



iranda Domeneghetti

It is obvious that our technology exceeds our humanity.

## RabbitMQ + .Net Core 3.1 + Docker

SEPTEMBER 2, 2020 / [FERNANDODOMENEGHETTI](#)

Neste exemplo você precisa já ter instalado em seu computador:

- .Net Core 3.1
- Docker instalado e configurado
- Editor de código, neste exemplo utilizei o VsCode durante o desenvolvimento

Primeiro passo é subir um ambiente RabbitMQ utilizando o Docker, para isso rode o comando abaixo em seu cmd\terminal:

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

Este comando irá baixar a imagem do RabbitMQ3 Management, e ao criar o container irá mapear a porta default 15672 para acessar o Rabbit via browser, a porta 5672 para o tráfego das mensagens via protocolo amqp e definimos o nome default do host para my-rabbit

Para validar se o container está em execução, basta executar o comando `docker container ls`

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS	PORTS
91d378d14ee0	rabbitmq:3-management	some-rabbit	"docker-entrypoint.s..."	About a minute ago	Up About a minute	4369/tcp, 5671/tcp, 0.0.0.0:15672->15672/tcp, 15671/tcp, 25672/tcp

(<https://github.com/domeneghetti/RabbitMQ-Exemplo01/blob/master/Imagens/dockerps.png>).

Agora basta acessar o RabbitMQ no navegador através da url <http://localhost:15672/> (<http://localhost:15672/>), o usuário default para o login é:

- Usuário: guest
- Senha: guest

### Configurando a Queue e o Exchange:

Embora seja explicado nos próximos tópicos o que é para que serve a Queue e a Exchange, acho importante já configurarmos uma e ver funcionando, assim quando vier a explicação, tudo fará mais sentido.

#### Advertisements



REPORT THIS AD

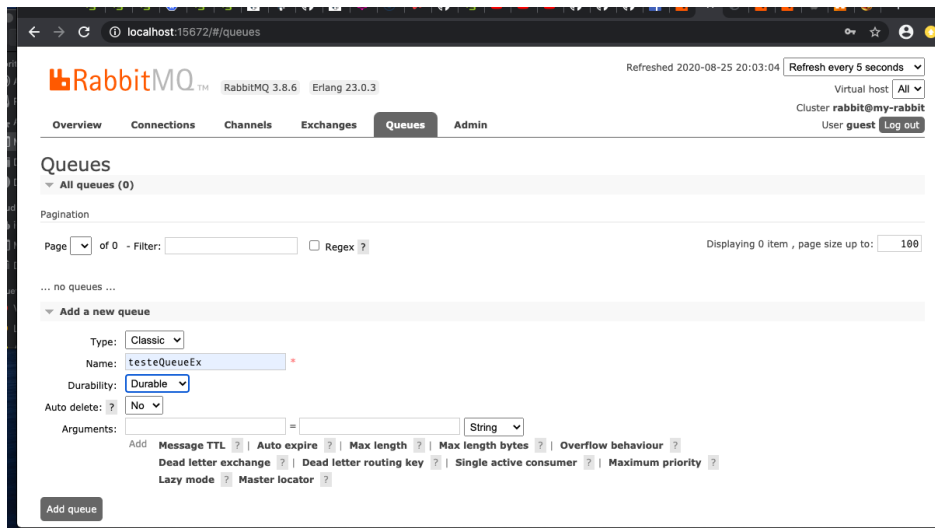
No projeto de Read da fila você verá que embora seja um projeto de leitura, existe uma declaração de uma Queue, isto não tem problema, desde que todos os projetos que utilizem a Queue estejam parametrizando elas com os mesmos valores:

```
channel.QueueDeclare(queue: "testeQueueEx",
    durable: true,
    exclusive: false,
    autoDelete: false,
    arguments: null);
```

(<https://github.com/domeneghetti/RabbitMQ-Exemplo01/blob/master/Imagens/DeclaracaoQueue.png>).

