Roteiro de Atividade Prática

Nome: Turma: .

**Atividade 1 – Padrões de mensageria avançados com RabbitMQ**

Exercício prático

**Objetivo**

Implementar um exemplo utilizando filas duráveis e trocas de tópicos.

**Passos:**

1. **Criar produtor com troca de tópicos:**
   * Modifique o produtor para enviar mensagens para uma troca de tópicos utilizando chaves de roteamento específicas.
2. **Criar consumidor com filas duráveis:**
   * Adapte o consumidor para escutar uma fila durável e vinculá-la a uma troca de tópicos.

**Exemplos de códigos**

**Produtor:**

import pika # Importa a biblioteca pika para interagir com RabbitMQ.

connection = pika.BlockingConnection(pika.ConnectionParameters('localhost')) # Estabelece uma conexão bloqueante com o RabbitMQ rodando localmente.

channel = connection.channel() # Cria um canal sobre a conexão estabelecida.

channel.exchange\_declare(exchange='topic\_logs', exchange\_type='topic') # Declara uma troca do tipo 'topic' chamada 'topic\_logs'.

routing\_key = 'mensagem.rápida' # Define a chave de roteamento para a mensagem a ser enviada.

message = 'Uma mensagem rápida!' # Define a mensagem a ser enviada.

channel.basic\_publish(exchange='topic\_logs', routing\_key=routing\_key, body=message) # Publica a mensagem na troca com a chave de roteamento especificada.

print(f" [x] Enviada {routing\_key}:{message}") # Imprime no console a mensagem enviada.

connection.close() # Fecha a conexão com o RabbitMQ.

**Consumidor**

import pika # Importa a biblioteca pika para interagir com RabbitMQ.

def callback(ch, method, properties, body): # Define a função de callback que será chamada ao receber mensagens.

print(f" [x] {method.routing\_key}:{body}") # Imprime a chave de roteamento e a mensagem recebida.

connection = pika.BlockingConnection(pika.ConnectionParameters('localhost')) # Estabelece uma conexão bloqueante com o RabbitMQ rodando localmente.

channel = connection.channel() # Cria um canal sobre a conexão.

channel.exchange\_declare(exchange='topic\_logs', exchange\_type='topic') # Declara novamente a troca do tipo 'topic' chamada 'topic\_logs'.

result = channel.queue\_declare('', exclusive=True) # Declara uma fila exclusiva e temporária.

queue\_name = result.method.queue # Obtém o nome da fila gerado pelo servidor.

binding\_key = 'mensagem.#' # Define a chave de ligação para a fila, ouvindo todas as mensagens que começam com 'mensagem.'.

channel.queue\_bind(exchange='topic\_logs', queue=queue\_name, routing\_key=binding\_key) # Liga a fila à troca com a chave de ligação especificada.

channel.basic\_consume(queue=queue\_name, on\_message\_callback=callback, auto\_ack=True) # Configura o consumo da fila, registrando a função de callback e habilitando a confirmação automática de mensagens.

print(' [\*] Esperando por mensagens. Para sair pressione CTRL+C') # Imprime no console que o consumidor está pronto para receber mensagens.

channel.start\_consuming() # Inicia o loop de consumo de mensagens.

**Tempo estimado:** 35 minutos.

**Lista de materiais**

* Computador com internet;
* Caderno para anotações;
* 1 caneta.

**Procedimento experimental**

1. A partir da implementação, descreva o resultado obtido na atividade e registre-o no AVA:

|  |
| --- |
|  |
|  |
|  |
|  |