

Chatbot – Sprawozdanie

Krzysztof Gądek

W ramach zadania został stworzony chatbot za pomocą technologii dialogflow. Chatbot potrafi umawiać na spotkania dwie wybrane osoby w wybranym czasie, oraz potrafi wylistować spotkania danej osoby na następny miesiąc. Spotkania są zapisywane w zewnętrznej bazie danych.

Założenia

Chatbot został stworzony dla hipotetycznej firmy, w której pracuje pięć osób:

Paweł Mrozowski, Mateusz Olszewski, Lena Pawłowska, Cezary Sobolewski, Iga Kowalska. Tylko dla tych 5 osób można wykonywać akcje, jeżeli użytkownik poda inną osobę, chatbot zwróci informację, że nie znalazł takiej osoby w bazie danych.

Zewnętrzne API

Dla tego projektu, został stworzony przeze mnie mikroserwis umieszczony na cloudzie heroku. Serwis został stworzony w technologii Ruby on Rails i jego kod wygląda następująco:

```
# frozen_string_literal: true
```

```
class MeetingsController < ApplicationController  
  skip_before_action :verify_authenticity_token
```

```
  def handleDialogFlowRequest  
    #Meeting.create time: DateTime.now + 2.day, invited_person: 6, invitor: 4  
    error_flag = false  
    initialize_variables  
    response = ""  
    data = extract_data_from_request_body params  
    puts data  
    if !data[0].nil? && !data[1].nil? && !data[2].nil?  
      unless person_valid? data[0]  
        response = "Niestety nie znalazłem takiej osoby w bazie jak #{data[0]}. Spróbuj poprosić o spotkanie jeszcze raz,  
        pamiętając aby wpisać pełne imiona oraz nazwiska :)"  
        error_flag = true  
      end  
      puts error_flag  
      if !error_flag && (!invitor_valid? data[1])
```

```

    response = "Niestety nie znalazłem takiej osoby w bazie jak #{data[1]}. Spróbuj poprosić o spotkanie jeszcze raz,
    pamiętając aby wpisać pełne imiona oraz nazwiska :)"
    error_flag = true
  end
  puts error_flag
  if !error_flag && (!date_valid? data[2])
    response = "Proponowany termin jest już zajęty przez osobę o imieniu #{@colliding_person.name}. Mają umówione
    spotkanie tego dnia na godzinę #{@colliding_date.strftime('%H:%M')}. Spróbuj poprosić o spotkanie z tą osobą w
    późniejszym terminie :)"
    error_flag = true
  end
  puts error_flag
  unless error_flag
    .create time: data[2], invitor: @invitor.id, invited_person: @invited_person.id
    response = 'Dobra, spotkanie zostało umówione :)'
  end
  elsif !data[3].nil?
    if !error_flag && (!invitor_valid? data[3])
      response = "Niestety nie znalazłem w bazie takiej osoby jak #{data[3]}. Spróbuj wysłać prośbę ponownie, pamiętając
      aby podać poprawnie imię i nazwisko :)."
      error_flag = true
    end
    unless error_flag
      meetings1 = Meeting.where({ time: (Time.now)..(Time.now + 1.month), invitor: @invitor })
      meetings2 = Meeting.where({ time: (Time.now)..(Time.now + 1.month), invited_person: @invited_person })
      meetings = meetings1 + meetings2

      if meetings.empty?
        response = 'Wygląda na to, że przez najbliższy miesiąc nie masz umówionych spotkań :)'
      else
        response = 'Na najbliższy miesiąc masz w planie następujące spotkania: \n'
        meetings.each_entry do |meeting|
          person = if meeting.invitor == @invitor.id
            meeting.invited_person
          else
            meeting.invitor
          end
          puts person.id
          response += "#{person.name} dnia #{meeting.time.strftime('%Y-%m-%d %H:%M')} "+'\n'
        end
      end
    end
  end
  render json: transform_json_response(response)
end

def create_people
  Person.create name: 'Paweł Mrozowski', position: 'Frontend Developer'
  Person.create name: 'Mateusz Olszewski', position: 'CEO'
  Person.create name: 'Cezary Sobolewski', position: 'Scrum Master'
  Person.create name: 'Lena Pawłowska', position: 'UX/UI Designer'
end

