#45

embarcadero

Mensageria com Delphi e RabbitMQ

Luiz Sfolia

Da sala de reunião à vida real

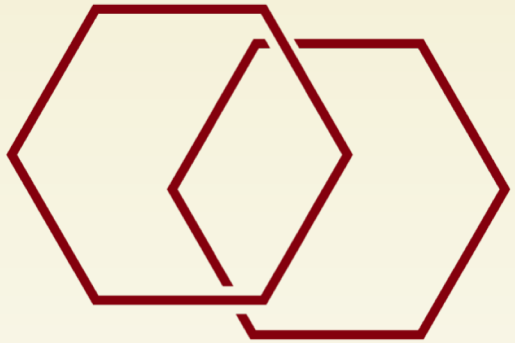
Embarcadero Conference 2019

Apresentação

Luiz Sfolia

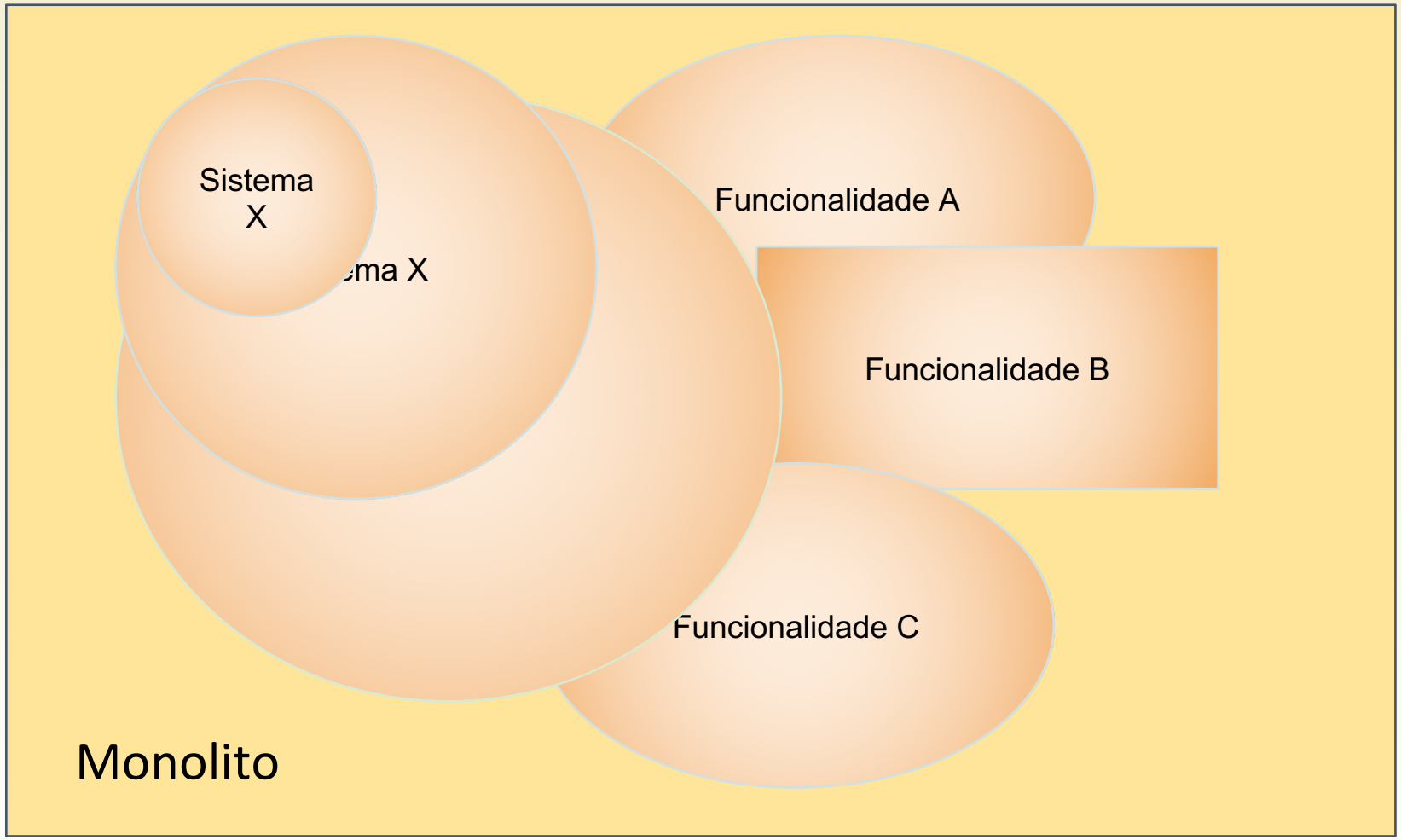
luizsfolia

luiz.sfolia@tmrti.com.br

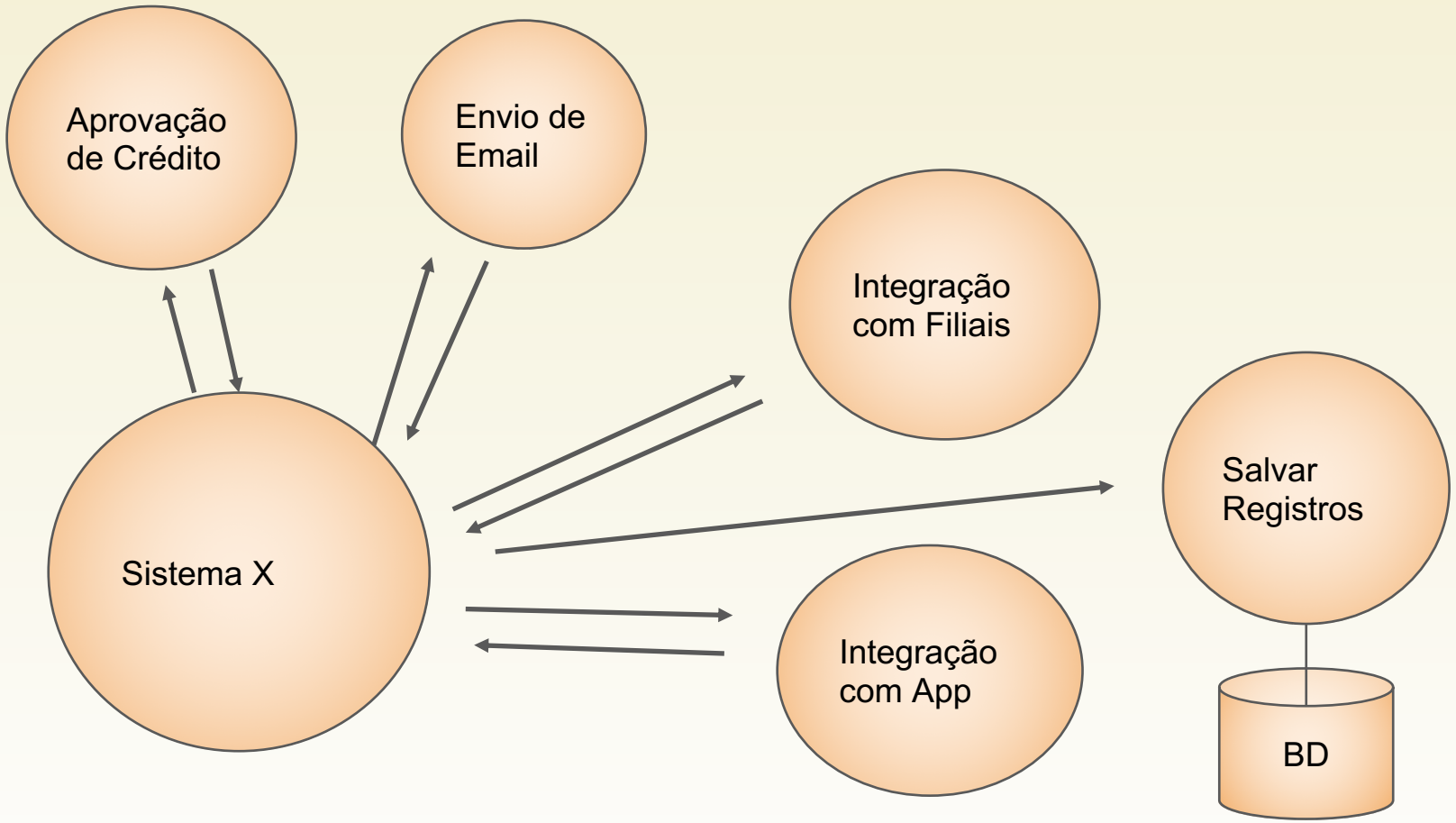


embarcadero

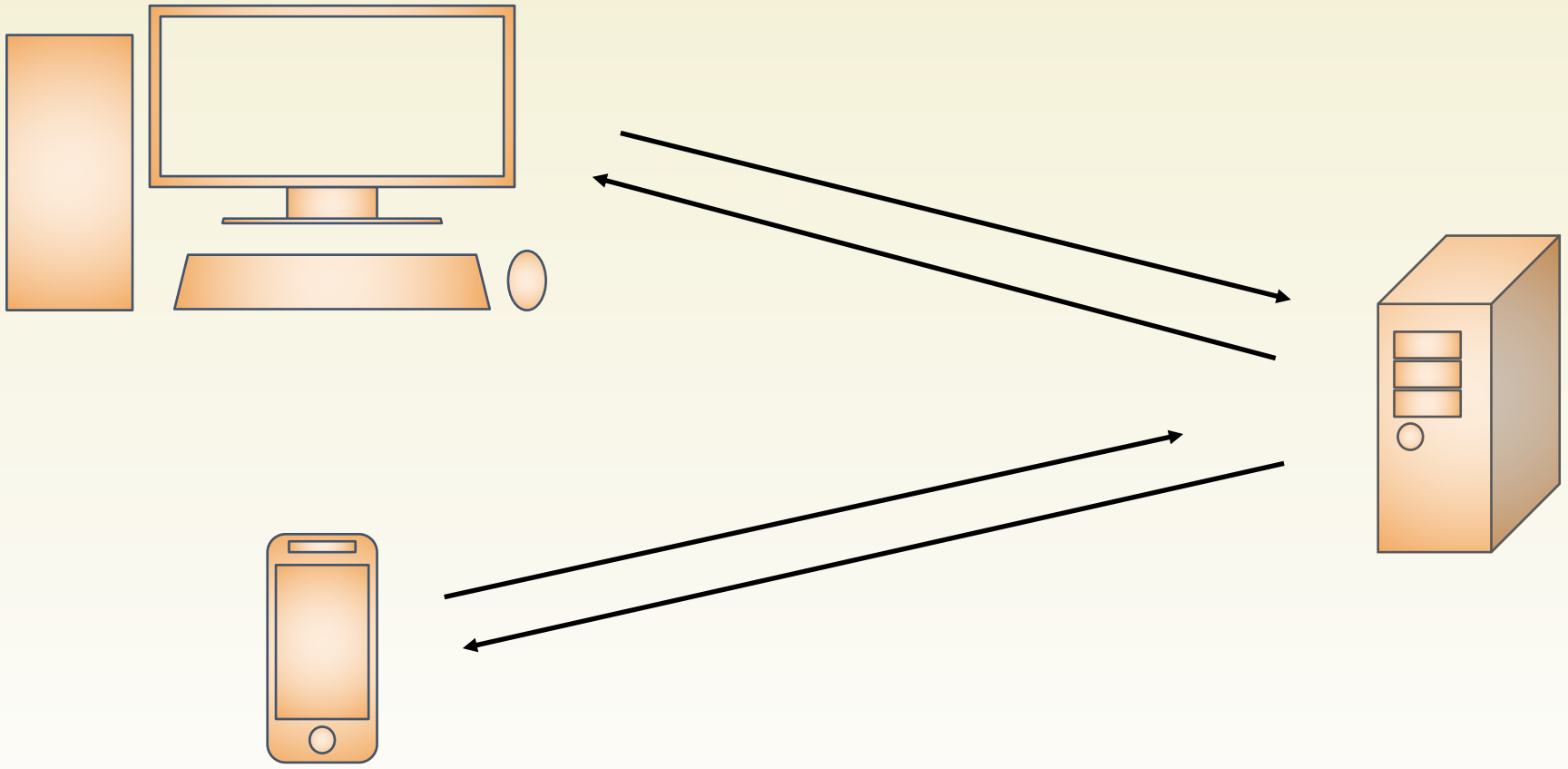
Início de um software



Microsservicos

