

Der Coaching Bot

The Coaching Bot

Maximilian Wellenhofer

Master-Projektstudium

Betreuer: Prof. Dr. Georg Schneider

Zürich, 28.02.2022

---

## Kurzfassung

*Provisionierung eines Bots, der die OnBoarding-Phase eines Coaching Programms automatisiert.*

In der Kurzfassung soll in kurzer und prägnanter Weise der wesentliche Inhalt der Arbeit beschrieben werden. Dazu zählen vor allem eine kurze Aufgabenbeschreibung, der Lösungsansatz sowie die wesentlichen Ergebnisse der Arbeit. Ein häufiger Fehler für die Kurzfassung ist, dass lediglich die Aufgabenbeschreibung (d.h. das Problem) in Kurzform vorgelegt wird. Die Kurzfassung soll aber die gesamte Arbeit widerspiegeln. Deshalb sind vor allem die erzielten Ergebnisse darzustellen. Die Kurzfassung soll etwa eine halbe bis ganze DIN-A4-Seite umfassen.

Hinweis: Schreiben Sie die Kurzfassung am Ende der Arbeit, denn eventuell ist Ihnen beim Schreiben erst vollends klar geworden, was das Wesentliche der Arbeit ist bzw. welche Schwerpunkte Sie bei der Arbeit gesetzt haben. Andernfalls laufen Sie Gefahr, dass die Kurzfassung nicht zum Rest der Arbeit passt.

---

## Abstract

The same in English.

---

## Inhaltsverzeichnis

---

## Abbildungsverzeichnis

---

## Tabellenverzeichnis

---

## Listings

# Einleitung und Problemstellung

## 1.1 Motivation

### *Warum einen Coaching Bot bauen?*

Viele junge Menschen die nach einem abgeschlossenen Studium in die Arbeitswelt einsteigen möchten, haben keine oder wenig Ahnung davon, wie sie sich vorbereiten sollen oder welche Schritte erforderlich sind, um einen erfolgreichen Start zu schaffen. Persönliche Beratungsleistungen und speziell Einzel-Coaching können dabei helfen, sich effektiv vorzubereiten und einen erheblichen Wettbewerbsvorteil bieten, sind aber für Berufseinsteiger ohne signifikante finanzielle Mittel meist weder zugänglich noch erschwinglich. Das sind in Gründung befindliche Unternehmen -wavehoover consult AG- aus Zürich hat es sich daher zur Aufgabe gemacht, in dieser Hinsicht einen Beitrag zu leisten. Interaktionen mit jungen Absolventen auf klassischen Websites waren bis dato wenig erfolgversprechend. Aus diesem Bedürfnis heraus und aufgrund meiner beruflichen Erkenntnis, dass die erste Kontaktaufnahme und die Vorbereitung auf die erste Sitzung meist sehr ähnlich oder gar einem Skript folgend ablaufen, ist die Idee zu einem standardisierten und automatisierten OnBoarding-Prozess entstanden. Vielen jungen Menschen wenden sich ab vom traditionellen Web-Browser und sind stärker in Messenger-Diensten wie WhatsApp, Signal, Telegram oder vor Allem auf Social Media Portalen wie Instagram, Facebook, TikTok, SnapChat, etc. zu finden. Um sich auf diese Zielgruppe einzulassen, hat man sich entschieden, einen Bot zu programmieren, der Informationen vom User abfragt und Der Instant Messenger Telegram (<https://telegram.org/>) ist (neben vielen anderen) besonders unter jungen Menschen, ein beliebtes Kommunikations- und Interaktionsmedium, das Funktionen weit über das einfache Nachrichten-austauschen hinaus abdeckt. Ziel des Projekts ist es, primär jungen Absolventen die Möglichkeit zu bieten, sich auf eine erste Live Coaching Session vorzubereiten und nicht "völlig blank" in eine bezahlte Beratungsleistung zu investieren, ohne überhaupt bereits zu wissen, was sie erwartet oder wie man sich vorzubereiten hat, um das Meiste für sich selbst heraus zu holen.



