Engenharia de Softwares Escaláveis

Domain-Driven Design (DDD) e Arquitetura de Softwares Escaláveis com Java

Agenda

Etapa 6: Design e Implementação de Microsserviços com Comunicação Assíncrona e Message Brokers em Java.

- Visão Geral.
- Usos Especiais de Mensageria.
- Message Broker.

Visão Geral

Integrações que usam mensageria os serviços se comunicam trocando mensagens **assincronamente**.

Um aplicativo baseado em mensagens normalmente usa um **Message Broker**, que atua como um intermediário entre os serviços.

Um cliente faz uma solicitação a um serviço enviando uma mensagem. Como a comunicação é assíncrona, o cliente não bloqueia a espera por uma resposta. Em vez disso, o cliente é escrito assumindo que a resposta não será recebida imediatamente.

Uma mensagem consiste em um **cabeçalho** e um **corpo de mensagem**.

O **cabeçalho** é uma coleção de pares nome-valor, metadados que descrevem os dados que estão sendo enviados.

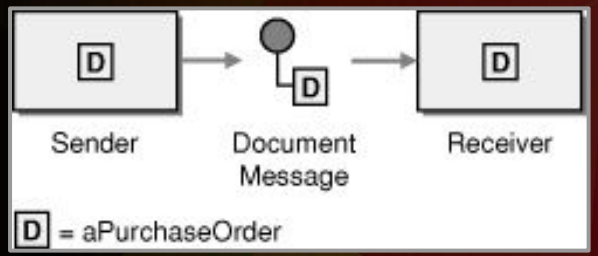
O **corpo da mensagem** são os dados que estão sendo enviados, em formato de texto ou binário.

Documento

Uma mensagem genérica que contém apenas dados.

O receptor decide como interpretá-la.

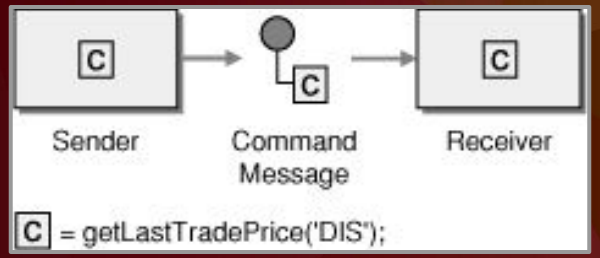
A resposta a um comando é um exemplo de uma mensagem de documento.



Command

Uma mensagem que é o equivalente a uma solicitação RPC.

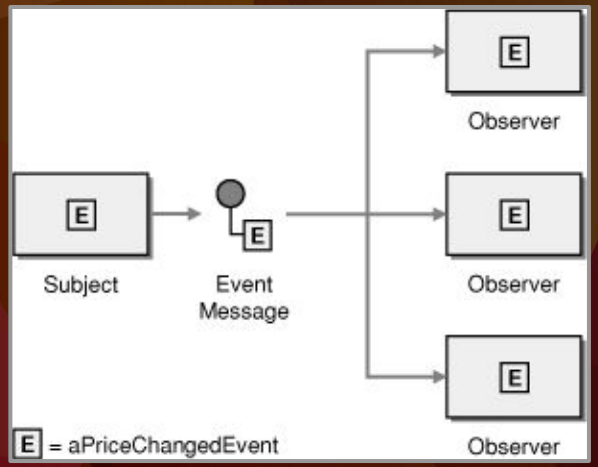
Ela especifica a operação a ser invocada e seus parâmetros.



Evento

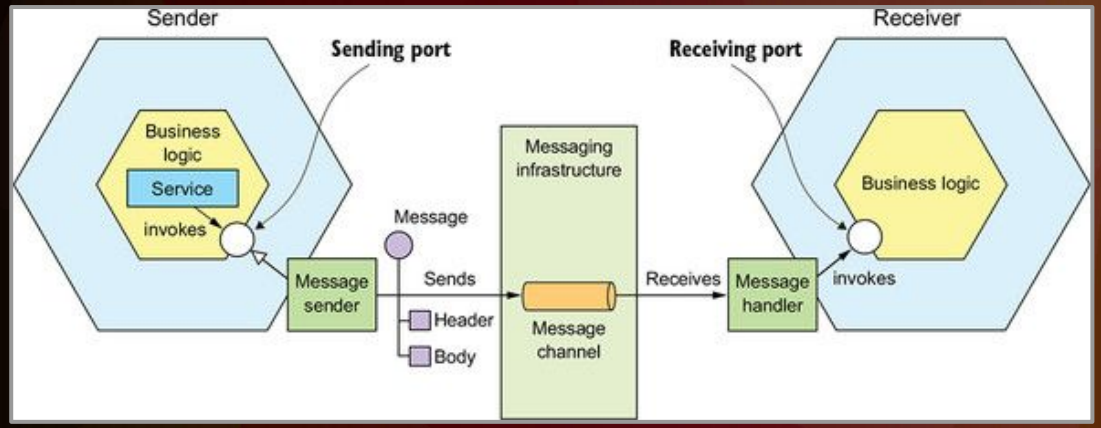
Uma mensagem indicando que algo notável ocorreu no remetente.