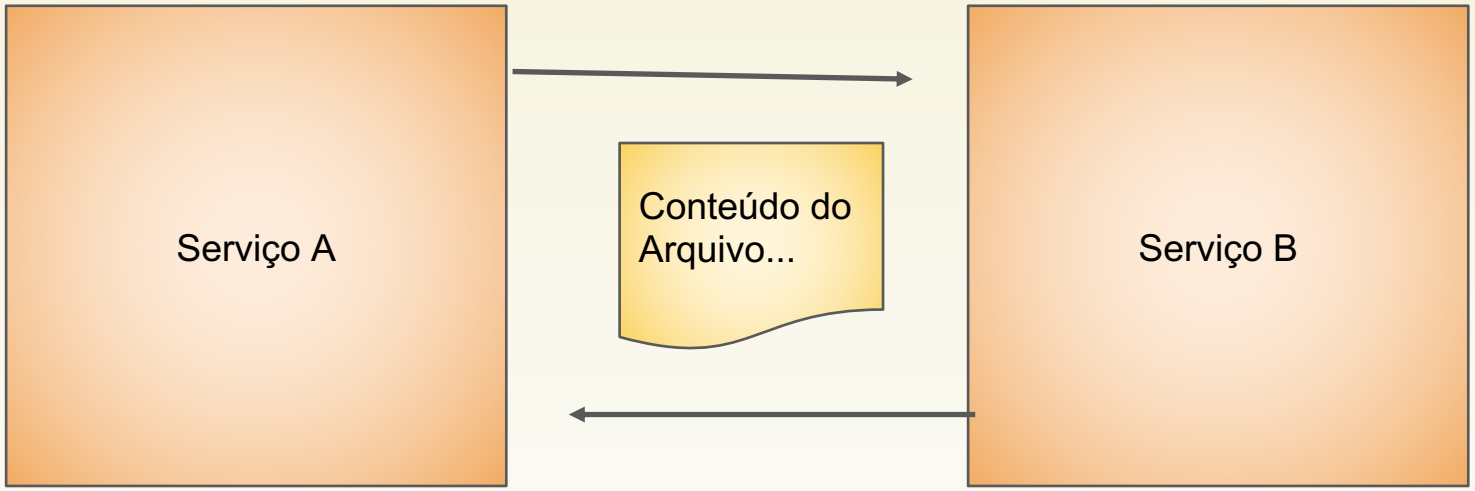


RPC - Remote Procedure Call



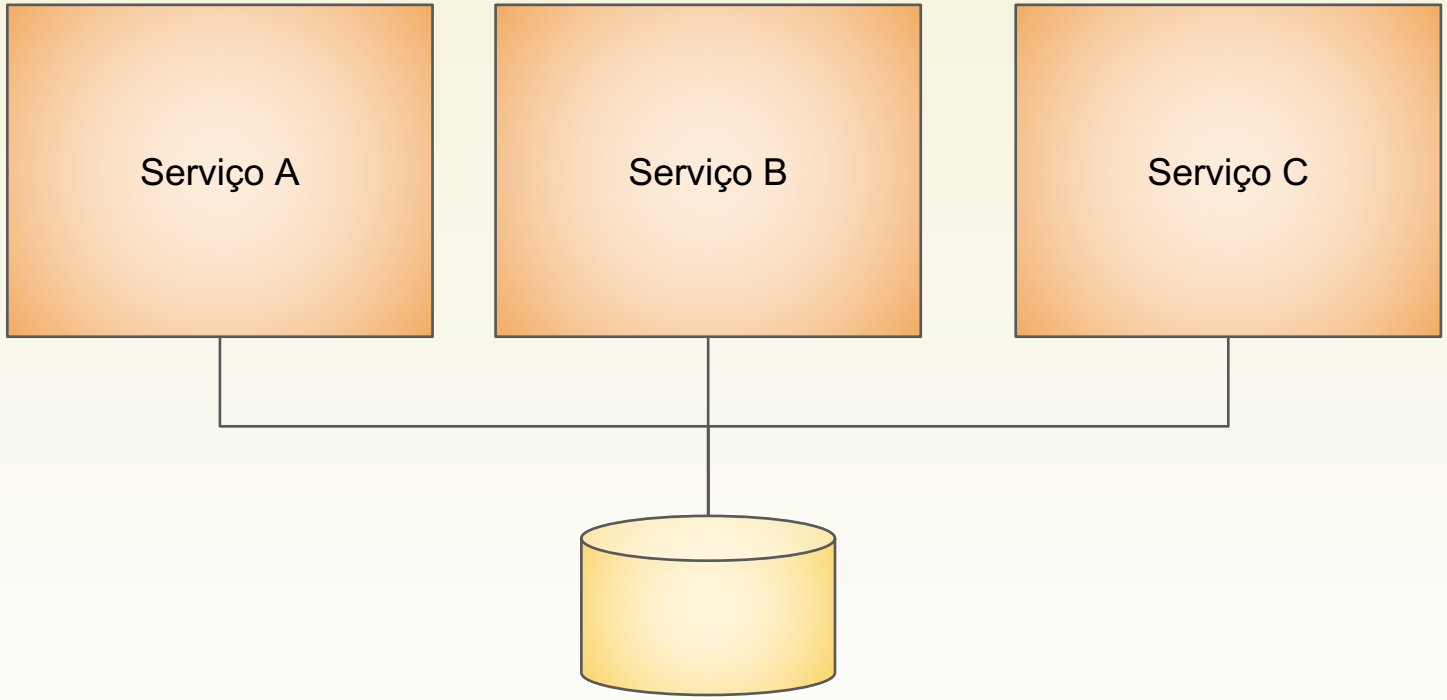
Integração de Informações

Transferência de Arquivos



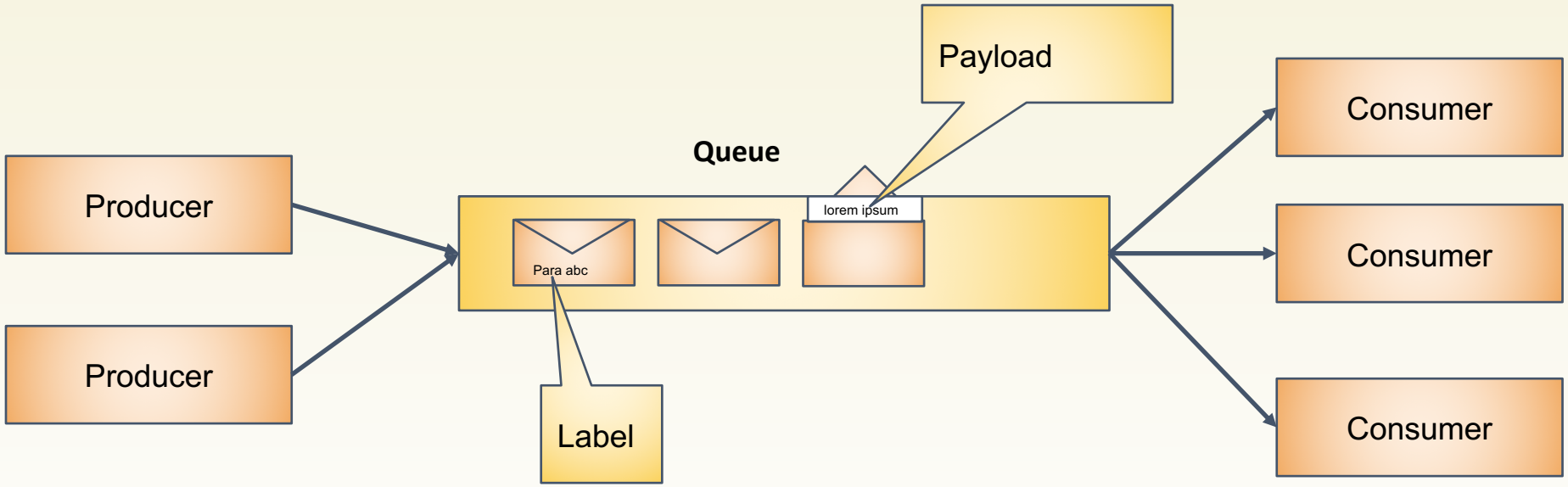
Integração de Informações

Banco de Dados

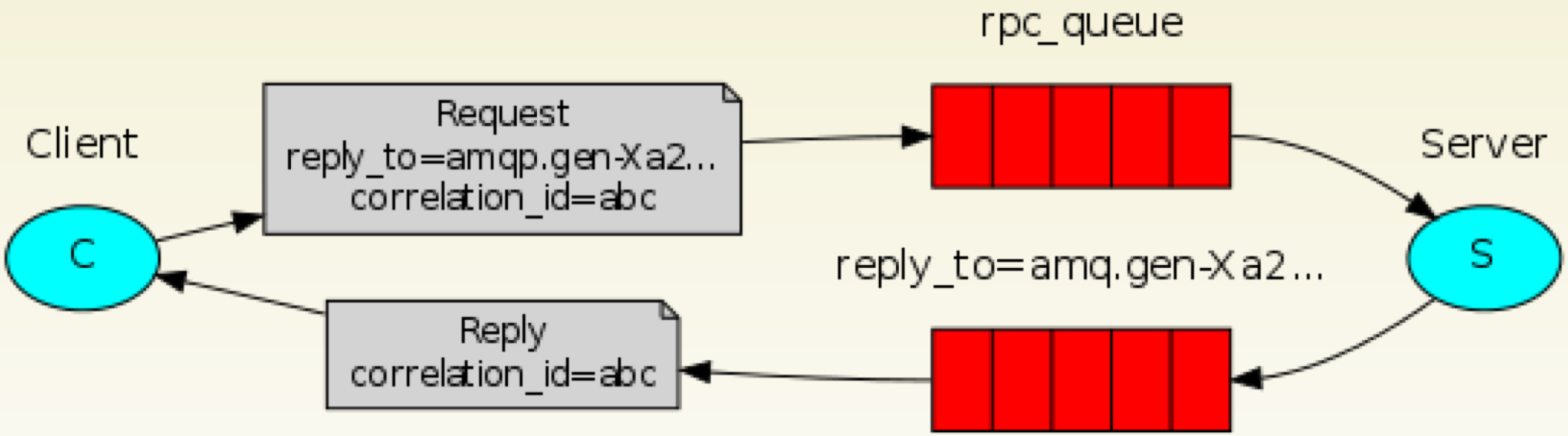


Integração de Informações

Message Broker - Mensageria



RPC - Remote Procedure Call



Message Broker

Producer/Publisher - Quem publica mensagem

