**Hotel Booking**

USER INTERFACE

External Users/Admins

**Event Queue Messaging**

HotelBooking Service

Hotel Service

User Service

API GATEWAY

Frontend Application

HotelsBooking

Logging

Mongodb Database

All microservices are exposed to same IP/address using API Gateway.

/users

User Service

/hotels

Hotel Service

**API GATEWAY**

**Internet**

http://localhost:8088

/hotelbooking

HotelBooking Service

For synchronous communication between services using Feign Client. And for asynchronous communication using ActiveMQ.

**Hotel Booking**

If user wants to book a hotel. Parameters will be passed accordingly to search hotels.

**API ENDPOINT TO SEARCH HOTELS** -

*http://localhost:8082/hotels/search?location=goa&checkin=2024-03-30&checkout=2024-03-31&noofrooms=2&hotelname=moti mahal&sortby=rating*

***Return List of available hotels***

Here the parameters **checkindate**, **checkoutdate** and either **location** or **hotelName** is required and other parameters if not passed will take default value.

Default value for **noofrooms** user can book is 1, **sortBy**(user can sort hotel on the basis of either price or rating) default value is price.

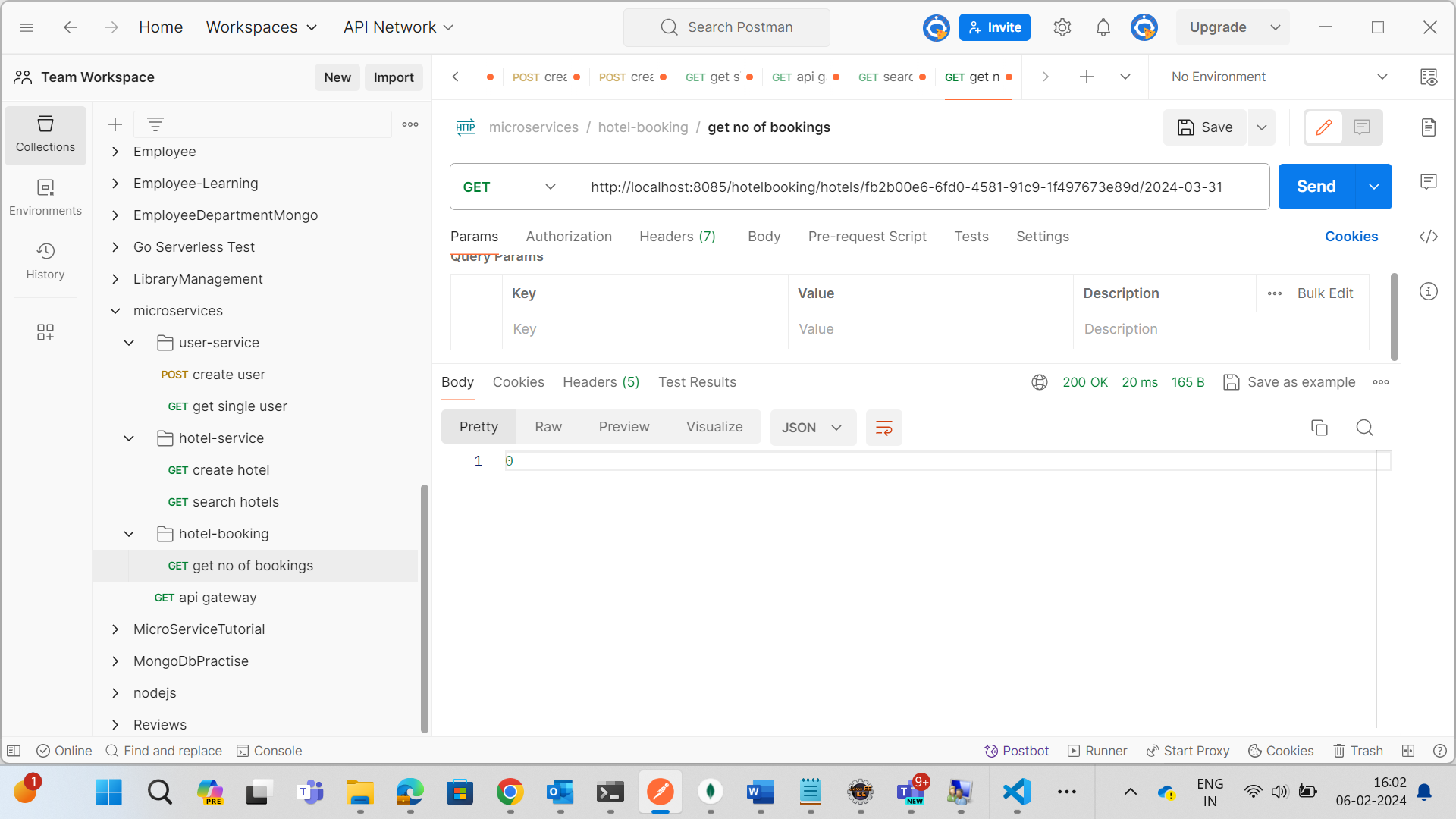
**Scenarios covered –**

User can book **multiple** rooms in a hotel.

**Checkindate** and **checkoutdate** cannot not extend greater than one year from current date i.e. we can’t search hotels for checkindate **2024-06-12** and checkoutdate **2025-06-14**

Searching return the hotels that are available from **checkindate** to **checkoutdate(not included)**. First the searching API will fetch all hotels from database based on location or hotelName(returns arrayList of hotels)

Then takes each hotel and implements a check on all dates starting from checkindate and less than checkoutdate to check for rooms availability. For eg checkindate is 2024-03-30 and checkoutdate is 2024-04-01 , now it checks which hotels are available on date 30th & 31th. For checking hotel availability it calls API **getBookingsByCheckoutAndId** from **HotelBookingService** to check how many rooms are available on date 30th & 31th (based on checkout date i.e mapping of checkindate to checkoutdate in HotelBookingService).



Hitting getBookingsByCheckoutAndId API on HotelBooking Service, here we pass date 2024-03-31, it checks on this date how many rooms are available(based on checkout). In above image, it return 0 means there is no room that is being checkout on the date **2024-03-31**.Similarily there is check for date **2024-03-30.**

If the **hotel** is available on each date then its added to **arrayList** that is returned.

Integer checkoutRooms = bookingService.getBookingsByCheckoutAndId(hotel.getHotelId(), checkInDate);

**if** ((hotel.getRoomsAvailable() != **null**) && (hotel.getRoomsAvailable() + checkoutRooms) >= noOfRooms) {

availability = availability + 1;

}.

If everything works fine return list of available hotels -:

A screenshot of a computer

Description automatically generated

After searching hotels, User can **book hotel** according to his desire.

Post request will hit on **HotelBookingService**(**bookHotel API**). All the required parameters will be sent in the body.

**API ENDPOINT TO BOOK A HOTEL IS -**

[*http://localhost:8085/hotelbooking/users/{userId}/hotel/{hotelId}*](http://localhost:8085/hotelbooking/users/%7buserId%7d/hotel/%7bhotelId%7d)

**Scenarios covered –**

First It checks that **checkindate** and **checkoutdate** is not any Date which extends greater than one year from current date i.e user can’t book hotel for checkindate **2024-03-30** and checkoutdate **2025-03-31**.

A screenshot of a computer

Description automatically generated

Then it checks whether the user and hotel exists or not by hitting **getUserById API** from **UserService** and **getHotelById** **API** from **HotelService** using **Feign Client**.