## Verwandte Arbeiten

Schauen Sie nach – ob es bereits existierende Arbeiten und Systeme, die ähnliche Probleme bearbeiten gibt. Dies ist eigentlich das „Herzstück“ einer wissenschaftlichen Arbeit, weil man seine eigenen Beiträge zum aktuellen State-of-the-Art in Verbindung setzt. Beschreiben Sie (wenn möglich) mindestens 2 dieser Arbeiten. Danach, sozusagen als Resümee können Sie dann sagen, was Sie anders (besser!) machen wollen (also hier kommen Argumente hin, warum Sie nicht ein existierendes System nehmen, sondern selber eins programmieren und was Sie an Ideen übernehmen).

Analysiert man den Hintergrund der meisten Bots, so wird schnell klar - es besteht in einer überwältigenden Mehrheit der Fälle ein monetärer Beweggrund, einen Bot zu erstellen. Neben Coaching Bots zur Optimierung des eigenen Investment-Portfolios sowie diversen Bezahlmodellen für Gesundheits- und Personal Training Programme, existiert auf Social Media Portalen eine Vielzahl an Chatbots, die technisch sehr umfangreich und mächtig sind, aber fast keine quelloffenen und kostenlosen Beispiele, die unserem Zweck genügen.

*Im Folgenden werden 2 etablierte Coaching-Bots betrachtet, die keinen monetären Zweck verfolgen.*

### 2.1 CoachPTBS - der Chatbot der Bundeswehr

Seit jeher haben Soldaten mit posttraumatischen Belastungssyndromen zu kämpfen. Die Bundeswehr hat mit dem CoachPTBS ein digitales Experiment gewagt, das versucht, Betroffenen eine möglichst niedrige Einstiegshürde zu bieten, sich mit ihren Problemen auseinanderzusetzen. Auch für den Coaching-Bot benötigen wir eine niedrige Hemmschwelle, damit möglichst viele junge Menschen teilnehmen und sich um einen Platz im Programm bewerben.

Die Technologie hinter der CoachPTBS ist sicherlich weit fortgeschritten, jedoch erhalten wir als gemeinnützige Organisation ohne finanzielle Mittel aus einem nicht quelloffenen Projekt - so wegweisend es auch sein mag - keinen direkten Startvorteil. Allerdings berstärkt uns der CoachBot der Bundeswehr in unserer Annahme, dass Individuen einen Bot als einfachen Weg sehen, eine erste Kontaktaufnahme mit einem Coachee zu etablieren - vor allem in Situationen, in denen Menschen noch nicht bereit sind, direkt mit jemandem zu sprechen.

## 2.2 Telegram Chat Bots

Telegram Chat Bots sind Applikationen, die auf einer quelloffenen API [?] basieren, auf allen Komplexitätsstufen adaptierbar sind und es jedermann ermöglichen, einen Chatbot zu bauen. Die einzige suboptimale Einschränkung besteht im Vendor Lock-In der Telegram-App. (Der Bot ist nur in Verbindung mit der Telegram-App [?] nutzbar.) Die meisten Menschen in Deutschland und der DACH-Region verwenden immer noch WhatsApp [?] und doch bietet uns der populärste Messenger nicht die Freiheiten und den Funktionsumfang, den wir uns für unseren CoachingBot wünschen. Telegram jedoch hält genau diese Offenheit für uns bereit. [?]. So bietet der Dienst, die Möglichkeit, via einer API direkt in die Entwicklung einzusteigen und hält sogar basale State-Machines für uns bereit, die uns die Komplexität für den Kern des Bots nicht komplett abnehmen, aber als Gerüst für einen ChatBot dienen können.

## Grundlagen

Die folgenden Sprachen und Systeme dienen als Grundlage für die im Rahmen dieses Projekts entwickelte Applikation. Eine Liste aller eingebundenen Bibliotheken kann dem Pipfile des Projekts entnommen werden.

### 3.1 Python

Der größte Teil der Applikation ist in Python 3.8.6 [?] geschrieben. Das war zum Stand des Entwicklungsbeginns 2021 die aktuell stabile Python-Version.

### 3.2 Telegram API

Die Applikation basiert auf der API des Instant Messaging Dienstes Telegram [?] sowie deren Extension [?]. Als Grundgerüst der für den Bot erforderlichen State Machine wurde der ConversationBot von Leandro Toledo et. al. genutzt. [?] Das Repository enthält eine Vielzahl basaler Bot-Implementierungen, die als Startpunkt für jegliche Bot-Implementierung einen guten Überblick über das Grundgerüst und die Funktionsweise eines Bot geben.