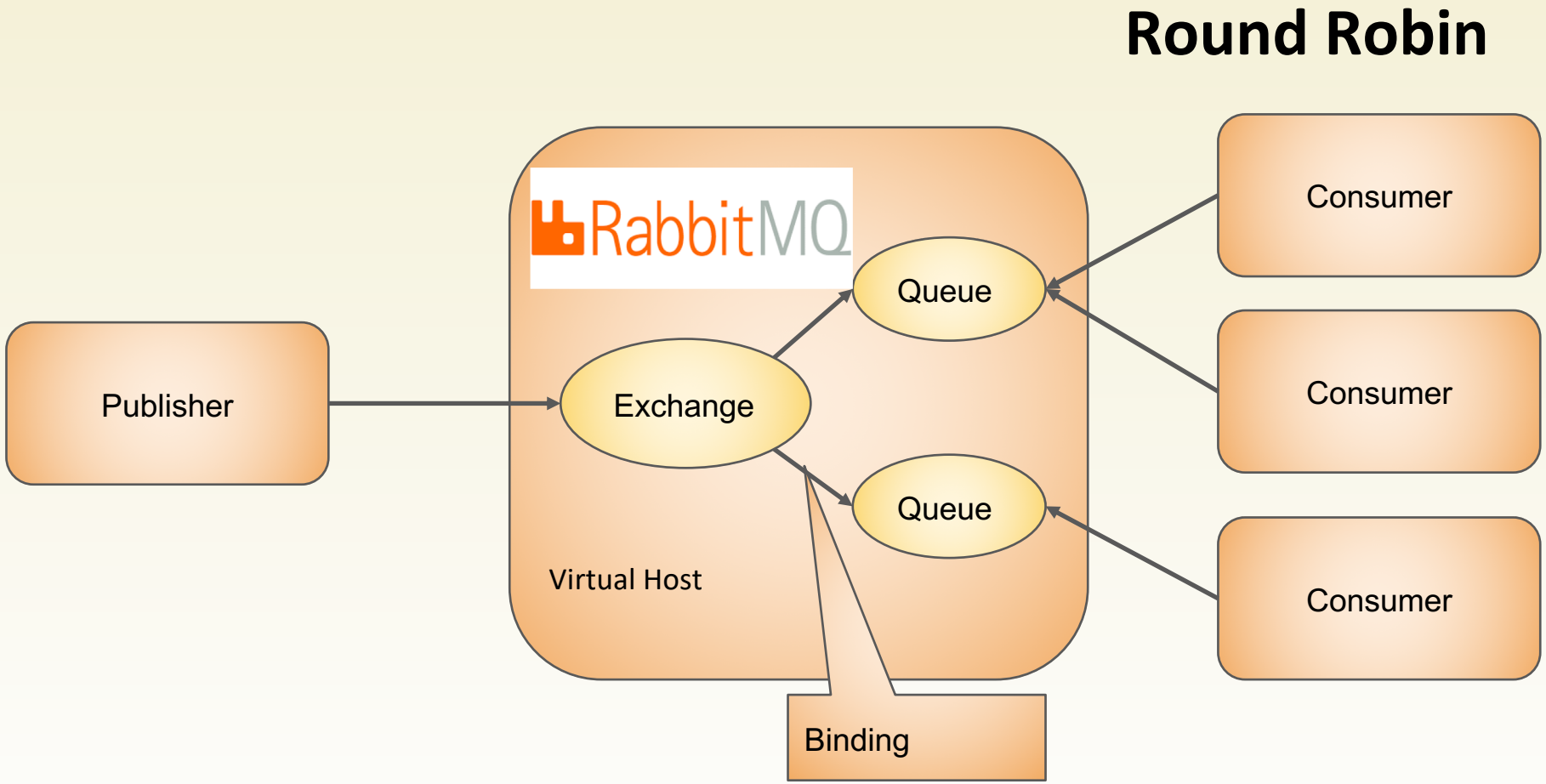
Consumer - Quem consome mensagem

- Resiliência
- Escalabilidade
- Confiabilidade

Assíncrono

Message Broker

Rabbit Message Queue



Exchange

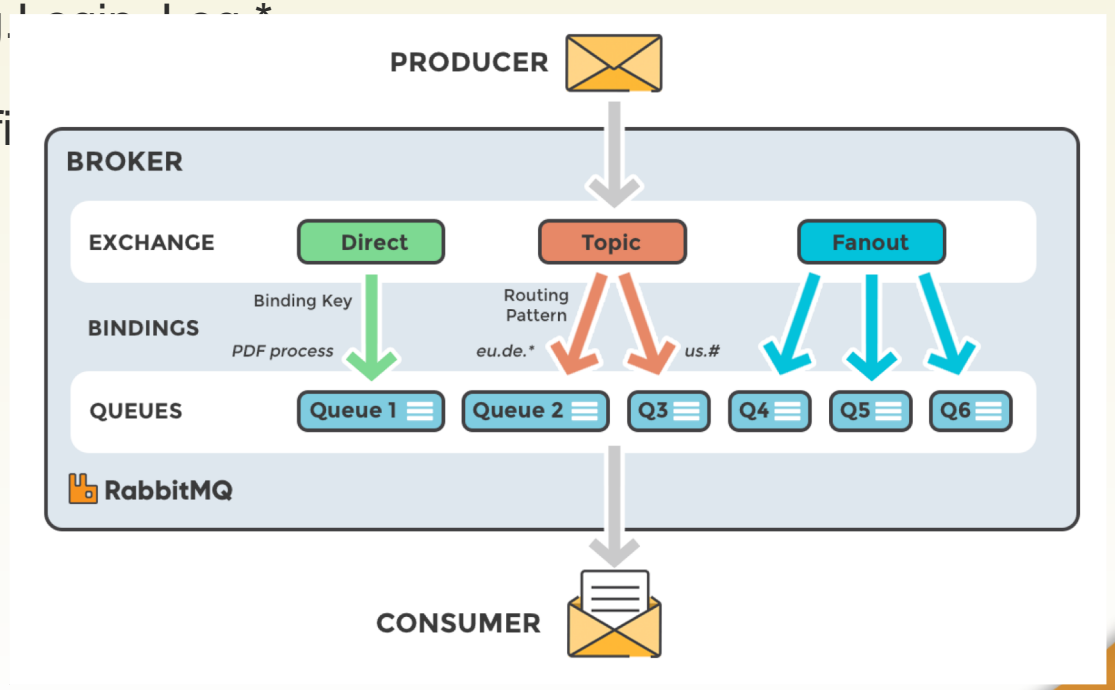
Default = Fila

Direct = Encaminhamento para a fila de mesmo nome

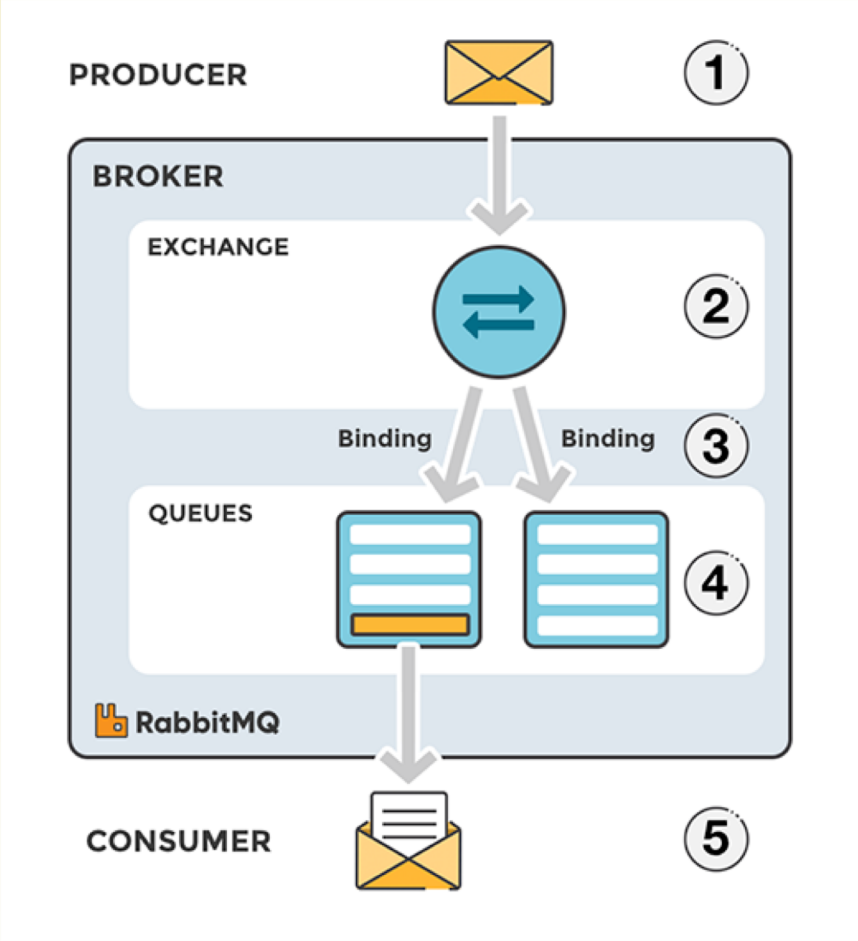
Fanout = Encaminha a mensagem para todas as filas associadas

