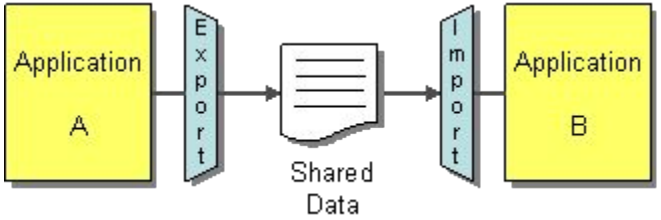
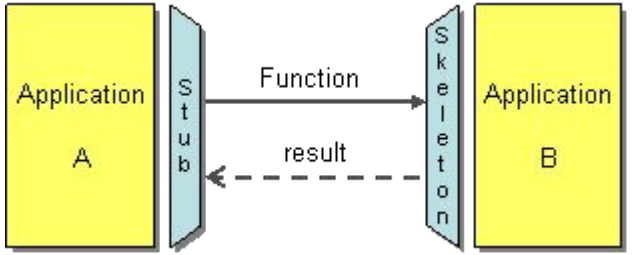
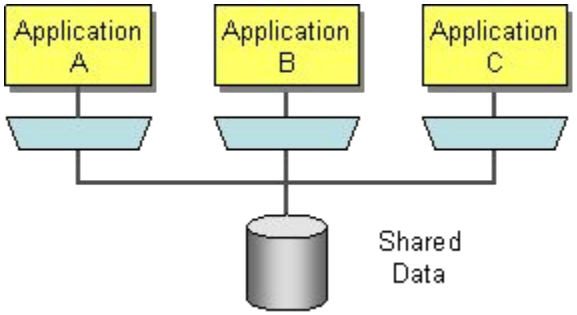
Mensageria

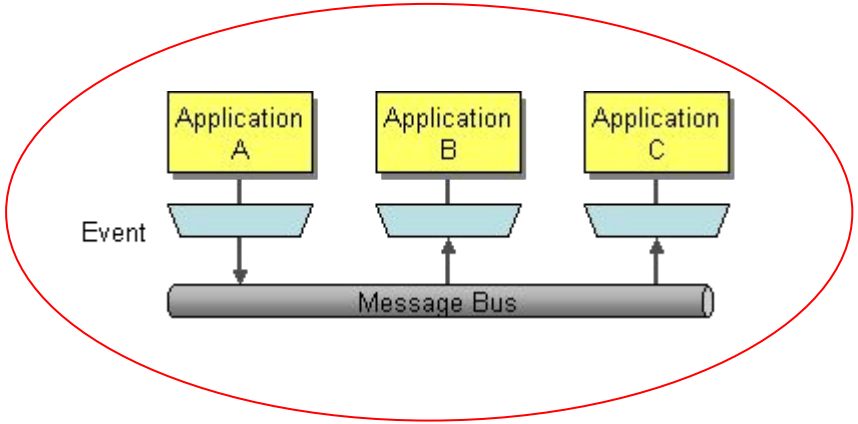
Álvaro Ferreira Pires de Paiva (20211028885)
Italo Oliveira Fernandes (20211028965)