Neste exemplo vamos configurar a Queue via browser e já realizar o vínculo com a Exchange, para criar a Queue basta acessar o menu Queues você verá na página que foi aberta no lado esquerdo um botão com o texto "Add queue", basta clicar sobre ele e preencher os dados da seguinte maneira:

- Type: Classic
- Name: testeQueueEx
- Durability: Durable
- Auto Delete: No
- Arguments: [Não preencher]



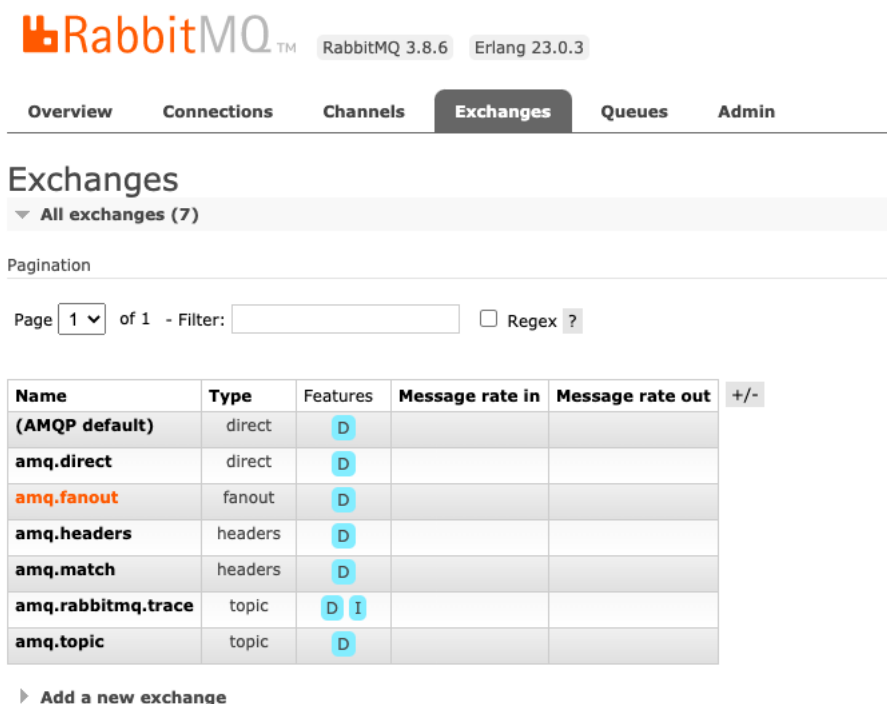
(<https://github.com/domeneghetti/RabbitMQ-Exemplo01/blob/master/Imagens/CriacaoQueue.png>).

Após basta clicar novamente sobre o botão “Add queue”, que a mesma será salva e você será redirecionado para as queues existentes, contendo agora a sua:

Overview				Messages			Message rates				+/-
Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver	get	ack	
testeQueueEx	classic	<span>D</span> <span>Args</span>	idle	0	0	0					

