Z5145006 Chuguan Tian

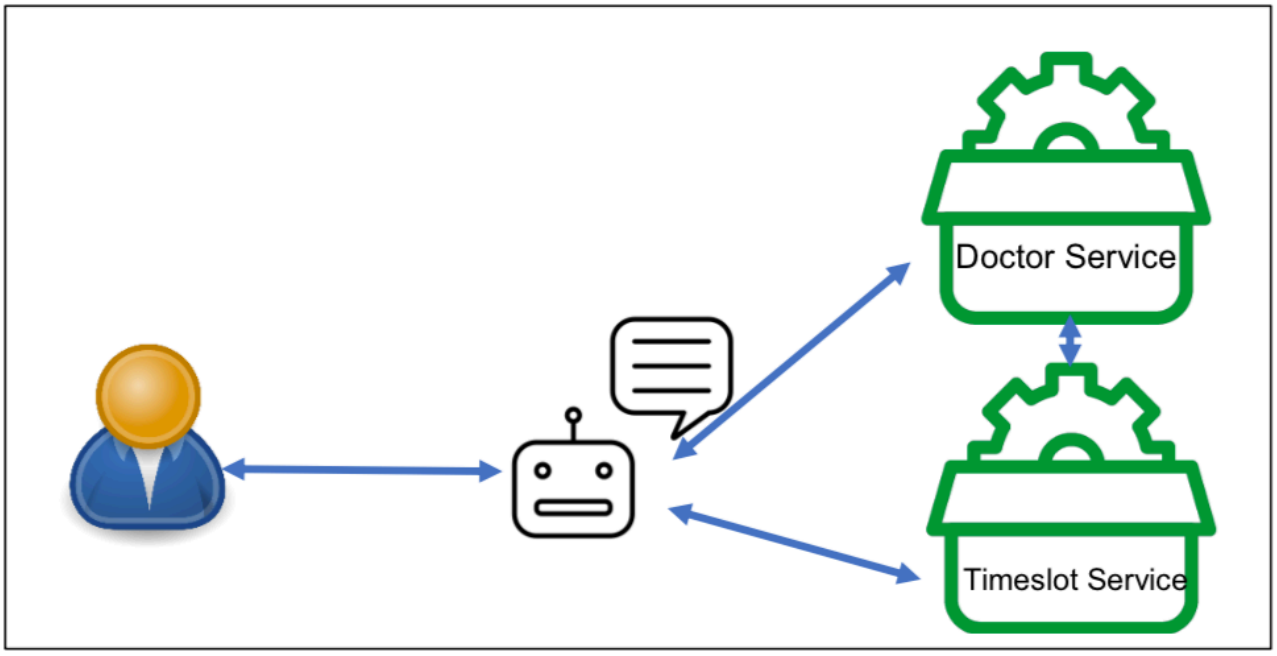
COMP9322 Assignment 1 Report

In this assignment, I am implementing a Chatbot that rely on a set of REST API based Micro-services to perform booking in a dental clinic using web UI based on riaescripts messaging platform.

The scenario here is the patient contact the Chatbot to request for a booking. If the patient needs to specify the doctor or ask for the list of doctors available. After the selection of the dentist the patient needs to specify the preferred timeslot the bot check if the timeslot is available and if not, provide list of available timeslots. The patient will select the timeslot available and the booking is made.

API Implementation (8 Marks)

The system overview



The diagram above shows the high level components of the system you are building. The dentist and Timeslot services are to be deployed in a docker

container and the Bot should communicate with the services through REST API calls.

To summarise the main functionalities:

- Get available dentists
- Get dentist information
- Get available timeslots for each dentist
- Reserve timeslot
- Cancelling appointment

The following describes the API that implemented.

The resources and their URI patterns:

There are two types of resources to be managed: Dentist and Timeslot. Details as below:
Dentist: name, location, specialization (e.g., Paediatric Dentistry, Orthodontics, Oral Surgery...etc.)
Timeslots: Date, 1 hour timeslots from 9:00 AM to 5:00 PM, status flag (reserved, available)

Managing resources:

I use PUT, POST, DELETE, GET function to fulfil the function

Persisting resources:

I used mongoDB to store the data in Json format.

RESTful APIs are stateless:

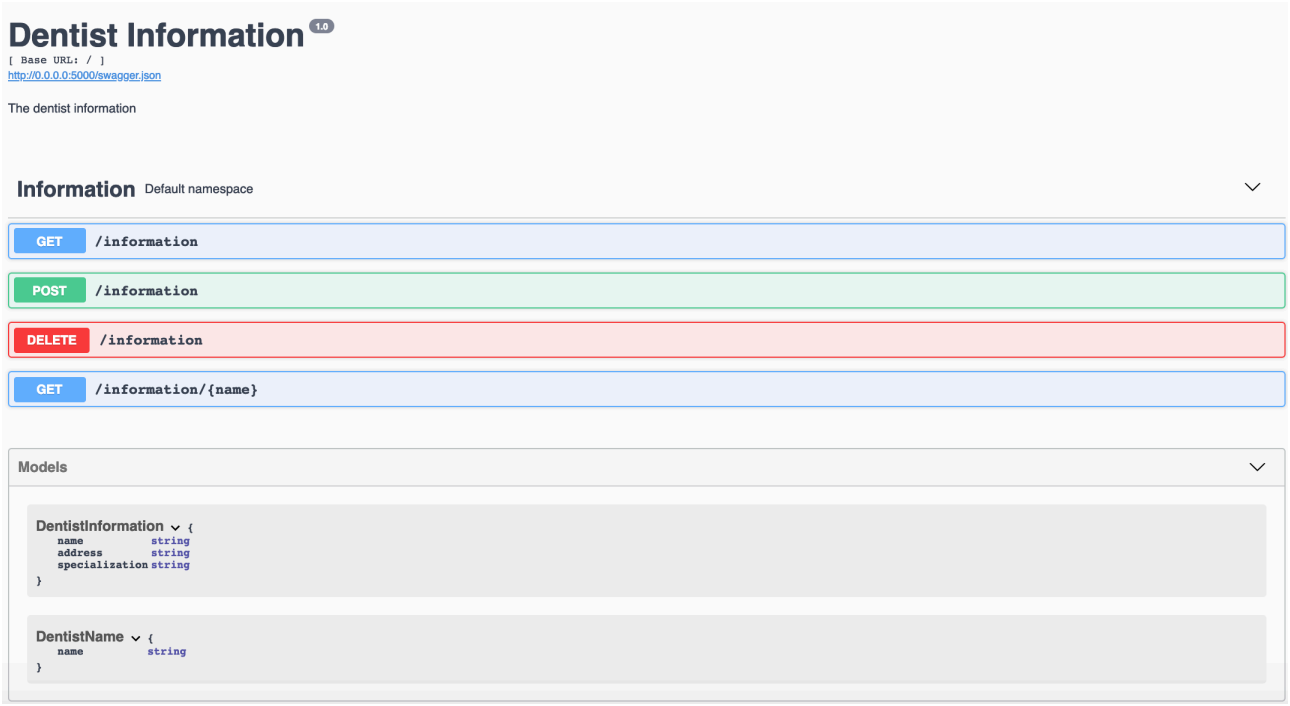
The APIs show that your APIs are stateless and all communications are self-contained with the messages.