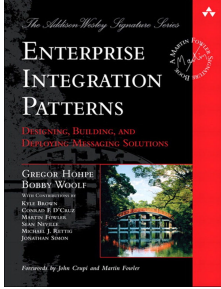
Comunicação entre aplicações





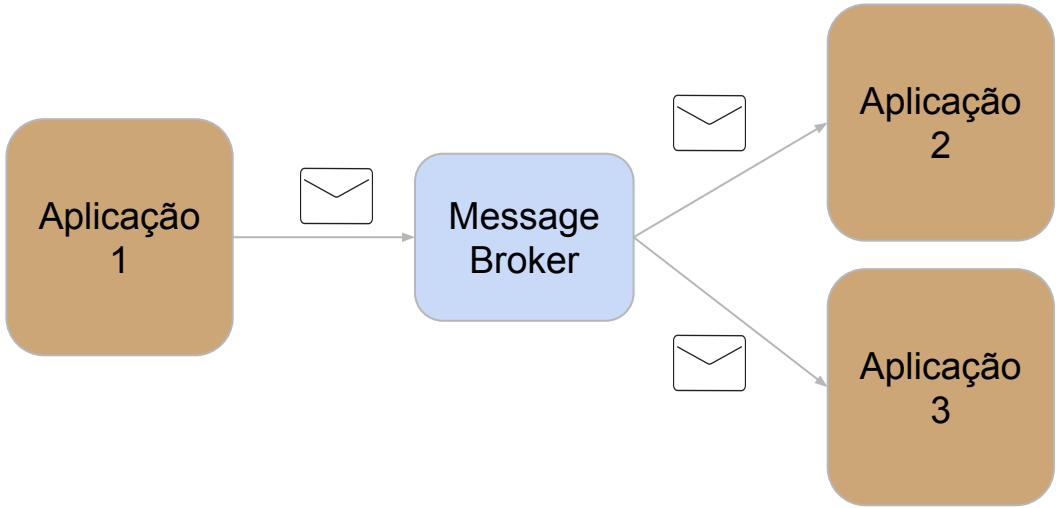


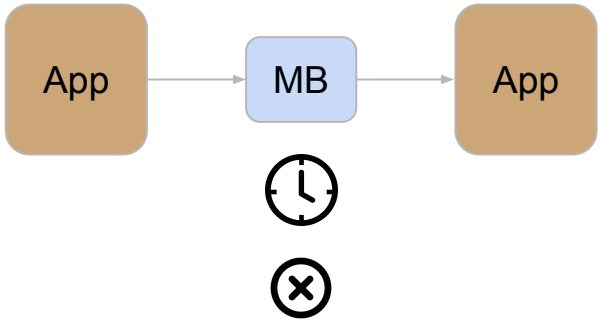




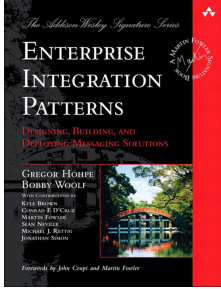
Mensageria (o que é?)

"*Mensageria é um conceito que define que sistemas distribuídos, possam se comunicar por meio de troca de mensagens (evento), sendo estas mensagens "gerenciadas" por um Message Broker (servidor/módulo de mensagens).*" *Fonte: https://medium.com/@devbrito91/mensageria-1330c6032049*

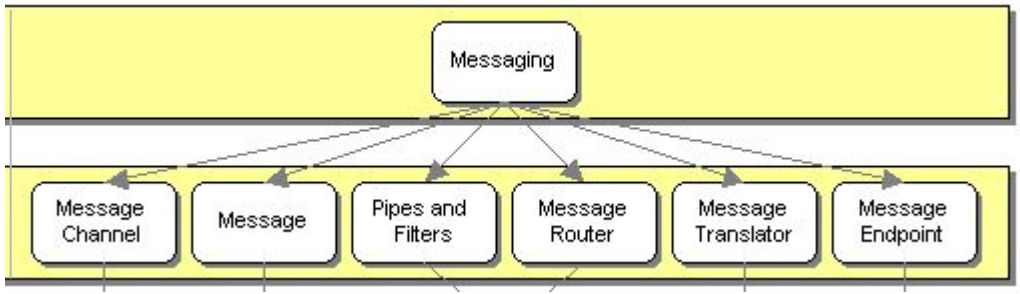




Mensageria (elementos e componentes)



- **Event/Message**: A mensagem.
- **Broker**: Recebe e envia mensagens.
- **Producer**: É quem produz a mensagem.
- **Consumer**: É quem consome a mensagem.



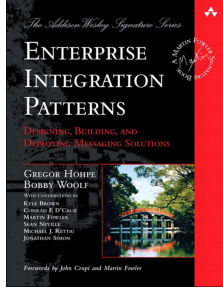
Mensageria (Quando usar)

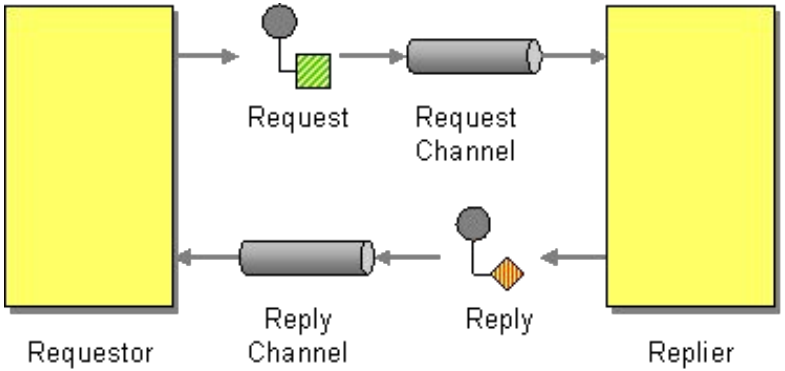
- Comunicação assíncrona;
- Desacoplamento;
- Escalabilidade;
- Resiliência;
- Aplicações distribuídas.

