

#45



Mensageria com Delphi e RabbitMQ

Luiz Sfolia

Da sala de reunião à vida real

Embarcadero Conference 2019

Apresentação

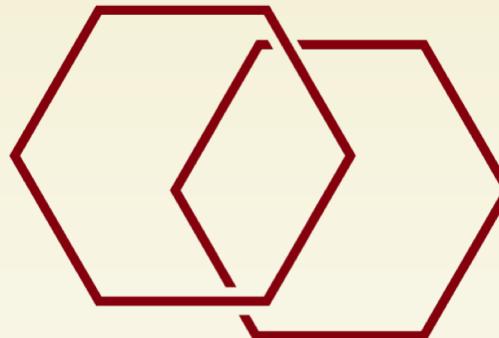
Luiz Sfolia



luizsfolia



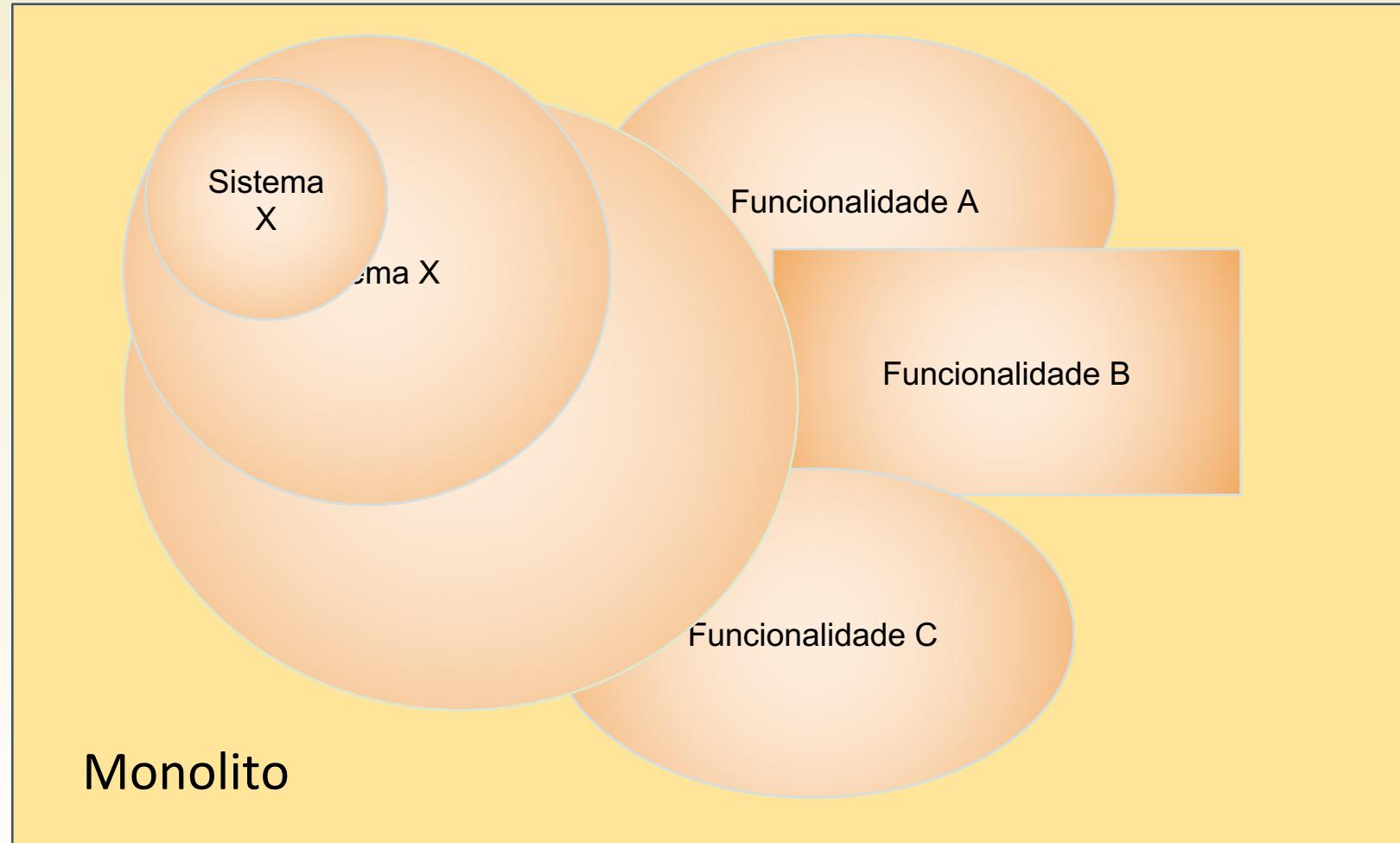
luiz.sfolia@tmrti.com.br



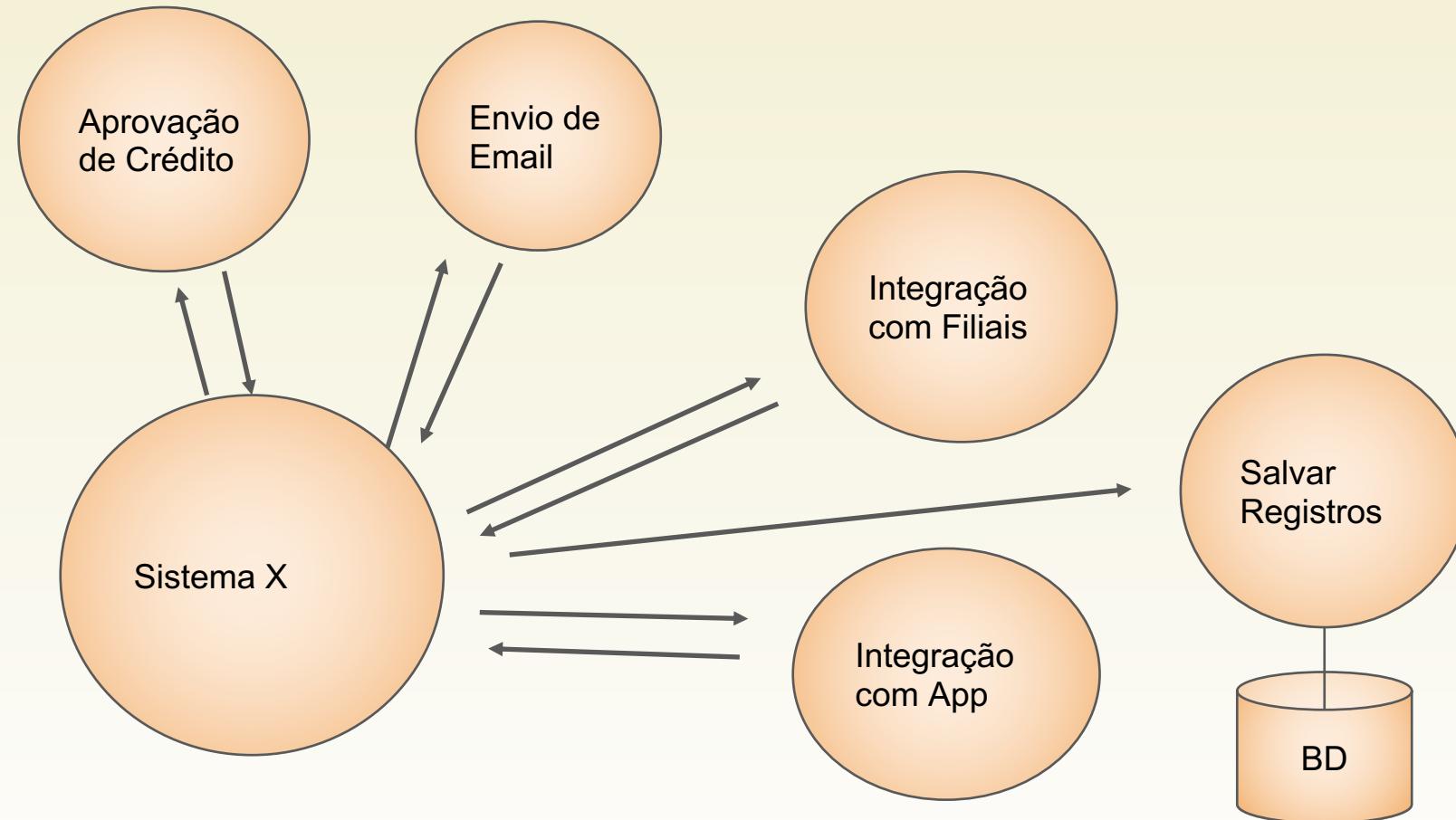
T M R

Consultoria e Desenvolvimento

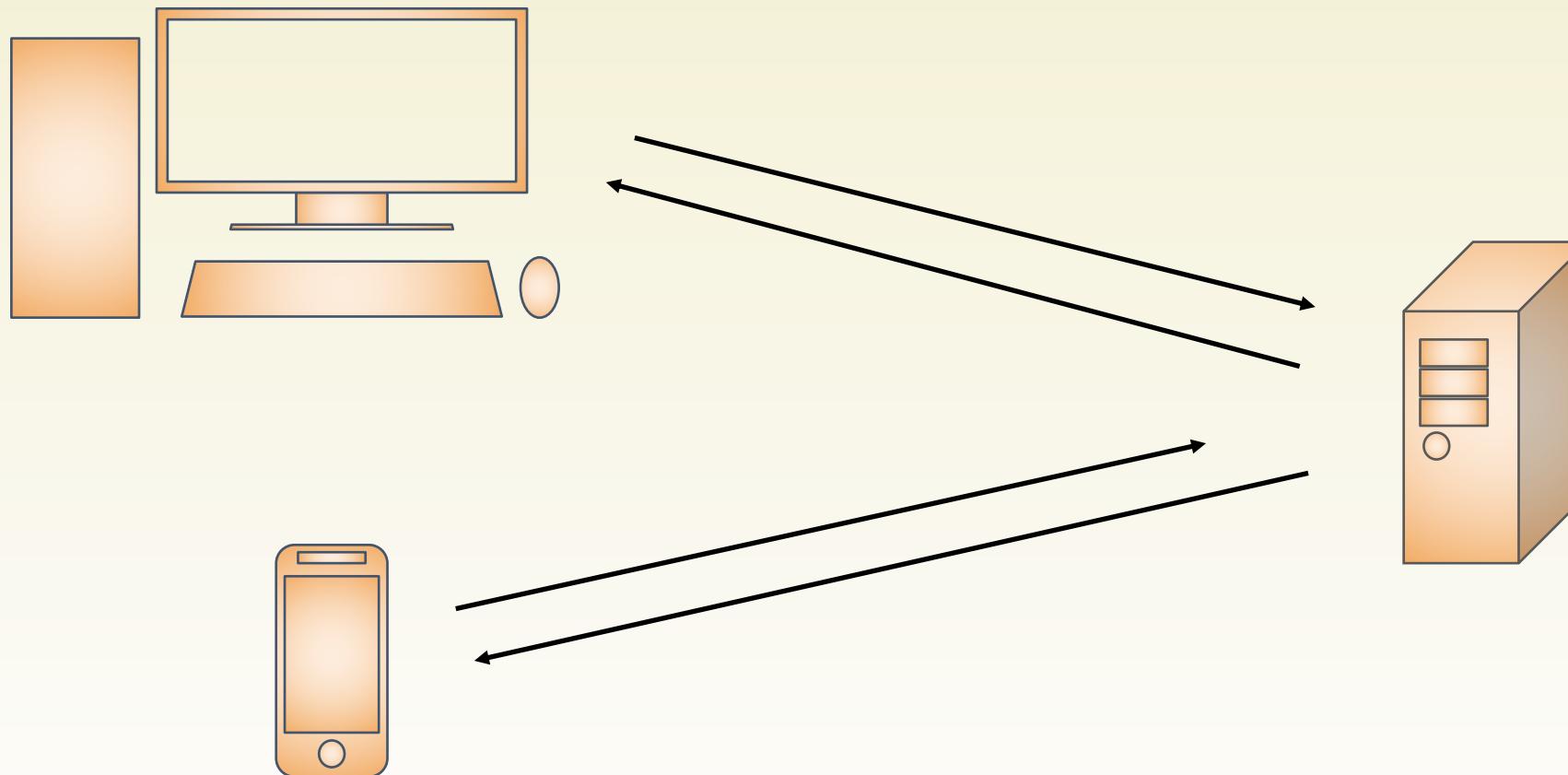
Início de um software



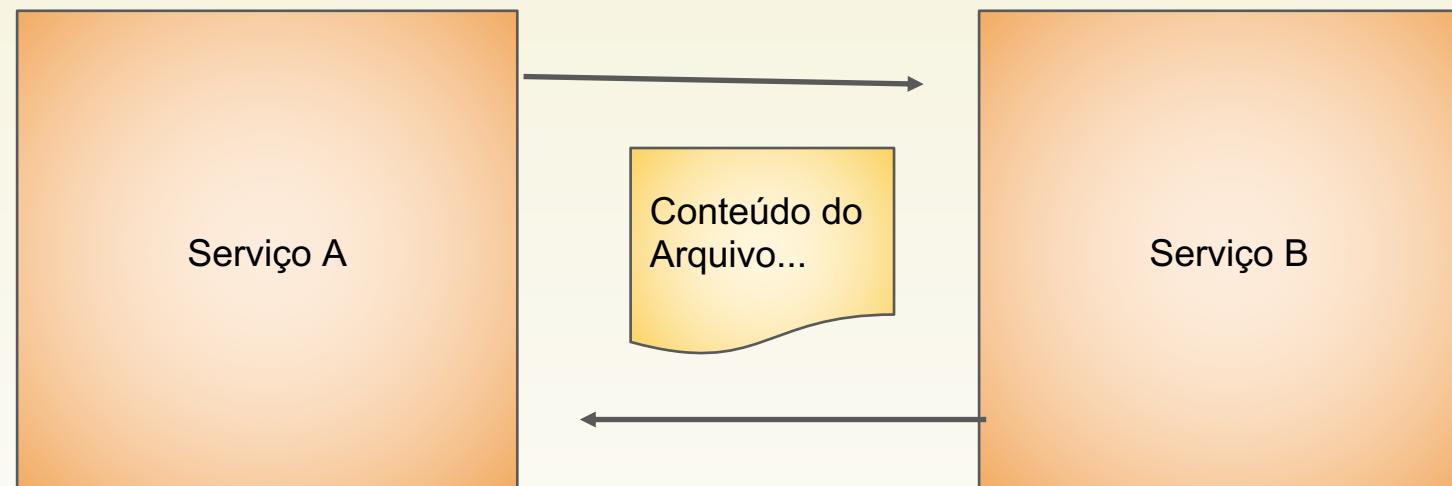
Microsservicos



