Arquiteturas Modernas
Mensageria e Filas de Mensagens

Prof. Msc. Francisco de Assis de Souza Rodrigues



🎯 Objetivos

- Entender o papel da mensageria em arquiteturas distribuídas modernas.
- Compreender o funcionamento de filas de mensagens.
- Conhecer os conceitos fundamentais e casos de uso de RabbitMQ e Apache Kafka.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

📦 1. Introdução à Mensageria

🧠 Por que mensageria?

Mensageria é uma técnica usada para comunicação entre partes de um sistema (geralmente serviços ou aplicações) por meio do envio e recebimento de mensagens. Ela resolve vários problemas típicos de sistemas distribuídos modernos.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

📦 1. Introdução à Mensageria

✅ Desacoplamento entre serviços

- **Sem mensageria:** O Serviço A precisa saber exatamente onde está o Serviço B e como chamá-lo (por exemplo, via HTTP). Isso gera dependência direta.

- **Com mensageria:** O Serviço A envia uma mensagem para uma fila. O Serviço B a consome quando estiver pronto. Eles não precisam se conhecer diretamente.

- Resultado: os serviços são mais independentes e fáceis de manter ou substituir.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

📦 1. Introdução à Mensageria

🚀 Escalabilidade

- Como os produtores e consumidores trabalham de forma independente, é fácil escalar horizontalmente os consumidores para lidar com maior volume de mensagens.

- Exemplo: 1 produtor pode gerar 1000 tarefas por minuto, então colocamos 5 consumidores para processá-las paralelamente.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

📦 1. Introdução à Mensageria

🔄 Resiliência

- Se um consumidor falhar, as mensagens continuam na fila e não são perdidas.

- Permite sistemas mais tolerantes a falhas, pois os componentes podem cair e voltar sem comprometer o fluxo de dados.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

📦 1. Introdução à Mensageria

🔑 Conceitos principais

👤 Produtor (Producer)

- É quem **envia** uma mensagem.

- Exemplo: um sistema de e-commerce que envia um pedido para processamento.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

📦 1. Introdução à Mensageria

🔑 Conceitos principais

📥 Consumidor (Consumer)

- É quem **recebe e processa** a mensagem.

- Exemplo: um serviço de faturamento que gera uma nota fiscal com base no pedido.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

📦 1. Introdução à Mensageria

🔑 Conceitos principais

🧠 Broker (Servidor de Mensagens)

- É o intermediário que **recebe mensagens dos produtores e as entrega aos consumidores**.

- Exemplos: RabbitMQ, Kafka, Redis Streams, AWS SQS.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

📦 1. Introdução à Mensageria

🔑 Conceitos principais

📫 Fila (Queue)

- Uma estrutura **FIFO (First In, First Out)** onde as mensagens esperam para serem consumidas.

- No RabbitMQ, as filas são o principal mecanismo de entrega.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

📦 1. Introdução à Mensageria

🔑 Conceitos principais

🔁 Tópico (Topic)

- Usado em sistemas de publicação/assinatura (pub/sub), como o Kafka.

- Produtores publicam mensagens em um tópico, e múltiplos consumidores podem receber essas mensagens.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

📦 1. Introdução à Mensageria

🔑 Conceitos principais

✉️ Mensagem

- A unidade de dados trafegada entre produtor e consumidor.

- Pode ser texto, JSON, XML, binário, etc.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

📦 1. Introdução à Mensageria

🔑 Conceitos principais

⏱️ Processamento síncrono vs. assíncrono

- **Síncrono**: o produtor espera o consumidor processar (ex: uma chamada HTTP que só retorna depois do processamento).

- **Assíncrono**: o produtor apenas envia e continua; o consumidor processa depois, sem bloquear o fluxo.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

📦 1. Introdução à Mensageria

🔑 Conceitos principais

📊 Exemplo visual (Diagrama simplificado)

[Serv. A - Pedido] ---> envia mensagem ---> **[FILA]** ---> **[Serv. B - Faturamento]**

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

1. Introdução à Mensageria

Conceitos principais



UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

RabbitMQ – Conceito e Aplicações

🟢 O que é RabbitMQ?

- **RabbitMQ** é um sistema de mensageria open-source, usado para enviar e receber mensagens entre sistemas de forma **assíncrona**.

- Ele **intermedia a comunicação entre serviços**, armazenando mensagens temporariamente até que o serviço consumidor esteja pronto para processá-las.

- É baseado no **protocolo AMQP (Advanced Message Queuing Protocol)** que define um padrão confiável e flexível para troca de mensagens.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

RabbitMQ – Conceito e Aplicações

⚙️ Componentes principais do RabbitMQ

🔁 Exchanges

As exchanges recebem mensagens dos produtores e as encaminham para as filas (queues), **de acordo com regras específicas** de roteamento.