### 3.3 SQLite

Zur Speicherung von Nutzerdaten wird eine SQLite Datenbank [?] genutzt.

### 3.4 SQLite3 API

Als Schnittstelle zwischen dem Python basierten Backend und der SQLite Datenbank nutzt die Applikation den sqlite3 Database-Connector [?].

### 3.5 HTML

HTML bietet uns die Möglichkeit die grafische Oberfläche in einem Standard Webbrowser auszugeben. [?]

## 3.6 CSS

Zur Aufbereitung der Web-GUI wird ein CSS-File genutzt.

## 3.7 Flask

Die HTML-GUI wird via Flask [?] an einen lokalen Web-Server übermittelt und kann so einfach mittels Python in gängigen Browsern präsentiert werden.

## 3.8 Google Calendar API

Die Applikation bindet die Google Calendar API [?] an, um es dem User zu ermöglichen, einen Termin mit dem Anbieter zu vereinbaren.

## 3.9 TheCoachingBot

Schließlich findet sich der gesamte Source-Code inklusive aller Abhängigkeiten in einem öffentlichen GitHub Repository: [?]

## Konzept

Zum Beispiel: - Blockbild der Architektur Ihrer Anwendung - Pseudocode für Algorithmen - mathematische Formeln - evtl. Diagramme auf hohem Abstraktionsniveau.

### 4.1 Grundkonzept

#### *Der Telegram Bot als Herzstück*

Das Herzstück der Applikation ist der Telegram-Bot selbst. Er wird von einem Benutzer angesprochen und reagiert auf seine Eingabe. So können verschieden Funktionen ausgelöst werden. Bspw. werden Antworten zurückgegeben, Informationen gespeichert oder es wird ein Vorschlag gemacht und an den Nutzer zurückgegeben. Der Bot soll mit mehreren Benutzern gleichzeitig sprechen können. Das wird ermöglicht, weil alle Reaktionen des Bots mit der Kennung des einzelnen Nutzers verbunden sind. So spricht der Bot den Nutzer mit Namen an oder kann sich daran erinnern, welche Fragen schon beantwortet wurden und welche nicht.

Abbildung 4.1: Architektur für das Projekt "Der Coaching Bot" auf hohem Abstraktionsniveau

Abbildung 4.2: Konversationsfluss des Bots

## Realisierung

Hier kommt hin, wie es gemacht haben.

### 5.1 Telegram Bot Framework

#### 5.1.1 Generierung Telegram Bot

Als ersten Schritt zur Erschaffung eines Telegram-Bots wird der Bot-Father (selbst ein Telegram-Bot [?]) konsultiert. Er erstellt das Framework, registriert den Bot und gibt ein API-Token zurück, das verwendet werden kann, um sich gegenüber der Telegram-Bot-API als Entwickler zu identifizieren. [?]

### 5.2 Vanilla Bot Implementierung

Als Basis (boiler plate) für den Coaching Bot nutzen wir die breit in der Community abgestützte Implementierung "Conversation Bot"[?]. Sie stellt uns die basale Anbindung an die State Machine zur Verfügung und ist einfach genug, um als Einstieg in einen vordefinierten Bot zu fungieren. Im Gegensatz dazu ist der "Nested Conversation Bot" schon zu umfangreich und zu mächtig für unsere Zwecke.

Die Kommunikationslogik des Bots basiert auf einer State Machine. Die Zustände, in denen der Bot sich befinden kann, sind vordefiniert und immer mit einer Aktion und einer Reaktion verbunden. Aktionen werden meist von Seiten des Benutzers durch eine Eingabe oder einen Befehl ausgelöst. Reaktionen sind in Funktionen vordefiniert. Deren Umfang wird im Folgenden funktional und im Kapitel "Implementierung" technisch beschrieben.

### 5.3 Meta-Funktionen

Neben den Hauptfunktionen des Bots (Funktionen, die zum Gesprächsfluss gehören), gibt es eine Reihe an Meta-Funktionen, die dem Nutzer zur Verfügung stehen, um eine Konversation zu starten, zu beenden, persönliche Daten zu löschen oder die Hilfe auszugeben.

