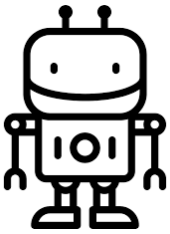
ADUNIS CHATBOT

AI Foundations: Miniprojekt



Fachhochschule OST, Studiengang Informatik

Autoren: Jan Huber, Yael Schärer
Abgabetermin: 29. Oktober 2021

Inhaltsverzeichnis

Einleitung

ADUNIS ist eine bestehende Schul-Administrations-Software, welche an der OST entwickelt wurde und gepflegt wird.[1] Über ADUNIS können die Studentinnen und Studenten vom Standort Rapperswil ihre Stundenpläne zusammenstellen, Informationen zum Studium abrufen oder Änderungen an ihrem Studienmodell beantragen. Das Tool wird als Webapplikation über den Browser gesteuert.

Ziel unserer Arbeit ist das Erstellen eines Chatbots für ADUNIS als Proof-of-Concept. Mit einem Bot lässt sich ein Grossteil der Anwendungsfälle des bestehenden Tools abdecken und vereinfachen. Für einzelne Funktionalitäten, wie das Zusammenstellen der Stundenpläne, ist die graphische Benutzeroberfläche jedoch besser geeignet. Unsere Applikation soll also als ergänzende ADUNIS-Oberfläche zum bereits bestehenden Tool funktionieren.

Der im Rahmen dieser Projektarbeit erstellte Chatbot bedient die folgenden Anwendungsfälle:

- Änderungen im Studium beantragen (Studiengangwechsel, Unterbruch, Exmatrikulation und Pausierung)
- Herunterladen von Semester- Leistungs- und Modulreporten
- Herunterladen von Stunden- und Prüfungsplänen
- Ändern persönlicher Daten wie Adresse, Telefon, etc.

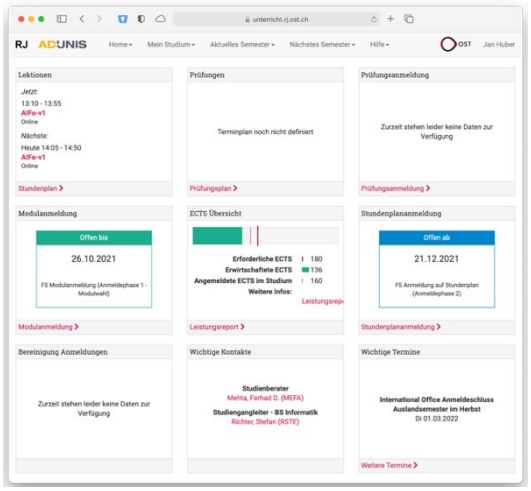


Abbildung 1: ADUNIS Weboberfläche

[1] https://www.adunis.ch/index.php?id=5064

Architekturübersicht

Die Grafik zeigt die Architektur inklusive der Umgebungssysteme. In den folgenden Kapiteln werden die einzelnen Komponenten genauer beschrieben.

- Der Chatbot arbeitet mit einer gemockten Version des ADUNIS Backends.
- Der Java Client kommuniziert über die Client Library von Dialog Flow mit dem darunterliegenden Model.
- Der Java Client wird über eine Weboberfläche angesteuert. Diese ist mit Hilfe des Java Spring Frameworks realisiert (Spring Web, Spring Boot, ...). Über die Weboberfläche interagiert der Student / die Studentin mit dem Chatbot und zusätzliche Informationen werden angezeigt. Auch ist eine Sprachausgabe der Roboterantworten implementiert.
- Über eine Webhook-Integration ist es alternativ möglich, über die App «Line Messenger» mit dem Chatbot zu kommunizieren.
- Über eine Google Cloud Function werden Emails versendet, um bei Änderungen am Studium den Studienberater zu informieren.