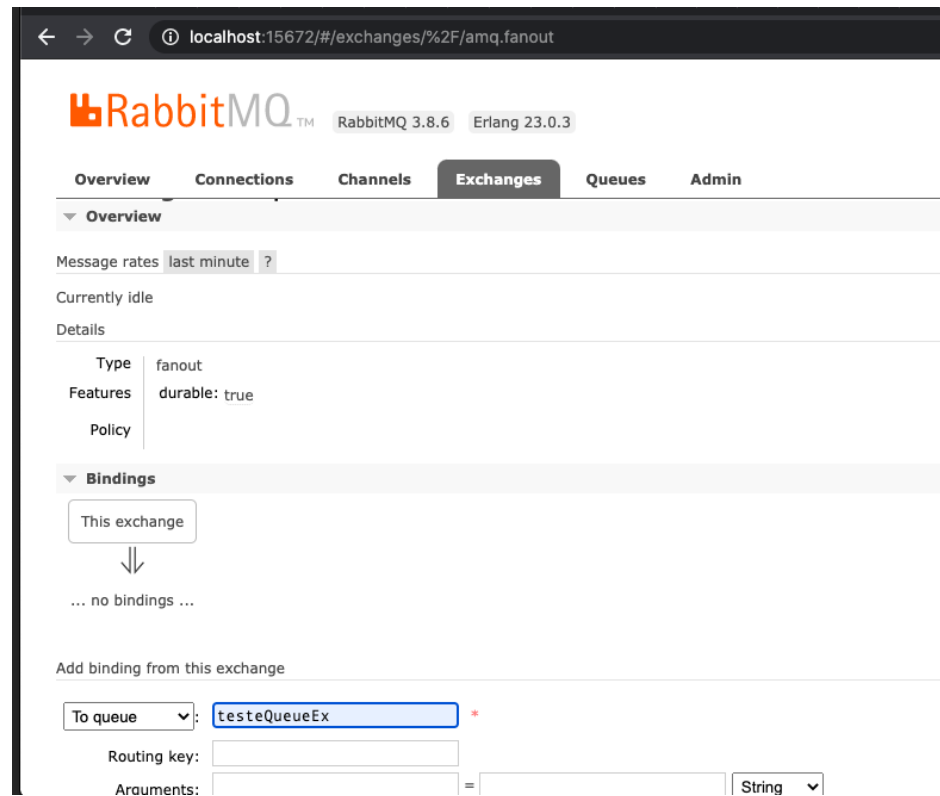
(<https://github.com/domeneghetti/RabbitMQ-Exemplo01/blob/master/Imagens/QueueCriada.png>).

Ao lado do menu da Queues, você tem acesso ao menu Exchanges, clique sobre ele para ver as Exchanges existentes, em seguida clique na opção “amq.fanout”



(<https://github.com/domeneghetti/RabbitMQ-Exemplo01/blob/master/Imagens/Exchanges.png>).

Ao clicar na opção “amq.fanout” você será redirecionado a uma outra página, nesta página você verá uma opção chamada “Bindings”, clique sobre ela para realizar um vínculo com a Queue que criamos no passo anterior, basta informar o nome da Queue e clicar em “bind”:



(<https://github.com/domeneghetti/RabbitMQ-Exemplo01/blob/master/Imagens/ExchangeRealizandoBindingQueue.png>)

Ao realizar o bind você verá uma imagem mostrando o vínculo criado:



(<https://github.com/domeneghetti/RabbitMQ-Exemplo01/blob/master/Imagens/ExchangeAposBinding.png>)

Pronto! Seu ambiente está configurado e já podemos testar, exato é somente isso!

Advertisements

REPORT THIS AD

abra o projeto RabbitMqPublish e no terminal basta digitar “dotnet run”, assim que for executado você no console um texto “Publish!”

```
Program.cs x
Program.cs > {} RabbitMq > RabbitMq.Program > Main(string[] args)
9
10 var connectionFactory = new ConnectionFactory()
11 {
12     Uri = new Uri("amqp://guest:guest@127.0.0.1:5672/"),
13     NetworkRecoveryInterval = TimeSpan.FromSeconds(10),
14     AutomaticRecoveryEnabled = true
15 };
16
17 using(var connection = connectionFactory.CreateConnection())
18 {
19     using(var channel = connection.CreateModel())
20     {
21         channel.ExchangeDeclare(exchange: "amq.fanout", type: ExchangeType.Fanout, durable: true, autoDe
22
23         byte[] messageBodyBytes = System.Text.Encoding.UTF8.GetBytes("{\" nome\":\"Fernando\", dataNascimen
24
25         channel.BasicPublish(exchange: "amq.fanout",
26                               routingKey: "",
27                               basicProperties: null,
28                               body: messageBodyBytes);
29
30         Console.WriteLine("Publish!");
31     }
32 }
```

(<https://github.com/domeneghetti/RabbitMQ-Exemplo01/blob/master/Imagens/Publish.png>).