Um evento é frequentemente um evento de domínio, que representa uma mudança de estado de um objeto de domínio.



Tipos de Mensagens

Canal de Mensagens



A lógica de negócios no remetente invoca uma interface de porta de envio, que é implementada por um adaptador de remetente de mensagem.

O remetente de mensagem envia uma mensagem para um destinatário por meio de um canal de mensagem.

O canal de mensagem é uma abstração da infraestrutura de mensagens.

Um adaptador de manipulador de mensagem no destinatário é invocado para manipular a mensagem. Ele invoca a interface de porta de recebimento implementada pela lógica de negócios do destinatário.

Ponto a Ponto

Um canal ponto a ponto entrega uma mensagem para exatamente um dos consumidores que está lendo do canal.

Os serviços usam canais ponto a ponto para os estilos de interação um-para-um.

Por exemplo, uma **mensagem de comando** é frequentemente enviada por um canal ponto a ponto.

Publish-Subscribe

Um canal publish-subscribe entrega cada mensagem a todos os consumidores anexados.

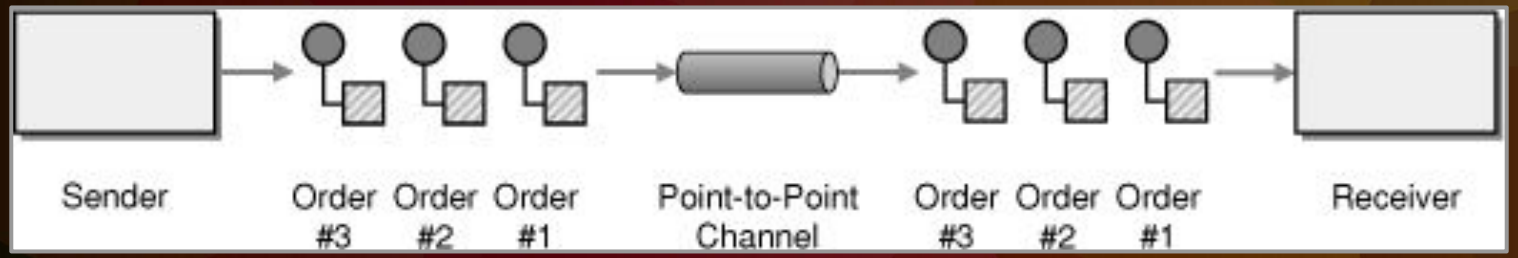
Os serviços usam canais publish-subscribe para os estilos de interação um-para-muitos.

Por exemplo, uma **mensagem de evento** é geralmente enviada por um canal publish-subscribe.