### 5.3.1 Start: Eine Konversation beginnen

Der Bot kann gestartet werden (Aktion) und gibt eine Begrüßungsnachricht zurück. Gleichzeitig erfasst er grundsätzliche Informationen des Nutzers und schreibt diese in eine Datenbank. Aber diesem Zeitpunkt, kennt die Applikation den Benutzer und kann weitere Informationen über ihn speichern oder individuell auf Eingaben reagieren.

#### *Die Loop-Back-Funktion*

Eine der großen Herausforderungen für den Bot besteht darin, einen Nutzer wiederzuerkennen und ihn am richtigen Punkt zurück in den Konversationsfluss zu platzieren. Diese Erfahrung soll für den Nutzer nicht angestrengt wirken, sondern so, als würde der Bot ihn schon kennen und einfach da weitermachen, wo man aufgehört hat. Die technische Komplexität besteht darin, dass das Feature besonders dann funktionieren soll, wenn der Bot neu gestartet wurde.

### 5.3.2 Ende: Konversation manuell beenden

Hat der Nutzer eine Konversation gestartet, so kann er diese auch wieder beenden. Die Konversation muss nicht zuende geführt worden sein. Über einen kurzen Befehl `/cancel` wird der Bot beendet und personenbezogene Daten werden aus der Datenbank gelöscht. Dabei ist darauf zu achten, dass der Nutzer nur seine eigenen Daten löschen kann. Hat der Nutzer seine Konversation bereits beendet, so ist `/cancel` nicht mehr verfügbar. Möchte der Nutzer seine Daten dennoch löschen, so steht ihm stattdessen der Befehl `/delete` zur Verfügung.

### 5.3.3 Persönliche Daten löschen

Zu jeder Zeit hat der Nutzer die Möglichkeit, die eigenen Daten via dem Befehl `/delete` zu löschen. Die Funktion ist mit einem "Reset-Knopf" zu vergleichen. Das Resultat ist nämlich, dass der Bot den Benutzer nicht mehr kennt. Er weiß nicht, dass er schon einmal da war und auch nicht, welche Angaben er gemacht hat oder nicht. So kann man den Bot nach fehlerhafter Eingabe oder, falls man neu anfangen möchte, einfach zurücksetzen.

### 5.3.4 Hilfe-Funktion aufrufen

Die Hilfe-Funktion gibt eine Beschreibung der Interaktionen-Optionen aus, die es gegenüber dem Bot gibt. So werden alle Befehle einfach erklärt und können auch direkt aus der Hilfe heraus aufgerufen werden.

### 5.3.5 Überspringen

Die meisten Zustände des Bots erlauben es dem Benutzer, die aktuelle Frage zu überspringen. Vor allem, wenn es um personenbezogene oder private Informationen geht, die der Nutzer preisgibt, ist der Befehl `/skip` verfügbar. Für jeden Zustand,

in dem /skip" verfügbar ist, ist eine individuelle Reaktion auf das Überspringen vorgesehen, die den Nutzer trotzdem abholt um in den nächsten Zustand leitet. Die einzelnen Übersprungsfunktionen werden im Kapitel "Implementierung" genauer erklärt.

## 5.4 Hauptfunktionen

### 5.4.1 Abfragen des Geburtsdatums

Um zu erfahren, wie alt der Bewerber ist, möchten wir das Geburtsdatum abfragen. Dabei ist wichtig, dass das Datum in einem sinnvollen Format eingegeben wird. (Siehe Eingabe-Validierung.)

### 5.4.2 Hintergrund des Nutzers

Für eine Coaching-Session ist es besonders wichtig, den Coachee besser kennenzulernen. Zu diesem Zweck hat der Nutzer die Möglichkeit, etwas über sich zu erzählen. Erwartet wird hier kein komplettes Motivationsschreiben, sondern einfache, kurz formulierte Beweggründe dafür, dass man gerne mit dem Personal Coaching beginnen möchte.

