung die sich um 2nd Level Themen «FTTH Verträge» und Rollout kümmert
	XTOC	-	Abteilung die sich um kundenspezifische 2nd Level Themen kümmert

Anhang E: Eigenständigkeitserklärung

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Eigenständigkeitserklärung

Erklärung

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.
- das ich keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt habe.

Ort, Datum: Rapperswil, 12.12.18

Name, Unterschrift:

Christian Lindauer, 

Felix Enchiparamban, 

Anhang F: Einverständniserklärung

Einverständniserklärung Publikation auf eprints.hsr.ch

○ SA

⊗ BA

Titel der Arbeit: Chatbot für das Customer Care Center von ewz

Team: Christian Lindauer, Felix Enchiparamban

Betreuer: Prof. Stefan Richter

Wir sind mit der Publikation unserer Arbeit auf eprints.hsr.ch einverstanden, sofern für diese Arbeit keine Geheimhaltungsvereinbarung unterzeichnet wurde.

Nach Bekanntgabe der Note haben wir die Möglichkeit innert 14 Tagen Einsprache zu erheben und das Einverständnis zur Publikation der Arbeit auf eprints.hsr.ch zurückzuziehen. In diesem Falle wird nur der Abstract publiziert.

Rapperswil, 13.12.2018

Anhang G: Urheber-Vereinbarung

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

Vereinbarung

1. Gegenstand der Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Bachelorarbeit von Christian Lindauer und Felix Enchiparamban unter der Betreuung von Stefan Richter geregelt.

2. Urheberrecht

Die Urheberrechte stehen der Studentin / dem Studenten zu.

3. Verwendung

Der Student/die Studentin erklärt hiermit von der Vereinbarung zwischen der HSR und Elektrizitätswerke der Stadt Zürich deren Kopie dieser Vereinbarung beigelegt ist, Kenntnis genommen zu haben und stimmt den Bedingungen, welche die Verwendung der Ergebnisse der Arbeit regelt, zu.

Beilagen:

Technischer Bericht
Projektplan
Programmcode
Sitzungsprotokolle
Dokumentation

Rapperswil, den 13.12.18 .....

Rapperswil, den 12.12.18 .....

Rapperswil, den 12.12.18 .....

1

Anhang H: Literaturverzeichnis

[1] E. Kaya, «Quora,» 1 3 2017. [Online]. Available: https://www.quora.com/What-is-the-best-way-to-teach-your-chat-bot-AI. [Zugriff am 1 10 2018].

[2] «Pareto Prinzip,» [Online]. Available: https://www.pareto-prinzip.net/. [Zugriff am 1 10 2018].

[3] «Arvato.com,» [Online]. Available: https://cloud-blog.arvato.com/die-arten-von-chatbots/. [Zugriff am 24 9 2018].

[4] «Quora,» [Online]. Available: https://www.quora.com/What-is-the-typical-architecture-of-an-AI-chatbot. [Zugriff am 25 9 2018].

[5] «Chatbotmagazine,» [Online]. Available: https://chatbotsmagazine.com/understanding-the-need-for-nlp-in-your-chatbot-78ef2651de84. [Zugriff am 26 9 2018].

[6] «Chatbotslife.com,» [Online]. Available: https://chatbotslife.com/conversational-ui-writing-chatbot-scripts-step-by-step-a78b611a5eba. [Zugriff am 27 9 2018].

[7] C. Villar, «Medium.com,» 12 12 2017. [Online]. Available: https://medium.com/landbot-io/creating-conversational-experiences-ii-build-and-design-20ac88d7ee72. [Zugriff am 28 9 2018].

[8] EWZ, «FTTH Präsentation,» Quellenordner Bachelorarbeit, 2018.

[9] N. R. a. M. Benton, «Quality Measurements for Chatbots,» Quellenordner der Bachelorarbeit, 2018.

[10] E. Seo, «chatbotmagazine.com,» 20 1 2017. [Online]. Available: https://chatbotsmagazine.com/19-best-practices-for-building-chatbots-3c46274501b2. [Zugriff am 2 10 2018].

[11] Maruti Techlabs Pvt. Ltd., [Online]. Available: https://www.marutitech.com/complete-guide-bot-frameworks/. [Zugriff am 27 09 2018].

[12] «https://azure.microsoft.com,» Microsoft Inc., [Online]. Available: https://azure.microsoft.com/en-us/services/bot-service/. [Zugriff am 01 10 2018].

[13] «https://customers.microsoft.com,» Microsoft Inc., [Online]. Available: https://customers.microsoft.com/en-us/search?sq=%22Azure%20Bot%20Service%20%28AI%29%22&ff=&p=2&so=story_publish_date%20desc. [Zugriff am 01 10 2018].

[14] A. B. S. a. L. U. Team, «https://azure.microsoft.com,» Microsoft Inc., 13 12 2017. [Online]. Available: https://azure.microsoft.com/en-us/blog/conversational-bots-deep-dive-what-s-new-with-the-general-availability-of-azure-bot-service-and-language-understanding/. [Zugriff am 01 10 2018].

