Booking-Application-Microservices

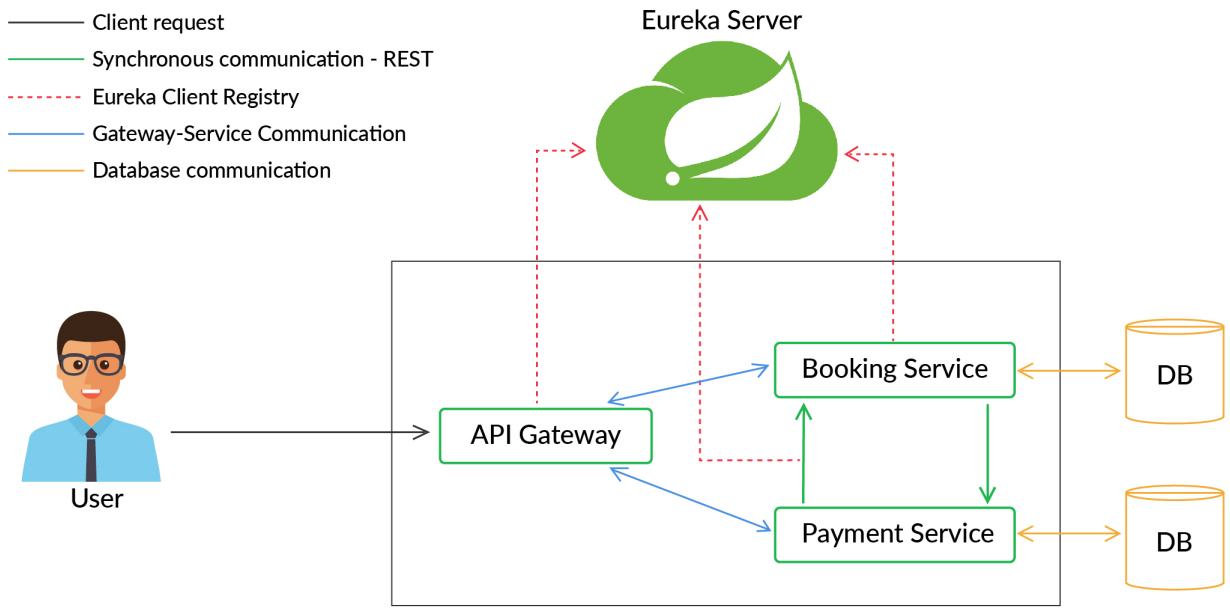
A microservice architecture based (hotel room) booking application

High Level Architecture

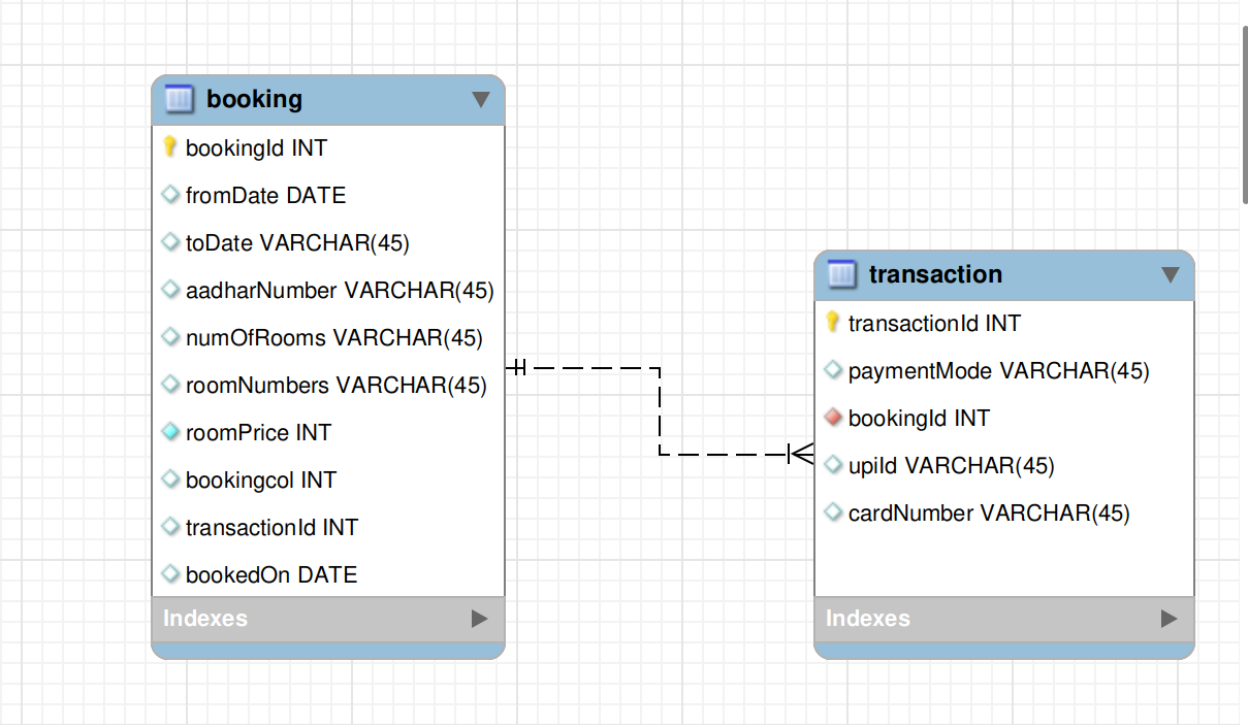
The application consists of the following three microservices:

- Booking Service
- Payment Service
- Eureka Server

Architecture Diagram -



Database Schema Diagram -



Booking Service

API-1: Make New Booking

Endpoint -

```
POST localhost:8080/booking
Content-Type application/json
```

Request Body Ex –

```
 {
    "fromDate": "2021-02-02",
    "toDate": "2021-02-10",
      "aadharNumber": "Sonu Kumar-Aadhar Number",
      "numOfRooms": 5
 }
```

Response Body Ex -

```
 {
      "id": 1,
      "fromDate": "2010-02-02",
      "toDate": "2010-02-10",
      "bookedOn": "2021-10-20 17:21:47",
      "aadharNumber": "Sonu Kumar-Aadhar Number",
      "roomNumbers": "40,61,62,59,3",
      "numOfRooms": 5,
      "roomPrice": 40000,
```

```
      "transactionId": 0
  }
```

API-2: Complete transaction

Endpoint -

```
POST localhost:8080/booking/{booking-id}/transaction
Content-Type application/json
```

Request Body Ex –

```
  {
     "paymentMode": "CARD",
   "bookingId": 1,
   "upiId":"",
   "cardNumber":"Test Card Number"
  }
```

Response Body Ex -

```
  {
      "id": 1,
      "fromDate": "2010-02-02",
      "toDate": "2010-02-10",
      "bookedOn": "2021-10-20 17:21:47",
      "aadharNumber": "Sonu Kumar-Aadhar Number",
      "roomNumbers": "40,61,62,59,3",
      "numOfRooms": 5,
      "roomPrice": 40000,
      "transactionId": 2
  }
```

Payment Service

API-3: Save Transaction and return transactionId

Endpoint -

```
POST localhost:8083/transaction
Content-Type application/json
```

Request Body Ex –

```
  {
     "paymentMode": "CARD",
   "bookingId": 1,
   "upiId":"",
   "cardNumber":"Test Card 2"
  }
```

Response Body Ex -

```
{
   "transactionId": 2,
}
```

API-4: Fetch details of a transaction

Endpoint -

```
GET localhost:8083/transaction/2
```

Response Body Ex -

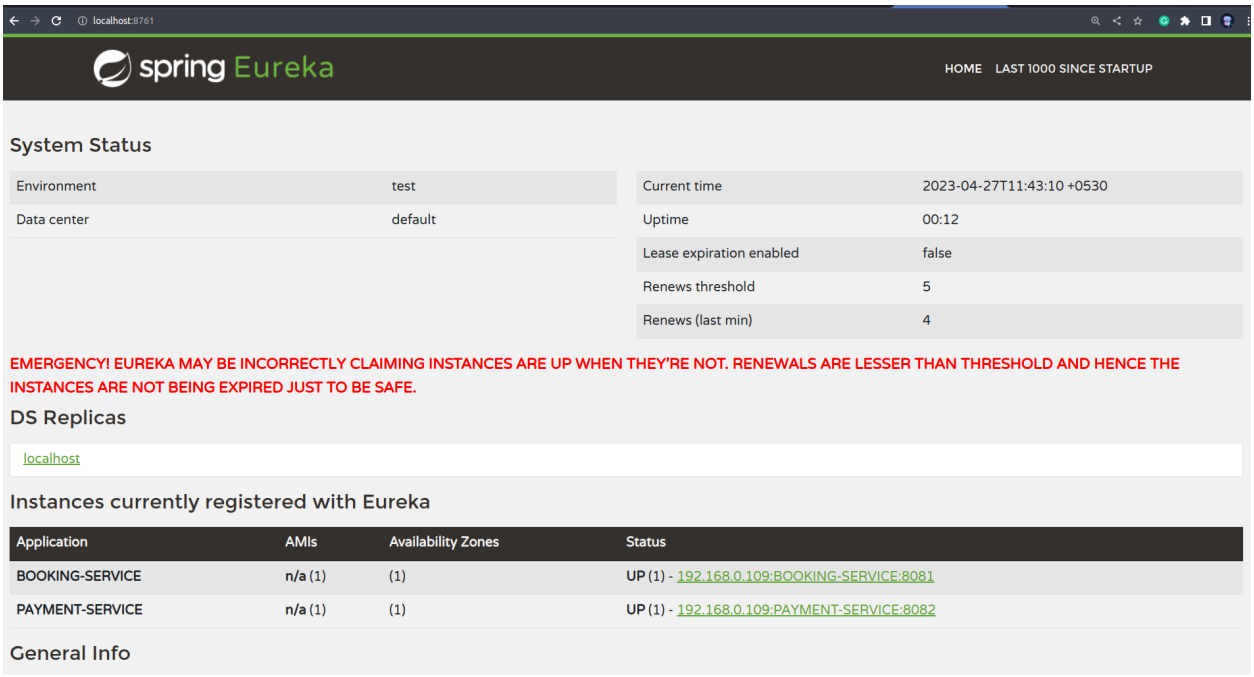
```
{
   "transactionId": 2,
     "bookingId": 1,
     "paymentMode": "CARD",
     "upiId": null,
     "cardNumber": "Test Card Number"
}
```