### 5.4.3 Abfragen des Geschlechts des Nutzers

Um den Nutzer in der Folgekommunikation korrekt anzusprechen, wird nach dem Geschlecht des Nutzers gefragt. Neben der Option, die Frage überspringen zu können, präsentiert der Bot den Nutzer mit mehr als 2 Optionen, um diversen Geschlechtern gerecht zu werden.

### 5.4.4 Abfragen der E-Mail Adresse des Nutzers

Um dem Nutzer eine E-Mail mit allen erfassten Daten zusenden zu können und dem eigentlichen Zweck des Bots nachzukommen - einen Termin vereinbaren zu können - benötigt der Bot eine valide E-Mail-Adresse des Nutzers. Um die Wahrscheinlichkeit zu erhöhen, dass bei dieser Eingabe keine Fehler passieren, ist auch hier eine Eingabe-Validierung hinterlegt.

### 5.4.5 Abfragen der Telefonnummer des Nutzers

Am Ende des Konversationsflusses hat der Nutzer die Möglichkeit, einen ersten Termin zu vereinbaren. Dabei handelt es sich um einen unverbindlichen Telefontermin. Um den Nutzer zu einer festgelegten Zeit erreichen zu können, wird hier die Telefonnummer des Nutzers erfasst. Da der Service aktuell nur in der DACH-Region angeboten wird, können hier nur Telefonnummern mit der Länderkennung Deutschland, Österreich und der Schweiz angegeben werden.



#### 5.4.6 Abfragen des Standorts des Nutzers

Der Coaching-Service soll primär und vorerst nur in der DACH-Region angeboten werden. Daher soll der Standort des Nutzers abgefragt werden. Eine Geo-Fencing-Funktion würde für unseren Zweck hier zu weit gehen, weil wir auch Personen die Chance geben wollen, sich für den Dienst anzumelden, die aktuell im Ausland sind. So bietet die Telegram-App dem Nutzer die Möglichkeit, den Ort, den er teilen möchte, spontan selbst auszuwählen.

#### 5.4.7 Abfragen des Bilds des Nutzers

Informationen aller Nutzer werden als Resultat der Teilnahme am On-Boarding in einer Web-GUI ausgegeben. Hier wird neben den Informationen zum Bewerber auch ein Bild angezeigt. So kann der Coach sich besser auf ein erstes Treffen einstellen.

#### 5.4.8 Zusammenfassungs-Funktion

Ziel des Bots ist ein hohes Maß an Transparenz auf allen Seiten. Der Nutzer weiß nicht nur, dass seine Daten erfasst wurden, sondern am Ende des Konversationsflusses werden diese auch automatisch zurückkommuniziert. Dies passiert auf zweierlei Wegen. Neben einer Telegram-Nachricht wird dem Nutzer auch eine Zusammenfassung in Form einer E-Mail an die angegebene Adresse gesendet. Darüber hinaus hat der Nutzer die Möglichkeit, die Zusammenfassung jederzeit manuell abzurufen. So kann er jederzeit einsehen, welche Informationen bereits übergeben wurden und welche noch fehlen.

### 5.5 Support-Funktionen

#### 5.5.1 Eingabe-Validierung

Bei einigen Angaben ist es besonders wichtig, dass Eingaben auf korrekte Formate geprüft werden. So müssen bspw. E-Mail-Adresse sowie Telefonnummer des Nutzers stimmen, um weitere Funktionen des Bots zu nutzen. Um die Wahrscheinlichkeit dafür, dass diese Eingaben korrekt sind, zu steigern, werden ausgewählte Eingaben auf Formatfehler geprüft und der Nutzer bei falscher Eingabe um eine erneute Eingabe gebeten.

#### 5.5.2 E-Mail zusammenbauen

Die E-Mail, die am Ende des Konversationsflusses ausgegeben wird, wird separat aus verschiedenen Bausteinen zusammengesetzt. Dafür kommen Informationsabfragen gegen die Datenbank mit der Ansprache eines Mail-Servers zusammen.

