Certified Tech Developer
The Ultimate Degree

Especialização Back end Java

Exercício Mesa de trabalho

A empresa de cosméticos “Bella Donna” teve seu sistema indisponibilizado durante uma black friday devido à altíssima taxa de requisições de cadastro de novos usuários em sua plataforma querendo aproveitar as promoções. Portanto, a pedido da empresa, criamos um serviço de cadastro à parte do serviço de persistência e, para evitar futuras indisponibilidades, implementaremos uma fila de requisições utilizando o RabbitMQ.

Dependências

- Estoque e Repositório:
  - Spring Boot Starter Ampq;
  - Spring Boot Starter Web;
  - Spring Boot Starter Data JPA;
  - Spring Boot Starter JDBC;
  - Spring Boot Autoconfigure;
  - Spring Cloud Starter OpenFeign;
  - H2 Database;
  - Lombok.

Instruções

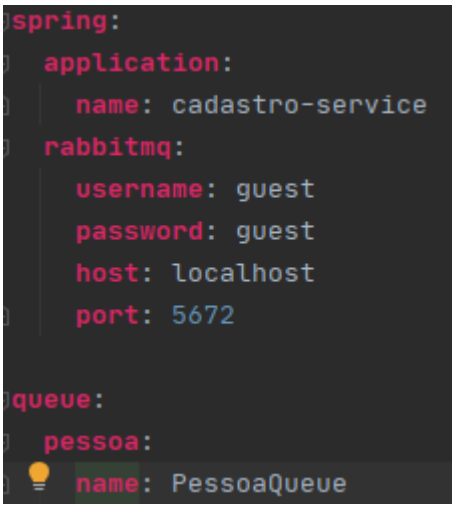
1. Crie o arquivo docker-compose.yml e inicie o servidor RabbitMQ;

2. Crie o microsserviço “pessoa-service” com uma REST API contendo os endpoints necessários para recuperar e salvar uma nova pessoa;

3. Crie o microsserviço "cadastro-service" com uma REST API para enviar os novos cadastros de usuários para a fila do RabbitMQ;
   3.1. O serviço “Cadastro” deve enviar os cadastros para a fila do serviço “Pessoa” através do OpenFeign;

4. Anote a classe Main da sua aplicação com **@EnableRabbit**;

5. Em seu **application.yml**, inclua as informações de login do RabbitMQ e o nome da fila utilizada entre os dois microsserviços;



6. Configure em ambos os serviços o template das informações enviadas na fila do RabbitMQ para que haja congruência entre as informações enviadas e recebidas, instanciando uma classe **RabbitTemplate**;

```java
@Configuration
public class RabbitTemplateConfig {

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

    @Bean
    public RabbitTemplate rabbitTemplate(ConnectionFactory connectionFactory) {
        RabbitTemplate rabbitTemplate = new RabbitTemplate(connectionFactory);
        rabbitTemplate.setMessageConverter(producerJackson2MessageConverter());
        return rabbitTemplate;
    }
}
```

7. Configure a fila que nossos microsserviços devem enviar ou esperar informações;

```java
@Configuration
public class RabbitMQSenderConfig {

    @Value ("${queue.pessoa.name}")
    private String pessoaQueue;

    @Bean
    public Queue queue() { return new Queue(this.pessoaQueue,  durable: true); }
}
```

8. Inicie as aplicações e envie novos cadastros para a fila, enquanto observa **localhost:15672**.