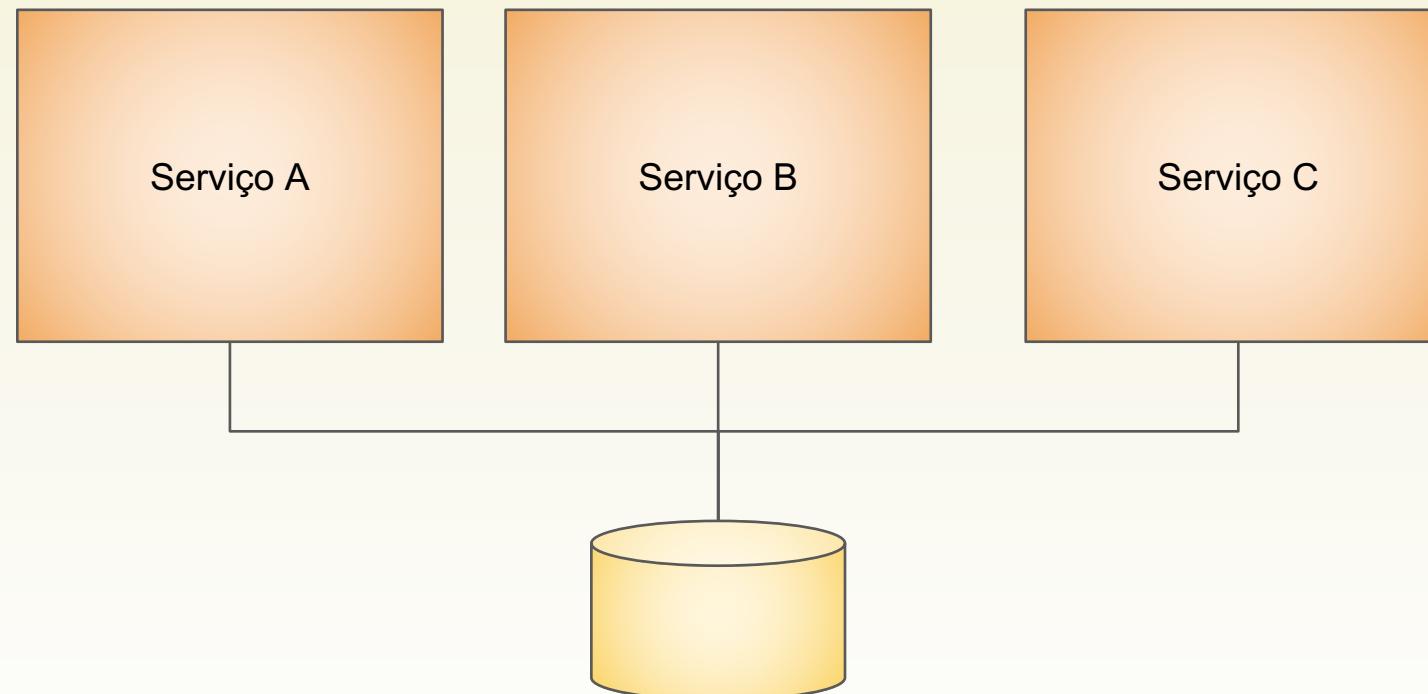
RPC - Remote Procedure Call



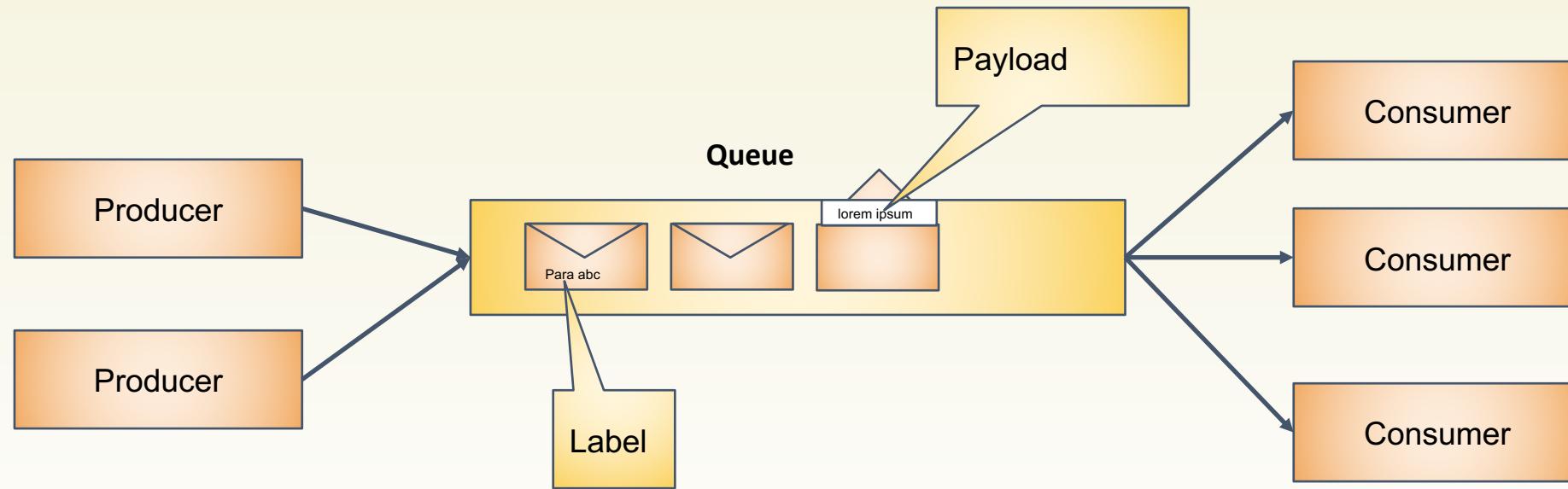
Transferência de Arquivos



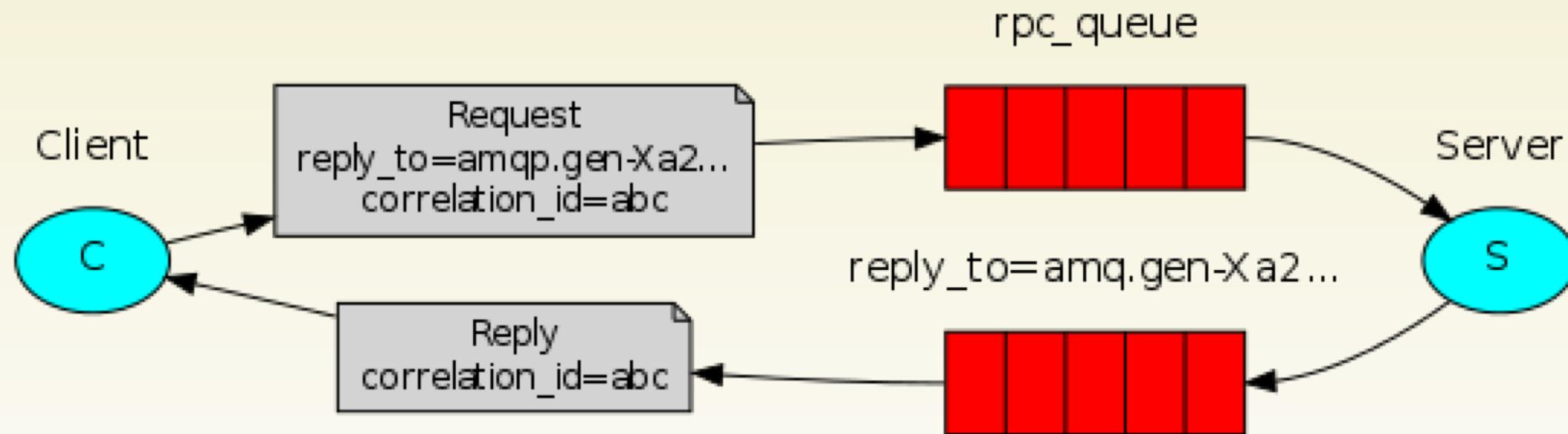
Banco de Dados



Message Broker - Mensageria



RPC - Remote Procedure Call



Message Broker

Producer/Publisher - Quem publica mensagem

Consumer - Quem consome mensagem

- Resiliência
- Escalabilidade
- Confiabilidade

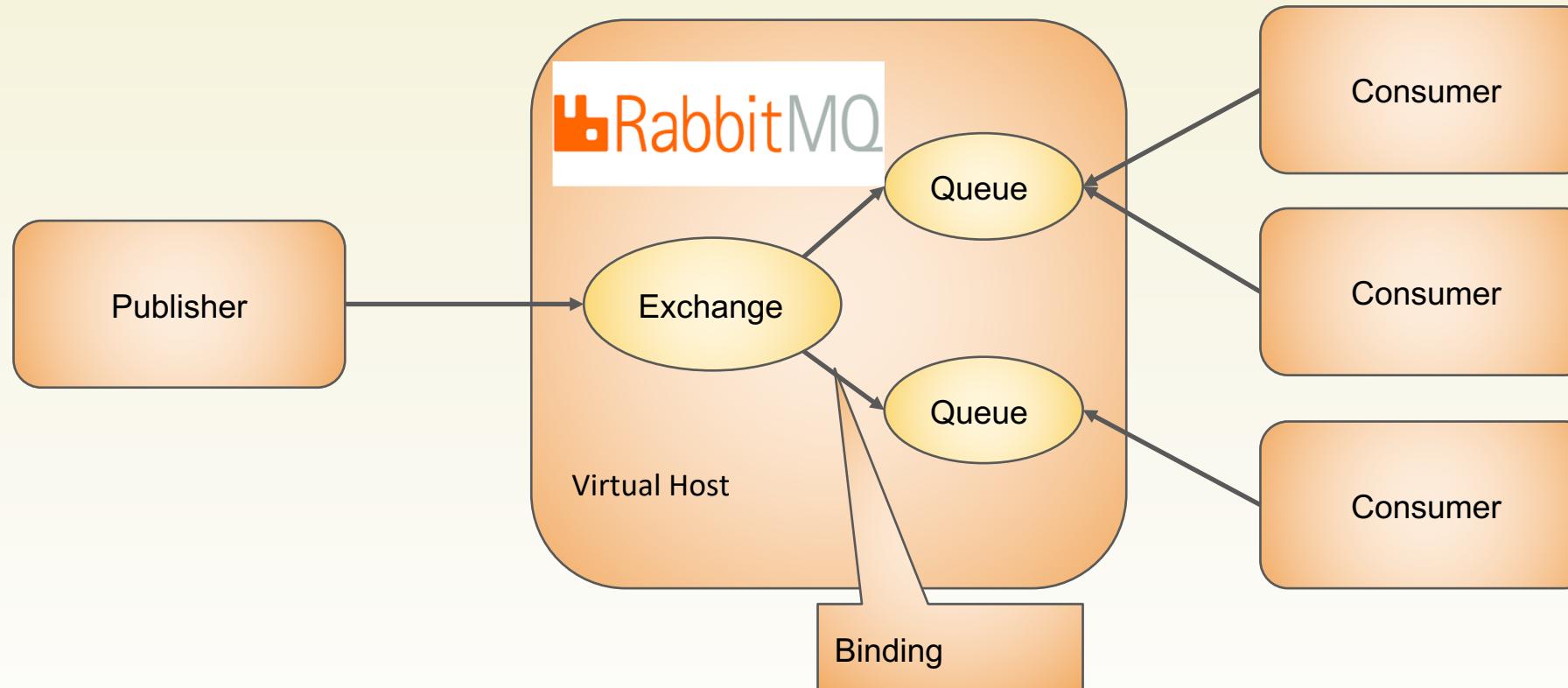
Assíncrono

Message Broker



Rabbit Message Queue

Round Robin



Exchange

Default

= Fila

Direct

= Encaminhamento para a fila de mesmo nome

Fanout
associadas

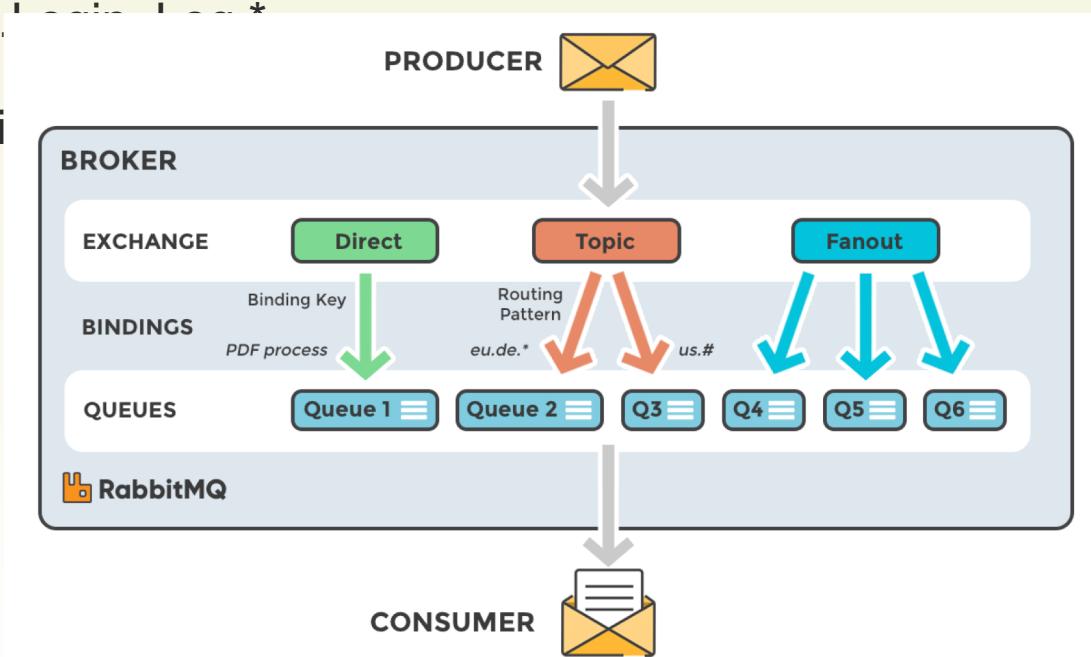
= Encaminha a mensagem para todas as filas

Topic

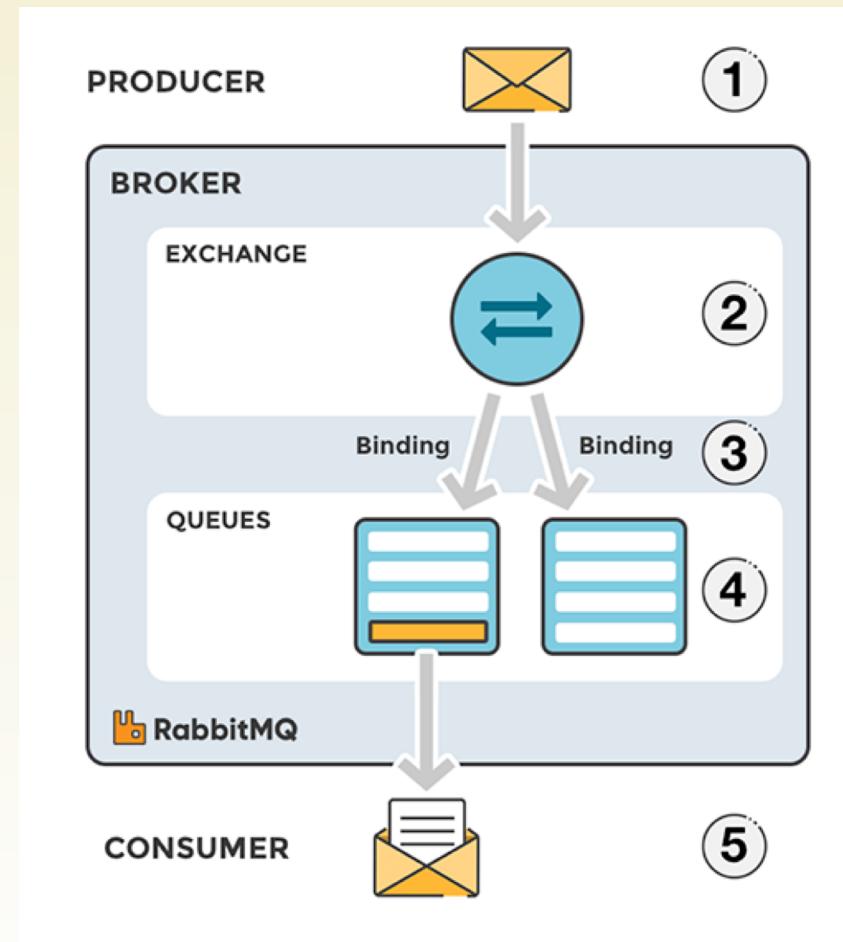
= Regras : Ex : Log.Error, Log.Info

Headers

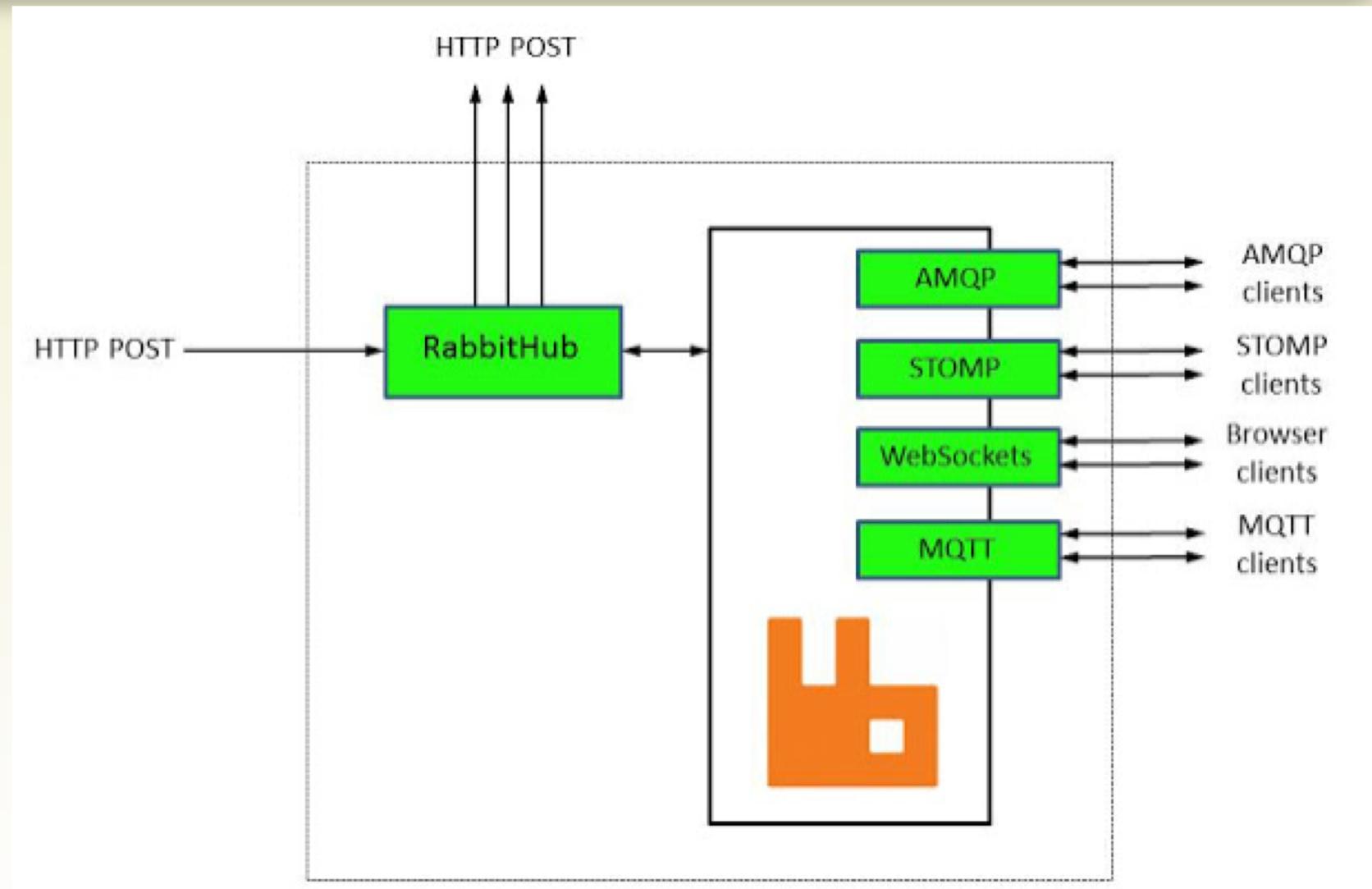
= Usa cabeçalhos para definir a fila



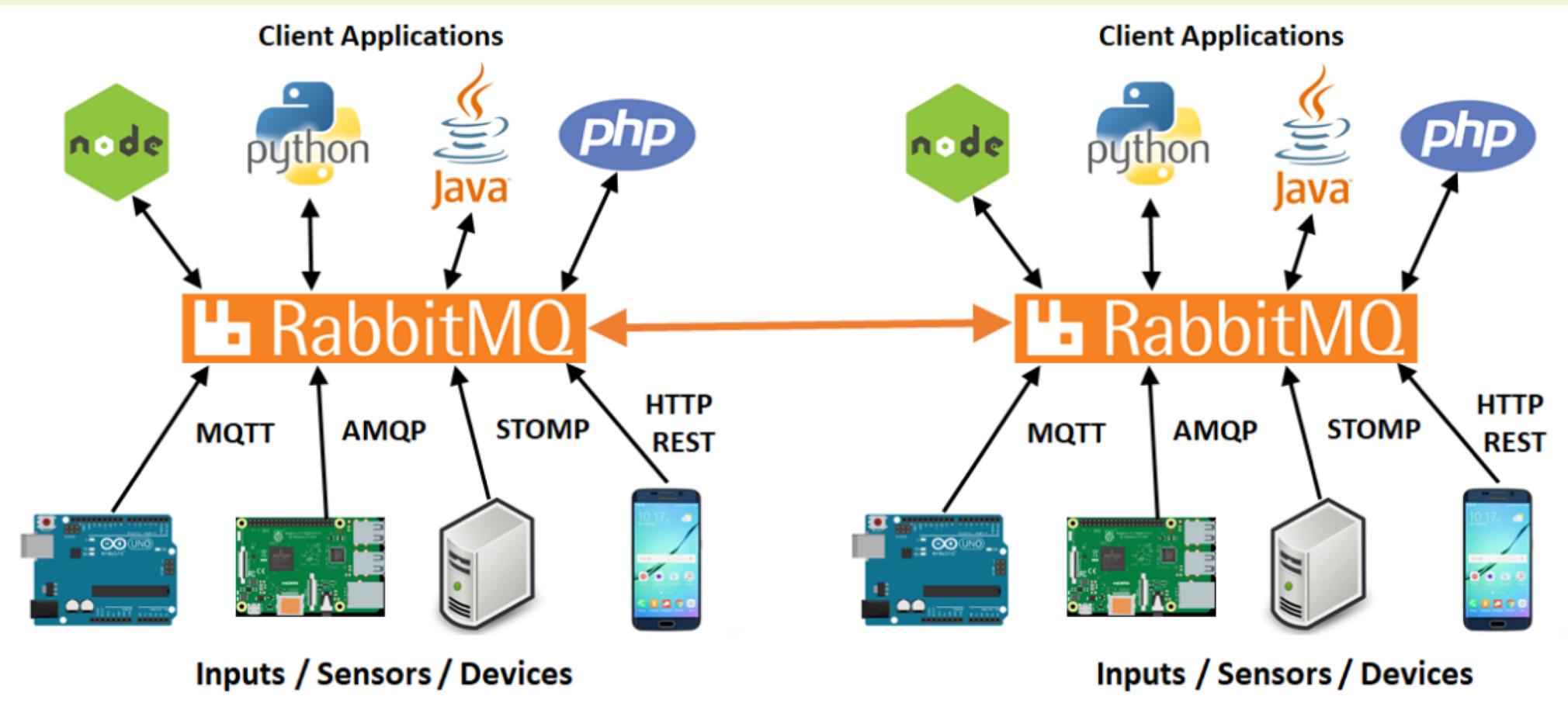
RabbitMQ - Funcionamento



Protocolos



Conectividade



Utilizando Com Delphi

https://www.habarisoft.com/habari_rabbitmq.html

<https://github.com/danieleteti/delphistompclient>

STOMP (Simple Text Oriented Messaging Protocol)

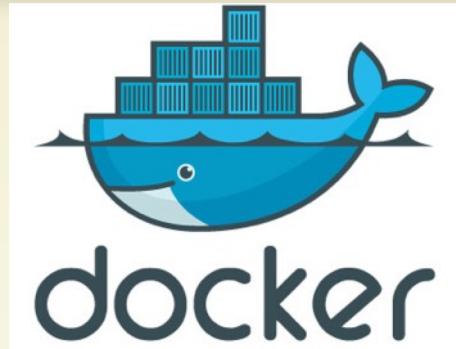


STOMP Simple Text
Orientated Messaging
Protocol [7].

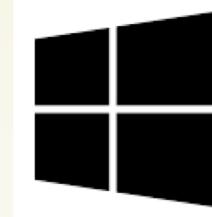
STOMP clients communicate
with any STOMP message
broker (eg. RabbitMQ).

Instalação

1



```
docker run -d --hostname my-rabbit --name rabbit13 -p 8080:15672 -p 5672:5672 -p 25676:25676 rabbitmq:3-management
```



- 2 Instalar plugin para Stomp
- 3 Acessar <http://localhost:8080>

Rabbit

localhost:8080/#/

User: guest Cluster: rabbit@776e7e3b387d (change) Log out
RabbitMQ 3.6.6, Erlang 19.1

Overview Connections Channels Exchanges Queues Admin

Overview

Totals

Queued messages (chart: last minute) (?)

Category	Value
Ready	4,200
Unacked	0
Total	4,200

Message rates (chart: last minute) (?)

Currently idle

Global counts (?)

Connections: 0 Channels: 0 Exchanges: 8 Queues: 55 Consumers: 0

Node

Node: rabbit@776e7e3b387d ([More about this node](#))

File descriptors	Socket descriptors	Erlang processes	Memory	Disk space	Rates mode	Info
89 1048576 available	0 943626 available	358 1048576 available	73MB 792MB high watermark	51GB 48MB low watermark	basic	Disc 2 Stats

Paths

Config file	/etc/rabbitmq/rabbitmq.config
Database directory	/var/lib/rabbitmq/mnesia/rabbit@776e7e3b387d
Log file	tty
SASL log file	tty

▶ Ports and contexts

▶ Import / export definitions

HTTP API | Command Line Update every 5 seconds ▾

Rabbit Message Queue

Propriedades da Fila

- Durable - A fila retorna em caso de queda
- Exclusive - Apenas um consumidor e a fila é excluída quando o mesmo desconecta
- Auto-delete - a fila é excluída quando o último consumidor desconecta
- Arguments
 - a. TTL - Tempo de vida da mensagem
 - b. Tamanho máximo de mensagem
- Prioridade - Define máximo de prioridade aceita

Propriedades da Mensagem

- DeliveryMode -
 - a. Non-persistent - Somente na memória
 - b. Persistent - Mensagem é armazenada até ser consumida

Propriedades do Consumidor

- Ack - Reconhecimento da mensagem
- Nack - Não reconhecimento da mensagem
- Prefetch - Quantidade de mensagens para cada inscrito na fila

Exemplo

Publicando na Fila

```
if FStompClientTerminal = nil then
begin
  FStompClientTerminal := StompUtils.StompClient;
  FStompClientTerminal.SetHost(edtServerRabbitMq.Text);
end;

FStompClientTerminal.Connect;
try
  FStompClientTerminal.Send('/queue/geradorsenha', TSenha.GerarSolicitacaoSenha(FTerminal, APrioritario));
finally
  FStompClientTerminal.Disconnect;
end;
...
```

Exemplo

Consumindo da Fila

```
FStompClient := StompUtils.StompClient;
FStompClient.SetHost(AEndereco);
FStompClient.Connect;

FStompClient.Subscribe('/queue/Terminal_' + ATerminal, amAuto);

while not Terminated do
begin
  if FStompClient.Receive(FStompFrame, 2000) then
  begin
    Sleep(100);
    Synchronize(ReceberSenha);
  end;
end;
```

ACK e NACK

ACK (acknowledgement) - Reconhecimento de Mensagem

- **Auto** - Sem confirmação de recebimento
- **Client** - Confirmação de todas as mensagens
- **Client-Individual** - Confirmação individual

NACK - Mensagem não reconhecida

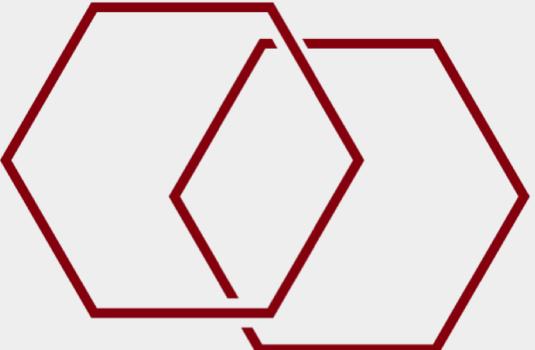
```
while not Terminated do
begin
  if FStompClient.Receive(FStompFrame, 2000) then
    begin
      FStompClient.Ack(FStompFrame.MessageID);
      Synchronize(ReceberSenha);
    end;
end;
```

Prefetch - Limite de quantidades de mensagens distribuídas para cada inscrito

```
FStompFrameSolicitacao := StompUtils.NewFrame();

lHeaders := FStompFrameSolicitacao.Headers
  .Add('x-max-priority', '10')
  .Add(StompHeaders.PREFETCH_COUNT, '1');

FStompClientSolicitacao.Subscribe('/queue/Solicitacao', amClient, lHeaders);
```



T M R

Consultoria e Desenvolvimento

Da sala de reunião à vida real

embarcadero®

Obrigado



luiz.sfolia@tmrti.com.br



luizsfolia

#45

Embarcadero Conference 2019