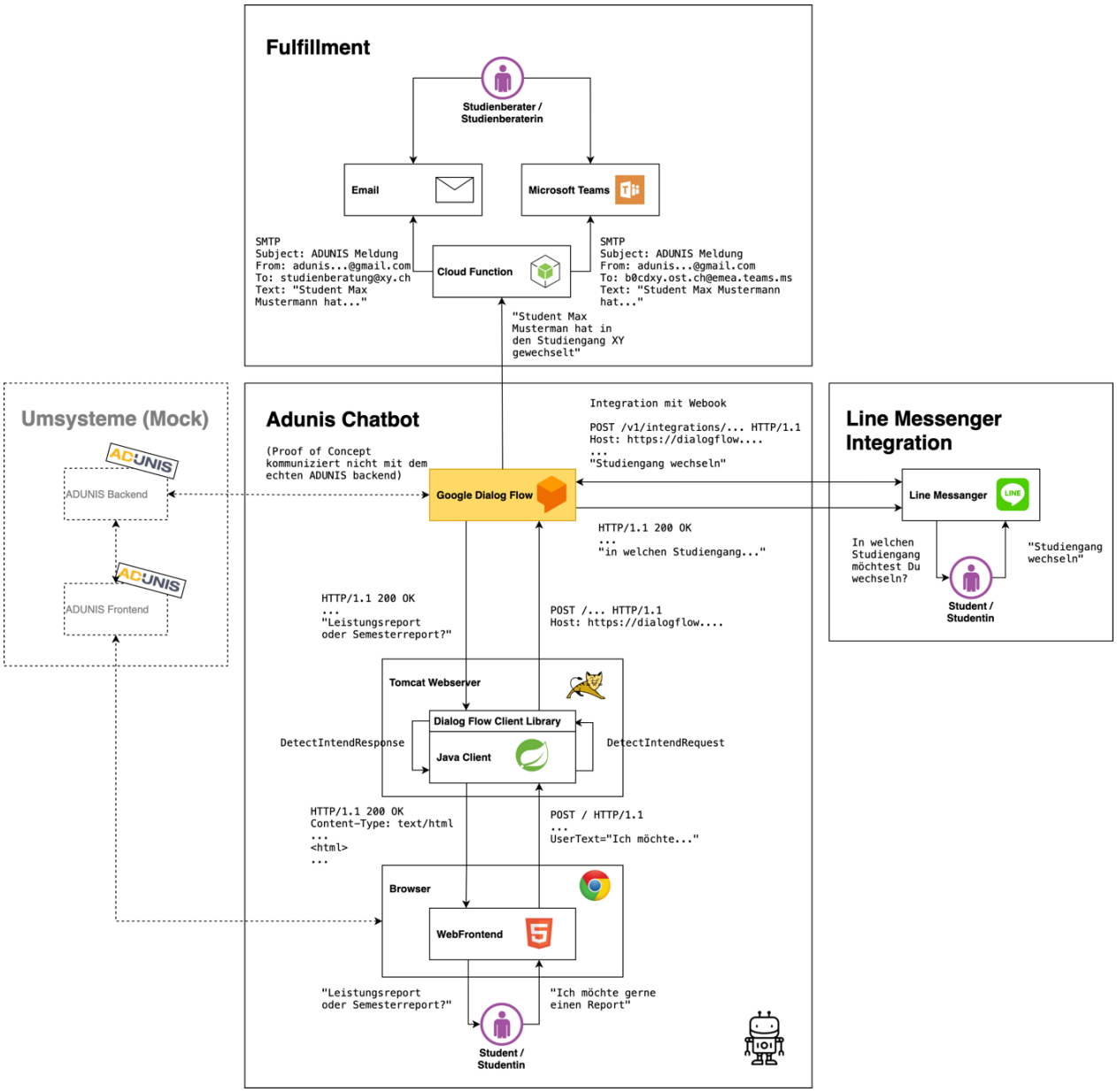


Abbildung 2: Architektur

Dialogflow Agent

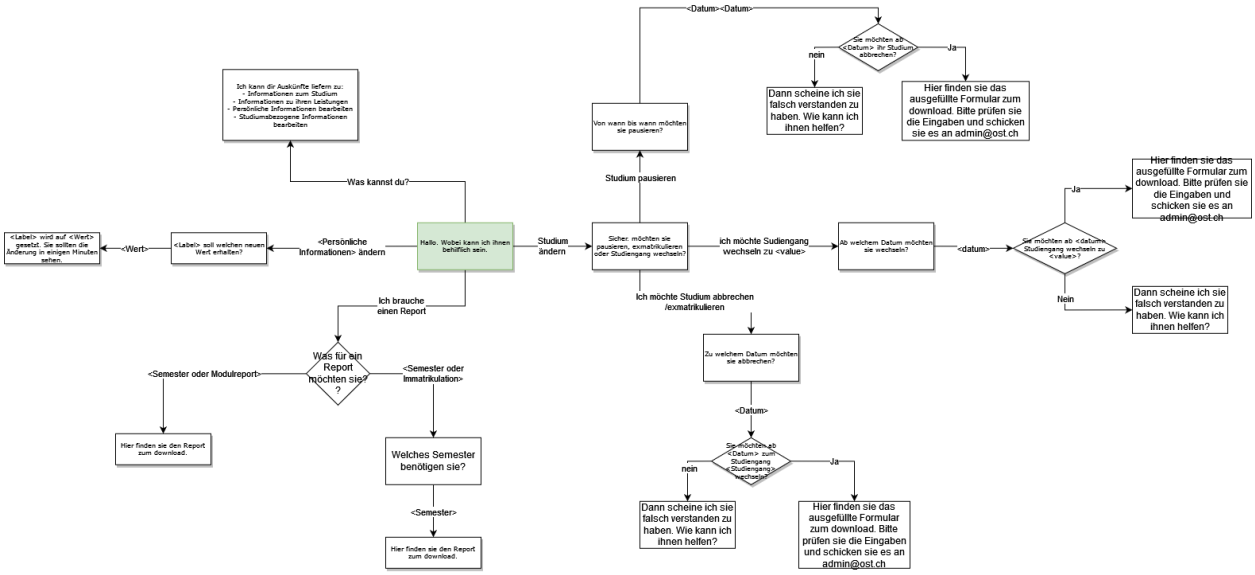


Abbildung 3: Diagramm der Dialogflow Intents

Als Basis für unser Dialogflow-Backend haben wir Aktionen vom ADUNIS Portal modelliert.

Beispielhafter Anwendungsfall: Ein Student, der einen Studiumsunterbruch melden will, meldet dies dem Chatbot. Der Chatbot fragt nach allen dazu nötigen Informationen. Das ausgefüllte Formular «Studienunterbruch» wird anschliessend zum Download bereitgestellt.

Insgesamt wurden 8 verschiedene Intents implementiert: info, changeData, reportSemester, studyReport, courseChange, exmatriculation, modelChange, pauseStudy. Um die gesammelten Daten strukturiert abzulegen, haben wir 11 Entities erstellt. (Semesterangabe, Reporttyp, Studiengangsmodell etc.)

Sentimentanalyse

Zur Analyse des Usererlebnisses wird im Dialogflow eine Sentimentanalyse vorgenommen. Somit soll das Erlebnis fortlaufend verbessert und Mängel aufgedeckt werden können. Für die Analyse der Benutzereingaben wird die Natural Language API verwendet. [2]

Stimmung	Beispielwerte
Eindeutig positiv*	`"score"`: 0,8, `"magnitude"`: 3,0
Eindeutig negativ*	`"score"`: -0,6, `"magnitude"`: 4,0
Neutral	`"score"`: 0,1, `"magnitude"`: 0,0
Gemischt	`"score"`: 0,0, `"magnitude"`: 4,0

Abbildung 4: Werte der Sentimentanalyse

[2] https://cloud.google.com/dialogflow/es/docs/how/sentiment

Java Client

Webserver

Der Java Client startet einen lokalen Webserver, der über die URL *http://localhost:8080* aufgerufen werden kann. Der Server stellt eine HTTPS-POST-Schnittstelle bereit, über welche die Eingaben der Benutzerin oder des Benutzers entgegengenommen werden.

Das Resultat in Form einer Webseite (HTML, CSS und JavaScript) wird Server-seitig gerendert. Um das Rendering zu vereinfachen, wird die Templating-Engine Thymeleaf eingesetzt. Aus dem Java-Code werden die zur Anzeige benötigten Informationen in einem Thymeleaf-Model abgelegt. Thymeleaf generiert aus diesen Informationen den HTML-Code.

Kommunikation mit Google Dialog Flow

Die Kommunikation mit dem Backend wurde in eine ChatBotService – Klasse ausgelagert. Dank dem Einsatz der Java Client Library für DialogFlow kann dabei direkt mit nativen Java-Objekten gearbeitet werden. Dem Controller, welcher die HTTPS-POST-Anfragen entgegennimmt, wird eine Service-Instanz injected, um mit dem Backend-Service zu kommunizieren. Dieser Service implementiert das Façade Pattern, in dem er die komplexen und umfangreichen Optionen der DialogFlow Library hinter einer einfachen Schnittstelle versteckt.

Bei den Anfragen an den Server wird neben der Eingabe des Benutzers auch eine OutputAudio Konfiguration mitgegeben. Diese wird später für die Sprachausgabe genutzt.

Die Antwort des Servers wird in eine Adapter-Klasse «ChatBotAnwser» gewrappt. Diese Klasse versteckt das eigentliche QueryResult-Objekt und stellt einige Hilfsmethoden zum Auslesen der interessanten Informationen zur Verfügung.

Dank dem ChatBotService, der ChatBotAnwser-Klasse sowie der Thymeleaf-Library ist im Controller selbst praktisch keine Logik notwendig:

```
@PostMapping(value = {"/"})
public String getBotAnswer(UserInput input, Model model) {
   ChatBotAnswer answer = chatBotService.sendMessage(input.getUserText());
   model.addAttribute("answer", answer);
   return "index";
}
```

Logik im Backend

Chat-Verlauf
Der Java Client persistiert den aktuellen Chat-Verlauf in einer Session, um diesen auf der Benutzeroberfläche ausgeben zu können.

Session Handling
Damit mehrere Personen gleichzeitig über die Java Applikation mit dem Chatbot kommunizieren können, ist ein Session Handling implementiert. Pro Browser-Session wird auch eine neue Session zwischen dem Java Client und dem Dialog Flow Backend aufgebaut.

Extrahieren relevanter Informationen
Aus der Antwort des Dialog Flow Backends werden bei jedem Aufruf die folgenden Informationen ausgelesen und für die Benutzeroberfläche aufbereitet:

- Antwort-Nachricht
- Intent
- Sentiment Score
- Intent Confidence Score
- Erkannte und Fehlende Felder für den aktuellen Intent

Anreichern der Sever-Antwort mit weiteren Informationen

Der Client kann erkennen, dass ein Intent abgeschlossen wurde und kann in diesem Fall die Fulfillment-Nachricht von Dialogflow mit weiteren Informationen anreichern. Dies ermöglicht es der Benutzeroberfläche, je nach ausgelöstem Fulfillment weitere relevante Informationen anzuzeigen (beispielsweise die Anzeige eines Download-Buttons). Je nach abgeschlossenem Intent wird dazu ein für diesen Anwendungsfall passendes ViewModel mit den benötigten Informationen abgefüllt und an die Benutzeroberfläche weitergegeben.