Eureka Server

Eureka Server is an application that holds the information about all client-service applications. It knows all the client applications running on each port and IP address. Eureka Server is also known as Discovery Server.

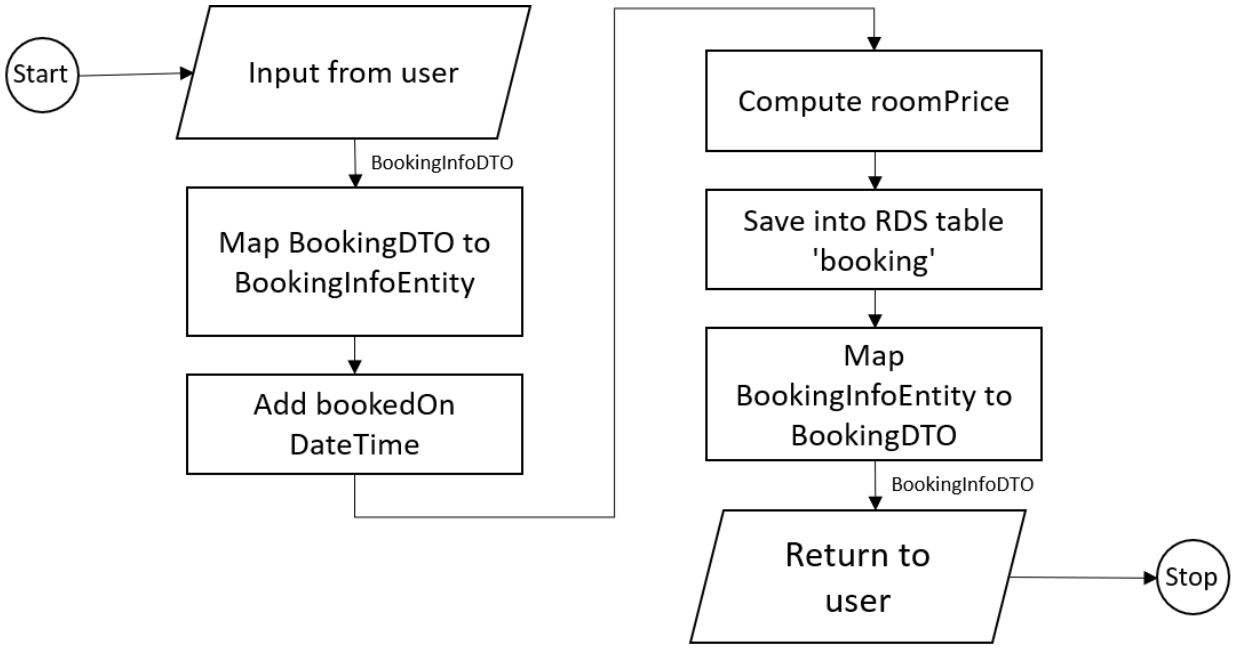
The Booking and Payment services registers themselves into the Eureka server.

The Eureka Server is started on port 8761. On the browser if we go to http://localhost:8761/ we can see the Booking and Payment services are up.

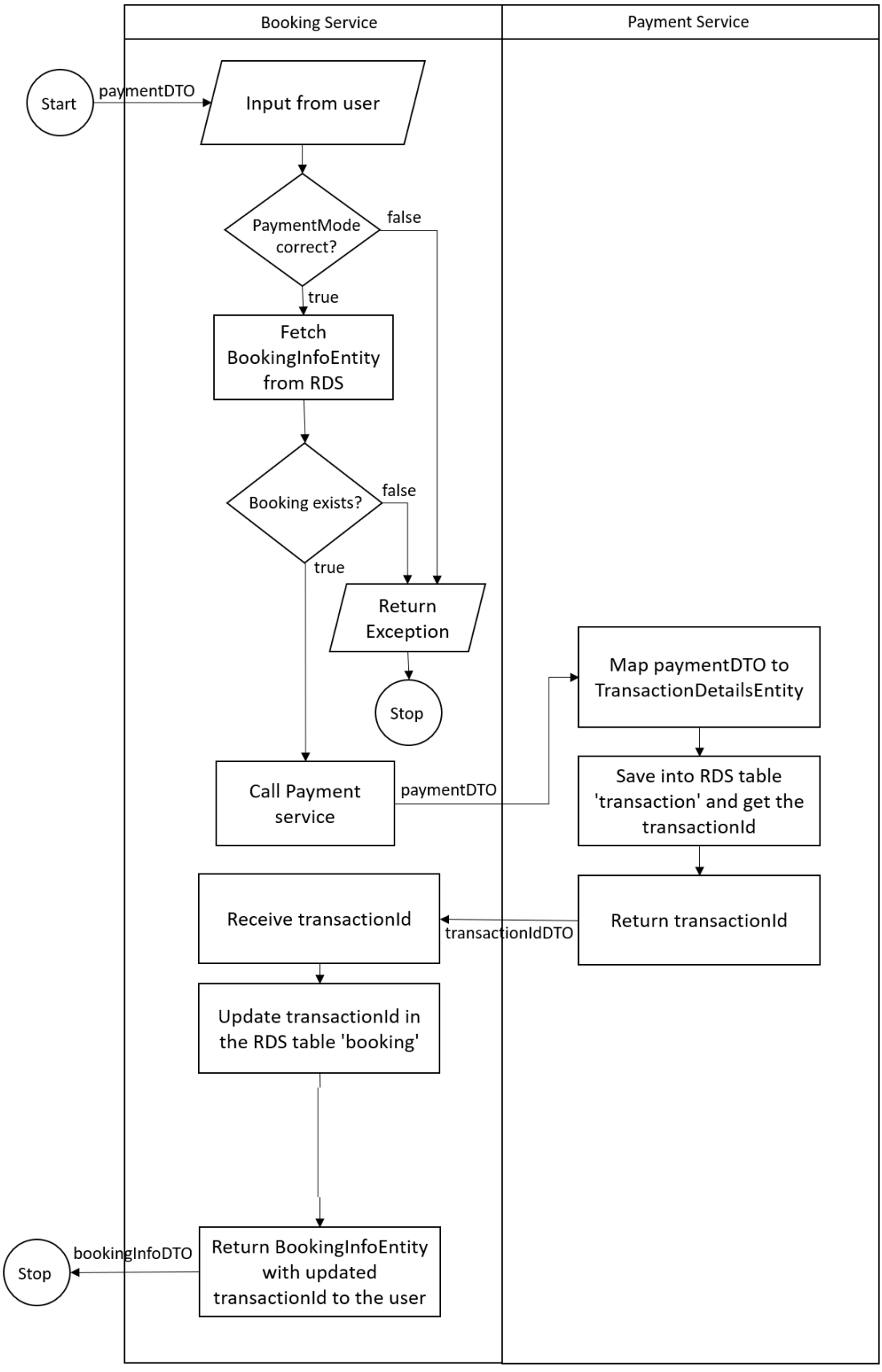


Logic Flow

Making a new booking -



Completing a transaction to confirm booking -


Booking Service
Payment Service
Start
paymentDTO
Input from user
PaymentMode correct?
false
true
Fetch BookingInfoEntity from RDS
Booking exists?
false
true
Return Exception
Stop
Call Payment service
paymentDTO
Map paymentDTO to TransactionDetailsEntity
Save into RDS table 'transaction' and get the transactionId
Return transactionId
transactionIdDTO
Receive transactionId
Update transactionId in the RDS table 'booking'
Return BookingInfoEntity with updated transactionId to the user
bookingInfoDTO
Stop

Future Enhancements

1. Add error and exception handling at all the possible points
2. Package all the services together
3. Provide single start-up script to bring up all the services
4. Implement API gateway and load balancer