Mensageria (desvantagens)

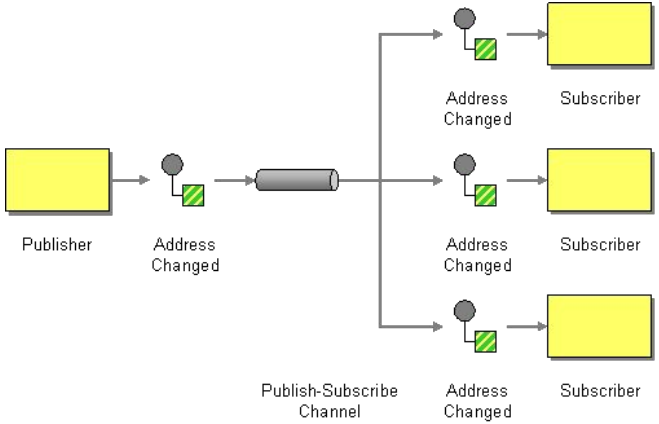
- Complexidade;
- Depuração em casos de erro;
- Monitoramento;
- Gerenciamento.

Mensageria (Alguns padrões)



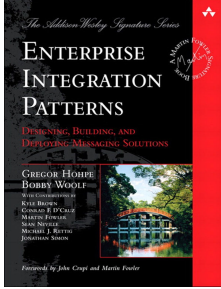


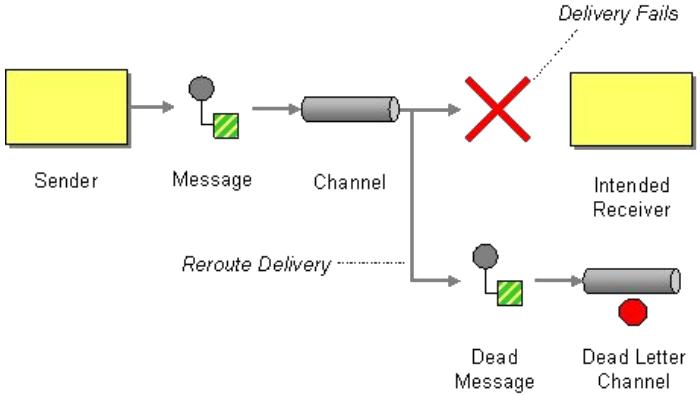
Request-reply



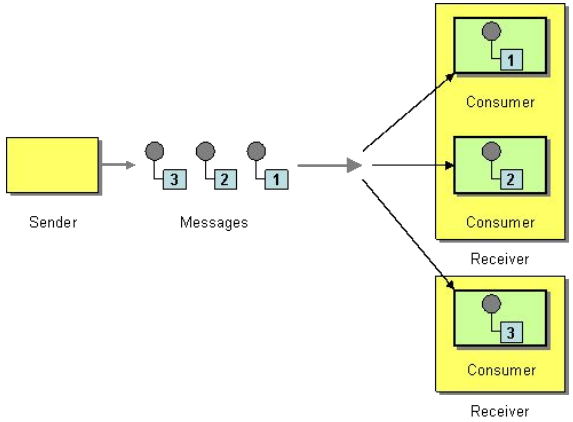
Publish-subscriber

Mensageria (Alguns padrões)





Dead Letter Channel



Competing consumers

Apache Kafka

Kafka (o que é?)