Weboberfläche

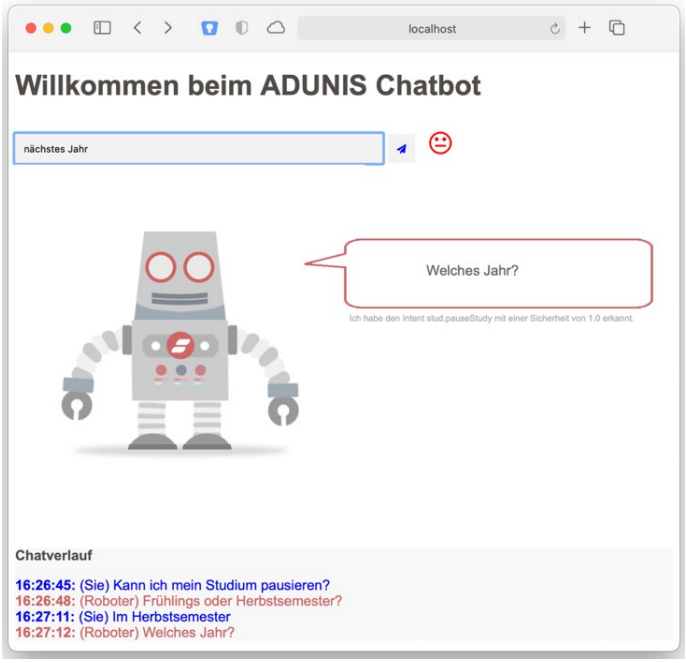


Abbildung 5: GUI der Webapplikation

Das GUI des Webservices ist einfach und übersichtlich gehalten. Prominent ist die Chateingabe, über die der Benutzer oder die Benutzerin mit dem Bot interagiert. Direkt neben der Eingabe ist die Sentimentanalyse durch einen Smiley abgebildet. Erkennt die Analyse einen gut gestimmten User, so wird ein glückliches Gesicht eingeblendet. Im Gegensatz dazu wird bei einem negativ gestimmten User ein negativer Smiley angezeigt.

Abbildung 6: Sentiment Smiley «Happy»

Unter der Eingabe steht die Antwort des Bots. Dieser wird mit einem GIF dargestellt, damit ein Interaktionsgefühl von Seiten des Benutzers, der Benutzerin entsteht. In der Sprechblase werden die Antworten des Dialogflow Agents platziert. Direkt unter der Blase werden der vom Bot erkannte Intent und die Erkennungs-«Confidence» ausgegeben. Damit können neugierige User hinter die Kulissen des Bots schauen.

Zuunterst ist der Chatverlauf dokumentiert, damit die gesamte Konversation nachvollzogen werden kann.

Sprachausgabe

Neben der visuellen Anzeige der Bot-Antwort wird diese auch mittels Text-to-Speech vorgelesen. Diese Funktion wurde über die Google Cloud Konsole konfiguriert. Zusätzlich muss bei jeder Anfrage an den Server eine OutputAudioEncoding-Konfiguration mitgegeben werden. Anschliessend sendet Dialog-Flow mit jeder Antwort Audio-Daten in Form eines ByteArrays zurück

```
detectIntendResponse.getOutputAudio().toByteArray();
```

Dieses ByteArray wird anschliessend als Audio wiedergegeben.

Benötigte Felder

Werden für ein Fullfillment mehrere Felder benötigt, wird ein Warnhinweis eingeblendet. So müssen zum Beispiel für einen Studiengangwechsel der neue Studiengang und das Datum des Wechsels ins Formular eingefügt werden.

Abbildung 7: Benötigte Felder für "Wechsel des Studiums"

Ausgabe von zusätzlichen Informationen beim Fulfillment

Ist ein Konversationszweig abgeschlossen und ein Fullfillment erreicht, wird dies im GUI graphisch angezeigt. Je nach Art des Fullfillments wird eine entsprechende Nachricht unter der Chat Eingabe-angezeigt oder ein Download-Knopf eingeblendet, damit das gewünschte Formular oder Report heruntergeladen werden kann.