Tipos de exchanges:

Tipo	Descrição
Direct	Roteia mensagens com base em uma chave exata (routing key).
Topic	Roteia com base em padrões (wildcards), útil para mensagens com múltiplos critérios.
Fanout	Ignora a chave de roteamento e envia para **todas as filas ligadas** à exchange.
Headers	Roteia com base em cabeçalhos personalizados nas mensagens, ao invés de chave.

RabbitMQ – Conceito e Aplicações

📥 Queues (Filas)

- As filas armazenam as mensagens até que um consumidor as processe.
- Cada fila pode ter **um ou mais consumidores**.
- Uma mensagem pode ser consumida por apenas **um** consumidor por padrão.

📤 Producer (Produtor)

- Envia mensagens para uma **exchange**.

📥 Consumer (Consumidor)

- Escuta uma ou mais filas e consome as mensagens.

RabbitMQ – Conceito e Aplicações

📦 Casos de uso comuns

- **Processamento assíncrono de tarefas**
  Exemplo: quando um usuário envia um formulário, a aplicação principal envia uma mensagem para o RabbitMQ, e um worker em segundo plano processa o envio de e-mail.

- **Comunicação entre microserviços**
  - Em uma arquitetura de microserviços, diferentes sistemas podem se comunicar via filas, evitando dependência direta e possibilitando mai resiliência.

RabbitMQ – Conceito e Aplicações

🧰 Ferramentas de Suporte

- **Painel Web (RabbitMQ Management Plugin)**:
  - Interface gráfica acessível via navegador (geralmente em http://localhost:15672) para monitorar filas, exchanges, mensagens, conexões e mais.
- **CLI (Command Line Interface)**:
  - Comandos como rabbitmqctl e rabbitmqadmin permitem gerenciar broker, usuários, permissões e filas.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

RabbitMQ – Conceito e Aplicações

🧪 Mini-demo teórica: Fluxo de uma mensagem

Exemplo conceitual do caminho de uma mensagem:

1. **Um produtor** envia uma mensagem para a **Exchange X**, com uma routing key = "email.send".
2. A **Exchange X** é do tipo direct e está configurada para enviar mensagens com routing key = "email.send" para a **fila "fila-emails"`.
3. A **fila "fila-emails"** armazena a mensagem até que um **consumidor** esteja pronto.
4. O **consumidor** (ex: um worker que envia e-mails) lê da fila, processa a mensagem, e a mensagem é então removida da fila.

RabbitMQ – Conceito e Aplicações

🔁 **Diagrama conceitual**

[Producer]

|

v

[Exchange] --(routing key: email.send)--> [Queue: fila-emails] --> [Consumer]

🧠 Kafka – Conceito e Aplicações

🟡 O que é o Kafka?

O **Apache Kafka** é uma **plataforma de streaming distribuída**. Ele é usado para construir **pipelines de dados** e **sistemas de processamento de eventos em tempo real**.

Ao contrário de ferramentas como o RabbitMQ (que são orientadas a filas), o Kafka é **orientado a logs/eventos**. Isso significa que ele **registra todos os eventos de forma sequencial e persistente**, como se fosse um grande "diário" de tudo que acontece no sistema.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🧠 Kafka – Conceito e Aplicações

⚙️ Orientado a logs/eventos (streaming)

Imagine um sistema onde você registra tudo que acontece:

- Um pedido foi feito
- Um produto foi adicionado ao carrinho
- Um usuário fez login

Esses eventos são armazenados **como registros contínuos** (log), e **vários sistemas podem "assinar" esse log e reagir a esses eventos em tempo real**.

🧠 Kafka – Conceito e Aplicações

🚀 Projetado para alto throughput e retenção

- Kafka é feito para **trabalhar com grandes volumes de dados** (milhares ou milhões de eventos por segundo).
- Diferente de outros sistemas de mensageria, **as mensagens não são apagadas depois que um consumidor lê**. Elas podem ser **mantidas por horas, dias ou semanas**, dependendo da configuração.