Apache Kafka é uma plataforma *open-source* de *streaming* de eventos distribuídos, sendo do tipo *publish-subscribe*. Ele possui 3 recursos principais:

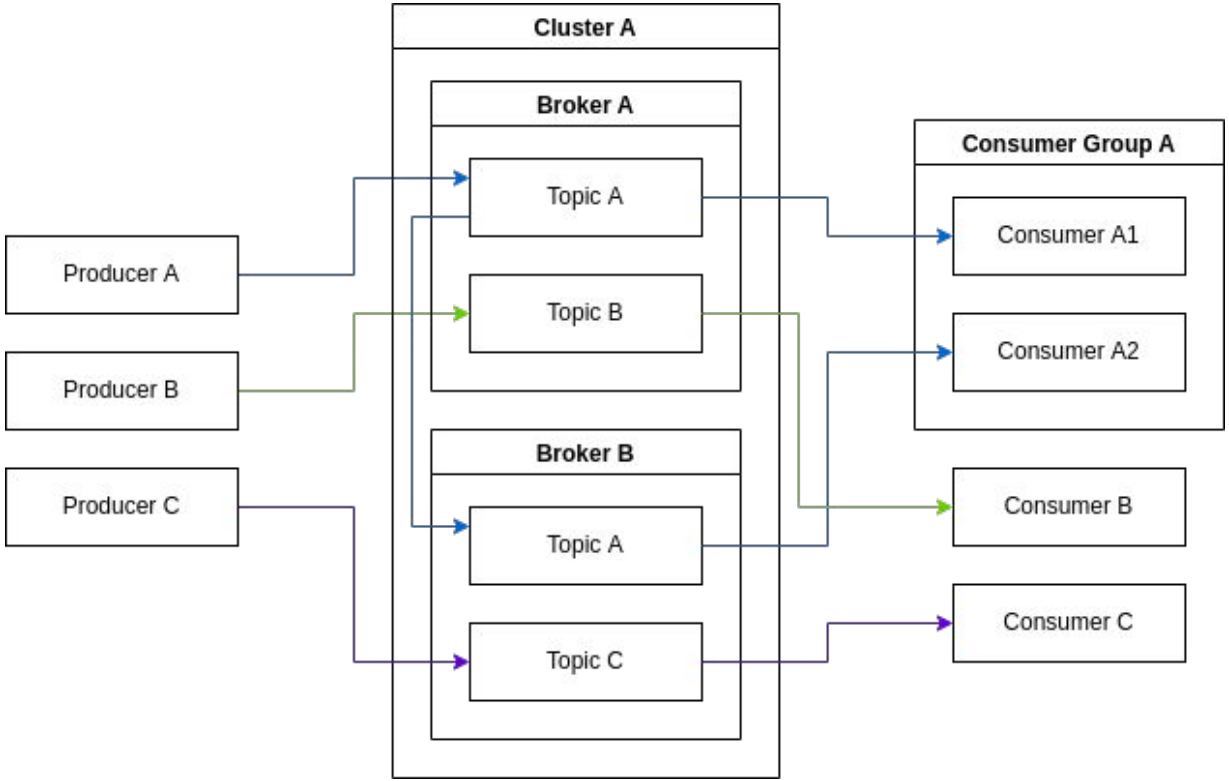
1. **Publicar** e **assinar** fluxos de eventos;
2. **Armazenar** fluxos de eventos;
3. **Processar** fluxos de eventos.

Além disso, possui mecanismos de tolerância a falhas.

Kafka (componentes)

- **Event**: Consiste em um registro de que algo acontece. Possui 3 partes principais: chave, valor e data e hora.
- **Topic**: Base de dados dos eventos.
- **Partition**: Permite a replicação de tópicos dentro do cluster.
- **Broker**: Recebe e envia eventos, além de armazenar às partições.
- **Cluster**: É um conjunto de *brokers*.
- **Producer**: É quem produz o evento.
- **Consumer**: É quem consome o evento.
- **Consumer group**: Grupo de consumidores de um determinado tópico.

Kafka (exemplo de arquitetura)



Kafka (vantagens e desvantagens)

Vantagens:

- Alta disponibilidade;
- Transmissões assíncronas;
- Armazenamento em disco rígido;
- Permite reprocessar os eventos;
- Maior garantia de entrega do evento.

Desvantagens:

- Configuração complexa;
- Mensagens não são entregues de maneira ordenada;
- Exige uma conexão boa.