Z5145006 Chuguan Tian

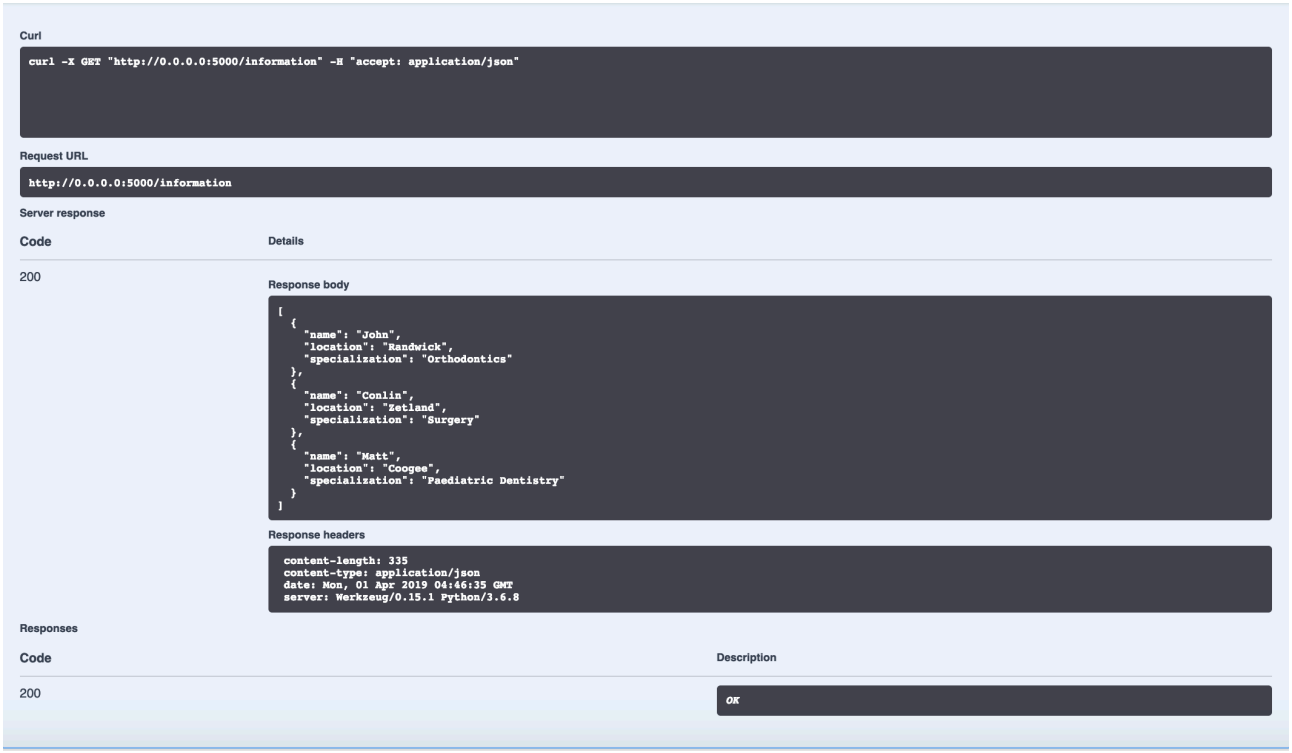
I divide these functions into two restplus server.

Dentist Information Server

This server store the information of the dentist, it could get all the available dentist information, add a dentist information, delete a dentist information and get the specific information of a dentist.

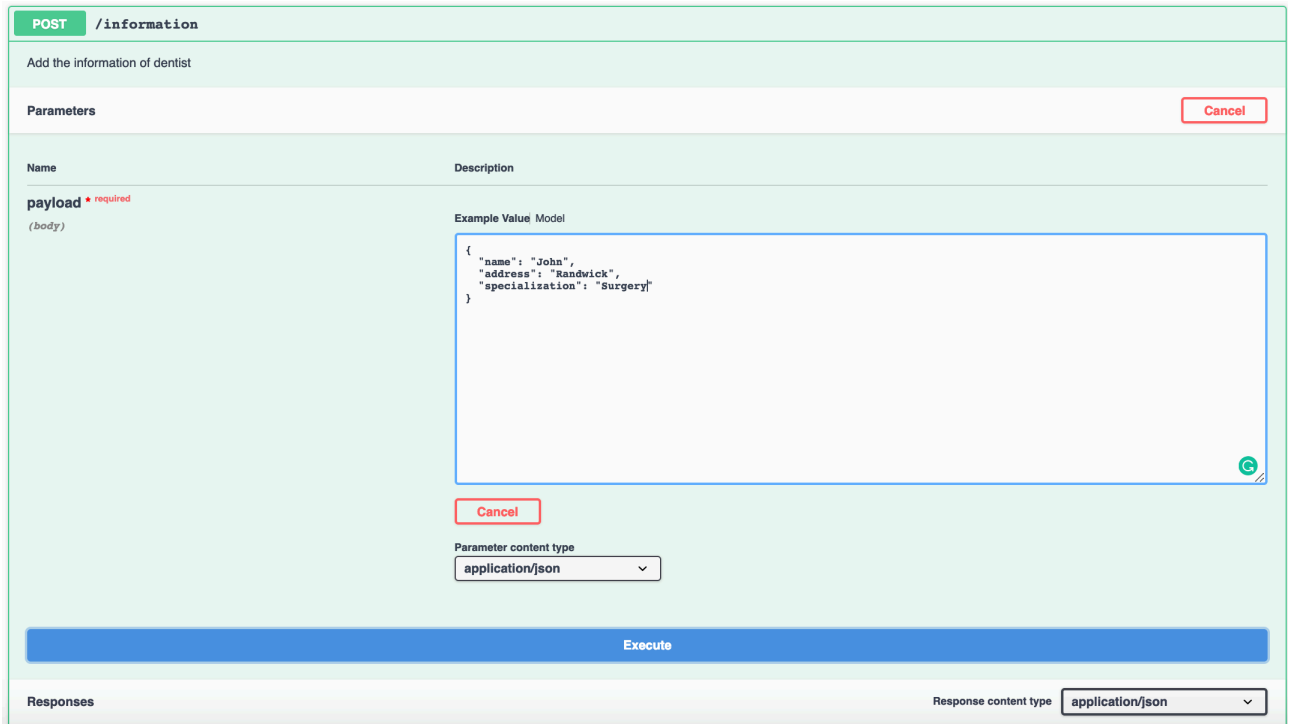


GET all the information of the dentist

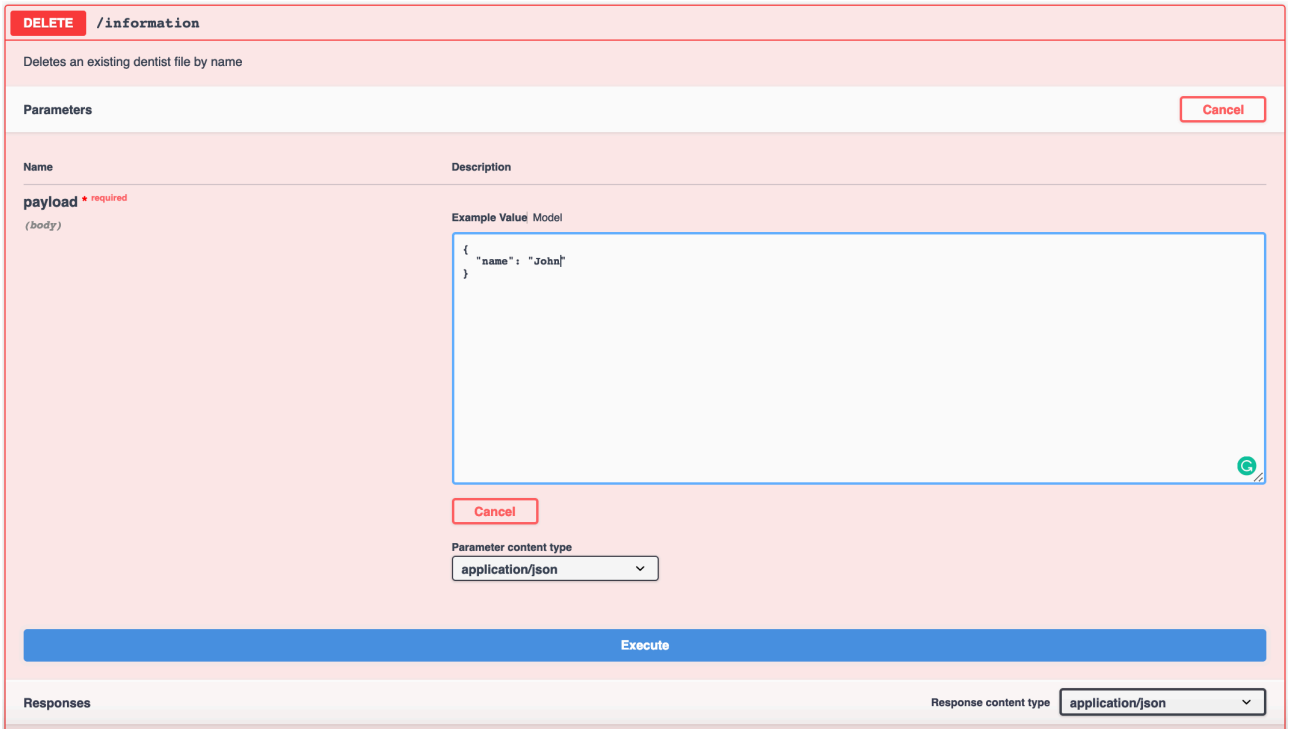


POST new dentist information

Z5145006 Chuguan Tian

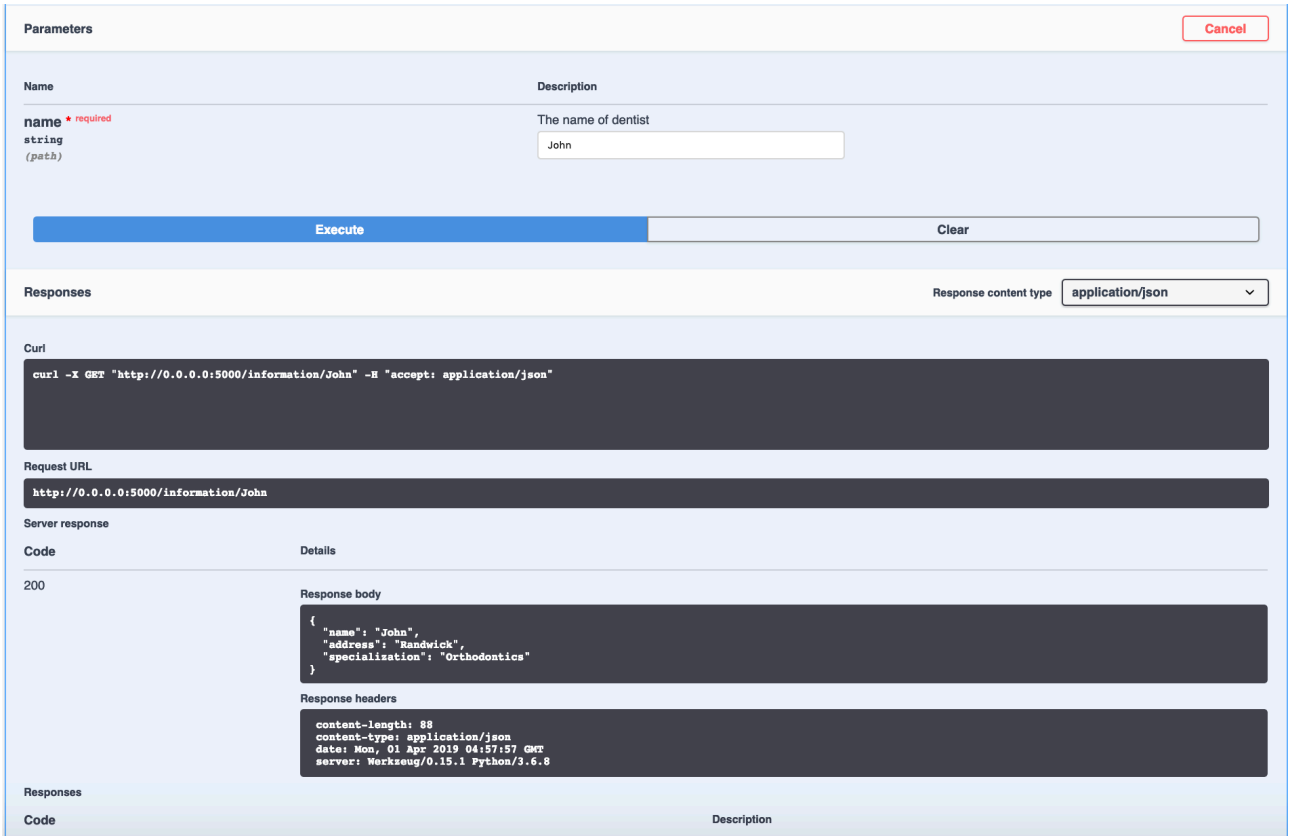


DELETE a dentist information by its name



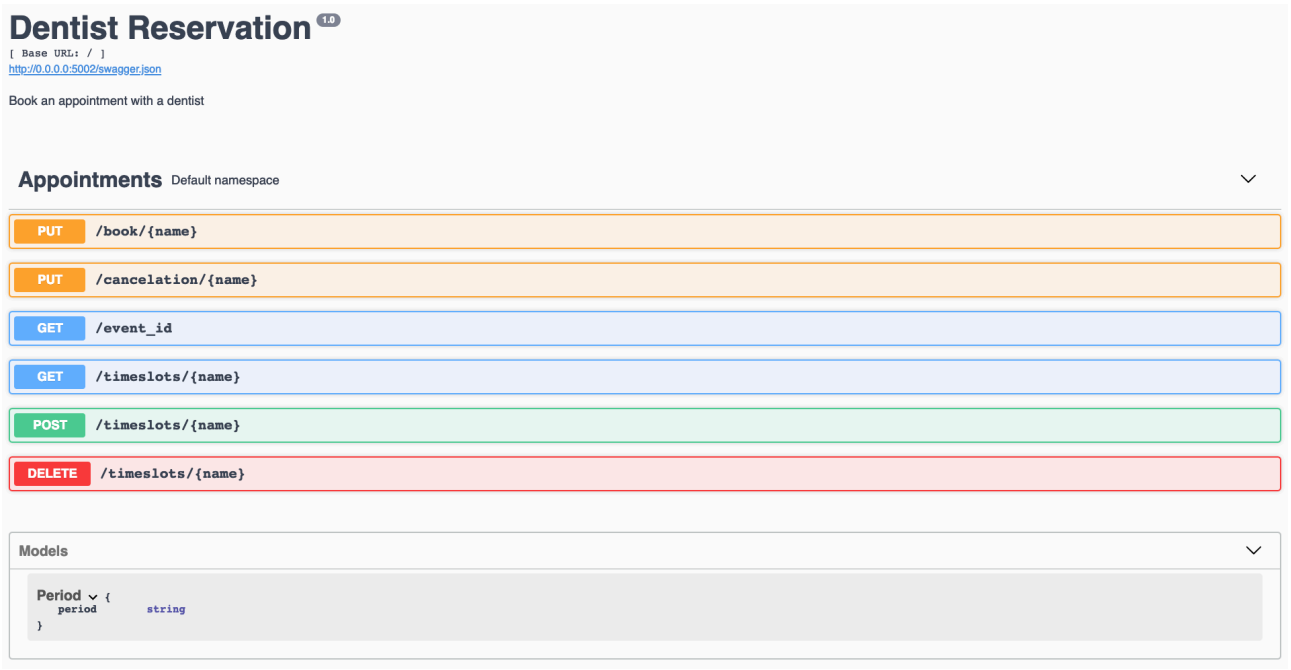
GET specific information of a dentist by its name