If all works fine, Returns the response with hotel booking status **PROCESSING.**

Using **ActiveMq** event will be fired to **HotelService** to check for availability of room in a hotel from checkindate to checkoutdate and payment is done or not.

jmsTemplate.convertAndSend("BookingRequestReceivedEvent",JsonSerializerUtil.*serialize*(b));

On **HotelService** , for all dates starting from checkindate and less than checkoutdate there is a check for availability of room. If the room is available, increasing the number of rooms booked according to date .

d.getNoOfRoomsBooked() + booking.getNoOfRooms() <= hotel.getRoomsAvailable()

Now checks the payment i.e.

booking.getPayment().equals(booking.getNoOfRooms() \* hotel.getCurrentPrice())

If the room is available and payment is successful, then **HotelConfirmEvent** is fired -:

jmsTemplate.convertAndSend("HotelConfirmEvent", bookingId);

If the room is not available on any date between checkindate and checkoutdate(not including checkoutdate), then **HotelNotAvailableEvent** is fired -:

jmsTemplate.convertAndSend("HotelNotAvailableEvent", bookingId);

If the room is available but payment is unsuccessful, then the number of rooms booked that we saved, again decreased to its original value so that it is available to other users to book.

**HotelPaymentDeclinedEvent** is fired -:

jmsTemplate.convertAndSend("HotelPaymentDeclinedEvent", bookingId);

If the ticket is not confirmed for any other reasons , then **HotelNotConfirmEvent** is fired -:

jmsTemplate.convertAndSend("HotelNotConfirmEvent", bookingId);

These events will be fired to **HotelBookingService** to change the booking status to **CONFIRMED, UNCONFIRMED, NOT\_AVAILABLE, PAYMENT\_DECLINED.** And set the payment to 0f accordingly(In case of **UNCONFIRMED, NOT\_AVAILABLE, PAYMENT\_DECLINED**  events) and save to **HotelBookings**.

From HotelBookingService events will be fired to **UserService** accordingly. With the help of **Logger** it will print the booking status whether the hotel is confirmed or not and reason for not confirmed. If confirmed then add that booking to **HotelBooking** arrayList in **User collection.**

For overlapping issues((i.e., room being taken when user is in the process of payment) used @Transactional annotation on the bookHotel method to ensure safety on our "Transactions"

The request body that is sent is -:

{

   "hotelName":"moti mahal",

   "checkInDate":"2024-03-30",

   "checkOutDate":"2025-03-31",

   "noOfRooms":1,

   "guestName":"Ridhi",

   "guestPhoneNo":"9812456100",

   "guestEmail":"ridhikagarg8@gmail.com",

   "payment":2000

}

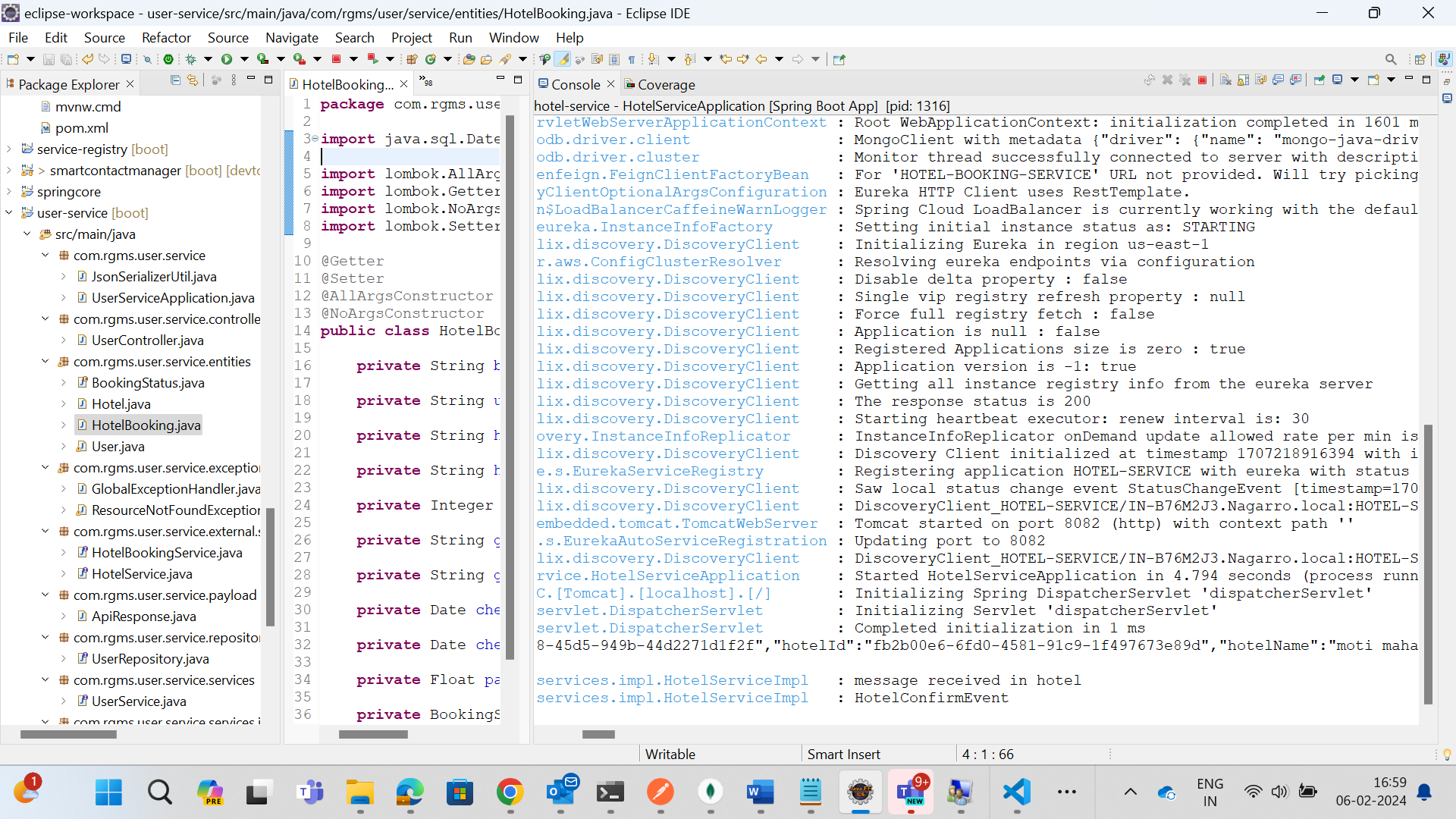
User wants to book hotel with hotelId **“fb2b00e6-6fd0-4581-91c9-1f497673e89d”.** Returns the response with status as PROCESSING -:

A screenshot of a computer

Description automatically generated

**With the help of Logger we can see events fired on different services.**

**Event fired on HotelService -:**



**Event fired on HotelBookingService -:**

A screenshot of a computer

Description automatically generated

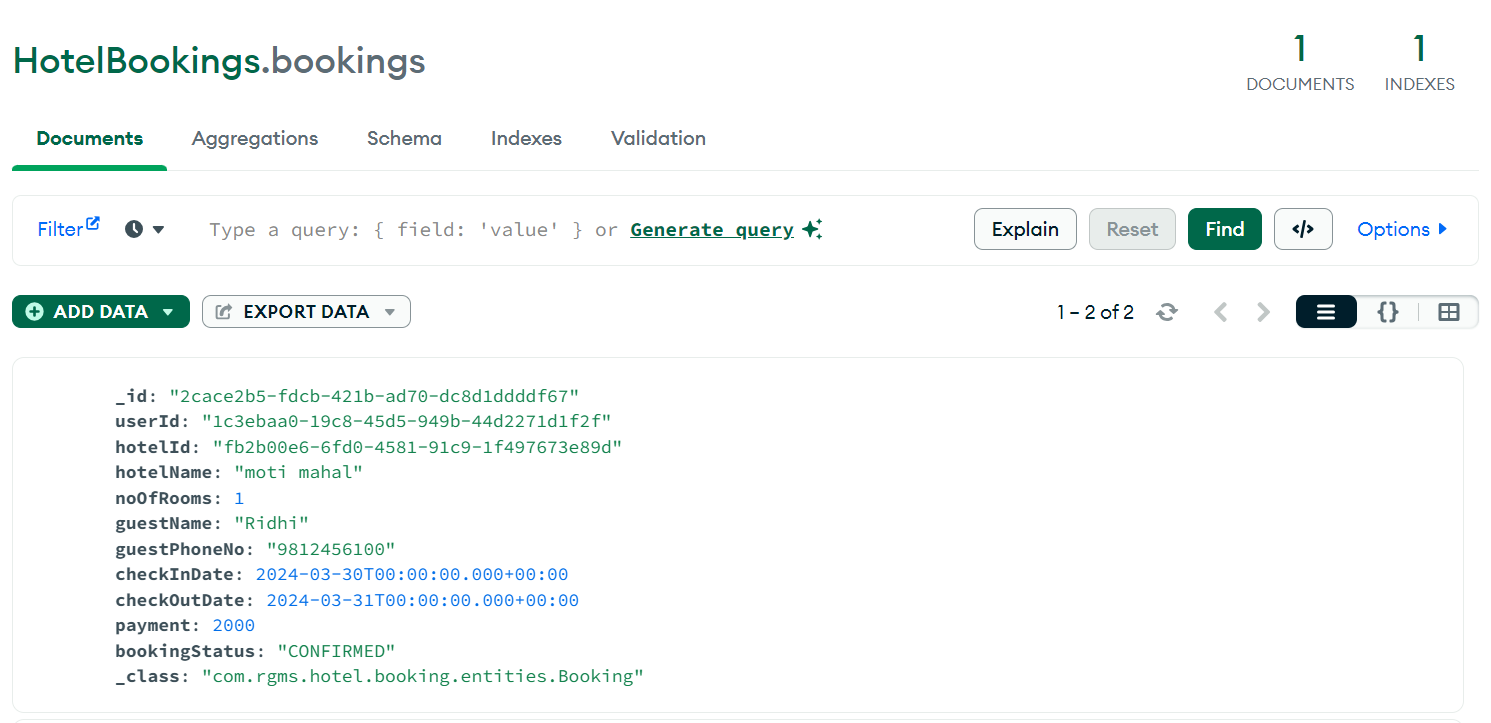
**Event fired on UserService-:**

A screenshot of a computer

Description automatically generated

**MongoDb data -:**

**HotelBookingCollection**



**HotelCollection**

For checkinDate 2024-03-30 **noOfRoomsBooked** is 1. Available rooms are 5-2=3

A screenshot of a computer

Description automatically generated

**UserCollection**

A screenshot of a computer

Description automatically generated

**If the room is not available in a hotel–**

**Event fired on HotelService -**

A screen shot of a computer

Description automatically generated

**Event fired on HotelBookingService –**

A screenshot of a computer

Description automatically generated

**Event fired on UserService –**

A screenshot of a computer

Description automatically generated

**MongoDb data –**

A screenshot of a computer

Description automatically generated

**In case of hotel room is available but payment declined –**

Event fired on HotelService –

A screenshot of a computer

Description automatically generatedEvent fired on HotelBookingService –

A close up of a computer code

Description automatically generatedEvent fired on UserService –

A screenshot of a computer

Description automatically generated

MongoDb data –

A computer screen shot of a computer code

Description automatically generated

In case of booking is not confirmed by any other reasons. **HotelNotConfirmEvent** is fired.

After booking hotel, if user wants to cancel it.

**API ENDPOINT TO CANCEL A HOTEL IS –**

***http://localhost:8085/hotelbooking/cancel/{bookingId}***

User can only cancel the hotel that is confirmed.

If user tries to cancel the hotel that is not confirmed yet , it returns error –

A screenshot of a computer

Description automatically generated

Using **ActiveMq** event will be fired to **HotelService** to make the rooms that was booked from **checkindate** to **checkdoutdate**(not including) is made **available**.

jmsTemplate.convertAndSend("BookingCancelRequestReceivedEvent",JsonSerializerUtil.*serialize*(b));

On **HotelService** ,starts by the checkindate and search for the checkindate in Hotel ,once the date is found we make the rooms available for that date. Keeps on increasing checkindate by 1 day.( iterates until its less than checkoutdate).