[15] C. Campbell, «https://theflyingmaverick.com,» 13 03 2018. [Online]. Available: https://theflyingmaverick.com/2018/03/13/give-your-solutions-a-more-human-side-with-microsoft-cognitive-services/. [Zugriff am 27 09 2018].

[16] «Microsoft Cognitive Services,» [Online]. Available: https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/overview/overview. [Zugriff am 26 9 2018].

[17] «zuerinet.ch,» ewz AG, [Online]. Available: https://zuerinet.ch/faq. [Zugriff am 01 10 2018].

[18] «https://docs.microsoft.com,» Microsoft Inc., 12 09 2018. [Online]. Available: https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/how-to/language-knowledge-base. [Zugriff am 01 10 2018].

[19] «https://azure.microsoft.com,» Microsoft Inc., [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/app-service/windows/. [Zugriff am 01 10 2018].

[20] «https://azure.microsoft.com,» Microsoft Inc., [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/search/. [Zugriff am 01 10 2018].

[21] «https://azure.microsoft.com,» Microsoft Inc., [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/monitor/. [Zugriff am 02 10 2018].

[22] D. Berry, «Microsoft Docs,» 12 9 2018. [Online]. Available: https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/concepts/confidence-score. [Zugriff am 1 10 2018].

[23] D. Berry, «https://docs.microsoft.com,» Microsoft Inc., 15 08 2018. [Online]. Available: https://docs.microsoft.com/en-us/azure/cognitive-services/luis/what-is-luis. [Zugriff am 03 10 2018].

[24] «https://azure.microsoft.com,» Microsoft Inc., [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/cognitive-services/language-understanding-intelligent-services/. [Zugriff am 03 10 2018].

[25] Botpress Inc., [Online]. Available: https://botpress.io/docs/latest/getting_started/. [Zugriff am 27 09 2018].

[26] botpress Inc., [Online]. Available: https://drive.google.com/drive/folders/0B76CSS2MqCRmenE0WE9GTGF1bkU. [Zugriff am 27 09 2018].

[27] Google Inc., 10 08 2018. [Online]. Available: https://cloud.google.com/dialogflow-enterprise/docs/editions. [Zugriff am 27 09 2018].

[28] V. o. E. Scott Huffman, Google, 19 09 2016. [Online]. Available: https://developers.googleblog.com/2016/09/making-conversational-interfaces-easier-to-build.html. [Zugriff am 27 09 2018].

[29] Google Inc., [Online]. Available: https://dialogflow.com/docs/reference/language. [Zugriff am 27 09 2018].

[30] «https://cloud.google.com,» Google Inc., [Online]. Available: https://cloud.google.com/products/ai/. [Zugriff am 02 10 2018].

[31] «https://aws.amazon.com,» Amazon Web Services Inc., [Online]. Available: https://aws.amazon.com/lex/. [Zugriff am 04 10 2018].

[32] K. Costello, «https://www.gartner.com,» Gartner Inc., 01 08 2018. [Online]. Available: https://www.gartner.com/newsroom/id/3884500. [Zugriff am 04 10 2018].

[33] «https://aws.amazon.com,» Amazon Web Service Inc., [Online]. Available: https://aws.amazon.com/lex/. [Zugriff am 04 10 2018].

[34] «https://aws.amazon.com,» Amazon Web Service Inc., [Online]. Available: https://aws.amazon.com/de/dynamodb/pricing/. [Zugriff am 04 10 2018].

[35] C. Coles, «https://www.skyhighnetworks.com,» Skyhigh Networks, 30 07 2018. [Online]. Available: https://www.skyhighnetworks.com/cloud-security-blog/microsoft-azure-closes-iaas-adoption-gap-with-amazon-aws/. [Zugriff am 04 10 2018].

[36] «https://azure.microsoft.com,» Microsoft Inc., [Online]. Available: https://azure.microsoft.com/en-us/services/bot-service/. [Zugriff am 04 10 2018].

[37] «https://azure.microsoft.com,» Microsoft Inc., [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/cosmos-db/ . [Zugriff am 05 10 2018].

[38] G. Stafford, «https://programmaticponderings.com,» 11 08 2018. [Online]. Available: https://programmaticponderings.com/2018/08/11/building-serverless-actions-for-google-assistant-with-google-cloud-functions-cloud-datastore-cloud-storage/. [Zugriff am 04 10 2018].

[39] «https://cloud.google.com,» Google Inc., [Online]. Available: https://cloud.google.com/natural-language/pricing. [Zugriff am 05 10 2018].

[40] «https://cloud.google.com,» Google Inc., [Online]. Available: https://cloud.google.com/functions/pricing. [Zugriff am 05 10 2018].

[41] «https://cloud.google.com,» Google Inc., [Online]. Available: https://cloud.google.com/sql/pricing#1st-gen-pricing. [Zugriff am 05 10 2018].