Z5145006 Chuguan Tian



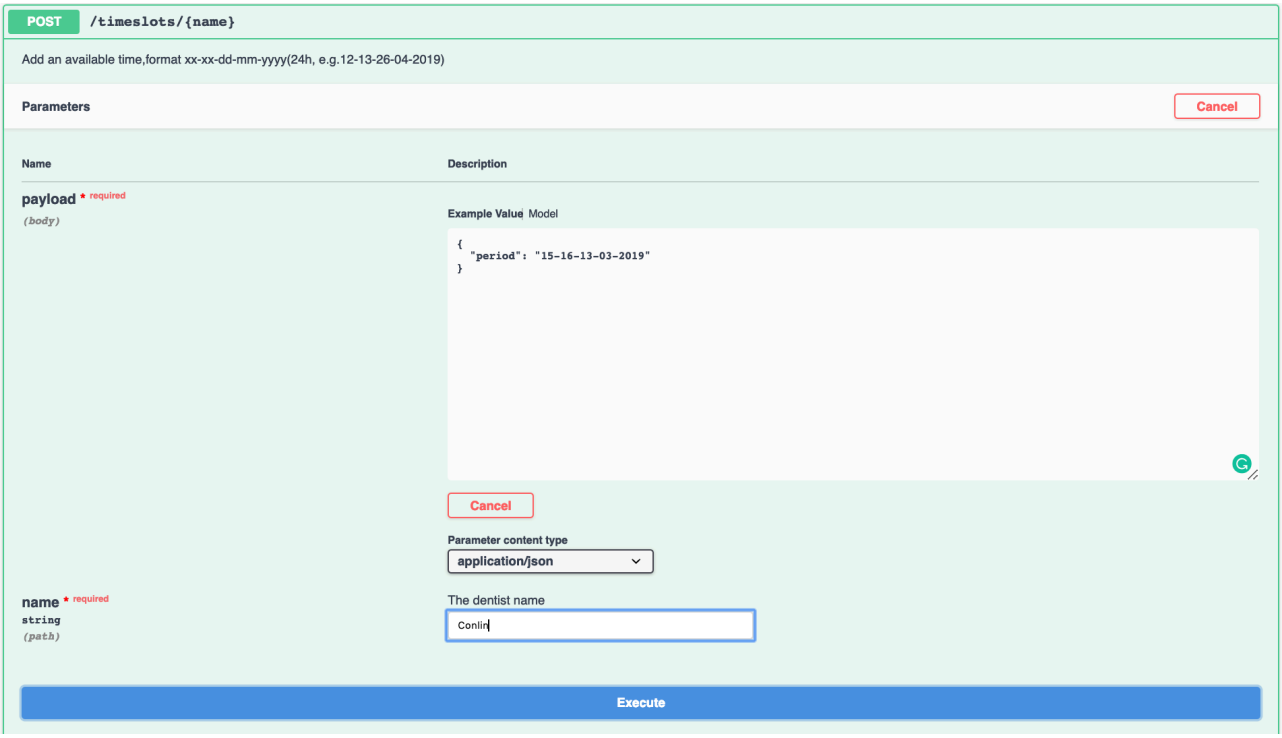
The Appointment Booking Server

It is a server for booking appointment which has the functions: post a new time slot by the dentist name, get the time list by the dentist name, delete a timeslot by the dentist name, book a time slot by the dentist name and cancel a time slot by the dentist name.

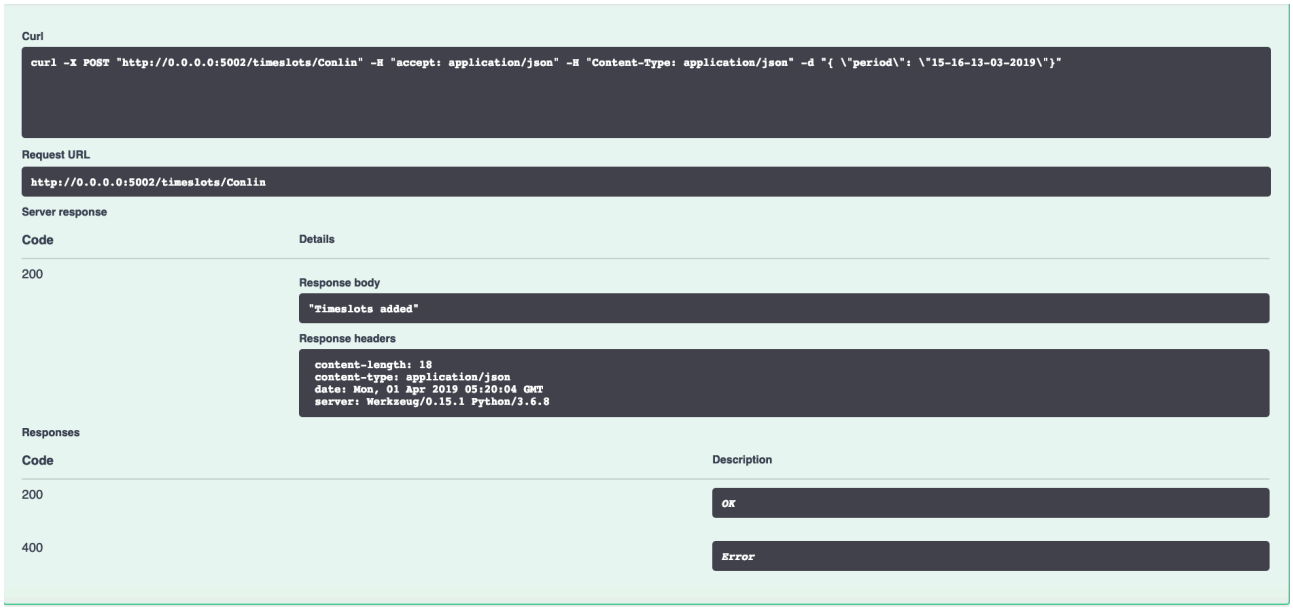


POST a new time slot by the dentist name

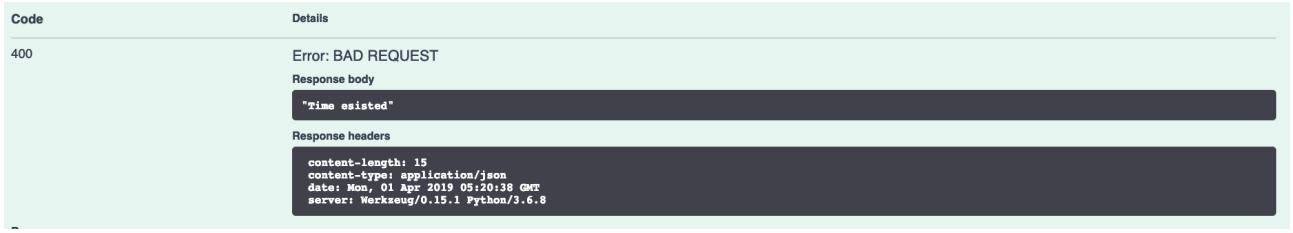
Z5145006 Chuguan Tian



Time slot added



When execute again, time existed.



When the name is not in the database, then give available name.

Z5145006 Chuguan Tian

Curl

```
curl -X POST "http://0.0.0.0:5002/timeslots/Conlinn" -H "accept: application/json" -H "Content-Type: application/json" -d "{ \"period\": \"15-16-13-03-2019\"}"
```

Request URL

```
http://0.0.0.0:5002/timeslots/Conlinn
```

Server response

Code	Details

400	Error: BAD REQUEST
	Response body
	`"Sorry, no Conlinn dentist here. Mayebe you mean John,Conlin,Matt,"`
	Response headers
	`content-length: 68`
	`content-type: application/json`
	`date: Mon, 01 Apr 2019 05:26:47 GMT`
	`server: Werkzeug/0.15.1 Python/3.6.8`

Responses

Code	Description
200	*OK*
400	*Error*

GET the time list by the dentist name

GET /timeslots/{name}

Available time of this dentist

Parameters Cancel

Name	Description
name * required string (path)	The dentist name John

Execute Clear

Responses Response content type application/json

Curl

```
curl -X GET "http://0.0.0.0:5002/timeslots/John" -H "accept: application/json"
```

Request URL

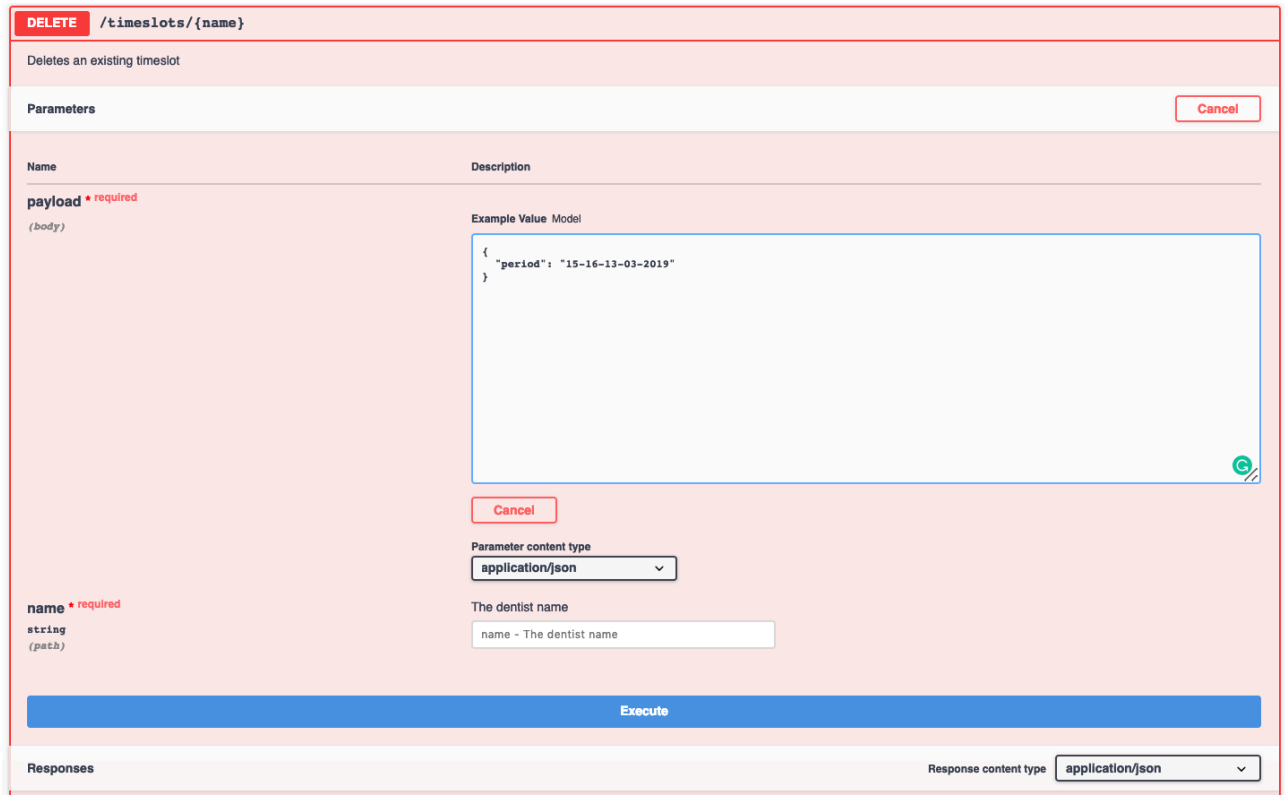
```
http://0.0.0.0:5002/timeslots/John
```

Server response

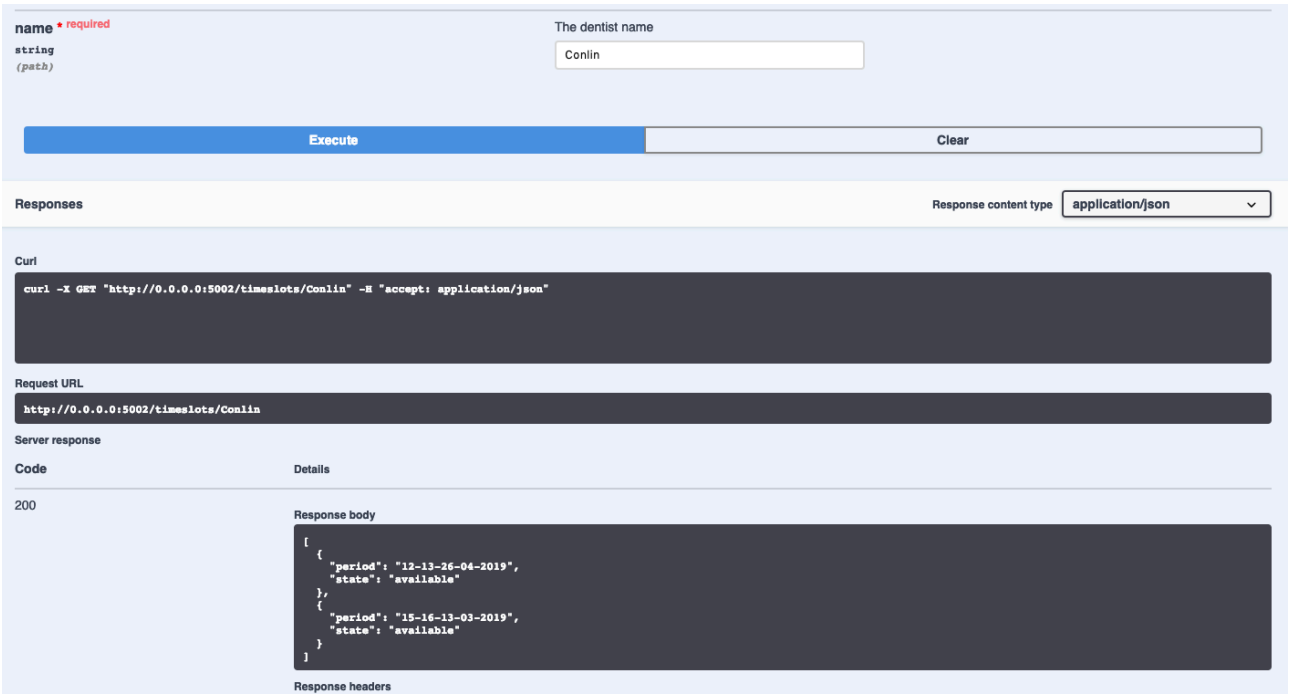
Code	Details
200	Response body `[ { "period": "12-13-26-04-2019", "state": "reserved" }, { "period": "14-15-26-04-2019", "state": "reserved" }, { "period": "14-15-26-04-2019", "state": "available" }, { "period": "15-16-20-04-2019", "state": "available" }, { "period": "12-13-19-04-2019", "state": "available" } ]` Response headers

Delete a timeslot by the dentist name

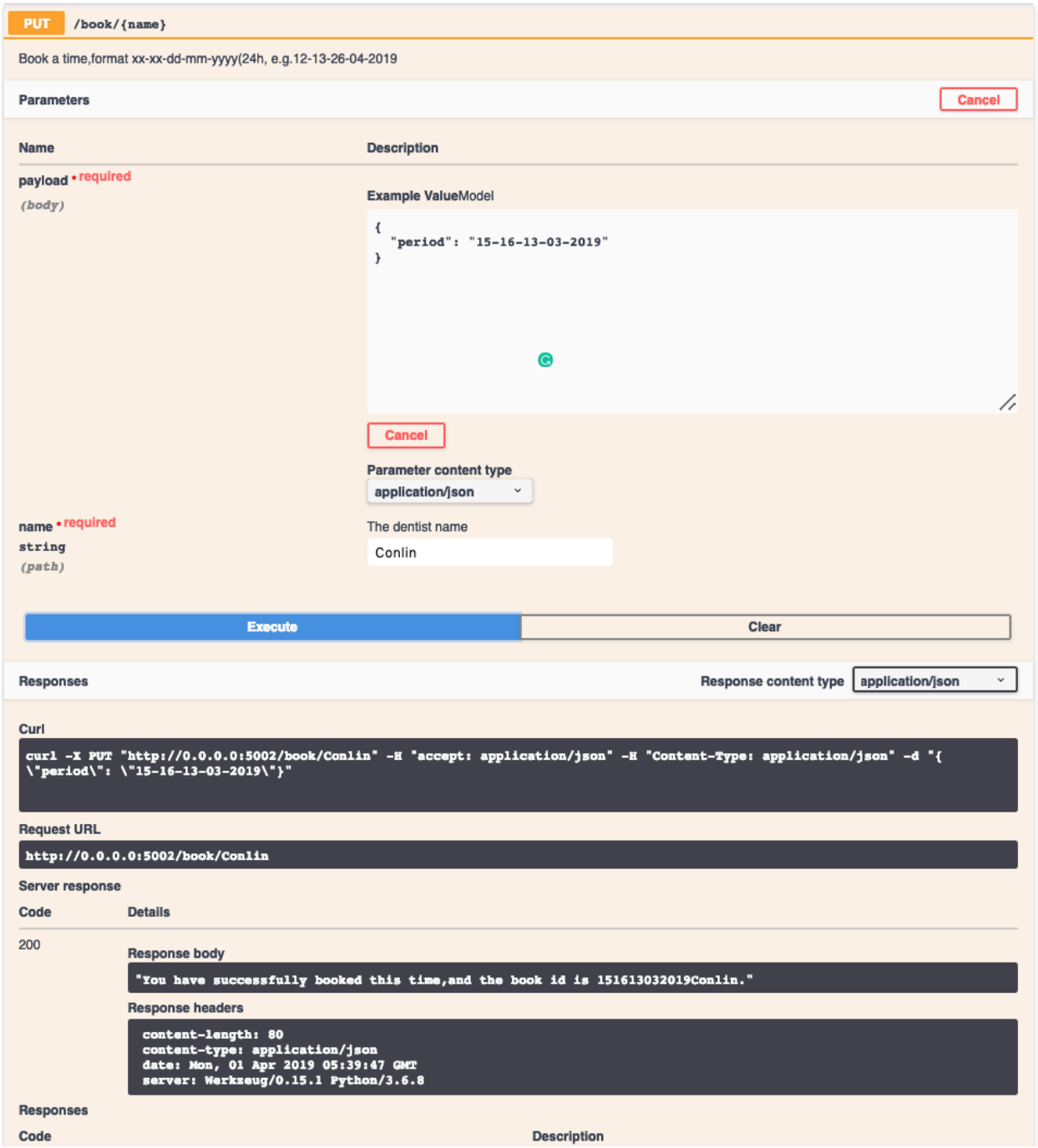
Z5145006 Chuguan Tian



PUT book a time slot by the dentist name



Z5145006 Chuguan Tian

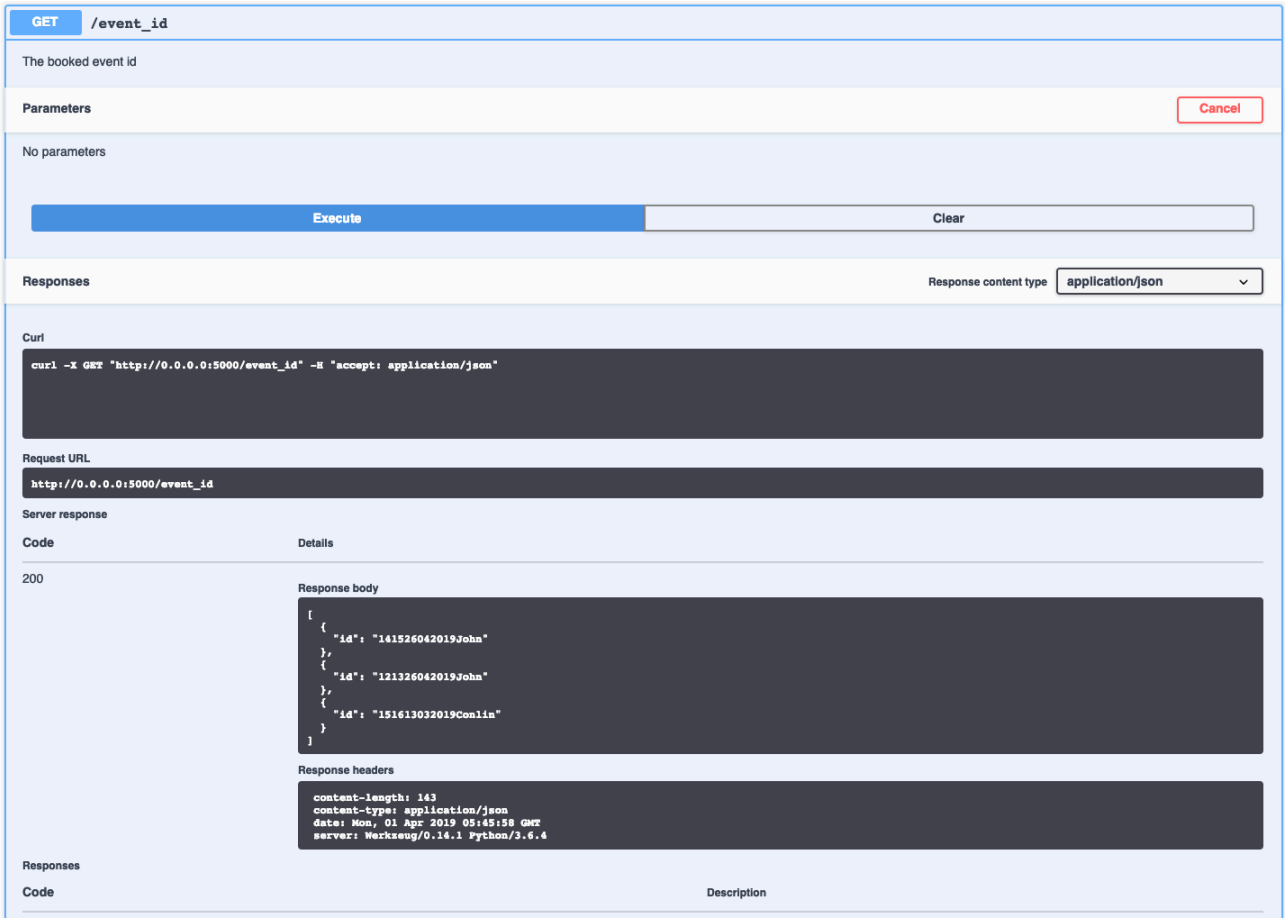


After the book action the book action, the state of this time slot will change to reserved.

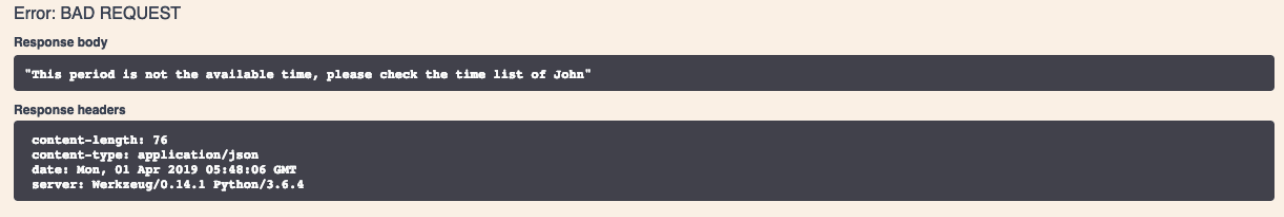

```
[
  {
    "period": "12-13-26-04-2019",
    "state": "available"
  },
  {
    "period": "15-16-13-03-2019",
    "state": "reserved"
  }
]
```


And an event log will be sent to the event id data collection.