Entre as linhas 10 e 15 está sendo criada a conexão com o Rabbit, mas precisamente na linha 12 onde informo na Uri

- Amqp é o protocolo de comunicação
- guest:guest são os dados de login no Rabbit
- 127.0.0.0:5672 é referente ao servidor do Rabbit, como estamos rodando o Rabbit via Docker no próprio pc, informo apenas o ip seguido da porta mapeada no Docker
- / a última barra serve para informar o host, por estar utilizando o host padrão, não será informado nenhum valor após a /, caso você crie um novo host pode então passar qual o host estaria sendo utilizado neste processo

Na linha 21 declaro o Exchange utilizado

#### Advertisements

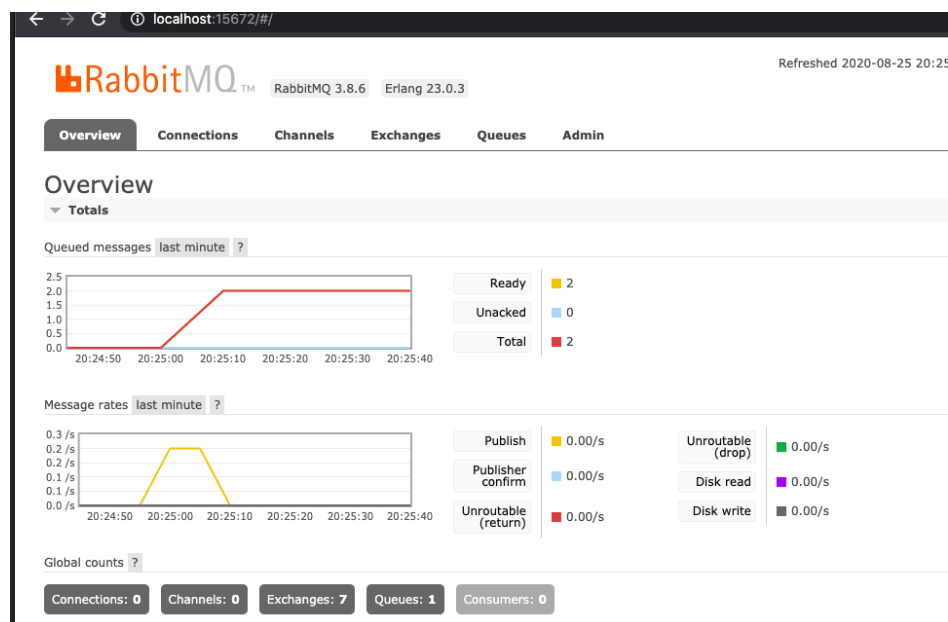


REPORT THIS AD

Na linha 23 é a mensagem que será enviada a fila, o ponto de atenção aqui é que o Rabbit necessita que o conteúdo esteja em uma base64

Na linha 25 é a publicação da mensagem em si informando o Exchange e o body.

Volte ao RabbitMQ, note na página overview que as mensagens já foram identificadas:



(<https://github.com/domeneghetti/RabbitMQ-Exemplo01/blob/master/Imagens/RabbitOverview.png>).

Ao acessar o menu Queues, você verá na queue que você criou que na área Messages a coluna Ready e Total possuem agora um valor referente a quantidade de publish realizados e que ainda não foram lidos

RabbitMQ™

RabbitMQ 3.8.6Erlang 23.0.3

Refreshed 2020-08-2

OverviewConnectionsChannelsExchangesQueuesAdmin

All queues (1)

Pagination

Page 1 of 1 - Filter:  ☐ Regex ? Disp

Overview				Messages			Message rates			+/-
Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack	
testeQueueEx	classic	D Args	idle	2	0	2	0.00/s			

Add a new queue

(<https://github.com/domeneghetti/RabbitMQ-Exemplo01/blob/master/Imagens/QueueWithMessage.png>).

Clique novamente sobre a queue, nesta página além de conseguir ver a quantidade de mensagens que estão na fila para serem lidos como mostra o print abaixo:

← → ↺ ⓘ localhost:15672/#/queues/%2F/testeQueueEx

RabbitMQ™

RabbitMQ 3.8.6Erlang 23.0.3

Refreshed 20

OverviewConnectionsChannelsExchangesQueuesAdmin

Queue testeQueueEx

Overview

Queued messages last minute ?

2.52.01.51.00.50.0

20:25:4020:25:5020:26:0020:26:1020:26:2020:26:30

Ready

Unacked

Total

2

0

2

Message rates last minute ?

1.0 /s0.0 /s

20:25:4020:25:5020:26:0020:26:1020:26:2020:26:30

Publish

0.00/s

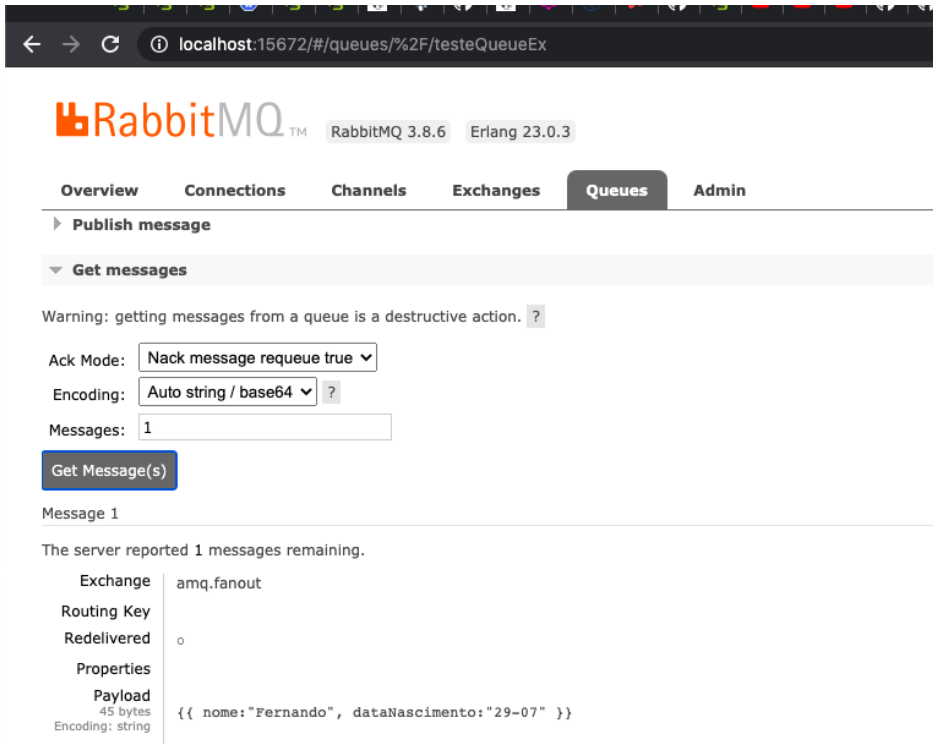
Details

(<https://github.com/domeneghetti/RabbitMQ-Exemplo01/blob/master/Imagens/QueueCountMessage.png>).

Você tem mais abaixo um menu na página como nome de “Get messages” para visualizar o conteúdo das mensagens, mantenha as configurações da tela como o print abaixo e clique em seguida no botão “Get Message(s)”

<https://fernandomirandadomeneghetti.wordpress.com/2020/09/02/rabbitmq-net-core-3-1-docker/>

