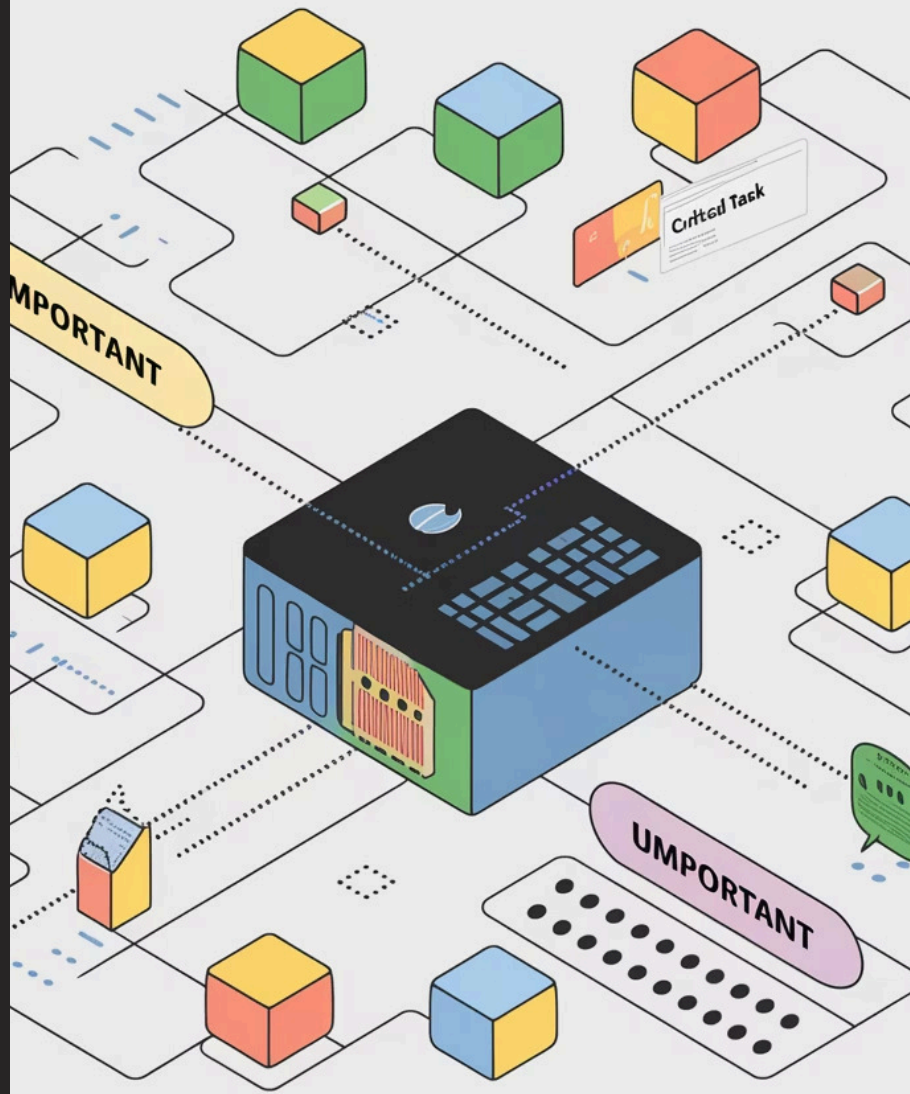
FullCycle

# Curso de RabbitMQ

Sistema de mensageria para comunicação entre aplicações



# Conteúdo do Curso

### Fundamentos do RabbitMQ

Arquitetura básica, exchanges, filas e bindings. Configuração de filas de trabalho e padrões publish/subscribe.

### Entrega e Persistência

Confirmações, persistência de mensagens e Dead Letter Exchanges. Garantia de resiliência nos sistemas.

### Plugins Essenciais

Shovel, Federation e Management. Monitoramento e transferência entre brokers.

