Certified Tech Developer
The Ultimate Degree

Especialización en Back End I

Nivel de complejidad intermedia:🔥🔥

Ejercicio grupal:👦👦👦👧

Ejercicio mesa de trabajo

La empresa de cosméticos Bella Donna tuvo su sistema no disponible durante un black friday debido al altísimo índice de solicitudes de registro de nuevos usuarios en su plataforma que querían aprovechar las promociones. Por ello, a petición de la empresa, debemos crear un servicio de registro aparte del servicio de persistencia y, para evitar futuras indisponibilidades, implementar una cola de peticiones utilizando RabbitMQ.

- Ejercicio individual:👩
- Nivel de complejidad intermedia:🔥🔥

Módulos

- Stock y Reposición:
  - Spring Boot Starter AMQP
  - Spring Boot Starter Web
  - Spring Boot Starter Data JPA
  - Spring Boot Starter JDBC
  - Spring Boot Autoconfigure
  - Spring Cloud Starter OpenFeign
  - H2 database
  - Lombok

Instrucciones

1. Crear el archivo **docker-compose.yml** e iniciar el servidor RabbitMQ.
2. Crear el microservicio **persona-service** con una API REST que contenga los endpoints necesarios para recuperar y guardar una nueva persona.
3. Crear el microservicio **registracion-service** con una API REST para enviar los nuevos registros de usuarios a la cola de RabbitMQ.
   3.1. El servicio **Registro** debe enviar las inscripciones a la cola del servicio **Persona** a través de OpenFeign.
4. Escribir la clase **Main** de la aplicación con **@EnableRabbit**.
5. En **application.yml**, incluir la información de acceso a RabbitMQ y el nombre de la cola utilizada entre los dos microservicios.

```yaml
spring:
 application:
   name: registracion-service
 rabbitmq:
   username: guest
   password: guest
   host: localhost
   port: 5672

queue:
 persona:
   name: personaQueue
```

6. Configurar en ambos servicios la plantilla de la información enviada en la cola de RabbitMQ para que haya congruencia entre la información enviada y la recibida, instanciando una clase **RabbitTemplate**.

```java
@Configuration
public class RabbitTemplateConfig {


   @Bean
   public Jackson2JsonMessageConverter producerJackson2MessageConverter(){
       return new Jackson2JsonMessageConverter();
   }

   @Bean
   public RabbitTamplate rabbitTamplate(ConnectionFactory connectionFactory){
       RabbitTamplate rabbitTamplate = new RabbitTamplate(connectionFactory);
```

```
        rabbitTamplate.setMessageConverter(producerJackson2MessageConverter());
        return rabbitTamplate;
    }

}
```

7. Configurar la cola en la que los microservicios deben enviar o esperar información.

```
@Configuration
public class RabbitMQSenderConfig {

    @value("${queue.persona.name}")
    public String personaQueue;

    @Bean
    public Queue queue(){
        return new Queue(this.personaQueue, true);
    }

}
```

8. Iniciar las aplicaciones y enviar los nuevos registros a la cola, mientras se observa **localhost:15672**.