Documentation

Introduction

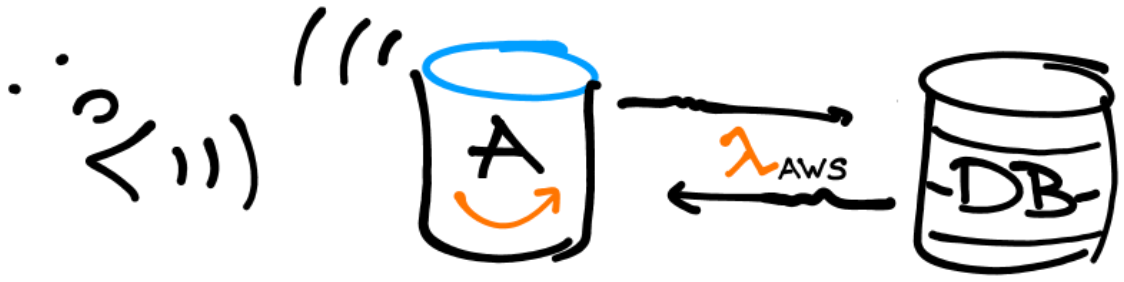
During the Synpulse Hackathon 2019, we were challenged with creating an Alexa Skill – *Dr. Who* – that helps users make appointments with a medical practitioner suiting their needs, while at the same time, making it an easy and comfortable experience. Taking advantage of the Amazon Echo, we focused on creating a flexible and responsive algorithm, capable of adapting to the user's wishes. It had to be able to capture as quickly as possible the essence of the need, without sacrificing the quality of its solution.

Procedure

As a first step, we had to make sure that we understood the core of our problem and what our minimal viable product would look like. For this we identified three core questions to address:

1. What output is essential to resolving the need?
2. What is the minimal input we need from the user?
3. How can we process this data as efficiently as possible?

For the first question we came to the conclusion, that we needed at least the name of the medical practitioner, their speciality and their location and available time slots.
The answer to the second question is directly related to the first one: speciality, location, when should the appointment take place.
For the third question, we came up with a scheme that would allow us to work efficiently and with possibilities for scalability. The user gives a voice input that activates Alexa. Alexa captures the input and sends it to a lambda function, which is in communication with a database. Alexa will continue asking questions until it can give a useful output.



Variables in the data base

Data base variables are:

- Name of the medical practicioner (name)
- medical speciality (type) [possibilities: orthopedist, cardiologist, general physician, emergency section]
- postal code (PLZ) [possibilities: 8046, 8047, 8048, 8049, 8050]

- time interval of availability (start_time and end_time).

We implemented an AuroraDB with MySQL compatibility, which are attached to . Our aim was to link AWS-lambda function to the AuroraDB by using AWS-CLI. Unfortunately, AWS doesn't have a built-in MySQL communication capability and we failed to implement it ourselves. Instead, we developed the following table as a basical data base and implemented in it directly in the AWS-lambda function. In order to save time, we had to keep the structure very simple. Therefore, we defined an availability time interval for each doctor, instead of creating an array of possible time slots.

id	name	type	plz	start_time	end_time
1	Dr. Müller	orthopedist	8046	2019-04-15	2019-06-30
2	Dr. Lu	cardiologist	8050	2019-04-20	2019-08-02
3	Dr. Frankenstein	general physician	8047	2019-05-02	2019-08-20
4	Dr. Rappen	general physician	8050	2019-04-15	2019-05-02
5	Dr. Mayer	general physician	8048	2019-04-20	2019-08-01
6	Dr. Herz	cardiologist	8049	2019-05-02	2019-07-10
7	Dr. Wiemer	general physician	8046	2019-04-22	2019-07-11
8	Dr. Boo	general physician	8047	2019-04-22	2019-06-21
9	Dr. Who	general physician	8049	2019-06-01	2019-06-20
10	Dr. Simic	emergency section	8047	2019-04-15	2019-08-02

Structure of the interaction

Alexa is implemented as follows:
It captures the information regarding the medical speciality (type), postal code (plz) and time interval of availability (start_time and end_time). If any of these variables is missing, Alexa asks for the missing variables. As mentioned in the previous section, the algorithm of finding

a possible appointment is implemented in AWS-lambda directly. The assigned appointment is a random date, chosen from the intersection of the doctor's time interval of availability and the time interval given by the user.

The original idea was to implement Alexa in a way that she gets all possible appointment from the data base and in case they are too many, she finds the most suitable appointment by asking further information. To this purpose, we defined different intents, which has only one mandatory slot.

User manual

1. Open the skill by commanding "Alexa, open Dr. Who"

2. Ask for an appointment by using one of these phrases:

    1. I need a doctor

    2. Book an appointment with a {type} in {plz} from {start_time} to {end_time}. Definition of each slot is limited to the possibilities given in the table.

Alexa asks for the missing slots, if one or some of them are missing and finds and confirms your appointment with a specific doctor at a given place and an available time.

Source Code

Lambda function
https://github.com/kromerh/synpulse-hackathon/blob/master/lambda.py

JSON-File
https://github.com/kromerh/synpulse-hackathon/blob/master/heiko.Alexa_skill.json