### Streaming de Dados

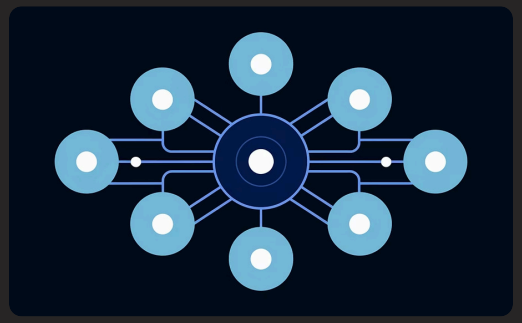
Técnicas para transmissão contínua de mensagens. Otimização de latência e throughput.

### Boas Práticas e Aplicações Avançadas

Padrões para idempotência, escalabilidade e alta disponibilidade. Integração com microsserviços.

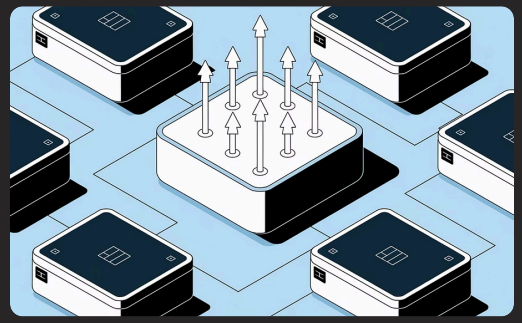
# O que é um Message Broker?

Um message broker é um intermediário que facilita a comunicação entre aplicações, permitindo troca de informações de forma desacoplada e resiliente.



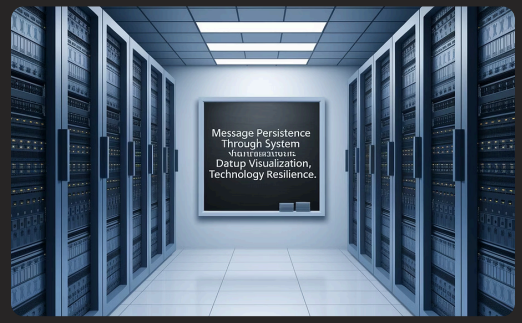
## Desacoplamento

Serviços se comunicam sem conhecer detalhes uns dos outros.



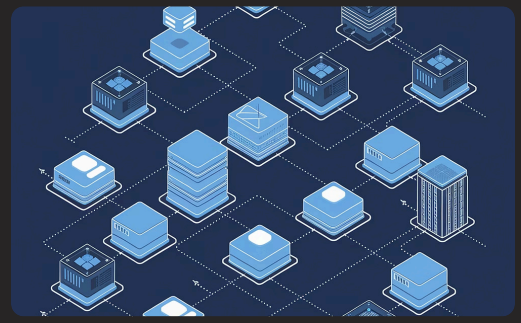
## Escalabilidade

Gerencia filas quando destinatários estão ocupados ou offline.




## Resiliência

Garante que mensagens não sejam perdidas durante falhas.



## Flexibilidade

Suporta diversos padrões de comunicação entre sistemas.

# História e Evolução do RabbitMQ

### 2007: Origens

Fundado pela Rabbit Technologies Ltd. no Reino Unido para implementar o protocolo AMQP em código aberto.

### 2010: Aquisição pela SpringSource

Posteriormente absorvido pela VMware, expandindo seu alcance e recursos técnicos.

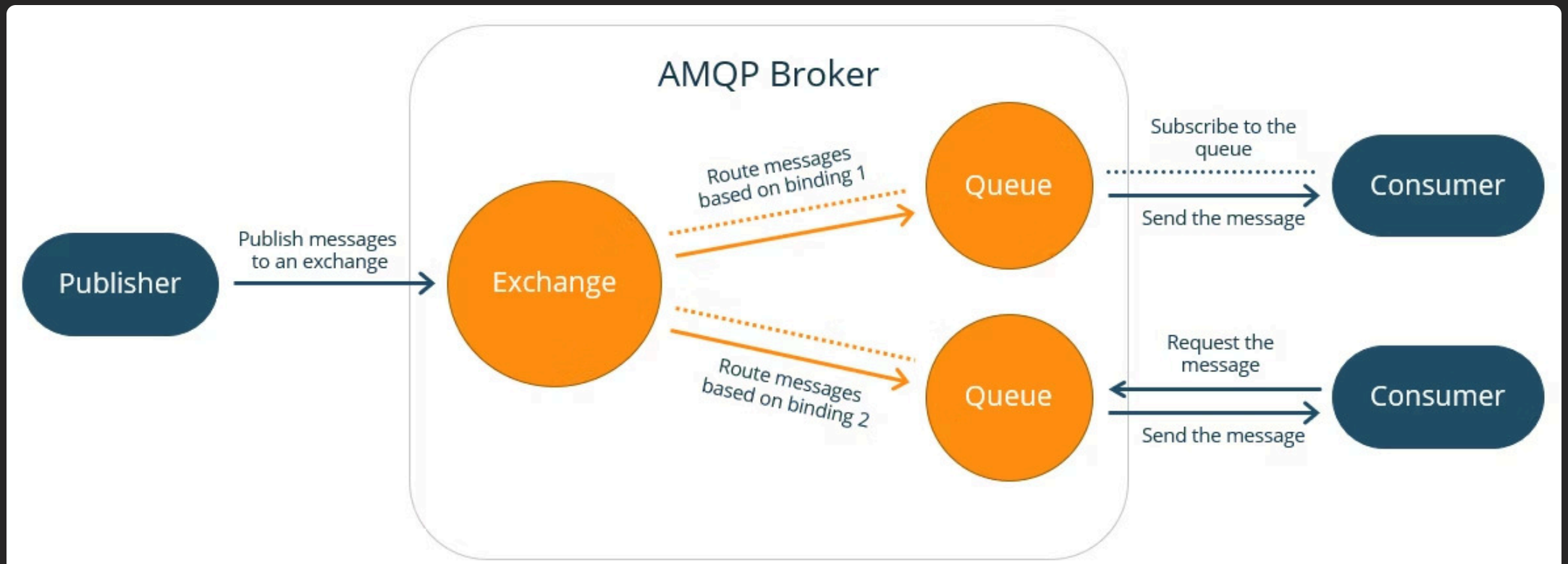
### 2013: Transição para Pivotal

Continuou seu desenvolvimento sob nova administração, mantendo a filosofia open-source.

### Atualmente

Um dos message brokers mais populares globalmente, utilizado desde startups até Fortune 500.

# O Protocolo AMQP




### Orientado a Mensagens

Projetado especificamente para troca de mensagens entre aplicações distribuídas.

### Roteamento Flexível

Define modelo publish/subscribe com regras de roteamento avançadas.

### Confiabilidade

Garantia de entrega através de confirmações e transações.

### Segurança

Autenticação, autorização e criptografia TLS integradas.

RabbitMQ implementa o AMQP 0-9-1 (Advanced Message Queuing Protocol), um padrão aberto para brokers de mensagens. Embora originalmente focado no AMQP, o RabbitMQ hoje suporta múltiplos protocolos como MQTT, STOMP e HTTP.

# RabbitMQ vs. Outras Soluções de Mensageria

Comparação das principais tecnologias de mensageria utilizadas no mercado


### RabbitMQ

Broker tradicional para roteamento flexível de mensagens

- Excelente em padrões complexos de integração
- Implementação AMQP com suporte a múltiplos protocolos
- Garantias robustas de entrega de mensagens


### Apache Kafka

Streaming de logs distribuído para alto throughput

- Superior para análise em tempo real
- Escalabilidade horizontal para volumes massivos
- Ideal para processamento de eventos em sequência

### Nats

Sistema de mensageria leve baseado em protocolo próprio

- Foco extremo em performance e baixa latência
- Menos recursos comparado do RabbitMQ.
- Versão Core para pub/sub. JetStream para persistência.


### Redis

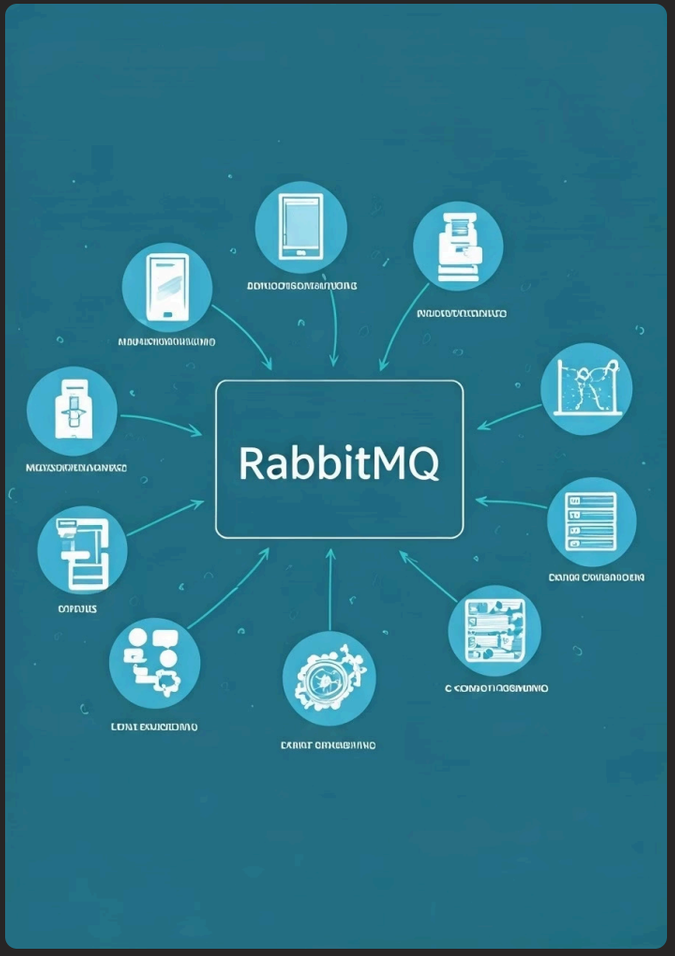
Banco em memória com recursos pub/sub

- Modelo mais simples, porém limitado
- Alto desempenho para casos de uso específicos
- RabbitMQ oferece persistência nativa superior

# Casos de Uso Comuns para RabbitMQ

## Aplicações Práticas

- Processamento assíncrono de tarefas pesadas e em background
- Comunicação entre microsserviços e sistemas legados
- Balanceamento de carga entre servidores distribuídos
- Coleta e processamento de dados IoT e telemetria
- Streaming de eventos e centralização de logs
- Arquiteturas de e-commerce para pedidos e estoque




### Flexibilidade

Suporte a múltiplos protocolos e padrões de mensageria

### Confiabilidade

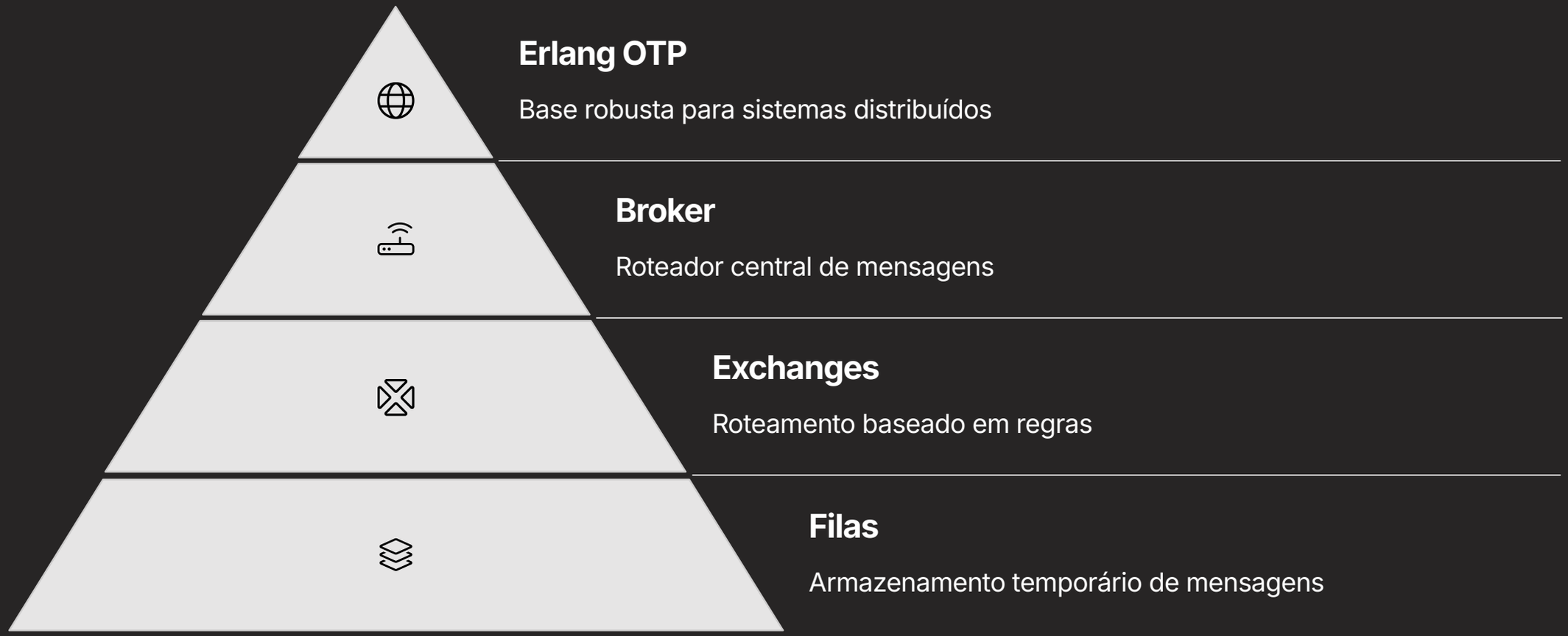
Confirmações, persistência e clustering para alta disponibilidade

### Administração

Interface web robusta e ecossistema com mais de 100 plugins

# Arquitetura Geral do Broker

O RabbitMQ funciona como um intermediário sofisticado entre produtores e consumidores de mensagens. Sua arquitetura baseia-se na plataforma Erlang OTP, especializada em sistemas distribuídos de alta disponibilidade.




Esta arquitetura permite total desacoplamento entre produtores e consumidores. Os componentes se comunicam de forma assíncrona sem dependências diretas.

Componentes Principais

# Virtual Hosts (vhosts)

Virtual hosts funcionam como instâncias lógicas isoladas dentro do RabbitMQ. Eles permitem separar completamente recursos entre diferentes aplicações no mesmo servidor.

### Isolamento Completo

Cada vhost mantém suas próprias filas, exchanges e bindings sem interferência entre aplicações.

### Permissões Granulares

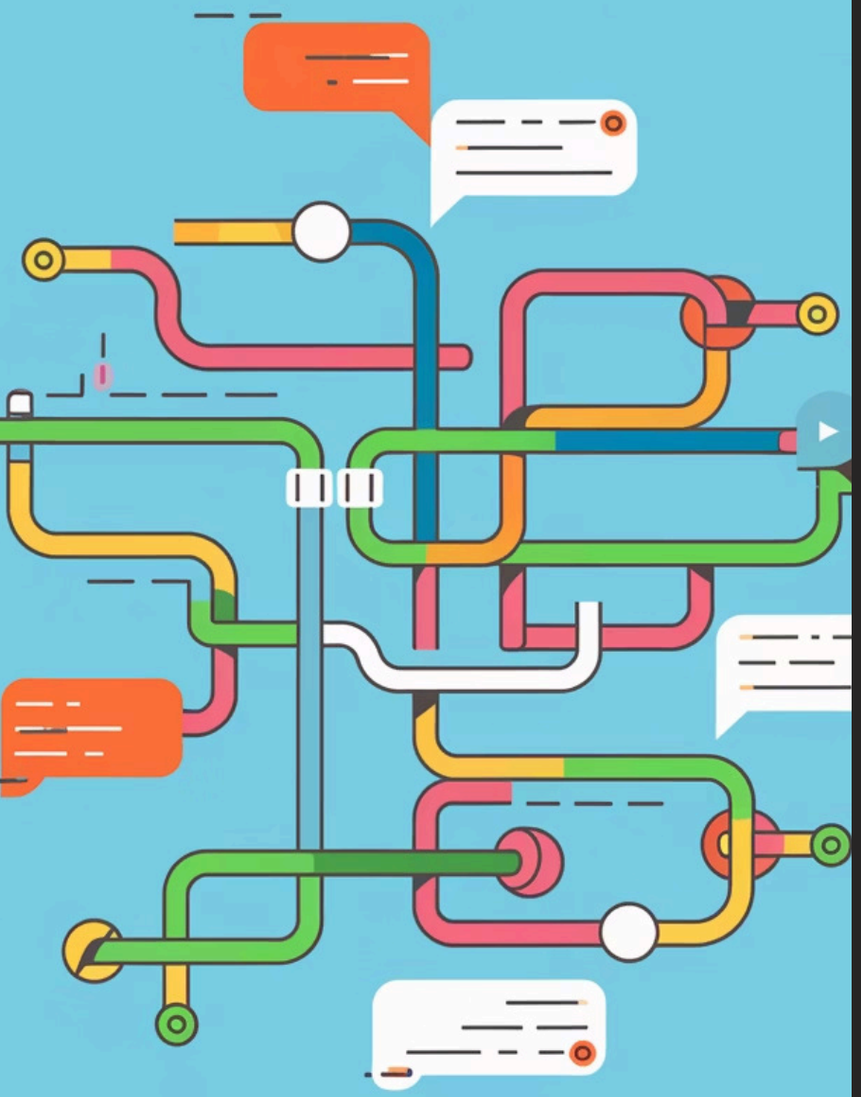
Usuários podem ter diferentes níveis de acesso em cada vhost, controlando segurança por aplicação.

### Configuração Padrão

O RabbitMQ vem com um vhost padrão denominado "/" pronto para uso.

### Conceito Análogo

Semelhantes a namespaces ou bancos de dados isolados em sistemas de banco de dados relacionais.



Componentes Principais

# Exchanges

Exchanges são os pontos de entrada para todas as mensagens no RabbitMQ. Eles determinam para quais filas as mensagens serão encaminhadas.

### Direct

Roteia mensagens baseado em correspondência exata da routing key.

### Fanout

Distribui mensagens para todas as filas vinculadas, ignorando routing keys.

### Topic

Roteia mensagens baseado em padrões de correspondência parcial da routing key.

### Headers

Utiliza atributos de cabeçalho para roteamento em vez da routing key.

**Componentes principais**

# Filas (Queues)

As filas são os armazenadores temporários de mensagens no RabbitMQ, seguindo o princípio FIFO (primeiro a entrar, primeiro a sair).

### Durabilidade

- Filas duráveis sobrevivem a reinicializações do broker
- Filas temporárias são perdidas quando o servidor reinicia

### Argumentos

- TTL (time-to-live) para mensagens
- Limites de comprimento para controle de fluxo
- Políticas de overflow para rejeitar ou descartar mensagens

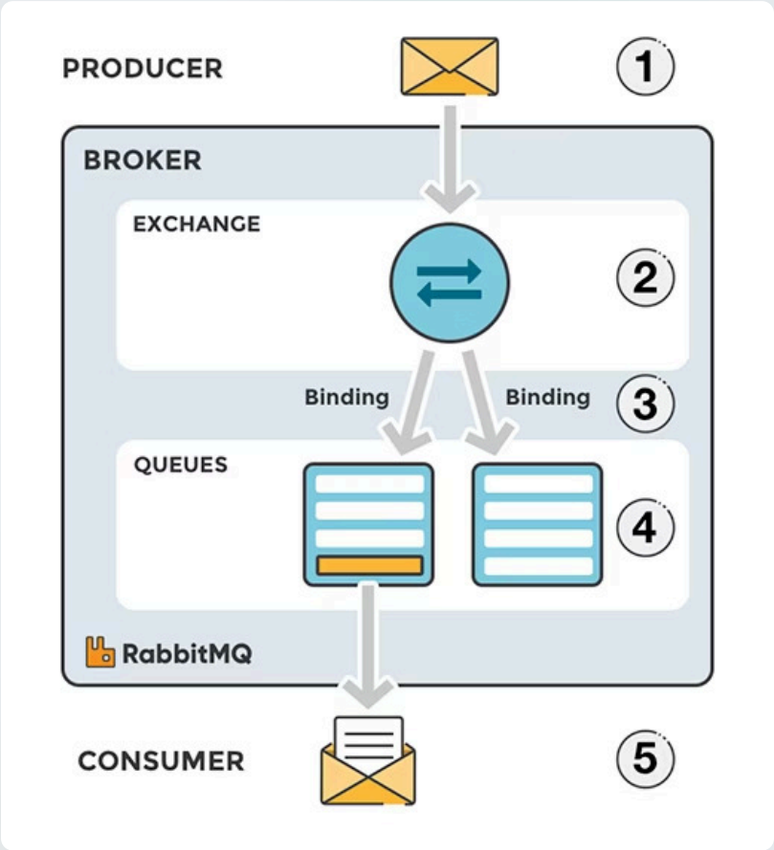
### Exclusividade

- Usadas por apenas uma conexão
- Automaticamente excluídas quando a conexão fecha
- Úteis para comunicação temporária cliente-servidor

### Auto-delete

- Removidas quando o último consumidor se desconecta
- Ideais para padrões de publicação/assinatura temporários

A configuração adequada das filas é essencial para o desempenho e a confiabilidade do sistema de mensageria.




Componentes Principais

# Bindings

Bindings são as regras que conectam exchanges às filas, definindo o fluxo de mensagens no RabbitMQ.

## Conexão Lógica

Estabelecem relações entre exchanges e filas, declarando o interesse da fila em mensagens específicas.