Tipos de Canais

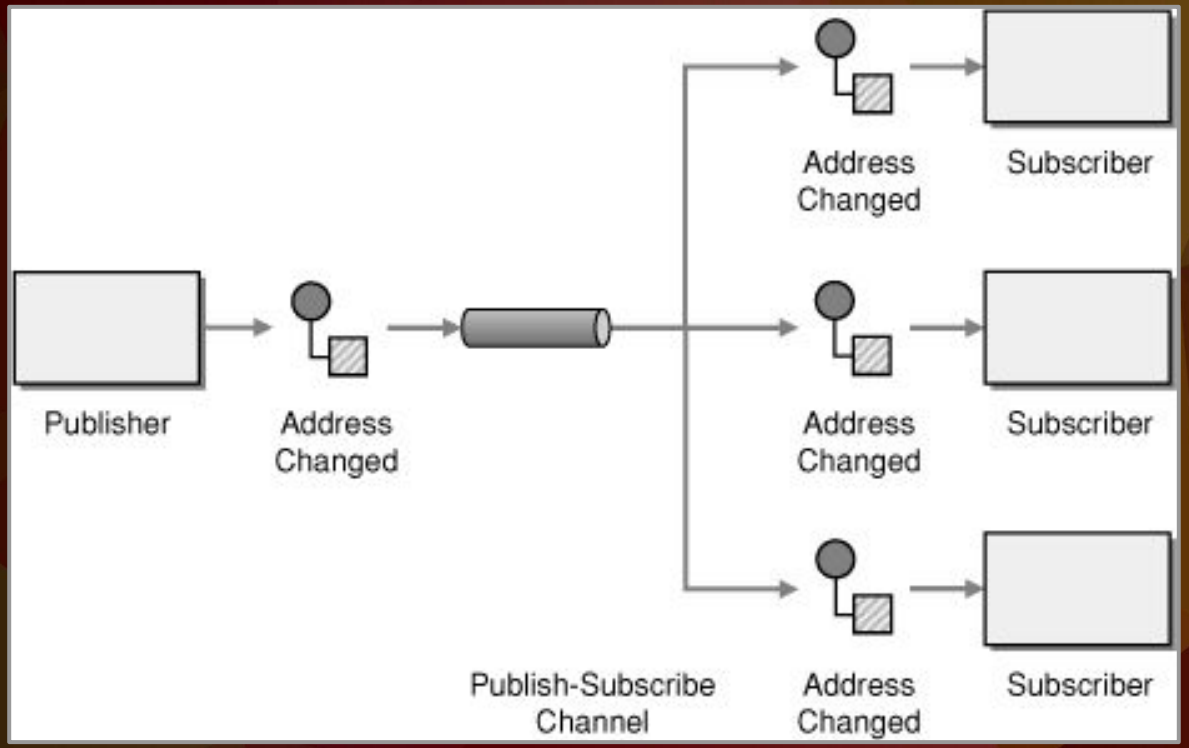
Tipos de Canais

Ponto a Ponto

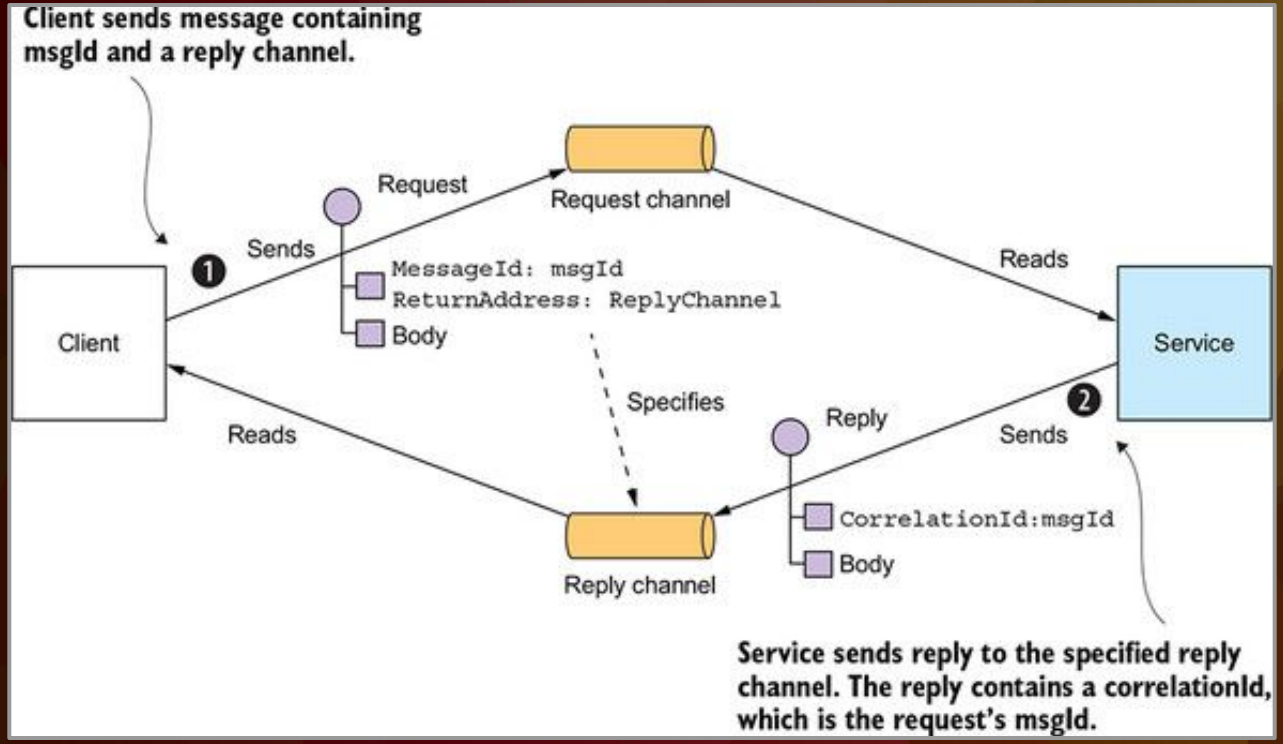


Tipos de Canais

Publish-Subscribe

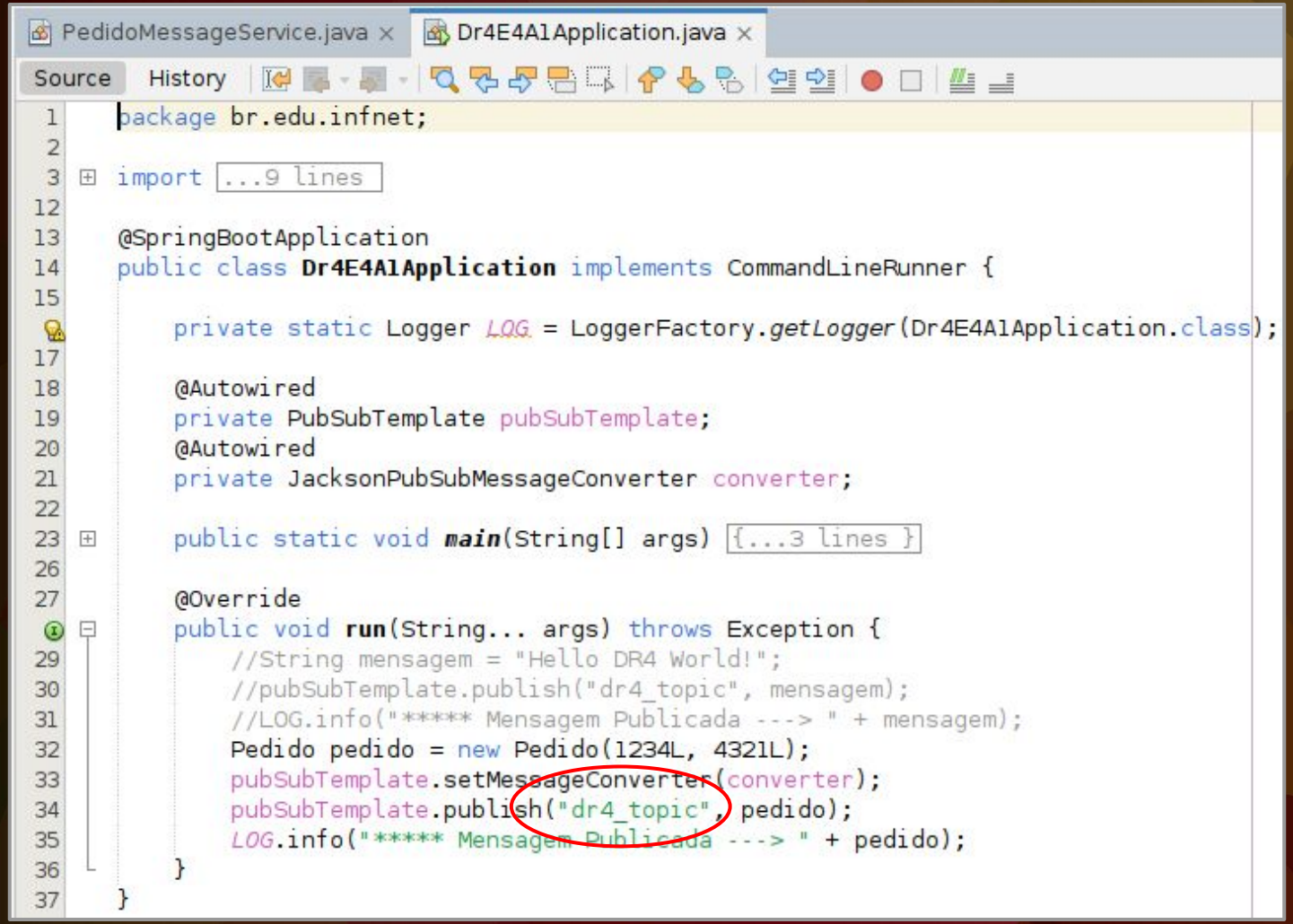


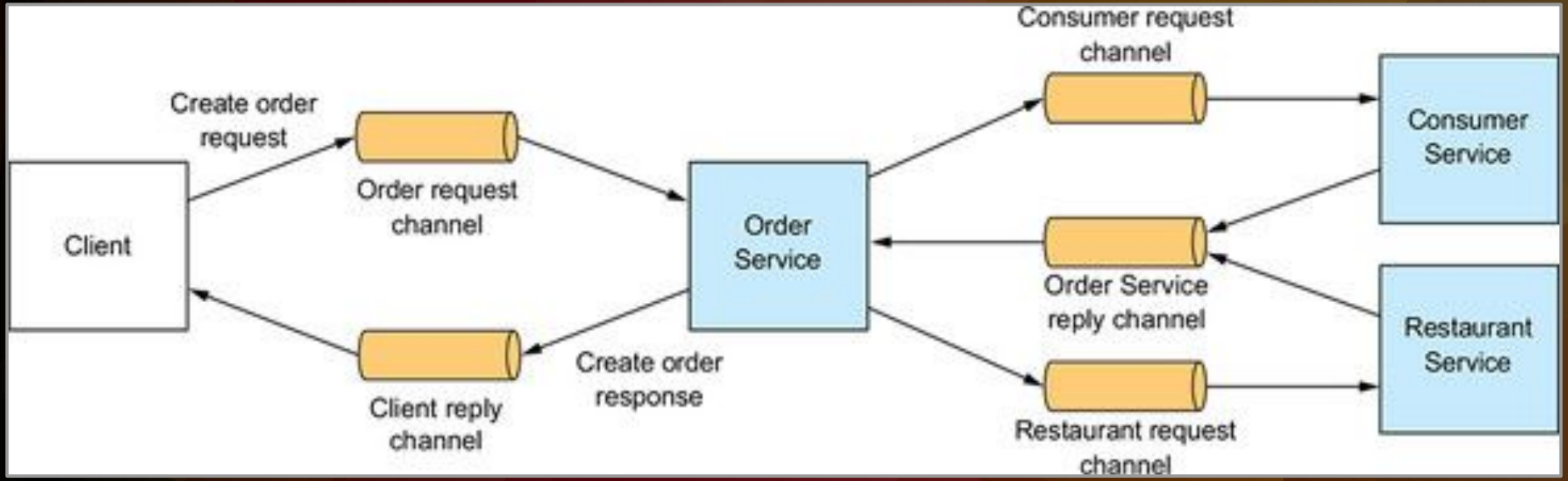
Usos Especiais de Mensageria



Implementando solicitação/resposta assíncrona **incluindo um canal de resposta e um identificador de mensagem** na mensagem de solicitação.

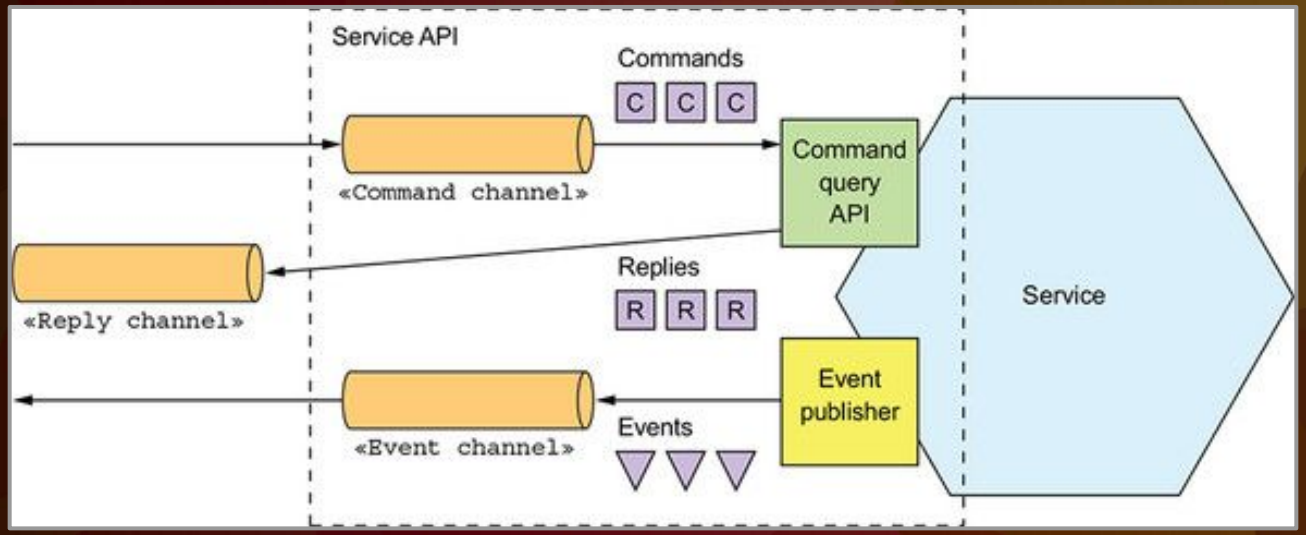
O receptor processa a mensagem e envia a resposta para o canal de resposta especificado.


PedidoMessageService.java
Dr4E4A1Application.java
Source
History
package br.edu.infnet;
import ...9 lines
@SpringBootApplication
public class Dr4E4A1Application implements CommandLineRunner {
private static Logger LOG = LoggerFactory.getLogger(Dr4E4A1Application.class);
@Autowired
private PubSubTemplate pubSubTemplate;
@Autowired
private JacksonPubSubMessageConverter converter;
public static void main(String[] args) {...3 lines }
@Override
public void run(String... args) throws Exception {
//String mensagem = "Hello DR4 World!";
//pubSubTemplate.publish("dr4_topic", mensagem);
//LOG.info("***** Mensagem Publicada ---> " + mensagem);
Pedido pedido = new Pedido(1234L, 4321L);
pubSubTemplate.setMessageConverter(converter);
pubSubTemplate.publish("dr4_topic", pedido);
LOG.info("***** Mensagem Publicada ---> " + pedido);
}
}



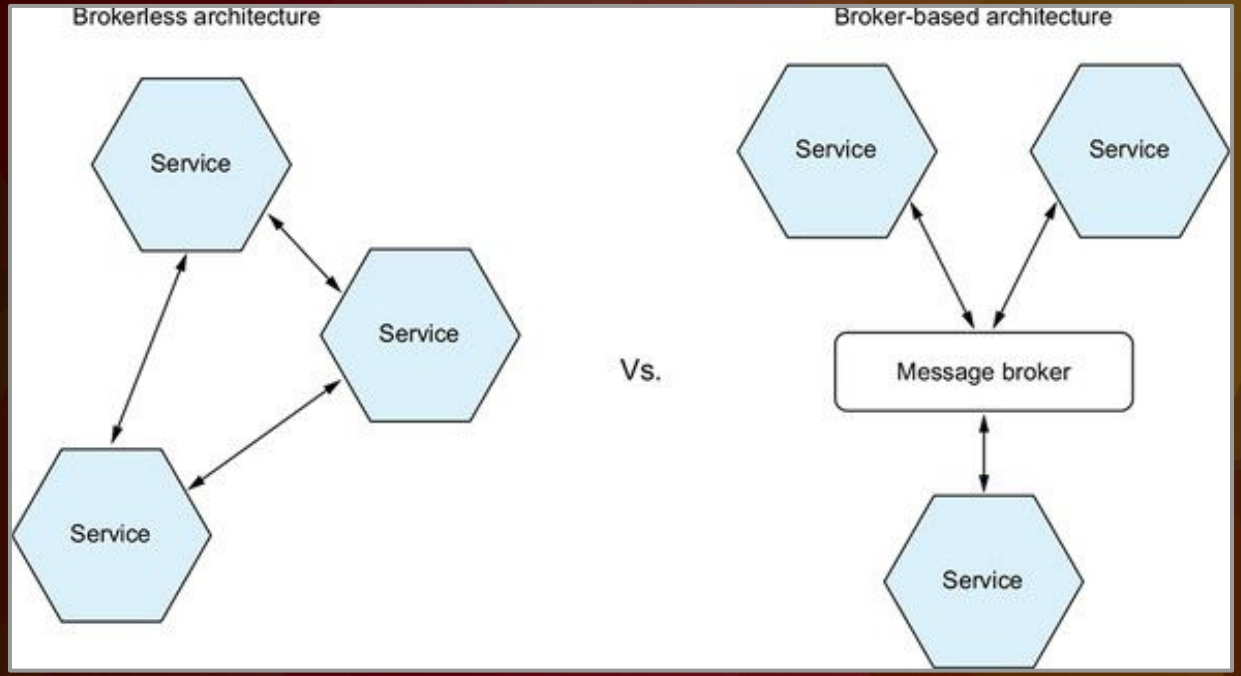
O cliente e os serviços se comunicam de forma assíncrona enviando mensagens por meio de canais de mensagens.

Nenhum participante dessa interação fica bloqueado esperando por uma resposta.



A API assíncrona de um serviço consiste em canais de mensagens e tipos de mensagens de comando, resposta e evento.

Message Broker



Um aplicativo baseado em mensagens normalmente usa um message broker, um serviço de infraestrutura por meio do qual o serviço se comunica.

Você também pode usar uma arquitetura de mensagens brokerless, na qual os serviços se comunicam diretamente.

Exemplos de message brokers populares de código aberto incluem os seguintes:

- ActiveMQ (http://activemq.apache.org)
- RabbitMQ (https://www.rabbitmq.com)
- Apache Kafka (http://kafka.apache.org)

Também existem serviços de mensagens baseados em nuvem:

- AWS SNS / SQS (https://aws.amazon.com/sqs/).
- Google PubSub.

Receptores Concorrentes e Ordenação de Mensagens

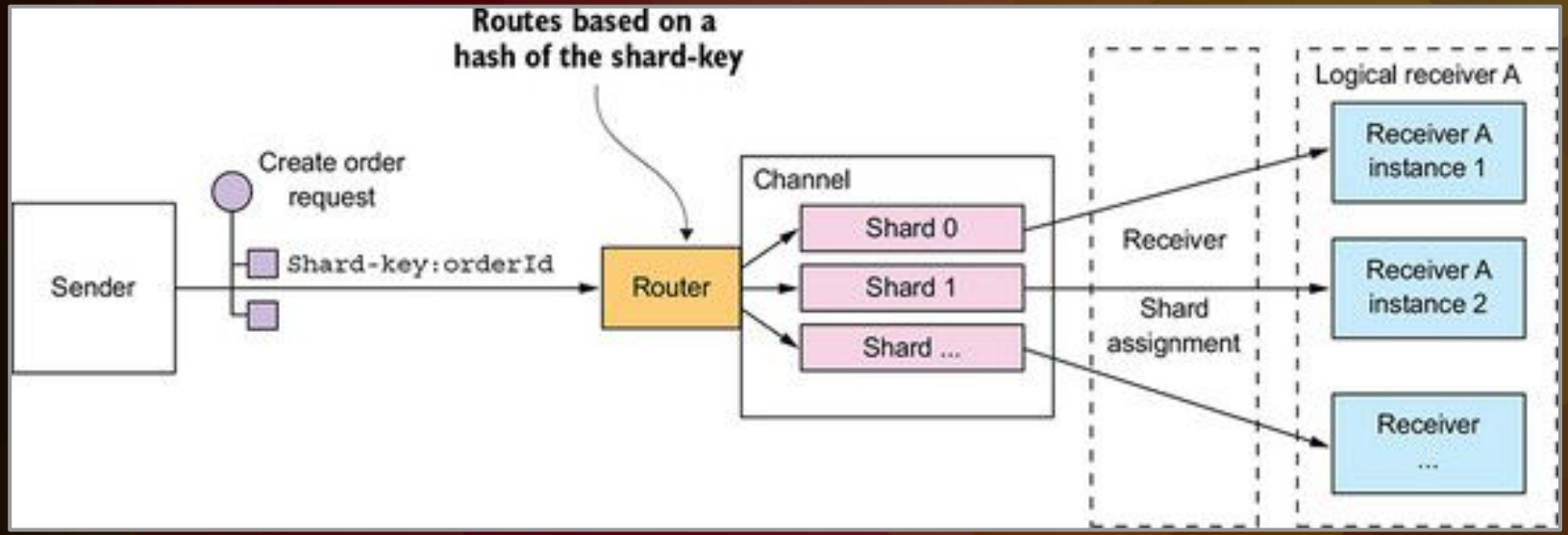
Um desafio é como escalar receptores de mensagens enquanto preserva a ordem das mensagens.

É um requisito comum ter várias instâncias de um serviço para processar mensagens simultaneamente.

Além disso, mesmo uma única instância de serviço provavelmente usará threads para processar várias mensagens simultaneamente.

Usar vários threads e instâncias de serviço para processar mensagens simultaneamente aumenta o rendimento do aplicativo.

Mas o desafio com o processamento de mensagens simultaneamente é garantir que cada mensagem seja processada uma vez e em ordem.



Dimensionando consumidores enquanto preserva a ordenação de mensagens usando um canal de mensagem fragmentado (particionado).

O remetente inclui a **chave de fragmento** na mensagem.

O broker de mensagem grava a mensagem em um fragmento determinado pela **chave de fragmento**.

O broker de mensagem atribui cada partição a uma instância do receptor replicado.

Lidando com Mensagens Duplicadas

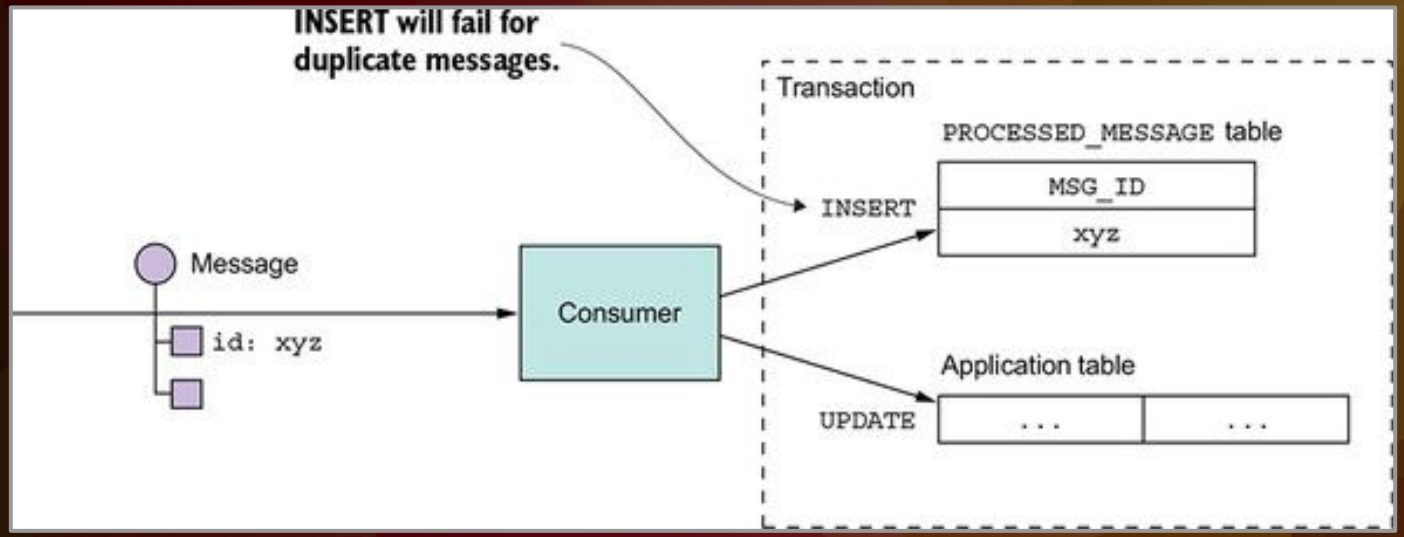
Um message broker deve idealmente entregar cada mensagem apenas uma vez, mas garantir mensagens exatamente uma vez é geralmente muito custoso.

Em vez disso, a maioria dos message brokers promete entregar uma mensagem pelo menos uma vez.

Mas uma falha de um cliente, rede ou do message broker pode resultar em uma mensagem sendo entregue várias vezes.

Digamos que um cliente trava após processar uma mensagem e atualizar seu banco de dados, mas antes de reconhecer a mensagem.