5/9



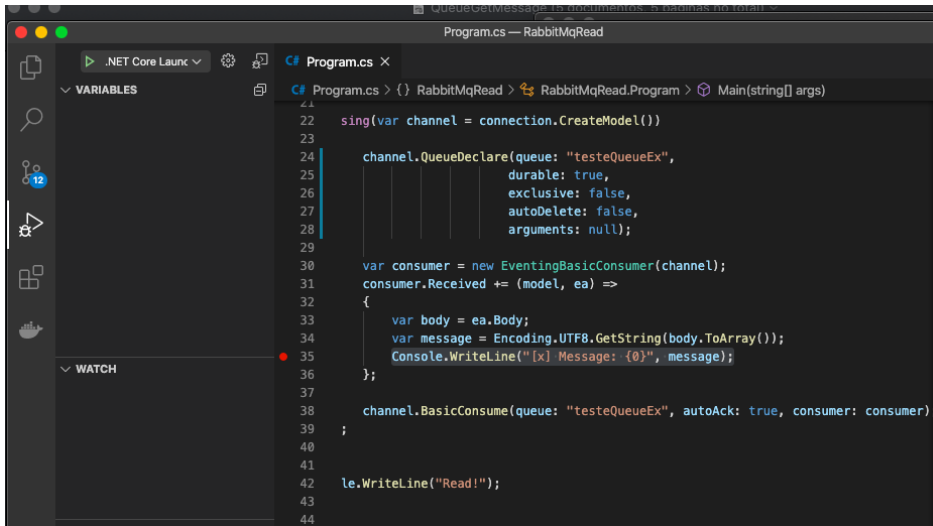
(<https://github.com/domeneghetti/RabbitMQ-Exemplo01/blob/master/Imagens/QueueGetMessage.png>).

Este passo apenas te ajuda a visualizar a sua mensagem, mas não remove ela da fila e também não existe nenhum processamento sobre a mesma ainda, isto será feito neste momento.

Advertisements

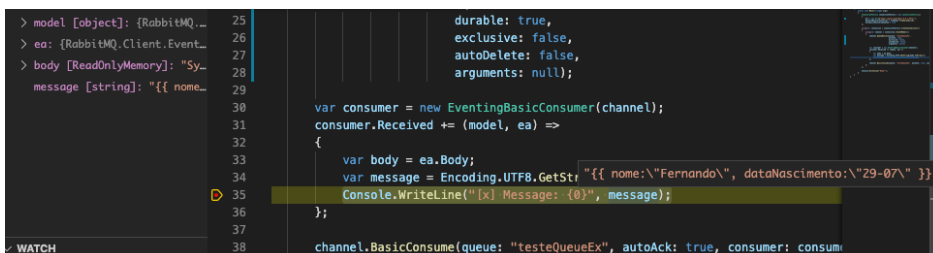
REPORT THIS AD

No VsCode coloque um breakpoint sobre Console.WriteLine na linha 35 e em seguida inicie o modo de depuração do vscode



(<https://github.com/domeneghetti/RabbitMQ-Exemplo01/blob/master/Imagens/AreaDebugVsCode.png>).

Assim que a mensagem for lida, ao passar o mouse sobre a propriedade message note que você já consegue ver o conteúdo



(<https://github.com/domeneghetti/RabbitMQ-Exemplo01/blob/master/Imagens/DebugAnalisandoMensagemLida.png>).

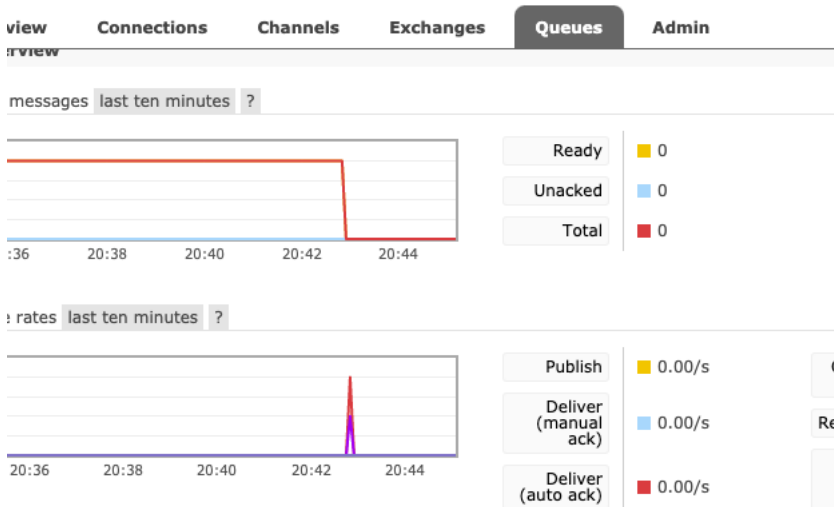
Assim que concluir o debug, você verá que as mensagens lidas, tem seu conteúdo escrito no console junto com a palavra "Read!":

```

11'. Carregamento de símbolos ignorado. O módulo está otimizado e a opção do depurador 'Apenas Meu Código' está habilitada.
Carregado 'C:\usr\local\share\dotnet\shared\Microsoft.NETCore.App\3.1.4\System.Threading.Timer.dll'.
Carregamento de símbolos ignorado. O módulo está otimizado e a opção do depurador 'Apenas Meu Código' está habilitada.
Carregado 'C:\usr\local\share\dotnet\shared\Microsoft.NETCore.App\3.1.4\System.Text.Encoding.Extensions.dll'. Carregamento de símbolos ignorado. O módulo está otimizado e a opção do depurador 'Apenas Meu Código' está habilitada.
[x] Message: [{ nome:"Fernando", dataNascimento:"29-07" }]
[x] Message: [{ nome:"Fernando", dataNascimento:"29-07" }]
Read!
O programa 'C:\[16322] RabbitMQRead.dll' foi encerrado com o código 0 (0x0).
>
  
