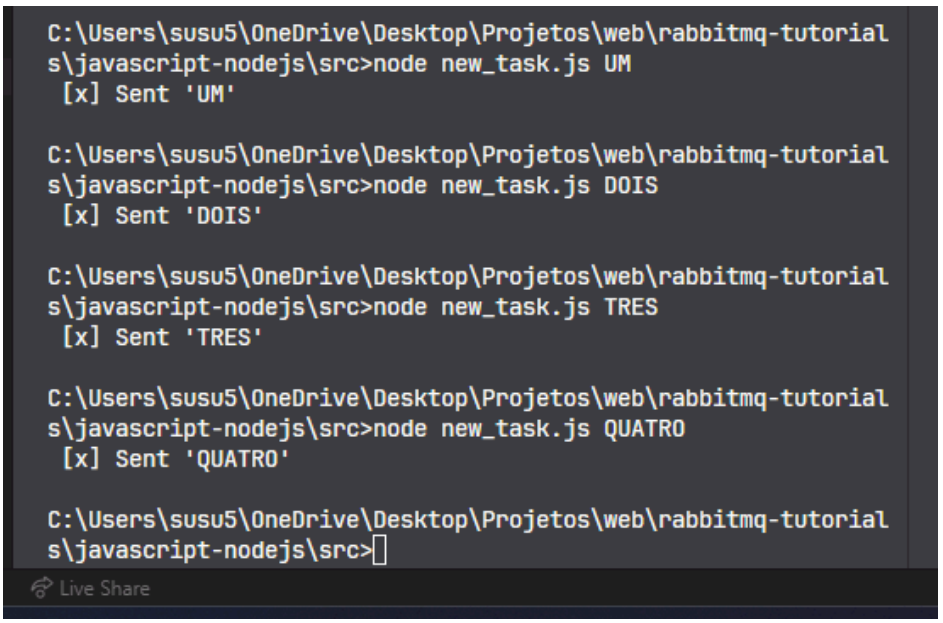
Aluna: Suellen Oliveira

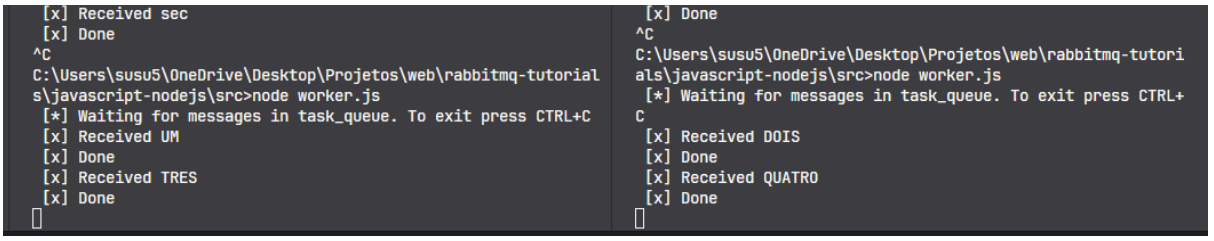
A linguagem escolhida foi JavaScript.

- 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



worker



- 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-QNmN5rKlvdyfsPR2O791Mw. 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 corr
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'
```

- 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 Hell
o
 [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 o
la
 [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'
```

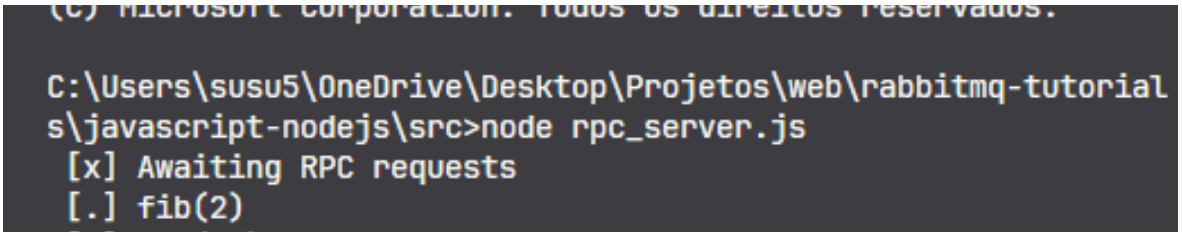
```
C:\Users\susu5\OneDrive\Desktop\Projetos\web\rabbitmq-tutori
als\javascript-nodejs\src>node receive_logs_topic.js "rabbit
.*"
 [*] Waiting for logs. To exit press CTRL+C
 [x] rabbit.green:'hi'
```

- 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-tutori
als\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-tutorials\javascript-nodejs\src>node rpc_server.js
[x] Awaiting RPC requests
[.] fib(2)