Kafka (curiosidade)

- O *broker* não envia os eventos para os consumidores, são os consumidores que solicitam os eventos ao *broker*.
- O Kafka faz acesso sequencial, devido a isso é mais interessante armazenar os dados em disco rígido.

RabbitMQ

RabbitMQ (o que é?)

RabbitMQ é um servidor de mensageria *open-source*, implementado o protocolo *Advanced Message Queueing Protocol* (AMQP).

Diferentemente do Kafka, o RabbitMQ trabalha com o conceito de filas. Após a mensagem ser consumida, ela é apagada permanentemente.

Atualmente suporta outros protocolos além do AMQP.

RabbitMQ (componentes)

- **Message**: Conjunto de dados que se deseja repassar. Possui 2 partes principais: payload (corpo da mensagem) e label (identifica o que tem na mensagem e quem irá consumir).
- **Queue**: Fila de mensagens esperando serem consumidas pelos consumidores.
- **Exchange**: Responsável por encaminhar a mensagem para a fila correta.
- **Binding**: Conjunto de regras que classificam às mensagens em filas.
- **Producer**: É quem publica a mensagem.
- **Consumer**: É quem consome a mensagem.

RabbitMQ (componentes - exchange)

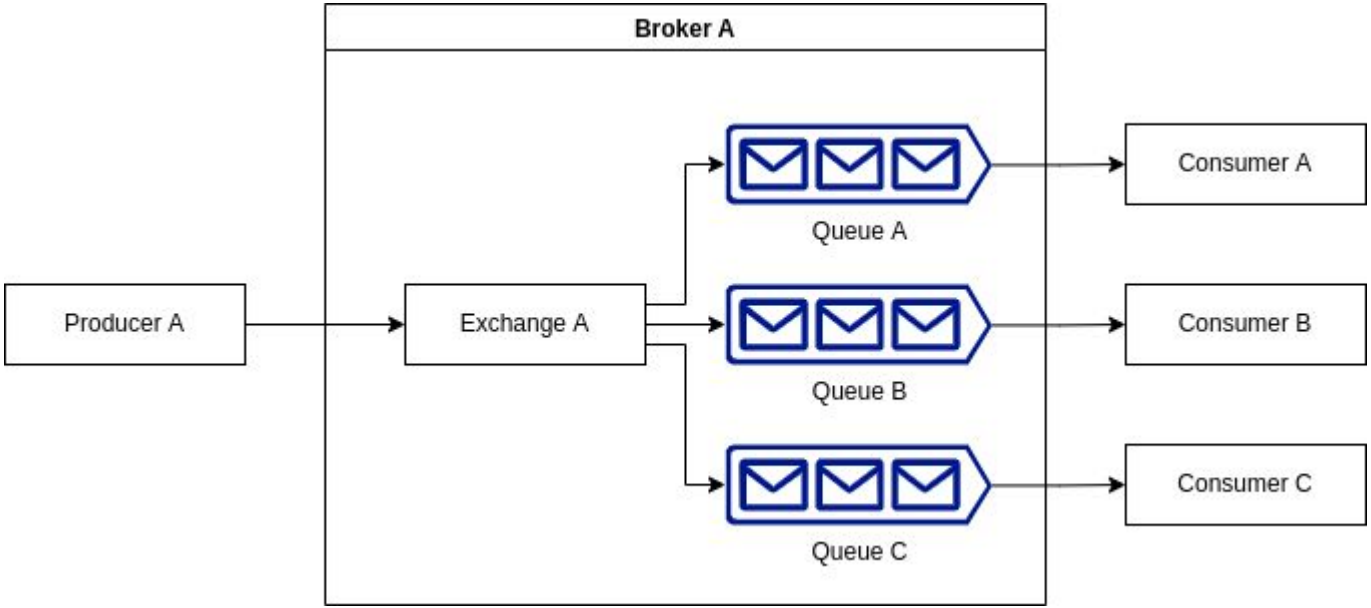
Identificação de ligações entre *message*, *exchange* e *queue*:

- **Binding key**: chave específica da ligação entre a *queue* e o *exchange*.
- **Routing key**: chave enviada junto a *message*.

Tipos de *Exchange*:

- **Direct**: Encaminha a message procurando uma *binding key* igual ao *routing key* fornecido.
- **Topic**: Encaminha para às filas com *binding keys* que atendem a um padrão especificado.
- **Fanout**: Envia para todas as filas vinculadas.
- **Headers**: Utiliza os valores do header para encaminhamento.