Z5145006 Chuguan Tian

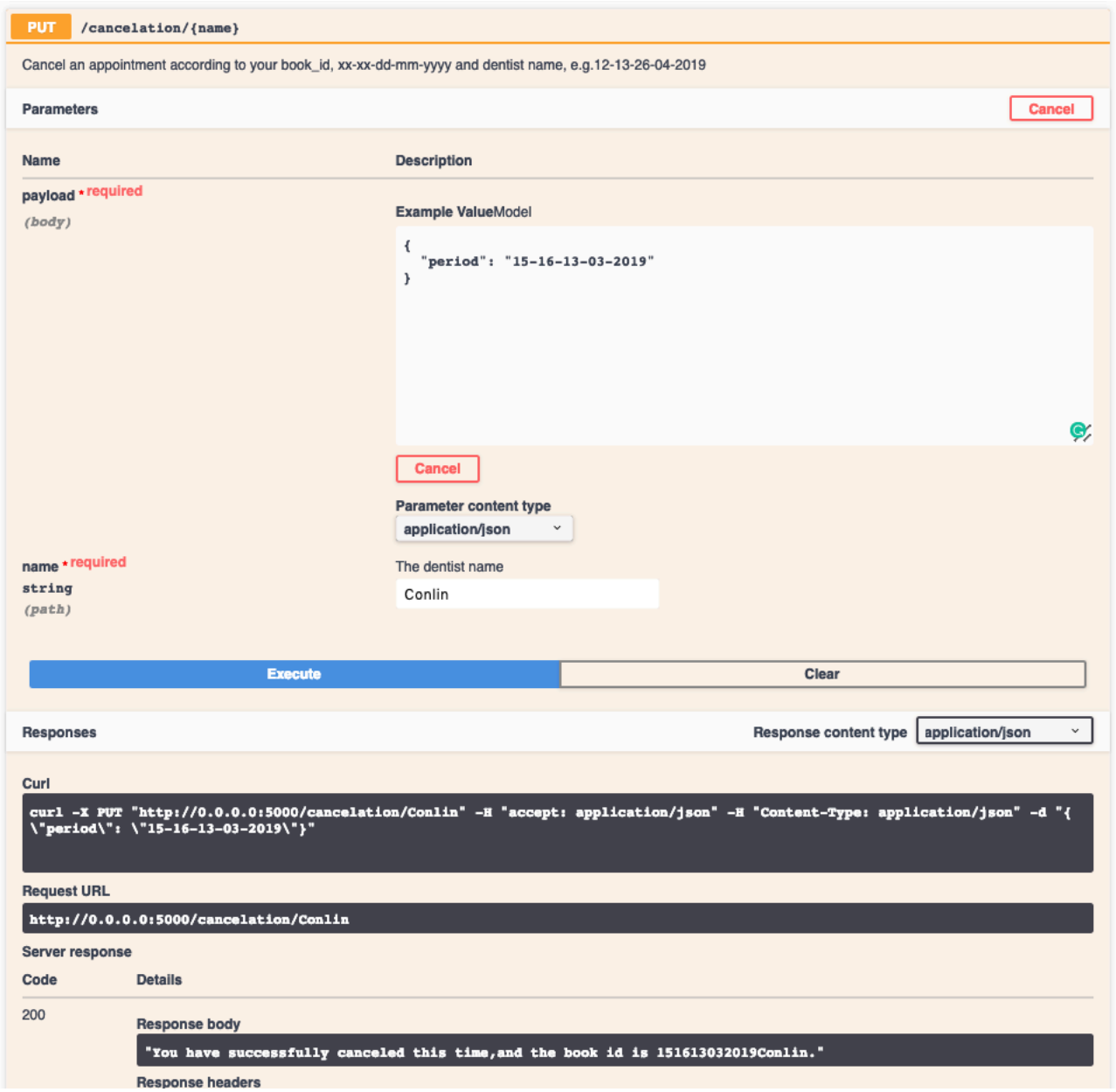


When the time slot is not in the list of this dentist

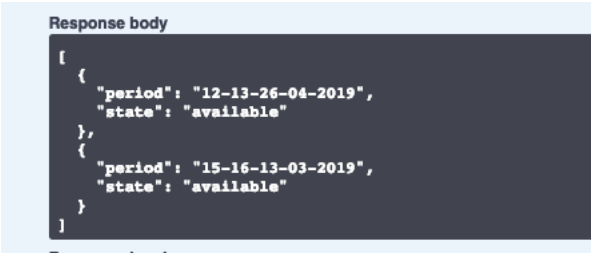


PUT cancel a time slot by the dentist name.

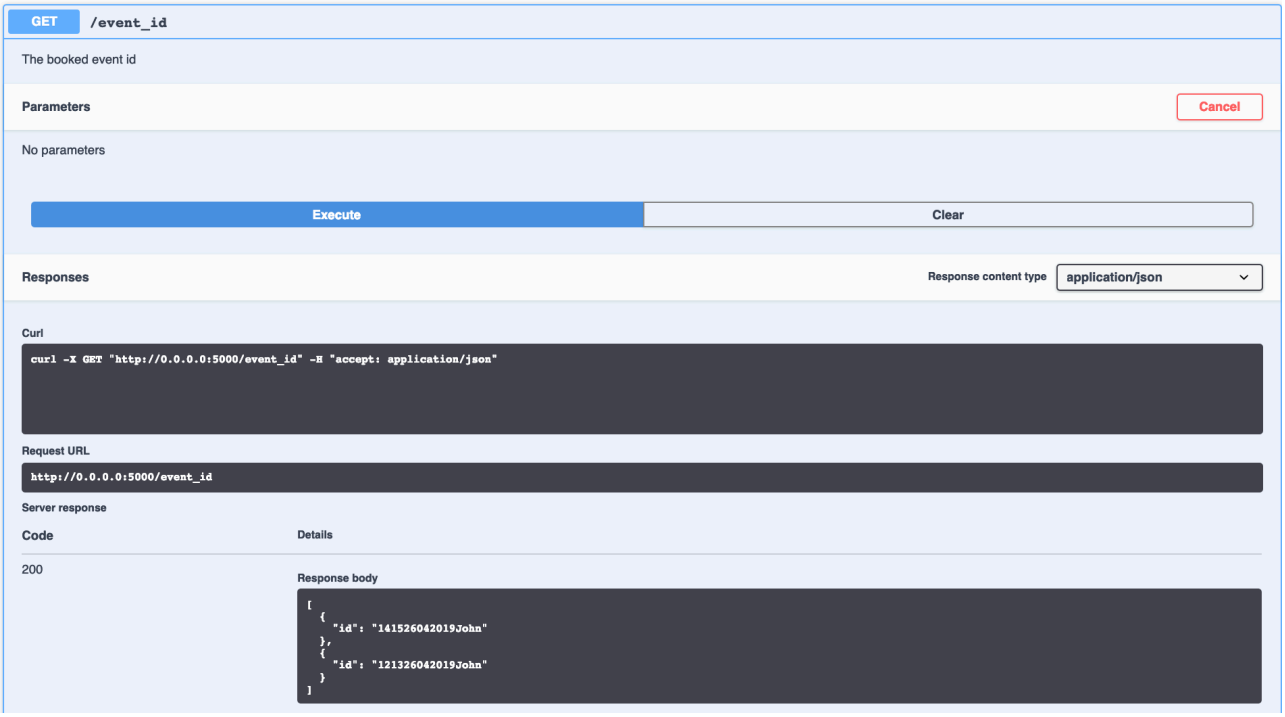
Z5145006 Chuguan Tian



The time state in this dentist's time list will change to available.



And the book log in event ID collection will also be deleted.



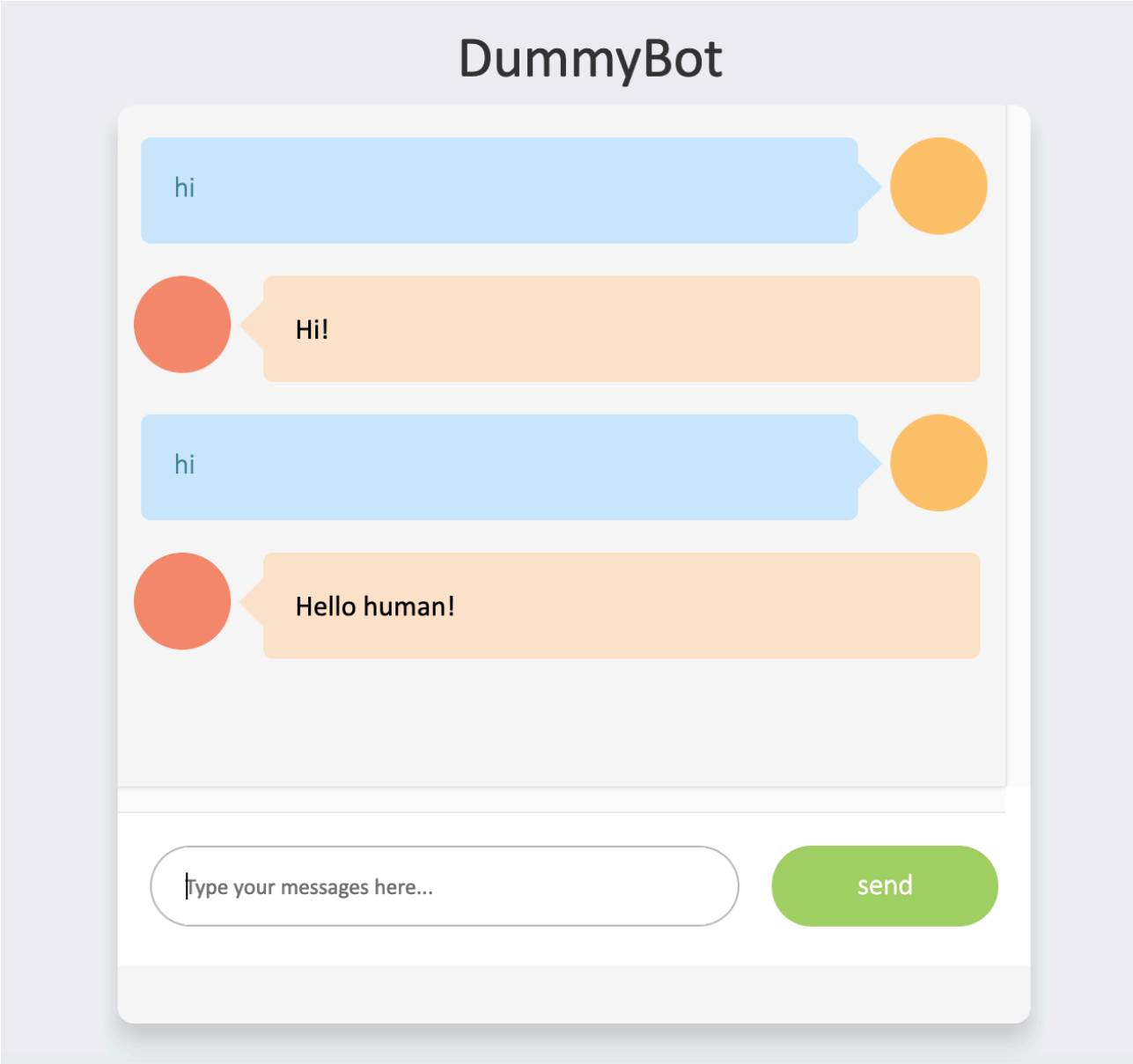
Bot Interaction

- The bot should be able to respond to basic greetings
- The bot asks the client for the preferred doctor and provide information about the doctor
- The bot can list all the available doctors in the clinic and the client can choose
- The bot can check if the selected timeslot is already reserved and suggest another timeslot
- The bot can provide a list of available timeslots for the selected doctor
- The bot can confirm the booking and summarize at the end.
- The bot can cancel the booking if the client requested it and ask for confirmation.

The bot system: rivescript

The UI: web UI

Z5145006 Chuguan Tian



Z5145006 Chuguan Tian

Documentation

Dentist Information Server

The path of dentist information server: ../Dentist/app/demo/api/dentist.py

The swagger doc of dentist information server: ../Dentist/app/demo/static/swagger.json

The requirements file of dentist information server: ../Dentist/app/requirements.txt

The docker file of dentist information server: ../Dentist/Dockerfile

Timeslots Server

The path of dentist timeslots server: ../Timeslots/app/demo/api/timeslots.py

The swagger doc of timeslots server: ../Timeslots/app/demo/static/swagger.json

The requirements file of timeslots server: ../Timeslots/app/requirements.txt

The docker file of timeslots server: ../Timeslots/Dockerfile

Bot

The path of swagger version bot server: ../Chatbot/app/demo/api/swagger_bot.py

The swagger doc of swagger version bot server:../Chatbot/app/demo/api/static/swagger.json

The path of Web-UI version bot server: ../Chatbot/app/demo/api/ui_bot.py

The requirements file of bot: ../Chatbot/app/requirements.txt

The rivescript file of bot: ../Chatbot/app/demo/api/brain/rules.rive

Running Instruction

1.Go to the right path of dentist timeslots server, input theses lines:
pip3 install -r requirements.txt
docker build -t dentist .
docker run -p 5003:5000 -t dentist

2.Go to the right path of timeslots server, input theses lines:
pip3 install -r requirements.txt
docker build -t timeslot .
docker run -p 5002:5000 -t timeslot

3.Go to the right path of bot, input theses lines:
pip3 install -r requirements.txt
python3 ui_bot.py

4.The Sample Video will be attached about the implementation.

Z5145006 Chuguan Tian

Reference of the bot chat (rivescript):

```
+ hi bot
- hi user!

+ how are you
- I am good, how are you?

+ where are you from
- I am from my country :)

+ hi
- Hello human!
- Hello!
- Hi there!
- Hey!
- Hi!

+ hello bot
- Hello human!
- Hello!
- Hi there!
- Hey!
- Hi!

+ my name is _
- oh hey <star>

+ (i am happy|i am excited|i am thrilled)
- I am happy too for you.

+ (what|which) (doctor|dentist) [do] [you] have
+ (what|which) dentist [can|should] i choose here
- intent=dentistName

+ which dentist should i (choose|book)
- intent=dentistInfo

+ tell me about something
- intent=dentistInfo

+ tell me about these [dentist|doctor|dentists|doctors]
- intent=dentistInfo

+ i want to know [something] [about] [dentist|dentists]
- intent=dentistInfo

+ i want to know [something] [about][dentist|doctor] *
- intent=dentistSinInfo,entity=<star>

+ tell me about [dentist|doctor] *
- intent=dentistSinInfo,entity=<star>
```

```
+ [could] you tell me about [dentist|doctor] *
- intent=dentistSinInfo,entity=<star>

+ can i know the available time of *
- intent=timetable,entity=<star>

+ which time is available of *
- intent=timetable,entity=<star>

+ tell me about the [available] time of *
- intent=timetable,entity=<star>

+ what is the available time of *
- intent=timetable,entity=<star>

+ what is the [available] timetable of *
- intent=timetable,entity=<star>

+ tell me about the [available] timetable of *
- intent=timetable,entity=<star>

+ i want to book [the time] at * with * [please]
- intent=book,entity1=<star1>,entity2=<star2>

+ [could|can] you help [me] [to] book the time at * with *
[please|pls]
- intent=book,entity1=<star1>,entity2=<star2>

+ i want to book with * at [the time] * [please]
- intent=book,entity1=<star2>,entity2=<star1>

+ i want to cancel the time at * with * [please]
- intent=cancel,entity1=<star1>,entity2=<star2>

+ [could|can] you help [me] [to] cancel the time at * with *
[please|pls]
- intent=cancel,entity1=<star1>,entity2=<star2>

+ i want to cancel with * at [the time] * [please]
- intent=cancel,entity1=<star2>,entity2=<star1>

+ is the time at * with * available
- intent=checktime,entity1=<star1>,entity2=<star2>

+ can i book the time at * with *
- intent=checktime,entity1=<star1>,entity2=<star2>

+ what is the weather [like] [today]
- It is a bright day, you could take a walk outside.

// Capture the user's name: letters only!
+ my name is _
```

```
- It's nice to meet you, <star>.
- <star>, nice to meet you.
- Pleased to meet you, <star>.

// What if the user says "my name is 5"? 5 isn't a real name!
+ my name is #
- Nobody has the name of <star>.
- <star> isn't a real name.
- Names don't have numbers in them, <star>.

// If they say their name is something that contains both numbers
// and letters, match this trigger:
+ my name is *
- Your name has a number in it?

// See how old the user is
+ i am # years old
- A lot of people are <star> years old.

// But don't let them give us their age in words!
+ i am _ years old
- Can you say that again using a number?

// Both numbers and letters?
+ i am * years old
- You told me numbers and letters? Tell me only numbers.

// Let them tell us where they're from. Numbers and letters are
OK!
+ i am from *
- What is it like to live in <star>?

// This one has multiple wildcards in it
+ _ told me to say *
- So did you say "<star2>" because "<star1>" told you to?

+ what is your (home|office|cell) phone number
- You can call my <star> number at 1 (888) 555-5555.

+ i (can not|cannot) *
- Have you tried?
- Why can't you <star2>?
- Do you really want to <star2>?

+ who (is your master|made you|created you|programmed you)
- I was developed by a RiveScript coder; you don't need to know
his name!

+ (what is your name|who are you|who is this)
- My name is Aiden, I'm a chatterbot running on RiveScript!

+ (happy|merry) (christmas|xmas|valentines day|thanksgiving)
```

```
- Wow! Is it really <star2> already?

// Now they don't even need to say the word "phone"!
+ what is your (home|office|cell) [phone] number
- My <star> number is: 1 (888) 555-5555.

+ i do not have [any] friends
- Aw. I'll be your friend!

+ am i [a] (boy|guy|male) or [a] (girl|female)
- I can't tell with any degree of certainty whether you are a
<star1> or <star2>.

// If the user begins a message with "google" it will create
// a google search link.
+ google *
- Google Search: <call>google <star></call>

// If the user ends their message with "or something", the
// bot will simply say "Or something." and drop the topic.
+ * or something
- Or something.

// Here is the Google search object. We'll cover objects in more
// depth later in the tutorial.
> object google javascript
    var query = escape(args.join(" "));
    return "<a href=\"http://www.google.com/search?q=" + query +
"\">Click Here!</a>";
< object
```