Sweet Home

This project is divided into 4 microservices i.e., Eureka Server, API Gateway, Booking Service and Payment Service. Each of the microservices are further divided into 3 categories namely –

1) Database Layer
2) Service Layer
3) Controller Layer

I will talk about these layers in a while. Let me introduce with all the microservices that I made for this application and also the functioning of these microservices.

The first microservice is Eureka Server –

This microservice maps all the request that are coming to it with the respective microservices registered with the Eureka Server. It also acts as a load balancer for the client i.e., it equally maps the requests between the multiple instances of the same application that are registered with the Eureka Server. For this functionality, every microservice must be registered with the Eureka Server.

Dependencies Used –

```
<dependencies>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
   </dependency>

   <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-netflix-eureka-
server</artifactId>
      <version>3.0.4</version>
   </dependency>

   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
   </dependency>
</dependencies>
```

Port Number - 8761

The Second microservice API Gateway –

This microservice helps the user to directly interact with rest of the microservices without remembering the end points of each microservice. This service contains all the endpoints in itself.

Dependencies Used –

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-webflux</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-config</artifactId>
    <version>3.0.5</version>
  </dependency>

  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-gateway</artifactId>
    <version>3.0.5</version>
  </dependency>
```

```
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-eureka-
client</artifactId>
        <version>3.0.4</version>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
```

Port Number – 9191

The third microservice Booking Service –

This microservice is actually performing all the logics at its end. This microservice helps the user to book the required rooms in the hotel as well as the it also offers users to complete the payment for the required booking.

Dependencies Used –

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-
client</artifactId>
    <version>3.0.4</version>
  </dependency>
  <dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
  </dependency>
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
  </dependency>
  <dependency>
```

```
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Port Number – 8081

End Points –

1) For booking rooms in hotel – http://localhost:8081/hotel/booking
2) For payment of the booking – http://localhost:8081/hotel/booking/{bookingId}/transaction

The fourth microservice Payment Service –

This service is responsible for the completion of the payment with the corresponding booking Id. This service generates a unique transaction Id for each booking id. This service is not directly used by the user but this service is used by the Booking service. The transaction id is provided to the Booking Service and after that Booking

Service update the transaction id corresponding to the booking id in the Database.

Dependencies Used -

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-eureka-
client</artifactId>
    </dependency>
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Port Number – 8083

Now let me talk about the 3-layer architecture of each microservice

1) Database Layer
   This layer contains 2 packages, Entities and Repositories. Entities contains all the entity classes corresponding to the Database and Repositories contains all the Repository interfaces.

2) Service Layer

   This layer contains 2 packages, Service Classes and Service Interfaces. Service Classes contains all the service classes corresponding to the required functionality of the application and Service Interfaces contains all the Service Interfaces.

3) Controller Layer
   This layer contains the controller class which maps the Rest Calls to the specific classes by the help of respective end points/URL.

Note: - For database functionality I have used the in-memory database called h2 database.

Note: - Before importing the project to the IDE, delete the .idea file from all the project structure. It is the cached file.

Note: - All the necessary comments are included in the project code.