Springboot Microservices

RabbitMQ

Dependência necessária

```
<dependency>
        <groupId>org.springframework.amqp</groupId>
        <artifactId>spring-rabbit</artifactId>
</dependency>
```

Configuração no application.properties do projeto JavaSpringBoot

```
spring.application.name=PREENCHER DO PROJETO UTILIZADO


spring.rabbitmq.host=localhost          # RabbitMQ host.
spring.rabbitmq.username= admin         # Usuario Login do broker.
spring.rabbitmq.password=admin123 # senha para autenticação do broker.
spring.rabbitmq.port=5672               # porta do RabbitMQ.
```

configuration/MQConfig.java (parte 1)

```
import org.springframework.amqp.core.AmqpAdmin;
import org.springframework.amqp.core.Binding;
import org.springframework.amqp.core.DirectExchange;
import org.springframework.amqp.core.Queue;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import jakarta.annotation.PostConstruct;

@Component
public class MQConfig {
        @Autowired
        private AmqpAdmin amqpAdmin;
        private Queue queue;
        private Queue queue (String queueName){
                return new Queue(queueName, true, false, false);
        }
        private DirectExchange createDirectExchange(){
                return new DirectExchange("ecommercermq");
        }
```

configuration/MQConfig.java (parte 2)

```
        @PostConstruct
        private void Create (){
                this.queue = new Queue("fila-ecommerce");

                // Create the direct exchange
                 DirectExchange directExchange = createDirectExchange();

                // Create the binding
                Binding binding = new Binding(queue.getName(), Binding.DestinationType.QUEUE,
                directExchange.getName(), queue.getName(), null);
                amqpAdmin.declareQueue(queue);
                amqpAdmin.declareExchange(directExchange);
                amqpAdmin.declareBinding(binding);
        }
}
```

Chamada envio de parâmetro na fila do RMQ

* Class DTO precisa ter implements Serializable

```
class produtoService{
...
        @Autowired
        private RabbitTemplate rabbitTemplate;

        private create(DTO dto){
                consulta = repository.save(dto)
                rabbitTemplate.convertAndSend("fila-ecommerce", consulta);
        }
}
```

Consumo dos elementos da fila

```
class produtoService{
...
        @Autowired
        private RabbitTemplate rabbitTemplate;

        @RabbitListener(queues = "fila-ecommerce")
        private void subscribe(Municipio municipio){
                System.out.println(municipio.getNome());
        }
}
```