```

(<https://github.com/domeneghetti/RabbitMQ-Exemplo01/blob/master/Imagens/DebugConsole.png>).

Importante voltar ao navegador do RabbitMQ e ver como ficou a Queue após a leitura, acesse o menu Queue, você verá que a quantidade de mensagens agora da fila é 0 novamente, ao acessar sua Queue veja no gráfico que após as mensagens serem lidas elas foram removidas e o gráfico ajuda inclusive a ver este momento:



(<https://github.com/domeneghetti/RabbitMQ-Exemplo01/blob/master/Imagens/MensagemSaiuQueue.png>).

Agora que você já configurou e já viu uma mensagem sendo enviada e também já fez a leitura do conteúdo das mensagens na fila, vamos entender melhor o que são cada coisa e como elas se relacionam...

## (<https://github.com/domeneghetti/RabbitMQ-Exemplo01#vis%C3%A3o-geral-do-publish--consumer--subscribe>) Visão Geral do Publish \ Consumer \ Subscribe

- Publish: São os responsáveis pela publicação da mensagem na Queue \ Exchange
- Consumer: São os ouvintes das Queues, os mesmos capturam a mensagem na Queue gerada pelo Publish e processam ela.
- Subscribe: Em um modelo mais simples um Publish publicaria a mensagem diretamente em uma Queue que teria seu(s) consumidor(es), já no modelo Publish \ Subscribe o Publish manda a mensagem em uma Exchange do tipo Fanout e a mensagem será clonada e entregue a todas as Queues ligadas a ela.

Advertisements

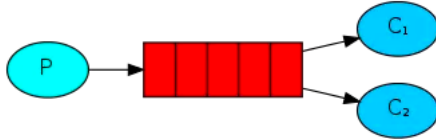
REPORT THIS AD

Com base no que foi falado acima, o caminho mais simples seria, um Publish simplesmente envia a mensagem em uma queue que terá um Consumer ouvindo ela e processará a mesma, como no desenho abaixo:



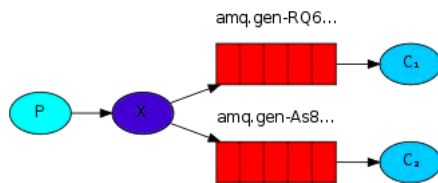
(<https://github.com/domeneghetti/RabbitMQ-Exemplo01/blob/master/Imagens/Simples.png>)

Porém a quantidade de mensagens a serem processadas pode em muitos casos ser muito alta, neste caso você poderá colocar mais Consumer ouvindo a mesma fila, isso não deverá ser um problema, pois o RabbitMQ trabalha de forma atômica, entregando uma mensagem por vez aos consumers, além de garantir que a mensagem seja entregue somente a um Consumer.



(<https://github.com/domeneghetti/RabbitMQ-Exemplo01/blob/master/Imagens/Worker.png>)

Já no modelo Publish\Subscribe, você entregará sua mensagem não a uma Queue em específico mas sim a uma Exchange que distribuirá a mesma entre as Queues (exemplo do fonte deste artigo) e que será processada por um ou mais Consumers.



(<https://github.com/domeneghetti/RabbitMQ-Exemplo01/blob/master/Imagens/PublishSubscribe.png>)

## <https://github.com/domeneghetti/RabbitMQ-Exemplo01#documenta%C3%A7%C3%A3o-rabbitmq>) Documentação RabbitMQ

Por fim, o site do próprio RabbitMQ tem muita informação, muitos exemplos em diversas linguagens e que com certeza vai te ajudar.

### Advertisements



REPORT THIS AD

Link: <https://www.rabbitmq.com/getstarted.html> (<https://www.rabbitmq.com/getstarted.html>)

Todo conteúdo e fonte estão no meu Github <https://github.com/domeneghetti/RabbitMQ-Exemplo01> (<https://github.com/domeneghetti/RabbitMQ-Exemplo01>).

Valeu!

### Sponsored Content

**Descoberta Que Fortalece a Memória de Idosos Choca Jurados do SharkTank - [Assista]** Dr. Rafael Freitas - Neurocirurgião | Sponsored ([https://info.doutornature.com/sfunnel/5826/?utm\\_source=outbrain&utm\\_medium=cpc&dn\\_campanha\\_id=006b075521e65592246357d83efc7e88f6&dn\\_ca03%5D%5Bfunnel%3D5826%5D+Co&ob\\_ad\\_id=00119f70d651d9214cb849013e1a2fe456&ob\\_ad\\_title=Descoberta+Que+Fortalece+a+Mem%C3%B3ria+de+](https://info.doutornature.com/sfunnel/5826/?utm_source=outbrain&utm_medium=cpc&dn_campanha_id=006b075521e65592246357d83efc7e88f6&dn_ca03%5D%5Bfunnel%3D5826%5D+Co&ob_ad_id=00119f70d651d9214cb849013e1a2fe456&ob_ad_title=Descoberta+Que+Fortalece+a+Mem%C3%B3ria+de+)

**Brasil: diga adeus aos caros painéis solares se você mora em Cuiabá** Painéis solares | Links patrocinados | Sponsored

([https://tracking.mb-trk.com/?flux\\_fts=tipxaxztzoitplptaoettxooqxqooqlpaqlxoz57887&reqid=\\$req\\_id&obcid=\\$ob\\_click\\_id&adtitle=Brasil%3A+diga+adeus+aos+caros+pain%C3%A9is](https://tracking.mb-trk.com/?flux_fts=tipxaxztzoitplptaoettxooqxqooqlpaqlxoz57887&reqid=$req_id&obcid=$ob_click_id&adtitle=Brasil%3A+diga+adeus+aos+caros+pain%C3%A9is)

[https://tracking.mb-trk.com/?flux\\_fts=tipxaxztzoitplptaoettxooqxqooqlpaqlxoz57887&reqid=\\$req\\_id&obcid=\\$ob\\_click\\_id&adtitle=Brasil%3A+diga+adeus+aos+caros+pain%C3%A9is](https://tracking.mb-trk.com/?flux_fts=tipxaxztzoitplptaoettxooqxqooqlpaqlxoz57887&reqid=$req_id&obcid=$ob_click_id&adtitle=Brasil%3A+diga+adeus+aos+caros+pain%C3%A9is)

**Confusão em posto de Cuiabá após homem promover aparelho que economiza combustível** G9 News | Sponsored

(<https://g9news.ci/homem-causa-confusao-em-posto-de-gasolina-promover-produto?obOrigUrl=true>)

### Uncategorized

[.NET CORE 3.1](#)



[CREATE A FREE WEBSITE OR BLOG AT WORDPRESS.COM.](https://www.wordpress.com)