```

```

    @person.create name: 'Iga Kowalska', position: 'HR head'
  end

  private

  def transform_json_response(response)
    string_json = "{\"fulfillmentText\": \"#{response}\", \"fulfillmentMessages\": [{\"text\": {\"text\": \"#{response}\"}}]}"
    @response.parse(string_json)
  end

  def extract_data_from_request_body(params)
    context = params['queryResult']['outputContexts']
    date = context.detect { |e| e['name'].include? '/day' }
    if !date.nil? && !date['parameters']['date-time']['date_time'].nil?
      date = date['parameters']['date-time']['date_time']
    else
      date = date['parameters']['date-time'] unless date.nil?
    end
    person = context.detect { |e| e['name'].include? '/person' }
    person = person['parameters']['person']['name'] unless person.nil?
    inviter = context.detect { |e| e['name'].include? '/me' }
    inviter = inviter['parameters']['person_me']['name'] unless inviter.nil?
    date_time = nil
    date_time = date_time.strptime(date, '%Y-%m-%dT%H:%M:%S') unless date.nil?
    person_to_find_meetings = context.detect { |e| e['name'].include? '/listmeetings' }
    unless person_to_find_meetings.nil?
      person_to_find_meetings = person_to_find_meetings['parameters']['person']['name']
    end
    [person, inviter, date_time, person_to_find_meetings]
  end

  def person_valid?(name)
    name = convert_to_polish_nominative(name.downcase)
    if name == 'didn\'t found' && !(@names.include? name)
      false
    else
      puts name
      @invited_person = @names.find { |person| person.name == name }
      puts @invited_person
      true
    end
  end

  def invitor_valid?(name)
    if convert_to_polish_nominative(name.downcase) == 'didn\'t found' && !(@names.include? name)
      return false
    else
      puts name
      @invitor = @names.find { |person| person.name == name }
      puts @invitor
      return true
    end
  end

```

```

    end
end

def date_valid?(date)
  colliding_date = @invited_person.where({ time: (date - 30.minutes)..(date + 30.minutes) })
  unless colliding_date.empty?
    @colliding_person = @invited_person.find(colliding_date[0].invitor)
    @colliding_date = colliding_date[0].time
    return false
  end
  true
end

def initialize_variables
  @names = ['Paweł Mrozowski', 'Mateusz Olszewski', 'Lena Pawłowska', 'Cezary Sobolewski', 'Iga Kowalska']
  @colliding_person = nil
  @colliding_date = nil
  @invitor = nil
  @invited_person = nil
end

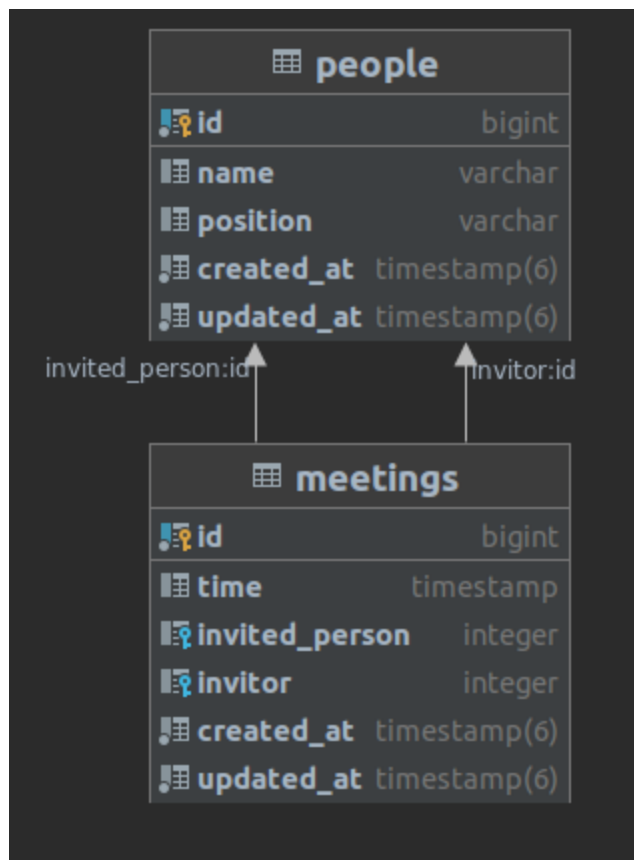
def convert_to_polish_nominative(name)
  case name
  when 'pawłem mrozowskim'
    'Paweł Mrozowski'
  when 'mateuszem olszewskim'
    'Mateusz Olszewski'
  when 'leną pawłowską'
    'Lena Pawłowska'
  when 'cezarym sobolewskim'
    'Cezary Sobolewski'
  when 'igą kowalską'
    'Iga Kowalska'
  else
    'didn\'t found'
  end
end

end
end

```

Serwis zawiera jeden endpoint, na który request wysyła chatbot.

Baza danych została postawiona na silniku PostgreSQL I zawiera dwie tabele:



Intents

Aby móc przeprowadzać rozmowy, zostały utworzone następujące intenty:

● Default Welcome Intent
● rejestracja-spotkan-osoba-po-czasie
● rejestracja-spotkan.dzien
● rejestracja-spotkan.godzina-po-osobie
● rejestracja-spotkan.jak-sie-nazywasz
● rejestracja-spotkan.od-razu-godzina
● rejestracja-spotkan.od-razu-osoba
● rejestracja-spotkan.wszystko-na-raz
● rejestracja-spotkań
● rejestracja-spotkań.osoba
● wylistuj-spotkania
● wylistuj-spotkania.imie

Każdy z nich będzie odpowiadał za innych fragment rozmowy, według ich nazwy.

Jak przeprowadzić rozmowę?

Chatbot obsługuje dwie sytuacje: gdy osoba chce umówić się na spotkanie oraz gdy osoba chce wiedzieć na jakie spotkania jest umówiona.