## 5.6 Datenbank

Fast alle Informationen über den Nutzer werden in einer Datenbank gespeichert. Ausgenommen ist nur das Bild, das der Nutzer hochlädt. So können einzelne Werte jederzeit verwendet werden, um Nutzer-spezifische Reaktionen zu gestalten. Dem Nutzer stehen die meisten Datenbank-Operationen implizit und wenige explizit zur Verfügung. Daten werden implizit gespeichert und abgerufen. Explizit können Daten gelöscht werden. Zur Realisierung wird eine SQLite Datenbank genutzt. Diese sehr einfache Datenbank ist für den Zweck des Coaching-Bots vollkommen ausreichend. Weder ist mit immensen Nutzerzahlen, noch mit vielen gleichzeitigen Operationen oder einer riesigen Datemenge zu rechnen, was für mächtigere Lösungen sprechen würde. Da keine komplexen Berechnungen auf den Daten ausgeführt werden, sondern nur basale CRUD-Operationen geplant sind, gibt es nur eine Tabelle, in der alle Nutzerdaten gespeichert sind.

## 5.7 Anbindung Datenbank an Python

Um eine individuelle Implementierung eines Database-Connectors zu vermeiden, bedient die Applikation sich der sqlite3-Bibliothek. Sie ermöglicht es, klassische Datenbank-Operationen direkt aus einem Python-Skript heraus anzustoßen und dient hier als Database-Connector. Die Operationen selbst werden in handelsüblichem SQL formuliert und übergeben.

## 5.8 Kalender

Um am Ende des Konversationsflusses einen ersten Termin mit einem Coach vereinbaren zu können, muss der Nutzer einen freien Termin auswählen können und für diesen eine Einladung beantragen. Zu diesem Zweck wurde die Google Calendar API angebunden. Der Nutzer wird zunächst gefragt, ob er überhaupt einen Termin vereinbaren möchte. Daraufhin wird die GCA abgefragt und dem Nutzer werden drei Termine vorgeschlagen. Mit einem Klick kann der gewünschte Termin dann ausgewählt werden. Kurz darauf erhält der Nutzer eine Termineinladung an die zuvor angegebene E-Mail-Adresse und kann diese im persönlichen Kalender-Client annehmen oder ablehnen.

## Implementierung

Kann mit 5. Zusammenfallen. Manchmal eignet es sich, 2 Abstraktionsschritte zu machen (Realisierung und Implementierung getrennt).

Generell für 5. Und 6.: Wenig Quellcode (wenn überhaupt)! Maximal 2/3 Seite und immer begleitet von Erklärungen, was zu sehen ist. Kommentare im Quellcode sind nicht ausreichend. Dies gilt für UML Diagrammen analog.

### 6.1 Setup

#### 6.1.1 Entwicklungsumgebung

#### 6.1.2 Microsoft Visual Studio Code

#### 6.1.3 pipenv

Die Applikation nutzt den Package Manager pipenv. Dieser bietet die Möglichkeit, ein projektspezifisches Dokument über alle Abhängigkeiten hinweg zu erstellen und im Projekt selbst zu speichern. So können andere Entwickler Abhängigkeiten leicht installieren und müssen dies nicht auf Systemebene tun, wo es ggf. zu Konflikten mit anderen Projekten kommen könnte. Um alle Abhängigkeiten einzusehen, pipenv [?] installieren und das *coaching<sub>bot</sub>/PipfileentsprechendderDokumentationnutzen, umalleau*

#### 6.1.4 Konstanten und Schlüssel

Der Coaching Bot hat einige Abhängigkeiten zu Umsystemen, die Zugangsdaten voraussetzen. Diese sind im Repository [?] aus Sicherheitsgründen abstrahiert und können durch kleine Anpassungen adaptiert werden.

#### 6.1.5 Dispatcher

Dispatcher liefern Nachrichten an den User aus. Pro Bot gibt es meist einen Dispatcher. Command- und ConversationHandler werden an diesen angehängt.

### 6.1.6 ConversationHandlers

ConversationHandler kontrollieren den Konversationsfluss zwischen dem User und dem Bot. Pro Bot kann es mehrere ConversationHandler geben. Der CoachingBot hat aber nur einen - den "conv\_handler". *Der ConversationHandler koordiniert alle CommandHandler*

### 6.1.7 CommandHandler

CommandHandler nehmen Nutzereingaben via eines CallbackContexts entgegen, prüfen diese auf vordefinierte Kriterien und führen prädefinierte Funktionen - sog. Handler Functions aus.

## 6.2 Coaching Bot Herzstück - main.py - State Machine

Die main.py ist das Herzstück des Coaching Bots.

Sie importiert alle Handler-Functions, authentifiziert sich durch den entsprechenden API-Schlüssel und beinhaltet den Dispatcher, an dem wiederum Conversation- sowie CommandHandler hängen. Darüber hinaus startet sie den Bot und aktualisiert die Handler in regelmäßigen Abständen via dem Updater.

Im Folgenden werden die CommandHandler für den Coaching Bot kurz aufgeführt:

### 6.2.1 start und cancel - Konversation starten und stoppen

Der in dieser Applikation umfangreichste ConversationHandler umfasst zwei Commands: /start und /cancel". Solange der Bot ausgeführt wird, lässt sich eine Konversation mit ihm über den Befehl /start starten und via /cancel beenden. Zur Funktionsweise von /start und /cancel", siehe start.py und "cancel.py unter "Handler Funktionen".

### 6.2.2 delete - Nutzerdaten löschen

Über den Befehl /delete" wird der CommandHandler "delete" ausgeführt. Zur Funktionsweise von /delete", siehe "delete.py unter "Handler Funktionen".

### 6.2.3 help - Hilfe ausgeben

Über den Befehl /help" wird der CommandHandler "help" ausgeführt. Zur Funktionsweise von /help", siehe "help.py unter "Handler Funktionen".

### 6.2.4 summary - Zusammenfassung ausgeben

Über den Befehl /summary" wird der CommandHandler "summary" ausgeführt. Zur Funktionsweise von /summary", siehe "summary.py unter "Handler Funktionen".

### 6.2.5 status - Status Quo ausgeben

Über den Befehl `/status` wird der CommandHandler `status` ausgeführt. Zur Funktionsweise von `/status`, siehe `status.py` unter "Handler Funktionen".

## 6.3 Handler Funktionen

Handler-Funktionen (siehe `coaching_bot/handler_functionsin[?]`) sind Funktionen, die auf Eingaben reagieren. Sie beschreiben den Umfang und Aufbau und erklären ihre Funktionsweise.

**6.3.1 start.py**

**6.3.2 birthdate.py**

**6.3.3 bio.py**

**6.3.4 cancel**

**6.3.5 confirmation<sub>m</sub>*ail.py***

**6.3.6 email.py**

**6.3.7 gender.py**

**6.3.8 help.py**

**6.3.9 location.py**

**6.3.10 photo.py**

**6.3.11 states.py**

**6.3.12 status.py**

**6.3.13 summary.py**

**6.3.14 telephone.py**

**6.3.15 validation.py**

**6.4 Datenbank**

**6.5 E-Mail-Versand**

**6.6 Tests**

**6.7**

**6.8**

**6.9**

**6.10**

**6.11**

**6.12 Kalender**

Google Cloud Console: <https://console.cloud.google.com/apis/credentials/consent?project=coaching-bot-339115> Calendar API Python Quickstart Guide: <https://developers.google.com/calendar/api/quickstart/python>  
Verify own website: <https://www.google.com/webmasters/verification/home?hl=en>

## Beispiele

Der User startet den Bot via dem Klick auf einen Link, den er auf einer Website findet oder der ihm zugesandt wird.