RabbitMQ (exemplo de arquitetura)



RabbitMQ (vantagens e desvantagens)

Vantagens:

- Garantia de assincronicidade entre aplicações;
- Fácil integração com outros componentes;
- Garantia de ordem de entrega;
- Real-time.

Desvantagens:

- Difícil manutenibilidade;
- Sem garantia de entrega;
- Baixo *throughput*.

Comparação entre Kafka e RabbitMQ

Retenção da mensagem	Após o fim de um período pré-determinado.	Após ser consumida.
Escalonamento	Horizontal	Vertical
Reprocessamento	Possui.	Não possui.
Prioridade de mensagem	Não possui.	Possui.
Throughput (MB/s)	605 MB/s	38 MB/s
P99 latency (ms)	5 ms	1 ms

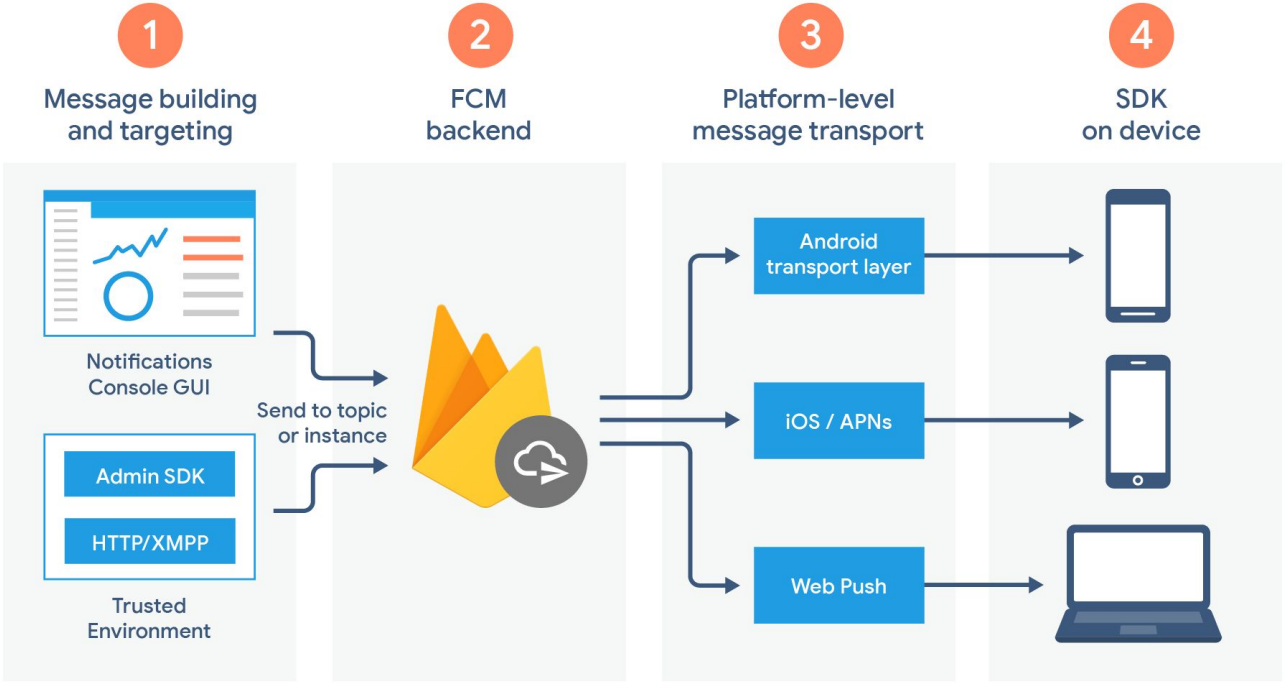
Firebase

Firebase (o que é?)

O Firebase Cloud Messaging (FCM) é uma solução de mensagens multiplataforma que permite enviar mensagens de forma confiável e sem custo.

Usando o FCM, você pode notificar um aplicativo cliente de que novos e-mails ou outros dados estão disponíveis para sincronização. Você pode enviar mensagens de notificação para estimular o reengajamento e a retenção do usuário.

Firebase (arquitetura)



Obrigado!