Aby umówić się na spotkanie, wystarczy napisać:

“Chcę umówić się na spotkanie”,

następnie chatbot zapyta o osobę i czas. Można również od razu podać czas lub osobę i napisać:

“Chcę umówić się na spotkanie z Cezarym Sobolewskim”,

W takim wypadku chatbot zapyta tylko o czas spotkania.

Można też od razu podać wszystkie dane:

“Umów mnie na spotkanie z Pawłem Mrozowskim w sobotę o godzinie 15:20”

W takim wypadku, chatbot od razu otrzyma wszystkie informacje.

Gdy ta faza przejdzie pomyślnie, chatbot zapyta o imię osoby, która umawia na spotkanie. Trzeba wtedy podać imię i nazwisko (w mianowniku liczby pojedynczej). Jeżeli wszystkie dane zostały podane prawidłowo, chatbot zapisze spotkanie w bazie danych i powiadomi użytkownika o pomyślnym zapisie.

Aby wylistować spotkania, wystarczy zapytać bota:

“Napisz mi, kiedy mam spotkania”, lub:

“Kiedy mam spotkanie?”

Chatbot zapyta imię i nazwisko osoby i gdy zostanie znaleziona w bazie danych, wypisze z kim i kiedy jest umówiona na spotkanie przez najbliższy miesiąc.

Błędy:

Umawianie spotkania nie uda się w następujących przypadkach:

- Nie znaleziono w bazie danych osób o podanych nazwiskach
- Nie podano poprawnego nazwiska lub czasu (zamiast tego np: “asd”)
- Osoba ma już umówione spotkanie w podanym czasie.

W przypadku gdy nie podano poprawnego imienia lub nazwiska, można od razu spróbować podać jeszcze raz, w przypadku pozostałych błędów trzeba rozpocząć rozmowę jeszcze raz.

Przykładowe rozmowy

1. Umówienie się na spotkanie podając wszystkie dane po kolei:

Chcę umówić się na spotkanie

Jasne, z kim mam Cię umówić?

Z Igą Kowalską

Dobrze, a kiedy?

10.11.2021 godzina 15:20

Świetnie, napisz mi jeszcze, jak się nazywasz?

Paweł Mrozowski

Dobra, spotkanie zostało umówione :)

2. Umówienie się na spotkanie podając od razu wszystkie dane:

Umów mnie na spotkanie z Cezarym
Sobolewskim na sobotę o godzinie 16:20

Dobrze, napisz mi jeszcze jak się
nazywasz?

Iga Kowalska

Dobra, spotkanie zostało umówione :)

3. Umówienie się na spotkanie, podając od razu imię i nazwisko:

Umów mnie na spotkanie z Igą Kowalską

Nie ma sprawy, na kiedy?

06.11.2021 godzina 15

Super, napisz mi jeszcze, jak się
nazywasz?

Lena Pawłowska

Dobra, spotkanie zostało umówione :)

4. Nieudana próba umówienia się na spotkanie: termin został już zajęty w rozmowie nr 3

Umów mnie na spotkanie z Leną
Pawłowską w dniu 06.11.2021 o godzinie
15:00

Dobrze, napisz mi jeszcze jak się
nazywasz?

Cezary Sobolewski

Proponowany termin jest już zajęty przez
osobę o imieniu Lena Pawłowska. Mają
umówione spotkanie tego dnia na godzinę
15:00. Spróbuj poprosić o spotkanie z tą
osobą w późniejszym terminie ;)

5. Wylistowanie umówionych spotkań w rozmowach 1-4

Kiedy mam spotkanie?

A jak się nazywasz?

Iga Kowalska

Na najbliższy miesiąc masz w planie
następujące spotkania: Cezary Sobolewski
dnia 2021-11-06 16:20 Paweł Mrozowski
dnia 2021-11-10 15:20 Lena Pawłowska
dnia 2021-11-06 15:00

6. Nieudane umawianie się na spotkanie: użytkownik podał osobę której nie ma w bazie. Niestety występuje problem z odmawianiem polskich przypadków.

umów mnie na spotkanie z Krzysztofem
Gądkiem w dniu 09.11.2021 o godzinie 19

Dobrze, napisz mi jeszcze jak się
nazywasz?

Lena Pawłowska

Niestety nie znalazłem takiej osoby w
bazie jak Krzysztofem Gądkiem. Spróbuj
poprosić o spotkanie jeszcze raz,
pamiętając aby wpisać pełne imiona oraz
nazwiska :)

Inne błędy

Niestety ze względu na to, że cloud heroku czasem działa dość wolno dla osób niepłatących za deployment, może się zdarzyć że odpowiedź nie dotrze na czas do chatbota i będzie widoczna jako [empty response]. Zazwyczaj błąd ten występuje, gdy chatbot długo nie był używany i tylko za pierwszym razem. Często jednak, mimo braku odpowiedzi, spotkanie zostanie umówione pomyślnie, aby to sprawdzić należy zapytać tak jak w scenariuszu rozmowy nr 5.

Link do wypróbowania chatbota

<https://bot.dialogflow.com/bc36146f-f346-4f32-9374-42197015efe9>