**Service Registry: Eureka Server**

1. Service Registry / Service Discovery

All the services will get registered here. Load balancing will be very easy

**Dependencies**:

1. Cloud Bootstrap
2. Eureka Server
3. Enable Eureka Server from ServiceRegistryApplication.java using @EnableEurekaServer annotation.
4. Add following codes in application.property file

eureka.instance.hostname= localhost  
  
eureka.client.register-with-eureka=*false*eureka.client.fetch-registry=*false*server.port=8085

**Service Registry: Eureka Client**

1. Service Registry Discovery Client

All the client services will get registered to discovery server

**Dependencies**:

1. Cloud Bootstrap
2. Eureka Discovery Client
3. Enable Eureka Client from MainServiceApplication.java using @EnableEurekaClient annotation.
4. Add the following codes in application.property file in each service

eureka.instance.prefer-ip-address=*true*eureka.client.fetch-registry=*false*eureka.client.register-with-eureka=*true*eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka  
  
spring.application.name=HOTEL-SERVICE

**Inter Service Communication: Web Client**

1. Add WebFlux dependency in pom.xml file

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

1. Create a new confix file named WebfluxConfig.java

*@Configuration  
public class* WebClientConfig {  
 *@Bean  
 public* WebClient webClient(){  
 *return* WebClient.builder()  
 .defaultHeader(HttpHeaders.CONTENT\_TYPE, MediaType.APPLICATION\_JSON\_VALUE)  
 .build();  
 }  
}

1. Call other services API, using webClient

*@Service  
public class* UserServiceImpl *implements* UserService {  
  
 *@Autowired  
 private* UserRepository userRepository;  
  
 *@Autowired  
 private* WebClient webClientBuilder;  
  
 *private* Logger logger = LoggerFactory.getLogger(UserServiceImpl.*class*);  
  
 *//get single user  
 @Override  
 public* User getUser(String userId) {  
 *//get user from database with the help of user repository* List<Rating> ratings = webClientBuilder.get()  
 .uri("localhost:8080/ratings/users/" + userId)  
 .header(HttpHeaders.CONTENT\_TYPE, MediaType.APPLICATION\_JSON\_VALUE)  
 .retrieve()  
 .bodyToMono(*new* ParameterizedTypeReference<List<Rating>>() {  
 })  
 .block();  
  
 User user = userRepository.findById(userId).orElseThrow(() -> *new* ResourceNotFoundException("User with given id is not found on server !! : " + userId));  
  
 List<Rating> ratingList = ratings.stream().map(rating -> {  
 Hotel hotel = webClientBuilder.get()  
 .uri("localhost:8081/hotels/" + rating.getHotelId())  
 .header(HttpHeaders.CONTENT\_TYPE, MediaType.APPLICATION\_JSON\_VALUE)  
 .retrieve()  
 .bodyToMono(Hotel.*class*)  
 .block();  
 rating.setHotel(hotel);  
 *return* rating;  
 }).toList();  
 user.setRatings(ratingList);  
 *return* user;  
 }  
}

1. API Gateway (Spring Cloud GateWay)