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

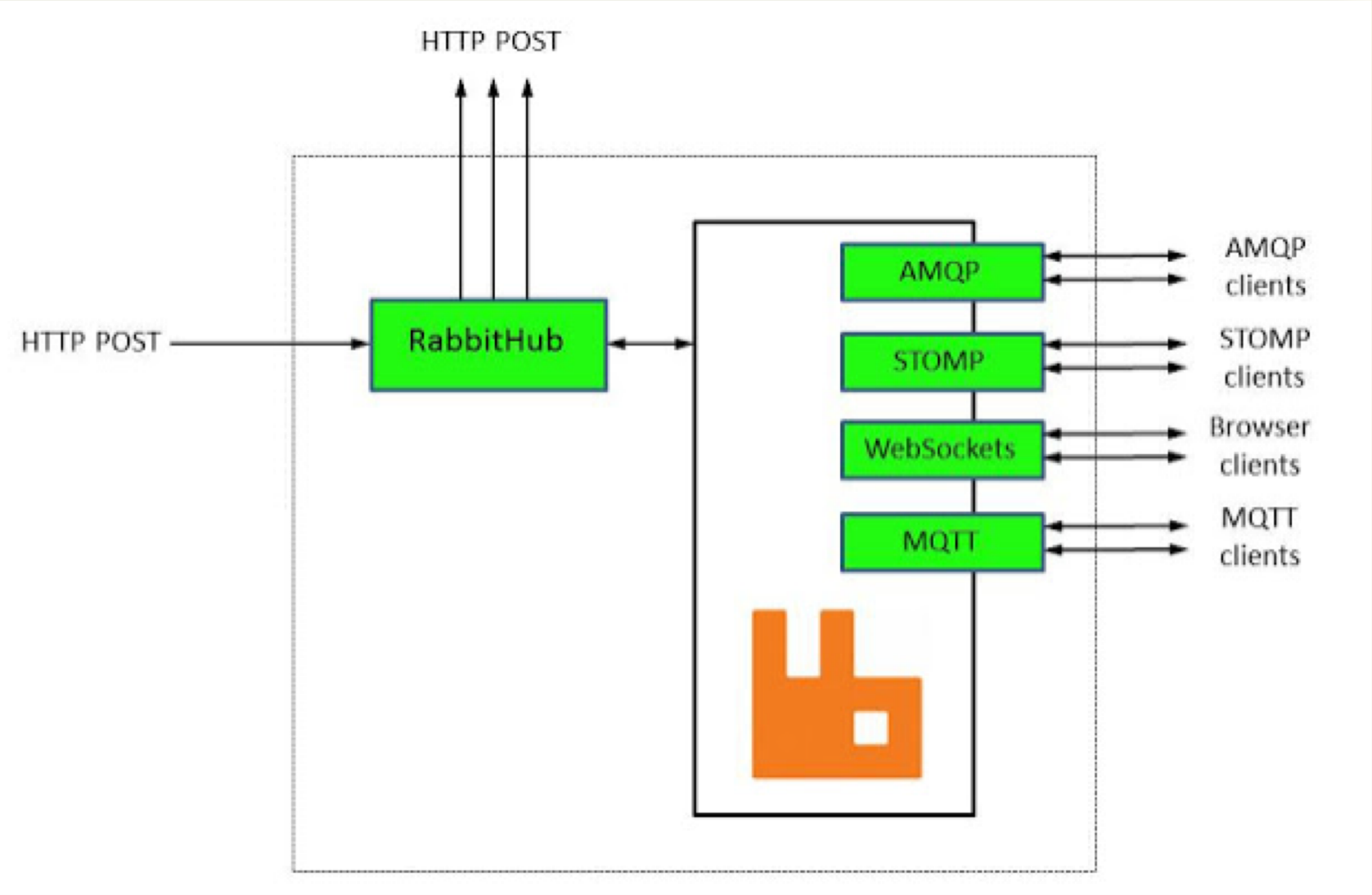
Headers = Usa cabeçalhos para definir a fi



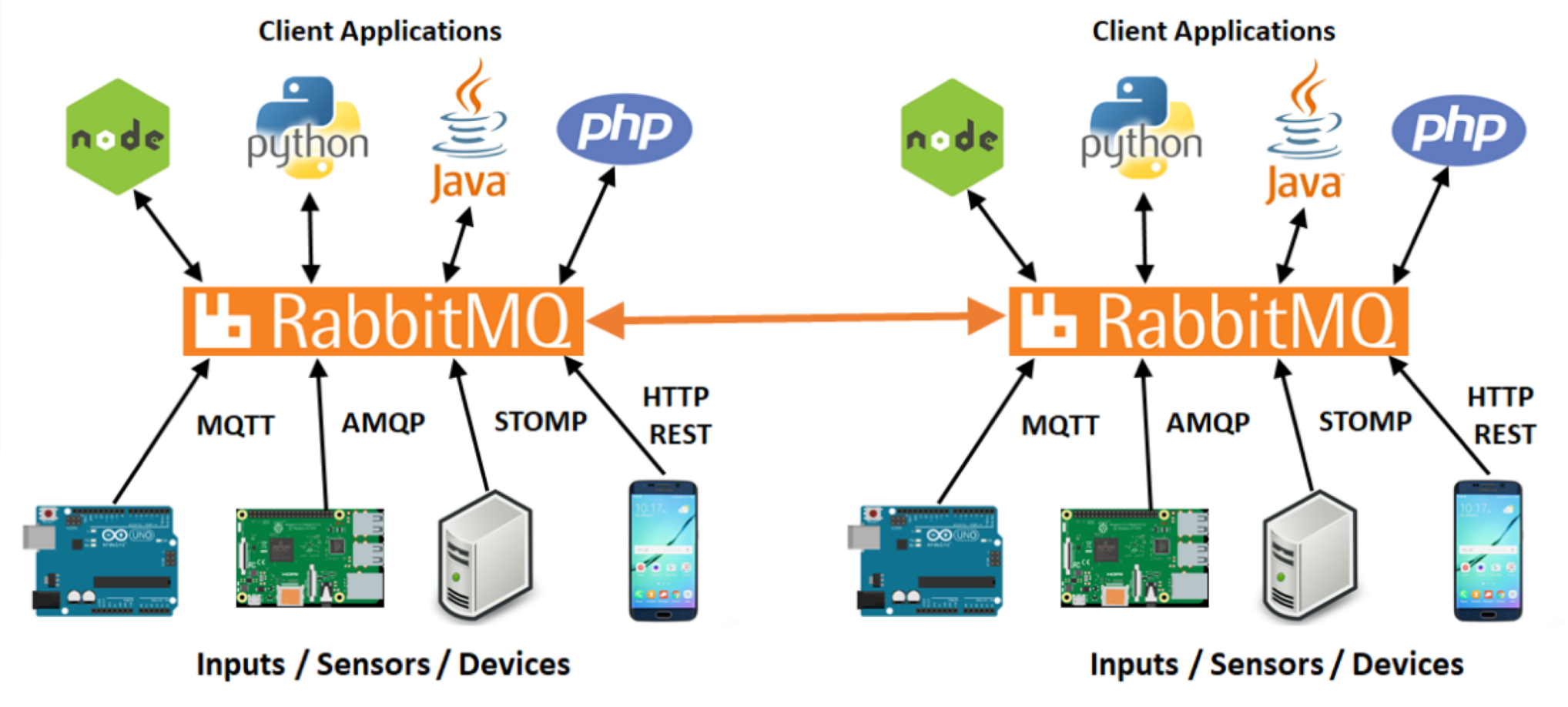
RabbitMQ - Funcionamento



Protocolos


HTTP POST
HTTP POST
RabbitHub
AMQP
STOMP
WebSockets
MQTT
AMQP clients
STOMP clients
Browser clients
MQTT clients