- Isso permite:
  - **Reprocessar dados** em caso de falhas
  - Fazer **análises históricas**
  - Criar sistemas **reprodutíveis e auditáveis**

🧠 Kafka – Conceito e Aplicações

🧩 Componentes principais

✅ Producer

- Aplica os dados ao Kafka

- Ex: um app envia um evento “usuário logou”

🧠 Kafka – Conceito e Aplicações

🧩 Componentes principais

✅ Broker

- Servidor Kafka que recebe, armazena e entrega mensagens
- Kafka pode ter múltiplos brokers em um cluster

🧠 Kafka – Conceito e Aplicações

🧩 Componentes principais

✅ Consumer

- Lê os dados do Kafka

- Pode ler em tempo real ou reprocessar dados antigos

🧠 Kafka – Conceito e Aplicações

🧩 Componentes principais

✅ Topic

- É como uma "categoria" de mensagens
- Exemplo: logins, pedidos, pagamentos

🧠 Kafka – Conceito e Aplicações

🧩 Componentes principais

✅ Partition

- Cada tópico pode ser dividido em **partições** (shards)
- Isso melhora o desempenho e permite **processamento paralelo**

💡 Exemplo visual:

Imagine o tópico pagamentos com 3 partições. Três consumidores diferentes podem ler esses dados simultaneamente, melhorando o desempenho.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🧠 Kafka – Conceito e Aplicações

kafka

📦 Casos de uso

🔄 Processamento de eventos em tempo real

- Um sistema de **fraude bancária** pode consumir eventos de transações e reagir em segundos

- Um sistema de **notificações** pode enviar alertas baseados em eventos do usuário

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🧠 Kafka – Conceito e Aplicações

📦 Casos de uso

📊 Analytics e Big Data

- Kafka se integra com ferramentas como:
  - **Apache Spark** → Processamento distribuído
  - **Apache Flink** → Processamento de fluxo com baixa latência
  - **Elasticsearch + Kibana** → Para dashboards e buscas
- Ele atua como um **canal centralizador de dados** para esses sistemas

📊 Tabela Comparativa

Característica	RabbitMQ	Kafka
Arquitetura	Fila baseada em broker (padrão de filas)	Baseado em log distribuído (padrão de pub/sub)
Modelo de Mensagem	Filas com roteamento (pode ser complexo)	Logs de eventos (mensagens são armazenadas como eventos)
Garantias de Entrega	Entrega garantida (acknowledgments e redelivery)	Garantia de entrega ao menos uma vez (com configurações)
Escalabilidade	Suporta escalabilidade, mas mais limitada em comparação	Altamente escalável e distribuído de forma nativa
Desempenho	Adequado para volumes menores, pode ser mais lento em larga escala	Alta performance em grandes volumes de dados
Persistência de Dados	Armazenamento de mensagens opcional (persistência configurável)	Persistência de mensagens é central para o design

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

kafka

📊 Tabela Comparativa

Característica	RabbitMQ	Kafka
Consumo de Mensagens	Modelos de consumo síncrono e assíncrono	Consumo de mensagens de múltiplos consumidores (partições)
Protocolos Suportados	AMQP (Advanced Message Queuing Protocol)	Protocolos de rede simples (como TCP)
Latência	Latência moderada (geralmente mais baixa que Kafka)	Latência geralmente mais alta devido à arquitetura de log
Facilidade de Implementação	Fácil de configurar e usar para pequenas/médias aplicações	Requer mais configuração e gerenciamento para alta escalabilidade
Suporte a Garantias de Ordem	Suporta ordem, mas depende do design das filas	Garantia de ordenação dentro de partições, não entre elas
Casos de Uso	Aplicações de mensagens com alta prioridade de entrega	Streaming de dados em grande escala, logs de eventos, análise de grandes volumes de dados

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

kafka

📊 Tabela Comparativa

Característica	RabbitMQ	Kafka
Ferramentas de Monitoramento	Ferramentas como Prometheus, Grafana, ou plugins próprios	Ferramentas de monitoramento integradas com o ecossistema (Kafka Manager, Confluent)
Escalabilidade Horizontal	Requer clustering manual e gerenciamento de filas	Escalabilidade horizontal nativa com partições

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

kafka

📊 Tabela Comparativa: RabbitMQ vs Kafka

Resumo:

- **RabbitMQ** é ideal para sistemas que exigem alta flexibilidade no roteamento de mensagens e garantias de entrega. É uma boa escolha para aplicações que não necessitam de alta escalabilidade ou para casos de uso com menos volume de dados.
- **Kafka**, por outro lado, brilha em cenários com grandes volumes de dados e necessidade de alta escalabilidade e persistência, sendo mais adequado para processamento de eventos e análise de dados em tempo real.

Ambos têm suas aplicações específicas, então a escolha entre um e outro depende das necessidades do seu sistema!

kafka

UniFAP

Experimento

https://colab.research.google.com/drive/1eHX1L7L8DS8BI-85R8Vs3P3alAo2jDYV?usp=sharing

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Atividade

- (Opcional) Criar um pequeno sistema com um produtor e um consumidor usando RabbitMQ (ex: produtor envia nomes, consumidor escreve em um arquivo).

