



Instituto Federal de Educação, Ciência e  
Tecnologia do Rio Grande do Norte  
Desenvolvimento de Sistemas Distribuídos  
Prof. Gracon Lima

# UTILIZAÇÃO DE MOM COM FILAS DE MENSAGEM

Alice Lima - alice.lima1@escolar.ifrn.edu.br



# INTRODUÇÃO

Uma fila de mensagens é uma estrutura que organiza e armazena mensagens em uma sequência (como uma fila de espera) para comunicação assíncrona entre sistemas ou processos. Ela funciona assim:

- Um produtor envia mensagens para a fila.
- A fila guarda as mensagens até que um consumidor esteja pronto para processá-las.
- Cada mensagem é entregue a apenas um consumidor (modelo ponto a ponto).

Benefícios:

- Desacoplamento: produtores e consumidores não precisam estar ativos ao mesmo tempo.
- Escalabilidade: balanceamento de cargas entre múltiplos consumidores.
- Persistência: mensagens podem ser salvas até serem processadas.





## Fila de Mensagens

# SOBRE O PROJETO

Sistema de filas de mensagens usando o RabbitMQ como Message-Oriented Middleware (MOM), com processos produtores e consumidores. Utilizando API Gateway para enviar mensagens para o RabbitMQ, e consumidores processam essas mensagens.



Aguardando mensagens. Pressione CTRL+C para sair.

Mensagem recebida: it works

Mensagem recebida: it works2

Mensagem recebida: it works4

# CÓDIGO

Produtor (Api Gateway) 

## Importações:

- Flask para criar a aplicação web.
- request para acessar dados das requisições HTTP
- jsonify para retornar respostas em formato JSON.
- biblioteca pika para interagir com o RabbitMQ.
- Criação da instância de Flask

```
1 from flask import Flask, request, jsonify
2 import pika
3
4 app = Flask(__name__)
```

```
7 def get_rabbitmq_connection():
8     credentials = pika.PlainCredentials('guest', 'guest')
9     parameters = pika.ConnectionParameters(
10         host='localhost',
11         port=5672,
12         credentials=credentials
13     )
14     return pika.BlockingConnection(parameters)
```

## Função para conectar ao RabbitMQ:

- Define uma função que cria e retorna uma conexão com o RabbitMQ.
- Cria credenciais usando o usuário e senha padrão
- Configura os parâmetros da conexão
- Estabelece uma conexão bloqueante

## Definição da rota /rabbitmq:

- Define uma rota para aceitar requisições POST. Quando alguém envia um POST para a url, a função publish\_message é chamada.
- Função que processa a requisição e publica a mensagem no RabbitMQ.

```
17 @app.route('/rabbitmq', methods=['POST'])
18 def publish_message():
```

# CÓDIGO

Produtor (Api Gateway)



Bloco try para processamento da requisição:

- Extraí o corpo JSON da requisição HTTP
- Obtém o valor do campo "message" do JSON. Se não existir, retorna None.
- Verifica se message é None ou uma string vazia. Se a validação falhar, retorna erro JSON com status HTTP 400 (Bad Request).

```
17 @app.route('/rabbitmq', methods=['POST'])
18 def publish_message():
19     try:
20         # Obtém os dados da requisição
21         data = request.get_json()
22         message = data.get('message')
23
24         if not message:
25             return jsonify({'error': 'message é obrigatório'}), 400
```

```
connection = get_rabbitmq_connection()
channel = connection.channel()

# Declara a fila (cria se não existir)
channel.queue_declare(queue='minha_fila', durable=True)
```

Publica a mensagem na fila:

- Usa o exchange padrão do RabbitMQ (direciona mensagens diretamente para a fila especificada).
- Especifica a fila de destino.
- Converte a mensagem (string) em bytes para envio.
- Define a mensagem como persistente (salva em disco, não apenas em memória).
- Fecha a conexão com o RabbitMQ após enviar a mensagem.

Publicação da mensagem no RabbitMQ:

- Chama a função para obter uma conexão com o RabbitMQ..
- Cria um canal de comunicação dentro da conexão.
- Declara a fila e a torna persistente.

```
channel.basic_publish(
    exchange='',
    routing_key='minha_fila', # Nome da fila
    body=message.encode(),
    properties=pika.BasicProperties(delivery_mode=2)
)
connection.close()
```

Reposta JSON de êxito e status HTTP de sucesso

```
return jsonify({'message': 'Mensagem publicada na fila com sucesso'}), 200

except Exception as e:
    return jsonify({'error': str(e)}), 500
```

Se qualquer erro ocorrer no bloco, retorna com status 500



# CÓDIGO

```
1 import pika  
2  
3 # Estabelece conexão com o RabbitMQ  
4 connection = pika.BlockingConnection(pika.ConnectionParameters(  
5     host='localhost',  
6     port=5672,  
7     credentials=pika.PlainCredentials('guest', 'guest')  
8 ))  
9 channel = connection.channel()
```

```
12 channel.queue_declare(queue='minha_fila', durable=True)  
13  
14 # Função para processar mensagens recebidas  
15 def callback(ch, method, properties, body):  
16     print(f'Mensagem recebida: {body.decode()}')  
17     ch.basic_ack(delivery_tag=method.delivery_tag)
```

Conversão dos bytes para string e impressão da mensagem.

Confirma ao RabbitMQ que a mensagem foi processada, remove-a da fila. O delivery\_tag é um identificador único da mensagem.

Importação da biblioteca para interagir com o RabbitMQ

Criação de conexão bloqueante com o RabbitMQ

Configura os parâmetros da conexão

Endereço do RabbitMQ

Porta AMQP

Definição das credenciais padrões

Criação de canal para o consumidor interagir com o RabbitMQ.

Declaração de fila e adiciona persistência em caso de reiniciar.

Define uma função que é chamada sempre que uma mensagem é recebida:

- ch: O canal usado para comunicação.
- method: Informações sobre a entrega (ex.: delivery\_tag).
- properties: Propriedades da mensagem (ex.: delivery\_mode).
- body: O conteúdo da mensagem (em bytes).



Consumidor

# CÓDIGO

Configuração do consumidor

```
20     channel.basic_consume(  
21         queue='minha_fila',  
22         on_message_callback=callback,  
23         auto_ack=False  
24     )
```

Configura o consumidor para escutar mensagens da fila:

- Especifica a fila a ser consumida.
- Define a função callback para processar mensagens.
- Desativa a confirmação automática (a mensagem só é removida após o basic\_ack).

```
26     print("Aguardando mensagens. Pressione CTRL+C para sair.")  
27     channel.start_consuming()
```

Exibe uma mensagem indicando que o consumidor está pronto.  
Inicia o loop de consumo e aguarda por mensagens.

# REFERÊNCIAS

**O que é uma fila de mensagens?**. IBM. Disponível em:

<<https://www.ibm.com.br-pt/topics/message-queues>>

Acesso em: Fev. 2025.

**O que é uma fila de mensagens?**. AWS. Disponível em:

<<https://aws.amazon.com/pt/message-queue/>>

Acesso em: Fev. 2025.