## Routing Keys

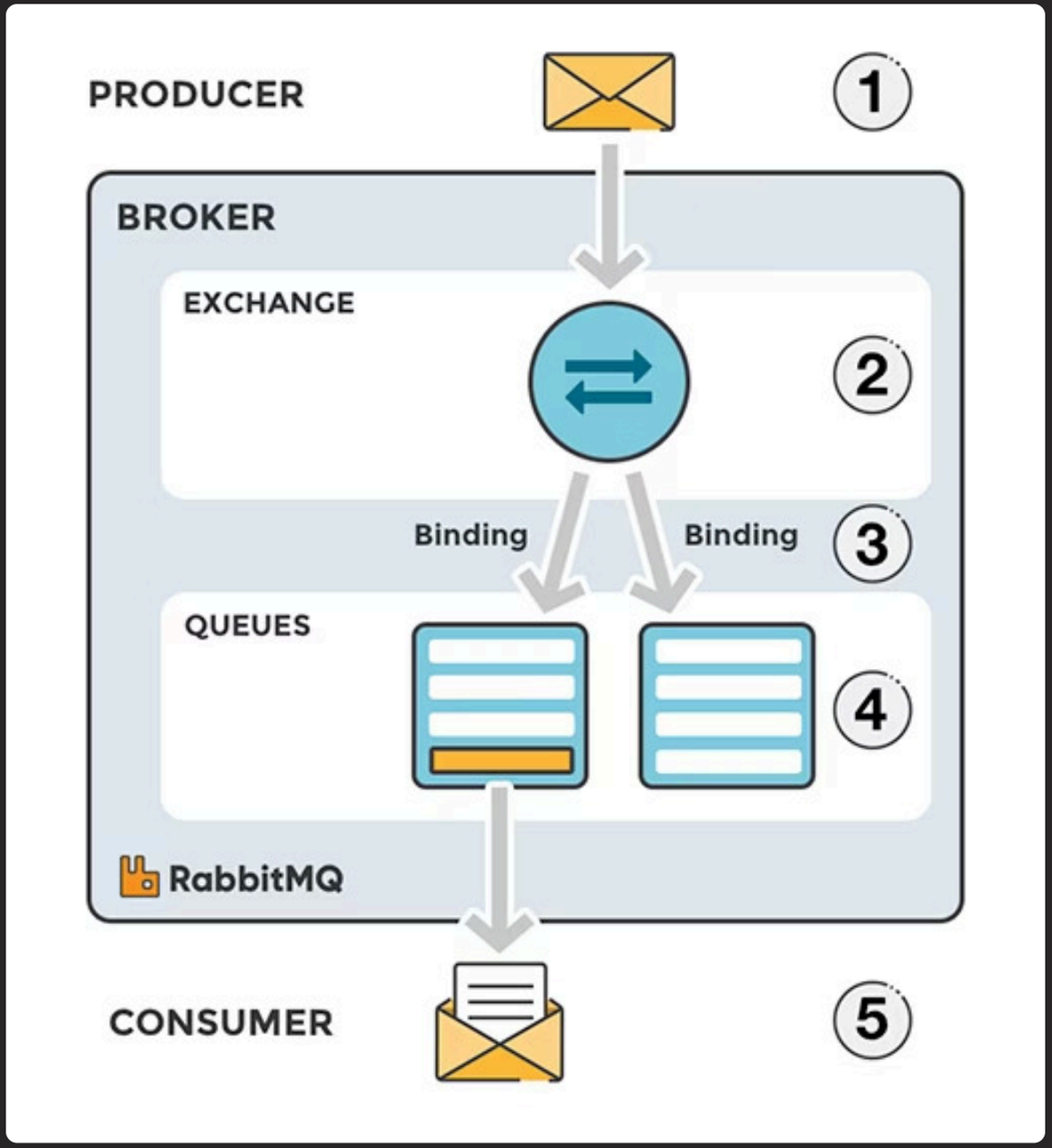
Utilizadas em exchanges direct e topic para filtrar mensagens com base em padrões de correspondência.

## Filtragem

Determinam quais mensagens devem ser encaminhadas para cada fila associada.

## Argumentos

Podem incluir parâmetros adicionais para personalizar o comportamento do roteamento.




# Fluxo de mensagens

## 1 Publicação de Mensagens

O produtor publica uma mensagem em um exchange, incluindo uma routing key.

## 2 Roteamento de Mensagens

O exchange avalia a routing key e envia a mensagem para as filas correspondentes, de acordo com os bindings configurados.

## 3 Armazenamento de Mensagens

As mensagens são armazenadas nas filas até serem consumidas ou expirarem.

## 4 Consumo de Mensagens

Os consumidores se conectam às filas e recebem as mensagens quando disponíveis.

# Modelo de Comunicação AMQP

O RabbitMQ implementa o AMQP (Advanced Message Queuing Protocol), definindo um modelo padronizado para troca de mensagens.

### Canais

Conexões virtuais TCP que abrigam múltiplos canais virtuais para operações concorrentes.

### Mensagens

Compostas por cabeçalhos (propriedades) e corpo (payload) com metadados.

### Exchanges e Filas

Pontos de entrada e armazenamento de mensagens no sistema.

### Bindings

Regras que determinam o fluxo das mensagens entre exchanges e filas.

## Propriedades de Mensagens

### Controle de Entrega

- delivery_mode (persistência)
- expiration (TTL)

### Conteúdo

- content_type
- headers personalizados

### Padrões RPC

- reply_to (fila de resposta)
- correlation_id (rastreamento)