Conectividade



Utilizando Com Delphi

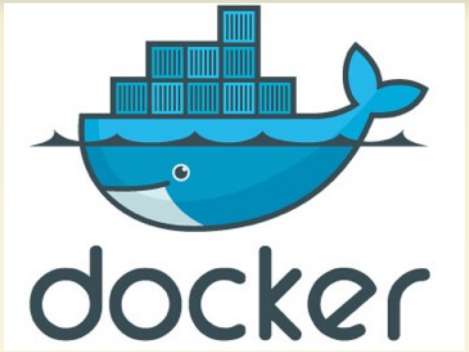
https://www.habarisoft.com/habari_rabbitmq.html

https://github.com/danieleteti/delphistompclient

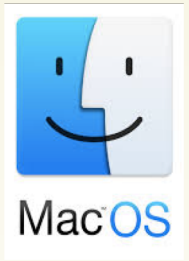
STOMP (Simple Text Oriented Messaging Protocol)

Instalação

1



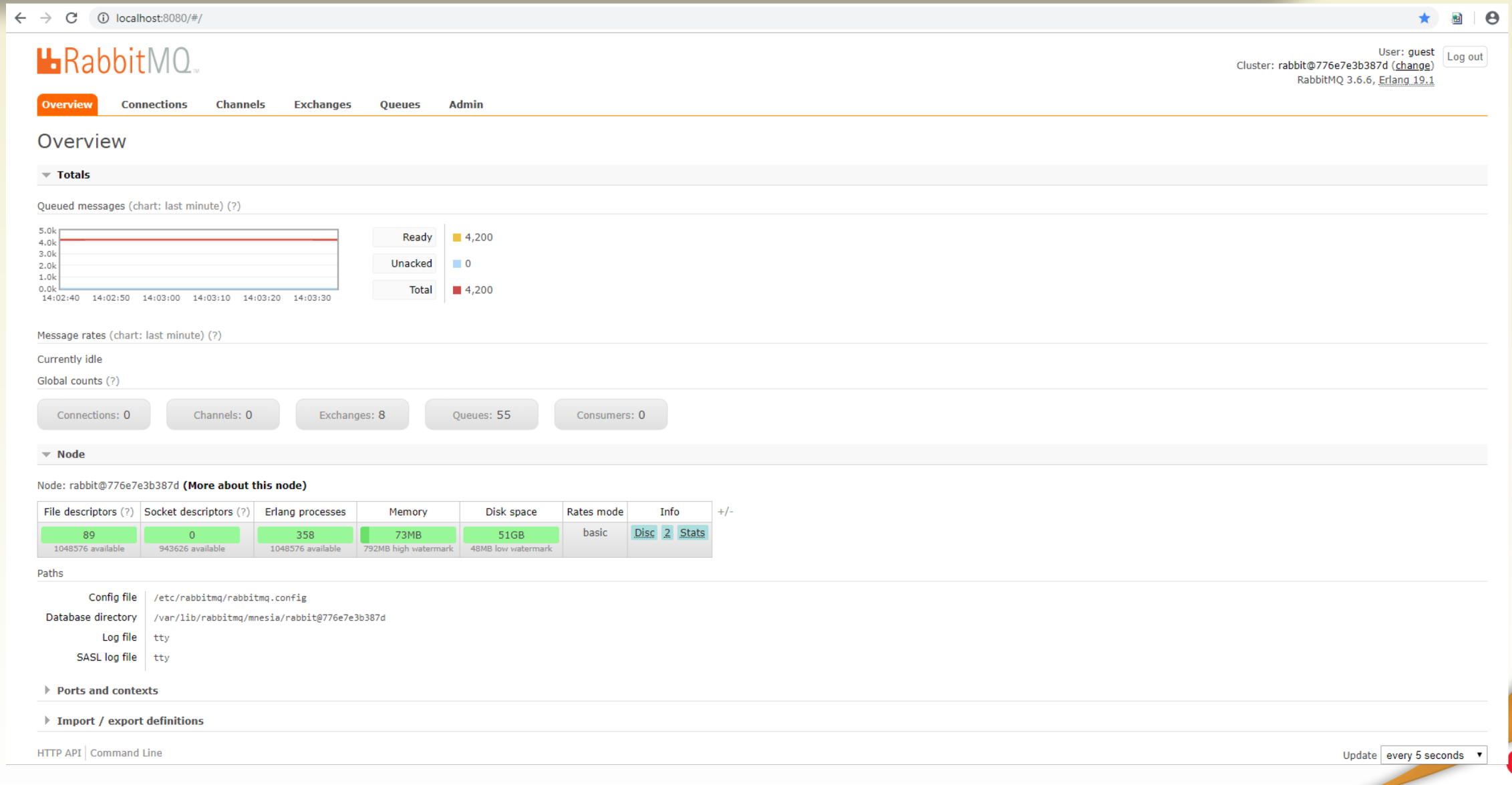

```
docker run -d – hostname my-rabbit – name rabbit13 -p 8080:15672 -p
5672:5672 -p 25676:25676 rabbitmq:3-management
```



2 **Instalar plugin para Stomp**

3 **Acessar http://localhost:8080**

Rabbit



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 me 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

NACK - Mensagem não reconhecida

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

```
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);
```

embarcadero

Obrigado

luiz.sfolia@tmrti.com.br

luizsfolia

#45

Da sala de reunião à vida real

Embarcadero Conference 2019