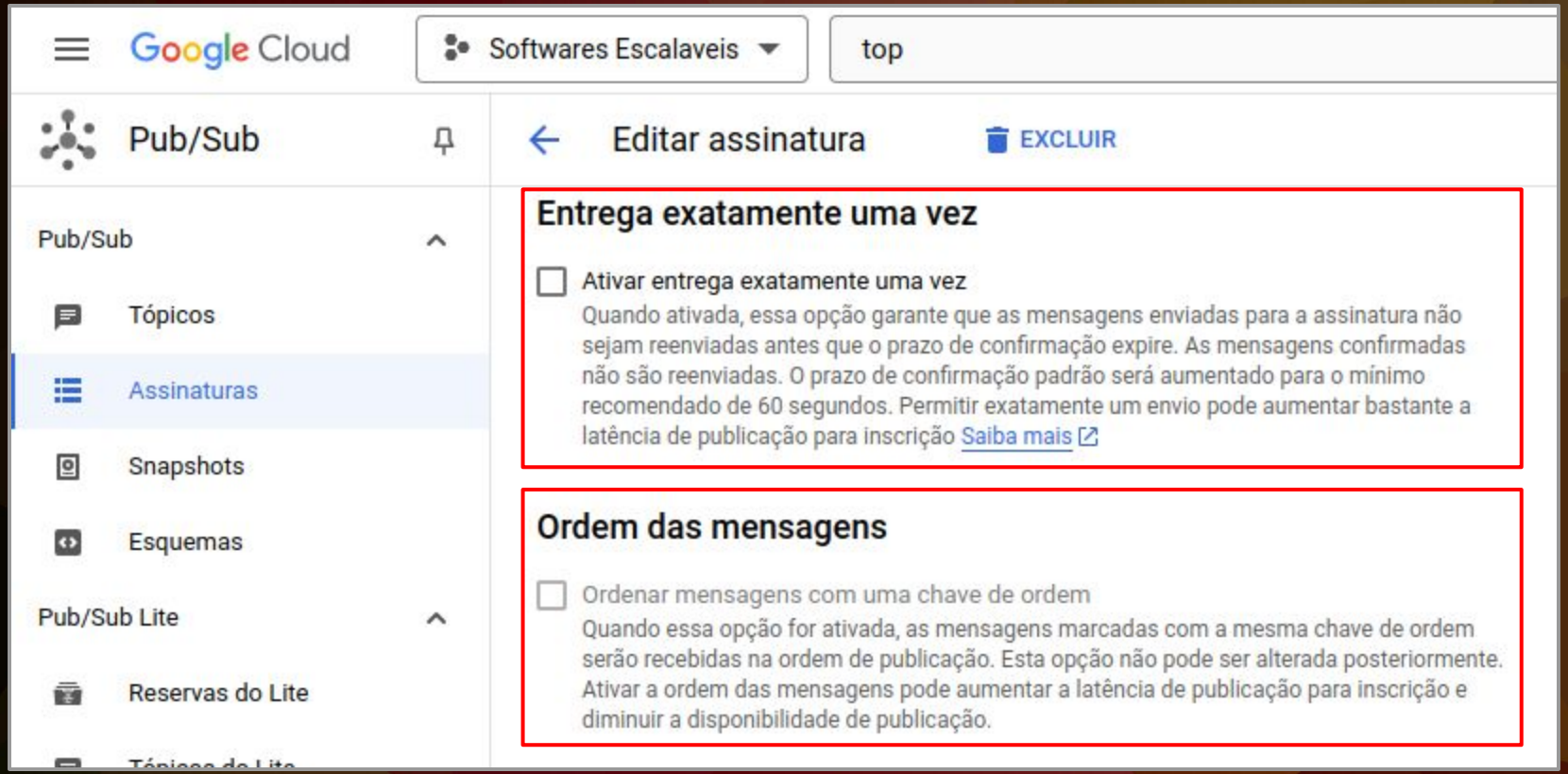
O message broker entregará a mensagem não reconhecida novamente, seja para esse cliente quando ele reiniciar ou para outra réplica do cliente.



Um consumidor detecta e descarta mensagens duplicadas registrando os IDs de mensagens processadas em uma tabela de banco de dados.

Se uma mensagem tiver sido processada antes, o INSERT na tabela PROCESSED_MESSAGES falhará.

Outra opção é que um manipulador de mensagens registre message ids em uma tabela de aplicativo em vez de uma tabela dedicada.


Google Cloud
Softwares Escalaveis
top
Pub/Sub
Editar assinatura
EXCLUIR
Pub/Sub
Tópicos
Assinaturas
Snapshots
Esquemas
Pub/Sub Lite
Reservas do Lite
Entrega exatamente uma vez
Ativar entrega exatamente uma vez
Quando ativada, essa opção garante que as mensagens enviadas para a assinatura não sejam reenviadas antes que o prazo de confirmação expire. As mensagens confirmadas não são reenviadas. O prazo de confirmação padrão será aumentado para o mínimo recomendado de 60 segundos. Permitir exatamente um envio pode aumentar bastante a latência de publicação para inscrição Saiba mais
Ordem das mensagens
Ordenar mensagens com uma chave de ordem
Quando essa opção for ativada, as mensagens marcadas com a mesma chave de ordem serão recebidas na ordem de publicação. Esta opção não pode ser alterada posteriormente. Ativar a ordem das mensagens pode aumentar a latência de publicação para inscrição e diminuir a disponibilidade de publicação.