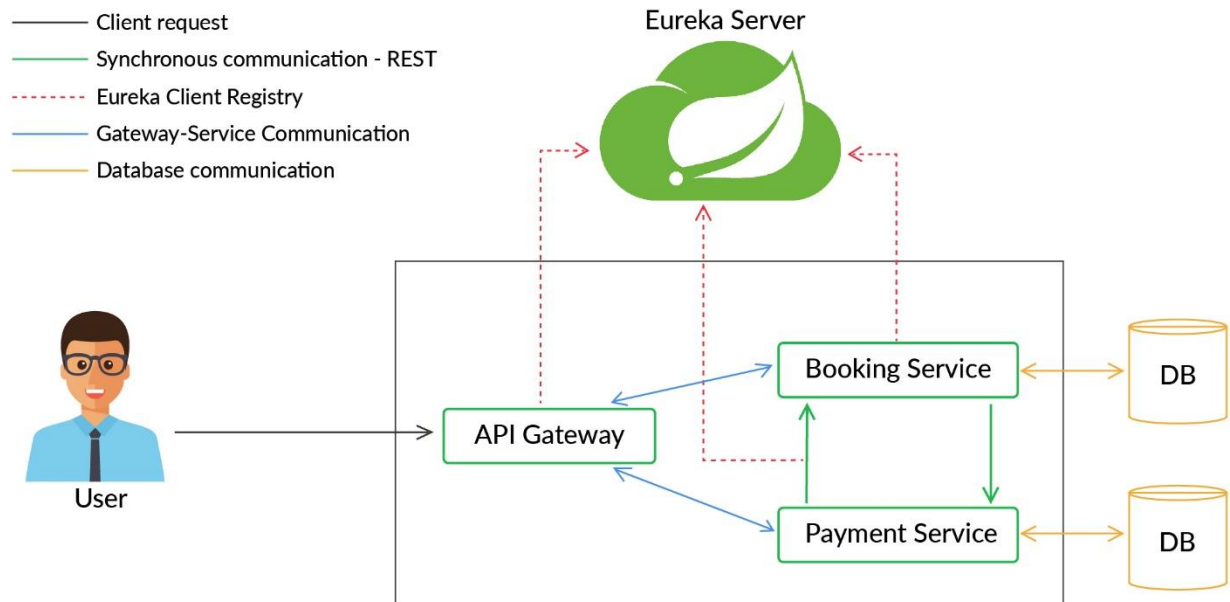


SWEET HOME- Hotel room booking application


Workflow as below:



1. Created API Gateway, Booking Service and Payment service.
2. Registered them on Eureka server.

```
eureka:
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://localhost:8761/eureka/
  instance:
    hostname: localhost

spring:
  application:
    name: PAYMENT-SERVICE
  cloud:
    discovery:
      enabled: true
```

HOME LAST 1000 SINCE STARTUP

System Status

Environment	N/A	Current time	2021-11-24T22:19:15 +0530
Data center	N/A	Uptime	00:32
		Lease expiration enabled	true
		Renews threshold	5
		Renews (last min)	8

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (1)	(1)	UP (1) - WS-9196.netcracker.com:API-GATEWAY:9191
PAYMENT-SERVICE	n/a (1)	(1)	UP (1) - WS-9196.netcracker.com:PAYMENT-SERVICE:8083

- If user wants to make any request, request will first land on API Gateway.

Created configurations in API gateway accordingly.

```
spring:
  application:
    name: API-GATEWAY
  cloud:
    gateway:
      routes:
        - id: BOOKING-SERVICE
          uri: lb://BOOKING-SERVICE
          predicates:
            - Path=/hotel/**
        - id: PAYMENT-SERVICE
          uri: lb://PAYMENT-SERVICE
          predicates:
            - Path=/payment/**
    discovery:
      enabled: true
```

- Data storage

Databases for Booking service and Payment service created and made configurations accordingly.

```

spring.datasource.url=jdbc:mysql://localhost:3306/bookingdb
spring.datasource.username=root
spring.datasource.password=
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=create
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
server.port=8081
api-gateway-url=http://localhost:9191

```

Similarly made configurations for payment service as well.

5. APIs for accessing Booking Services

The user initiates room booking using the 'Booking' service by providing information such as toDate, fromDate, aadharNumber and number of rooms required (numOfRooms).

Logic to calculate room prices.

```

} public BookingInfoEntity addBooking(BookingInfoEntity bookingInfoEntity) {
    int totalDays;
    int noOfRooms = bookingInfoEntity.getNumOfRooms();
    int roomPrice;

    long difference = bookingInfoEntity.getToDate().getTime() - bookingInfoEntity.getFromDate().getTime();
    totalDays = (int) (difference / (1000 * 60 * 60 * 24));
    roomPrice = 1000 * noOfRooms * (totalDays);

    ArrayList<String> roomNumbers = getRandomNumbers(bookingInfoEntity.getNumOfRooms());
    String roomNumber = String.join( delimiter: " ", roomNumbers);

    bookingInfoEntity.setRoomNumbers(roomNumber);
    bookingInfoEntity.setRoomPrice(roomPrice);
    //Setting current Date as booked date
    bookingInfoEntity.setBookedOn(new Date());

    bookingDao.save(bookingInfoEntity);
    return bookingInfoEntity;
}

```

Logic added as below to get random room numbers

```
//Method to generate Random room numbers
public static ArrayList<String> getRandomNumbers(int count) {
    Random rand = new Random();
    int upperBound = 100;
    ArrayList<String> numberList = new ArrayList<>();

    for (int i = 0; i < count; i++) {
        numberList.add(String.valueOf(rand.nextInt(upperBound)));
    }

    return numberList;
}
```

6. APIs for accessing Payment Service

If the user wants to go ahead with the booking, then they can simply provide the payment related details like bookingMode, upId/ cardNumber to the 'Booking' service which will be further sent to the 'Payment' service. Based on this data, the payment service will perform a dummy transaction and return back the transaction Id associated with the transaction to the Booking service. All the information related to transactions is saved in the database of the payment service.

```
@Override
public int addPaymentDetails(PaymentDto paymentDto) {
    TransactionDetailsEntity transactionDetailsEntity =new TransactionDetailsEntity();
    transactionDetailsEntity.setTransactionId(paymentDto.getTransactionId());
    transactionDetailsEntity.setPaymentMode(paymentDto.getPaymentMode());
    transactionDetailsEntity.setCardNumber(paymentDto.getCardNumber());
    transactionDetailsEntity.setUpiId(paymentDto.getUpiId());
    transactionDetailsEntity.setBookingId(paymentDto.getBookingId());
    paymentDao.save(transactionDetailsEntity);

    return transactionDetailsEntity.getTransactionId();
}

//METHOD TO GET PAYMENT DETAILS FOR GIVEN TRANSACTION ID
@Override
public TransactionDetailsEntity getPaymentDetails(Integer transactionId) {
    Optional<TransactionDetailsEntity> reqDetails = paymentDao.findById(transactionId);
    if (reqDetails.isPresent()) {
        return reqDetails.get();
    }
}
```

-----THANK YOU-----