### 7.0.1 Stage 00

/start

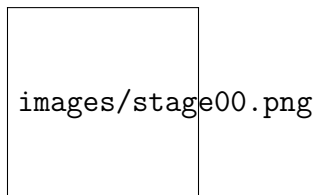


Abbildung 7.1: Bezeichnung der Abbildung

### 7.0.2 Stage 01

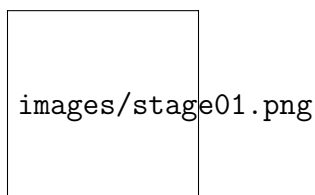


Abbildung 7.2: Bezeichnung der Abbildung

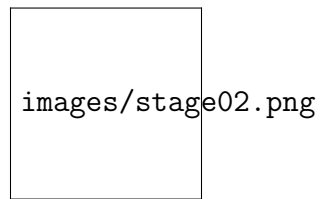


Abbildung 7.3: Bezeichnung der Abbildung

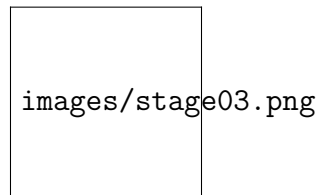


Abbildung 7.4: Bezeichnung der Abbildung

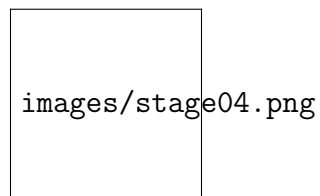


Abbildung 7.5: Bezeichnung der Abbildung

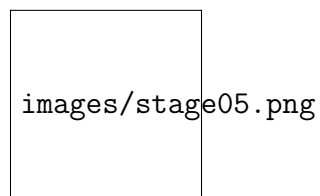


Abbildung 7.6: Bezeichnung der Abbildung

**7.0.3 Stage 02**

**7.0.4 Stage 03**

**7.0.5 Stage 04**

**7.0.6 Stage 05**


**7.0.7 Stage 06**

**7.0.8 Stage 07**

**7.0.9 Stage 08**


**7.0.10 Stage 09**





images/stage06.png

Abbildung 7.7: Bezeichnung der Abbildung




images/stage07.png

Abbildung 7.8: Bezeichnung der Abbildung



images/stage08.png

Abbildung 7.9: Bezeichnung der Abbildung



images/stage09.png

Abbildung 7.10: Bezeichnung der Abbildung

## Anwendungsszenarien

Der Coaching Bot kann in allerlei Szenarien angewandt werden. Die in den letzten Jahren stark angewachsene Zahl an Personal Coaches kann den Bot mit basalen Programmierfähigkeiten an die eigenen Bedürfnisse anpassen. Vor allem für Nebenerwerbstätige Coaches mit einem kleinen Kundenportfolio stellt der Coaching Bot eine einfache Möglichkeit dar, Neukunden onzuboarden. Der Prozess ist unkompliziert, unverbindlich und einfach zu adaptieren.

### *Ausbaupotenzial*

Sollte der Coaching Bot über die ersten Monate vielversprechende Ergebnisse liefern, sind folgende Ausbaustufen geplant:

1. Verteilung und Verlagerung in die Cloud für konstante und hohe Verfügbarkeit
2. Skripten und Automatisierung weiterer Coaching-Stufen. i.e. könnte die erste Session, die oft ähnlich abläuft auch vom Bot abgehandelt werden.
3. Ton-Aufnahmen für das Biography-Modul

## Zusammenfassung und Ausblick

In diesem Kapitel soll die Arbeit noch einmal kurz zusammengefasst werden. Insbesondere sollen die wesentlichen Ergebnisse Ihrer Arbeit herausgehoben werden. Erfahrungen, die z.B. Benutzer mit der Mensch-Maschine-Schnittstelle gemacht haben oder Ergebnisse von Leistungsmessungen sollen an dieser Stelle präsentiert werden. Sie können in diesem Kapitel auch die Ergebnisse oder das Arbeitsumfeld Ihrer Arbeit kritisch bewerten. Wünschenswerte Erweiterungen sollen als Hinweise auf weiterführende Arbeiten erwähnt werden.

Nochmal kurz sagen, was Sie gemacht haben - am besten die Ziele der Arbeit aus 1.2. nochmals nennen und kurz erklären, wie sie das in Ihrem System realisiert haben. (So was wie ein "management summary")

---

appendix

# A

---

## Glossar

DisASter	Distributed Algorithms Simulation Terrain, eine Plattform zur Implementierung verteilter Algorithmen [?]
DSM	Distributed Shared Memory
AC	Atomic Consistency (dt.: Linearisierbarkeit)
RC	Release Consistency (dt.: Freigabekonsistenz)
SC	Sequential Consistency (dt.: Sequentielle Konsistenz)
WC	Weak Consistency (dt.: Schwache Konsistenz)

## B

---

### Selbstständigkeitserklärung

- ☐ Diese Arbeit habe ich selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.
- ☐ Diese Arbeit wurde als Gruppenarbeit angefertigt. Meinen Anteil habe ich selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Namen der Mitverfasser:

Meine eigene Leistung ist:

---

Datum

---

Unterschrift der Kandidatin/des Kandidaten