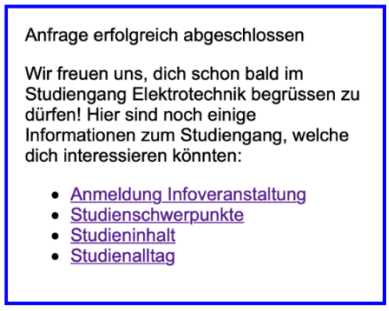


Abbildung 8: Abschluss der Anfrage "Wechsel des Studiums"

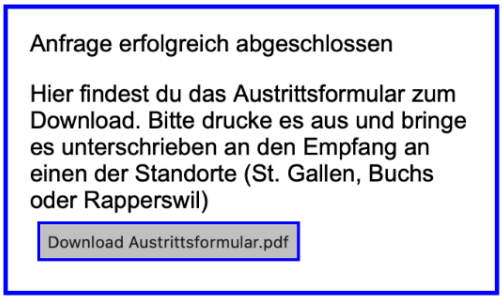


Abbildung 9: Abschluss der Anfrage "Austritt"

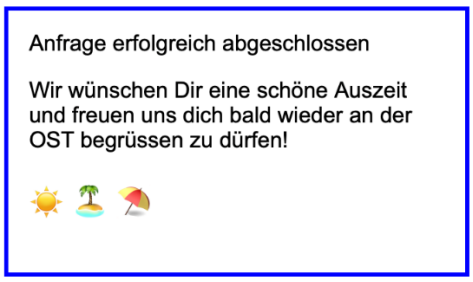


Abbildung 10: Abschluss der Anfrage "Studiums-Unterbruch"

Abbildung 11: Abschluss der Anfrage "Semesterreport"

Abbildung 12: Abschluss der Anfrage "Leistungsreport"

Integration mit der App «Line Messenger»

Abbildung 13: Chat

Über die Integrations-Seite auf der Dialog-Flow Konsole wurde als Alternative zum Webinterface eine Verknüpfung mit der Messanger-App «Line» erzeugt. Beim Erstellen der Verknüpfung wird von Google eine Webhook-URL sowie ein Secret erzeugt, mit welcher der Messenger mit dem Chatbot interagieren kann. Anschliessend wird ein neuer Line-Provider erzeugt, wo wiederum die Webhook-URL sowie das Secret in der Konfiguration hinterlegt ist.

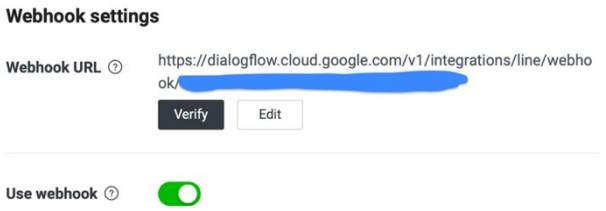


Abbildung 14: Webhook

Der Chatbot kann nun über die Benutzer-Suche des Line-Messengers gefunden und als Kontakt hinzugefügt werden.

Fullfilment mit Google Cloud Function

Wenn ein Student oder eine Studentin eine Änderung am Studium vornimmt, wird die Studienberatung automatisch informiert. Da es nun auch möglich ist, über den Line-Massenger mit dem Bot zu chatten, ist es nicht sinnvoll, diese Funktionalität nur in den Java Client einzubauen. Wir haben deshalb eine Google Cloud Function erstellt, die beim Fulfillment einer solchen Mutation automatisch ausgelöst wird. Der Node.js-Webhook verwendet das Package «nodemailer», um über eine Gmail-Adresse die Nachrichten zu versenden.

Benachrichtigungen werden ausgelöst, wenn ein Studiengang oder das Studien-Modell gewechselt wird, sowie wenn das Studium pausiert oder abgebrochen wird.

Einerseits gehen die Benachrichtigungen an eine (fake) Emailadresse der Studienberatung, andererseits an einen Teams-Kanal «Studienberatung». Die Benachrichtigungen im Teams-Kanal werden ebenfalls über die Kanal-Emailadresse gesendet.