- (Opcional) Explorar um exemplo básico com Kafka se tiverem interesse em Big Data.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Leitura Sugerida

System Design - Mensageria, Eventos, Streaming e Arquitetura Assíncrona - Matheus Fidelis**:** https://fidelissauro.dev/mensageria-eventos-streaming/



Bibliografia

LISBOA, Flávio. Arquitetura de software distribuído: boas práticas para um mundo de microsserviços. São Paulo, SP: Casa do Código, 2021. E-book. Disponível em: https://plataforma.bvirtual.com.br/Acervo/Publicacao/212795. Acesso em: 02 maio 2025.

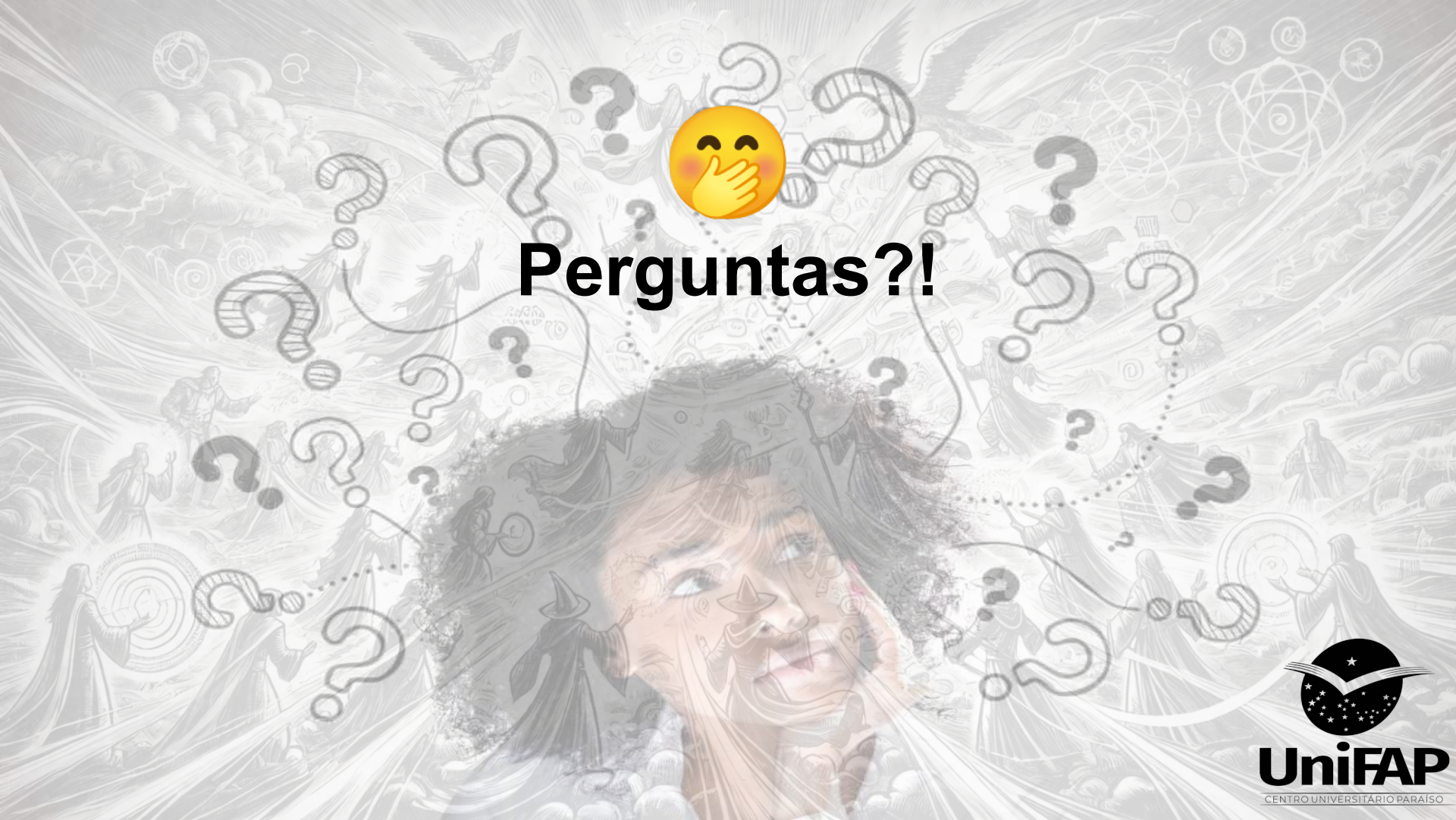


Referências

CHATGPT. **Fundamentos de Inteligência Artificial**. OpenAI, 2025. Disponível em: https://chat.openai.com. Acesso em: *20/03/2025*.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Perguntas?!



UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Eclesiastes 5:3 – "Da muita ocupação vêm os sonhos."

Sonhe alto, mas lembre-se de que o esforço constante é o que os realiza.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO