Ezdin Aslancı - 150112022
Kutalmış Coşkun - 150113018

1. About Assignment

   In this programming assignment, we designed and implemented a system that allows a user to plan a trip. A trip plan consists of hotel and flight reservations for requested number of travelers for a requested time interval.

   The procedure of planning a trip can be expressed as follows:

   a. A user uses the customer interface to specify her arrival and departure dates, preferred hotel and airline and the number of travelers.

   b. Customer interface communicates with the travel agency by using TCP sockets to transfer customer's request.

   c. Travel agency receives the request and contacts to the preferred hotel and airline by using HTTP to check whether they are available or not for requested dates.

   d. If one of preferred hotel and airline is not available, travel agency contacts all of the hotels and airlines and it generates a list of available itineraries. If there is no available itinerary, travel agency sends a message to customer about the case. If there is an available itinerary, travel agency keeps offering the itinerary to the customer until it runs out of options

   e. If both preferred hotel and airline are available, travel agency sends a message hotel and airline to reserve preferred dates.

2. Implementation Details

   We used Python (3.5.2) programming language to implement such a service. Here is a list of classes we used:

   - Customer Interface
   - Travel Agency
   - Hotel Server
   - Airline Server

   a. Customer Interface

      This is the interface that customer directly uses to plan a trip. It gets details from the customer and checks whether they are valid or not. Here is a list of checks applied:

      - Arrival should be after the current date
      - Departure should be after the arrival
      - Number of travelers should be a positive integer

      After checking the validity of details, Customer Interface wraps these by using JSON and sends it to Travel Agency by using a TCP socket.

      Here is an example JSON message from Customer Interface to Travel Agency:

```
{
        "arrivalDate": "01.01.2017",
        "departureDate": "05.01.2017",
        "preferredHotel": "Hotel Endurance",
        "preferredAirline": "Enterprise Airlines",
        "numOfTravellers": "3"
}
```

b. Travel Agency

Travel Agency receives the JSON package sent by the Customer Interface and sends the preferred dates and the number of travelers to preferred hotel and airline by using HTTP GET method.

Here is an example URL generated by Travel Agency to check the availability of one of hotels:

```
http://10.20.14.33:8081/?
type=check&arrivalDate=01.01.2017&departureDate=05.01.2017&numOfTravellers=3
```

The “type” attribute can be either “check” or “reserve”. As the name indicates, a message with “check” is sent only for checking the availability of hotel/airline for given dates and the number of travelers. Databases stay untouched when the type is set to “checked”. Type is set to “reserve” when the customer accepts the itinerary and the result should be written to the database of hotel/airline.

Depending on the response from the hotel/airline server, Travel Agency generates an alternative itinerary by sending HTTP GET requests to other hotels and airlines and keeps offering it to the customer (by sending the alternative itinerary to Customer Interface by using TCP sockets) until the customer accepts it or Travel Agency runs out of options.

c. Hotel/Airline Server

When a hotel/airline server receives such an HTTP GET request, it parses the URL to get the information (it splits the URL by “?” and “&”). At this point, the hotel/airline should check its database and generate a response according to the availability.

We used JSON for storing the reserved dates in hotel/airline databases. Here is an example hotel database:

```
{
    "hotelName": "Hotel Endurance",
    "numOfRooms": 3,
    "rooms": [
        {
            "reservedDates": [
                {
                    "arrivalDate": "01.01.2017",
                    "departureDate": "02.01.2017"
                },
                {
                    "arrivalDate": "09.01.2017",
                    "departureDate": "10.01.2017"
                }
            ],
            "roomID": 1
        },
        {
            "reservedDates": [
                {
                    "arrivalDate": "01.01.2017",
                    "departureDate": "02.01.2017"
                }
            ],
            "roomID": 2
        },
        {
            "reservedDates": [],
            "roomID": 3
        }
    ]
}
```

Here is an example airline database:


```
{
    "airlineName": "Ascendant Airlines",
    "numOfSeats": 3,
    "seats": [
        {
            "reservedDates": [
                {
                    "date": "01.01.2017"
                },
                {
                    "date": "02.01.2017"
                }
            ],
            "seatID": 1
        },
        {
            "reservedDates": [
                {
                    "date": "11.01.2017"
                },
                {
                    "date": "12.01.2017"
                }
            ],
            "seatID": 2
        },
        {
            "reservedDates": [],
            "seatID": 3
        }
    ]
}
```


The availability of a hotel for a requested time period is checked by the following method:

- Checking all rooms, a requested time period is not available for a room if,
  - Arrival date is in an already reserved time period OR
  - Departure date is in an already reserved time period

Also, the availability of an airline for requested dates is checked by the following method:

- Checking all seats, requested dates are not available for a seat if,
  - Arrival date overlaps with an already reserved date OR
  - Departure date overlaps with an already reserved date

After checking the availability, hotel/airline server generates a response, here is an example from an hotel server:


```
{
        "ifAvailable": true,
        "availableRooms": [2, 3, 4, 5],
        "status": "not reserved"
}
```


“ifAvailable” field holds the information of availability, “availableRooms” field holds a list of IDs of available rooms and “status” may be “reserved” or “not reserved” depending on the “type” of the HTTP GET request.

Also, here is an example response generated by an airline server:


```
{
        "ifAvailable": true,
        "availableSeats": [[5, 6], [5, 6]],
        "status": "not reserved"
}
```

The response of an airline server is similar to a hotel's one, the only difference is "availableSeats" field. It holds a list of two lists, the first one is for holding the IDs of available seats for arrival, the second one is for departure.

Communication Protocol Between Travel Agency and Customer Interface

After getting responses from preferred hotel and airline servers, Travel Agency sends a message involving the information that it obtained. Here is the list of status codes and their meanings & structure.

Status Code	Pref. Hotel	Pref. Airline	Structure
[1]	available	available	[1]
[2]	not available	available	[2]+ALT_HOTEL
[3]	available	not available	[3]+ALT_AIRLINE
[4]	not available	not available	[4]+ALT_HOTEL+ALT_AIRLINE

Status Code	Meaning	Structure
[5]	preferred hotel is not available and there is no alternative hotel available	[5]
[6]	preferred airline is not available and there is no alternative airline available	[6]
[7]	preferred hotel and airline are not available and there are no alternatives	[7]
[8]	successfully reserved	[8]+RESERVED_HOTEL+RESERVED_AIRLINE
[20]	unknown preferred hotel	[20]
[21]	unknown preferred airline	[21]

After getting the response from Travel Agency, Customer Interface shows messages to user according to their status code and expects user to decide whether she accepts the itinerary or not and contacts to the Travel Agency if needed.

Here is a sequence diagram showing how the communication between customer and our service takes place: