

Aluna: Suellen Oliveira

A linguagem escolhida foi JavaScript.

- No primeiro tutorial o send envia uma mensagem e o receive a recebe.
- Para o segundo é aberto 3 terminais, um com o new\_task, que é quem envia mensagem e dois com o worker.js que é quem a recebe. Quando mandadas as mensagens pelo new\_task elas são distribuídas igualmente no worker, então a primeira mensagem vai aparecer em um terminal a segunda no outro e assim por diante.

### new\_task

```
C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutorial
s\javascript-nodejs\src>node new_task.js UM
[x] Sent 'UM'

C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutorial
s\javascript-nodejs\src>node new_task.js DOIS
[x] Sent 'DOIS'

C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutorial
s\javascript-nodejs\src>node new_task.js TRES
[x] Sent 'TRES'

C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutorial
s\javascript-nodejs\src>node new_task.js QUATRO
[x] Sent 'QUATRO'

C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutorial
s\javascript-nodejs\src>
```

### worker

<pre>[x] Received sec [x] Done ^C C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutorial s\javascript-nodejs\src&gt;node worker.js [*] Waiting for messages in task_queue. To exit press CTRL+C [x] Received UM [x] Done [x] Received TRES [x] Done</pre>	<pre>[x] Done ^C C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutorial s\javascript-nodejs\src&gt;node worker.js [*] Waiting for messages in task_queue. To exit press CTRL+C [x] Received DOIS [x] Done [x] Received QUATRO [x] Done</pre>
---	--

- No terceiro tutorial o emit\_log envia a mensagem e o receive\_logs a recebe. Se abrirmos dois terminais, os endereços de onde os receive\_logs estão esperando a mensagem são diferentes, mas a mensagem enviada no emit\_log aparece para os dois, independente do endereço.

```
C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutorial
s\javascript-nodejs\src>node emit_log.js um
[x] Sent um

C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutorial
s\javascript-nodejs\src>node emit_log.js dois
[x] Sent dois
```

## receive\_logs

```
^C
C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutori
als\javascript-nodejs\src>node receive_logs.js
[*] Waiting for messages in amq.gen-biz_Su3y0FMxLCy4kza6Fg.
To exit press CTRL+C
[x] um
[x] dois
[]
```

```
C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutorial
s\javascript-nodejs\src>node receive_logs.js
[*] Waiting for messages in amq.gen-QNmN5rKLvdYfsPR20791Mw. T
o exit press CTRL+C
[x] um
[x] dois
[]
```

Live Share

- O quarto tutorial tem 3 ações possíveis para o `receive_logs_direct`, ele pode receber “error”, “info” e/ou “warning”. No meu teste eu usei dois terminais um recebendo o error e o outro recebendo info. E para enviar as mensagens utilizei o `emit_log_direct`.

```
C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutorial
s\javascript-nodejs\src>node emit_log_direct.js error run run
[x] Sent error: 'run run'

C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutorial
s\javascript-nodejs\src>node emit_log_direct.js error run run
[x] Sent error: 'run run'

C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutorial
s\javascript-nodejs\src>node emit_log_direct.js info plis corra
a
[x] Sent info: 'plis corra'
```

```
Microsoft Windows [versão 10.0.22631.3447]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutorial
s\javascript-nodejs\src>node receive_logs_direct.js
Usage: receive_logs_direct.js [info] [warning] [error]
```

```
C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutorial
s\javascript-nodejs\src>node receive_logs_direct.js info
[*] Waiting for logs. To exit press CTRL+C
[x] info: 'plis corra'
```

receive\_logs\_direct.js info warning error

```
C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutori
als\javascript-nodejs\src>
node receive_logs_direct.js error
[*] Waiting for logs. To exit press CTRL+C
[x] error: 'run run'
[x] error: 'run run'
[]
```

Ln 1, Col 1 - Espacos: 4 - UTF-8 - CRLF - {} - JavaScript

- Para o penúltimo tutorial, temos algumas opções para o `receive_logs_topic` podemos declarar que ele recebe um log, por exemplo, “kern.\*”, ou que ele recebe múltiplos logs, por exemplo “kern.\*” e “\*.critical”, e depois no `emit_log_topic` podemos mandar uma mensagem para aquele que recebe múltiplos ou para o que recebe apenas um, semelhante ao tutorial anterior. No exemplo testado, abri um terminal que receberia “\*.rabbit” e outro que receberia “rabbit.\*”.

1.

```
s:\javascript-nodejs\src>node emit_log_topic.js red.rabbit Hello
[x] Sent red.rabbit: 'Hello'

C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutorial
s\javascript-nodejs\src>node emit_log_topic.js orange.rabbit ola
[x] Sent orange.rabbit: 'ola'

C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutorial
s\javascript-nodejs\src>node receive_logs_topic.js "*.rabbit"
[*] Waiting for logs. To exit press CTRL+C
[x] red.rabbit:'Hello'
[x] red.rabbit:'Hello'
[x] orange.rabbit:'ola'
```

2.

```
C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutorial
s\javascript-nodejs\src>node emit_log_topic.js rabbit.green hi
[x] Sent rabbit.green: 'hi'

C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutorial
s\javascript-nodejs\src>node receive_logs_topic.js "rabbit.*"
[*] Waiting for logs. To exit press CTRL+C
[x] rabbit.green:'hi'
```

- O sexto e último experimento, mandamos no `rpc_client` uma requisição com um numero e o `rpc_server` retorna com uma sequência de fibonacci.

**rpc\_client**

```
C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutorial
s\javascript-nodejs\src>node rpc_client.js 2
[x] Requesting fib(2)
[.] Got 1
```

## rpc\_server

```
(c) Microsoft Corporation. Todos os direitos reservados.  
  
C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutorial  
s\javascript-nodejs\src>node rpc_server.js  
[x] Awaiting RPC requests  
[.] fib(2)
```

## Dificuldades:

A maior dificuldade foi em descobrir que precisaria baixar o server do RabbitMQ e achar onde fazer o download. Os experimentos foram bem tranquilos de executar.