[42] «mindtools.com,» [Online]. Available: https://www.mindtools.com/pages/article/newPPM_89.htm. [Zugriff am 27 11 2018].

[43] K. Hebenstreit, «Manymize,» 4 12 2017. [Online]. Available: https://www.manymize.com/chatbots-analyse#Was_bedeuten_Chatbots_fuer_die_Unternehmensstrategie. [Zugriff am 21 11 2018].

[44] B. Meyer, «CUSOLL Präsentation ewz,» 2018.

[45] «logistikknowhow.com,» 8 2014. [Online]. Available: https://logistikknowhow.com/informationssysteme/mandantenfaehigkeit/. [Zugriff am 11 27 2018].

[46] «initiative21,» [Online]. Available: https://initiatived21.de/publikationen/d21-digital-index-2017-2018/. [Zugriff am 29 11 2018].

[47] «Qnamaker.ai,» [Online]. Available: https://www.qnamaker.ai/old/Documentation/ActiveLearning. [Zugriff am 8 10 2018].

[48] «https://github.com,» github, 19 10 2017. [Online]. Available: https://github.com/Microsoft/BotBuilder/issues/3639. [Zugriff am 26 10 2018].

[49] L. Stott, «https://blogs.msdn.microsoft.com,» Microsoft Inc., 05 04 2016. [Online]. Available: https://blogs.msdn.microsoft.com/uk_faculty_connection/2016/04/05/what-is-microsoft-bot-framework-overview/. [Zugriff am 26 10 2018].

[50] G. Pretty, «https://www.garypretty.co.uk,» 21 02 2018. [Online]. Available: https://www.garypretty.co.uk/2018/02/21/whats-next-for-the-microsoft-bot-framework/. [Zugriff am 26 10 2018].

[51] R. Sahay, «https://www.c-sharpcorner.com,» 10 02 2017. [Online]. Available: https://www.c-sharpcorner.com/article/asp-net-core-middleware-in-a-nutshell/. [Zugriff am 26 10 2018].

[52] «Sendgrid,» [Online]. Available: https://sendgrid.com/. [Zugriff am 6 12 2018].

[53] «db-engines,» [Online]. Available: https://db-engines.com/en/system/Microsoft+Azure+Cosmos+DB%3BMicrosoft+Azure+Table+Storage . [Zugriff am 9 11 2018].

[54] «Azure Content Moderation,» [Online]. Available: https://azure.microsoft.com/en-us/services/cognitive-services/content-moderator/. [Zugriff am 9 11 2018].

[55] «Bing Spell Check,» [Online]. Available: https://azure.microsoft.com/en-us/services/cognitive-services/spell-check/. [Zugriff am 11 26 2018].

[56] «https://medium.com,» 13 04 2018. [Online]. Available: https://medium.com/@createdincode/simple-unit-testing-in-microsofts-c-bot-framework-with-nunit-and-moq-345b805ecd1b. [Zugriff am 01 12 2018].

[57] Davide, «https://www.davidezordan.net,» 08 05 2017. [Online]. Available: https://www.davidezordan.net/blog/?p=8119. [Zugriff am 26 10 2018].

[58] «https://github.com,» Github Inc., 04 02 2017. [Online]. Available: https://github.com/Microsoft/BotFramework-Emulator/issues/99. [Zugriff am 26 10 2018].

[59] M. Kearn, «Blogs Msdn,» 17 7 2018. [Online]. Available: https://blogs.msdn.microsoft.com/martinkearn/2018/07/17/bot-framework-v4-what-i-learnt-in-4-days-in-july-2018/. [Zugriff am 14 10 2018].

[60] «https://stackoverflow.com,» stackoverflow Inc., 29 06 2009. [Online]. Available: https://stackoverflow.com/questions/1056121/how-to-create-json-string-in-c-sharp. [Zugriff am 31 10 2018].

[61] «Stackoverflow,» 17 3 2017. [Online]. Available: https://stackoverflow.com/questions/42567165/botframework-to-disable-card-buttons-in-web-chat-under-bot-framework. [Zugriff am 7 12 2018].

[62] T. A. Donovan Brown, «msdn.com,» 21 03 2017. [Online]. Available: https://channel9.msdn.com/Series/DevOps-for-the-Bot-Framework/Testing-the-Bot-Framework. [Zugriff am 06 12 2018].

[63] xe.com Inc., [Online]. Available: https://www.xe.com/currencyconverter/convert/?Amount=0.002&From=USD&To=CHF. [Zugriff am 27 09 2018].

[64] C. Larman, «http://www.craiglarman.com,» [Online]. Available: http://www.craiglarman.com/wiki/downloads/applying_uml/larman-ch6-applying-evolutionary-use-cases.pdf. [Zugriff am 8 10 2018].

[65] «Azure Documentation,» [Online]. Available: https://azure.microsoft.com/en-us/services/cognitive-services/content-moderator/. [Zugriff am 8 11 2018].

Anhang I: Abbildungsverzeichnis

Anhang J: Tabellenverzeichnis