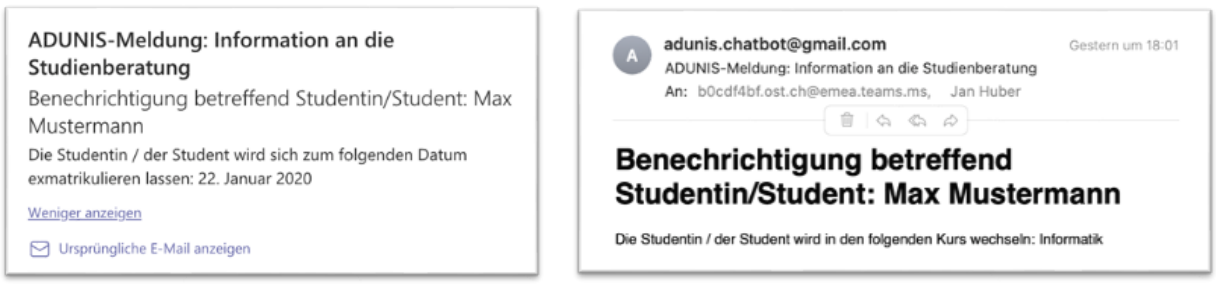


Abbildung 15: Benachrichtigung der Studienberatung über Microsoft Teams und Email

Fazit und Reflektion

Die Arbeit am Chatbot hat uns Spass gemacht, da wir uns ohne strenge Vorgaben ausleben und Neues ausprobieren konnten. Es war erstaunlich, wie sich mit verhältnismässig wenig Aufwand ein kleiner funktionstüchtiger Chatbot zusammenstellen lässt. Beim Erstellen des Agents mussten wir uns zuerst nochmals die Vorlesung in Erinnerung rufen (wir funktioniert ein Kontext? wir war das mit den Follow-Up-Intends?), wir glauben mittlerweile diese Konzepte gut verstanden zu haben.

Auf der Suche nach möglichen Implementations-Features und beim Lesen der Dialogflow-Dokumentation haben wir viele neue interessante Konzepte kennengelernt. Dabei sind wir auch über Konzepte gestolpert, die nicht direkt mit «AI» zu tun haben, aber trotzdem hilfreich sind: Wie wird ein Audio-Stream in Form eines Bytes-Arrays abgespielt? Wie funktioniert das Session-Management bei einer Spring Boot Web Applikation? Wie wird mit Node.js eine Email versendet?

Beim Versenden der Emails haben wir uns zuerst lange gefragt, warum die Email-Benachrichtigungen sporadisch nicht im Teams-Kanal eingetroffen sind. Wir konnten aber schlussendlich herausfinden, dass der Spam-Filter einen Teil der Mails eliminiert.

Etwas schade ist, dass der Chatbot nur mit einem Fake-Backend von ADUNIS kommuniziert. Es wäre spannend gewesen, über unsere Oberfläche das echte ADUNIS anzusteuern.

In einem nächsten Schritt wäre es interessant, die Sentiment-Analyse im Hintergrund zu persistieren und aus diesen Daten Verbesserungsmöglichkeiten auszuarbeiten.

Anhang

Quellenverzeichnis

Diagramme wurden mit der Software Draw.io gezeichnet

Titelbild: CC 3.0 BY, Freepik, http://www.flaticon.com/authors/freepik, via Wikimedia.org

Animated Robot Gif: showit.co via giphy.com (https://giphy.com/stickers/showit-wave-robot-showit-RJPBinhXAfOYpBtWWN)

Java Code-Fragement zur Ausgabe von Sprachsynthese-Audio :
http://www.java2s.com/example/java/applet/plays-the-sound-clip-contained-in-the-given-byte-array.html

JavaScript Code-Fragement zur Ausgabe von Sprachsynthese-Audio :
https://stackoverflow.com/questions/24151121/how-to-play-wav-audio-byte-array-via-javascript-html5

Sentiment Analyse: https://cloud.google.com/dialogflow/es/docs/how/sentiment

Abbildungsverzeichnis