After making rooms available, fire **HotelCancelEvent** to **HotelBookingService**

jmsTemplate.convertAndSend("HotelCancelEvent", bookingId);

In **HotelBookingService**, we set the **Payment** to 0f and change **Booking statu**s to **REFUNDED** and save to **hotelBookings**. From **HotelBookingService** event will be fired to **UserService**.

jmsTemplate.convertAndSend("HotelCanceled", JsonSerializerUtil.*serialize*(b));

In **UserService**, it will log that **hotel booking is cancelled and amount refunded successfully.** Accordingly finds that particular booking in user’s **hotelBooking arraylist** by **bookingId,** once found it will set the status of that particular booking to **REFUNDED** and set **Payment** to 0.

**Screenshots for the same -:**

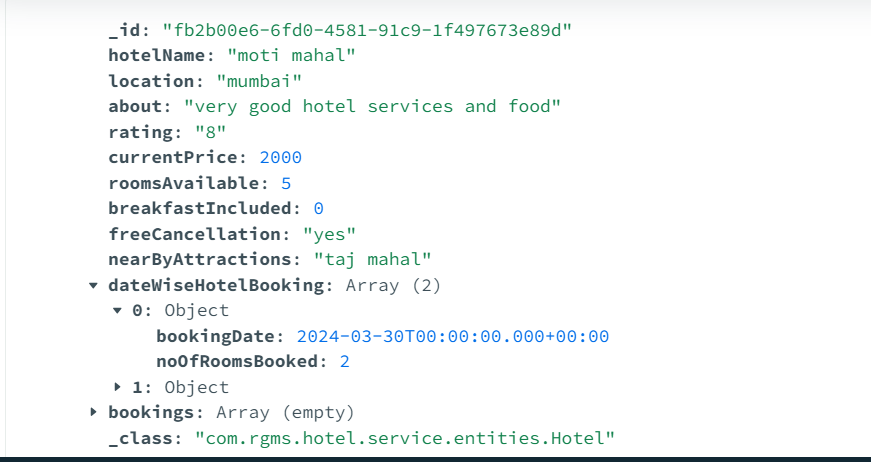
Before cancelling hotel, data in mongodb –

**HotelBookings**



One room is booked for checkindate **2024-03-30** and checkoutdate **2024-03-31** by a particular user.

Hotels



In **hotels collection** we can see that on date **2024-03-30** 2 rooms are booked.

Now if user **cancel** this booking –

A screenshot of a computer

Description automatically generated

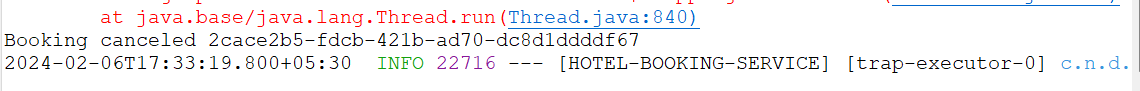
**With the help of Logger we can see events fired on different services.**

**Event fired on HotelService –**

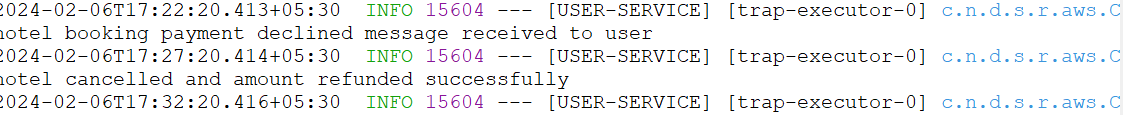
A close-up of a text

Description automatically generated

**Event fired on HotelBookingService –**



**Event fired on UserService –**



**Mongodb data after cancel –**

**HotelBookings collection** ,here the status is updated to REFUNDED which earlier was CONFIRMED and payment is set to 0.

A screenshot of a computer

Description automatically generated

**Hotels collection**, now the no of rooms booked for date 2024-03-30 is 1. As earlier there were 2 rooms booked and one is cancelled.

A screenshot of a computer

Description automatically generated

**User collection**, status of that particular hotel booking is changed to REFUNDED and payment is set to 0.

A screenshot of a computer

Description automatically generated

Every microservice is being registered with Service Discovery -:

A screenshot of a computer

Description automatically generated

Every request is landed to API Gateway (Port – 8084) -:

For logging used (org.slf4j.Logger). Used eureka-server for implementing service discovery.