Roteiro de Atividade Prática

Nome: Turma: .

**Atividade 1 – Criar um produtor (*publisher*) que envia mensagens para o RabbitMQ e um consumidor (*consumer*) que as recebe**

**Objetivo**

Criar um produtor (*publisher*) que envia mensagens para o RabbitMQ e um consumidor (*consumer*) que as recebe.

**Passos:**

1. **Criar um produtor de mensagens:**
   * Desenvolva um script Python que estabeleça uma conexão com o RabbitMQ e envie uma mensagem para uma fila.
2. **Criar um consumidor de mensagens:**
   * Escreva um script Python que se conecte ao RabbitMQ, escute a fila e processe as mensagens recebidas.

**Exemplos de códigos**

**Produtor**

import pika # Importa a biblioteca pika para interação com o RabbitMQ.

connection = pika.BlockingConnection(pika.ConnectionParameters('localhost')) # Cria uma conexão bloqueante com o RabbitMQ na máquina local.

channel = connection.channel() # Abre um canal na conexão para enviar e receber mensagens.

channel.queue\_declare(queue='hello') # Assegura que a fila 'hello' existe antes de enviar mensagens.

channel.basic\_publish(exchange='', # Usa a troca padrão do RabbitMQ.

routing\_key='hello', # Especifica a chave de roteamento (nome da fila) para onde a mensagem será enviada.

body='Olá, Mundo!') # Envia a mensagem "Olá, Mundo!".

print(" [x] Enviada 'Olá, Mundo!'") # Exibe uma confirmação no console de que a mensagem foi enviada.

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

**Consumidor**

import pika # Importa a biblioteca pika para interação com o RabbitMQ.

def callback(ch, method, properties, body): # Define a função de callback para processar mensagens recebidas.

print(f" [x] Recebida {body}") # Imprime a mensagem recebida.

connection = pika.BlockingConnection(pika.ConnectionParameters('localhost')) # Cria uma conexão bloqueante com o RabbitMQ na máquina local.

channel = connection.channel() # Abre um canal na conexão.

channel.queue\_declare(queue='hello') # Assegura que a fila 'hello' existe.

channel.basic\_consume(queue='hello', # Especifica a fila de onde as mensagens serão consumidas.

on\_message\_callback=callback, # Define a função de callback que é chamada com cada mensagem recebida.

auto\_ack=True) # Habilita a confirmação automática de que a mensagem foi processada.

print(' [\*] Esperando por mensagens. Para sair pressione CTRL+C') # Informa ao usuário que o consumidor está pronto e esperando mensagens.

channel.start\_consuming() # Inicia o consumo de mensagens, entrando em um loop que espera por 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:

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