sweet-home-backend

A spring boot application serving as a backend for a hotel booking application

The project has five microservices -

#1 apiGateway ->

- This serves as the gateway to the booking and payment microservices.
- Configured on http://localhost:9191

#2 eurekaServer ->

- This microservice helps maintain and track URLs for our microservices. Enables load balancing and provides usage stats for better optimization.
- Configured on http://localhost:8761

#3 configSever ->

- This microservices help us maintain application.properties for our various services running on different locations, which gives us the flexibility to change the configuration of either of the microservices at runtime.
- Configured on http://localhost:8888

#4 booking ->

- This is our leading service for booking rooms at the hotel.
- In this, I am using the h2 database to provide on-demand quick storage for our booking microservice, which has its h2-console enabled and can be accessed on localhost:8081/console
- Error handling in invalid payment mode or booking id cases while executing a transaction is handled using controller advice with a custom exception handler.

Endpoint for booking rooms -> **POST**: http://localhost:9191/hotel/v1/booking

Request Body :

```
{
  "fromDate" : "2021-06-20T10:22:15",
  "toDate" : "2021-06-25T10:22:15",
  "aadharNumber": "1234567890",
  "numOfRooms" : 3
}
```

Response Structure:

```
{
  "bookingId": 6,
  "fromDate": "2021-06-20T10:22:15",
  "toDate": "2021-06-25T10:22:15",
  "aadharNumber": "1234567890",
  "numOfRooms": 3,
  "roomNumbers": "[92, 26, 24]",
  "roomPrice": 15000,
  "transactionId": 0,
  "bookedOn": "2023-07-23T00:34:14.7786362"
}
```

This adds an entry to our database regarding a booking request from the user having the details mentioned above.

After receiving the booking request, the user has to hit the below API to confirm the transaction mentioning the payment details as well as the booking id the user is paying:

Endpoint for the execution of a transaction for a booking ->
POST: https://localhost:9191/hotel/v1/booking/2/transaction
Request Body:

```
{
  "paymentMode" : "UPI",
  "bookingId" : 2,
  "aadharNumber" : "1234567890",
  "cardNumber" : "44317283000920063"
}
```

Response Structure:
We get a validated booking with a valid transaction id which confirms our payment.

```
{
  "bookingId": 2,
  "fromDate": "2021-06-20T10:22:15",
  "toDate": "2021-06-25T10:22:15",
  "aadharNumber": "1234567890",
  "numOfRooms": 3,
  "roomNumbers": "[13, 39, 41]",
  "roomPrice": 15000,
  "transactionId": 8,
  "bookedOn": null
}
```

#5 payment (Dummy payment microservice)

- This is an internal microservice used by our booking service to execute transactions.
- It uses feign client to communicate with the booking microservice synchronously to acknowledge the transactions with a valid transaction id.

Endpoint to add a transaction called by booking service ->
POST: https://localhost:9191/payment/v1/transaction
Request Body ->

```
{
   "paymentMode" : "UPI",
   "bookingId" : 2,
   "aadharNumber" : "1234567890",
   "cardNumber" : "44317283000920063"
}
```

Response Body->

```
{
   “transactionId”: 1,
   "paymentMode" : "UPI",
   "bookingId" : 2,
   "aadharNumber" : "1234567890",
   "cardNumber" : "44317283000920063"
}
```

Endpoint to get transaction details ->
GET: https://localhost:9191/payment/v1/transaction/{transactionId}

Response Body->

```
{
   “transactionId”: 1,
   "paymentMode" : "UPI",
   "bookingId" : 2,
   "aadharNumber" : "1234567890",
   "cardNumber" : "44317283000920063"
}
```