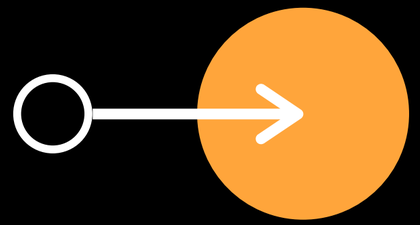
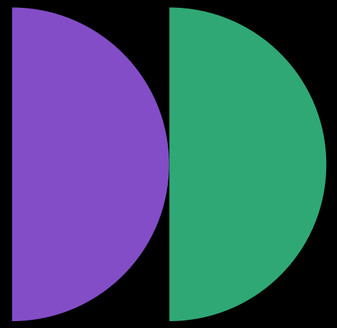
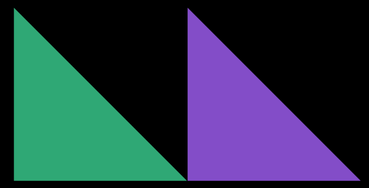
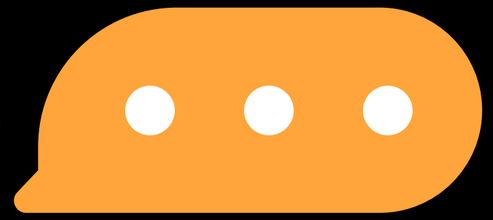


Protocolo AMQP e o broker RabbitMQ



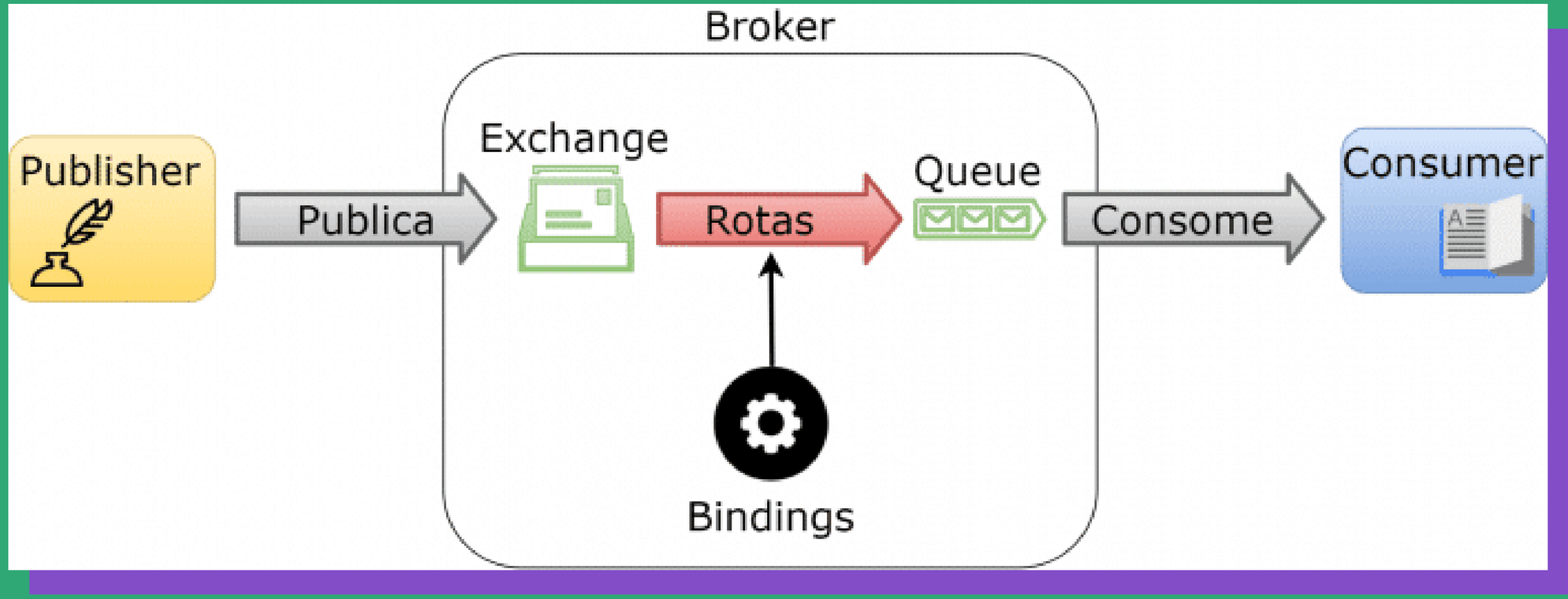




João Pedro Moura Oliveira - 190030879

Um pouco mais sobre o protocolo AMQP

- Sistema de correios assíncrono
- Independente de hardware, SO e linguagem de programação
- Protocolo bi-direcional para envio de mensagens através do broker
- Altamente programável (rotas, exchanges, filas, ...)



Fonte: Medicci T. S. 2018.

Recebimento de mensagens

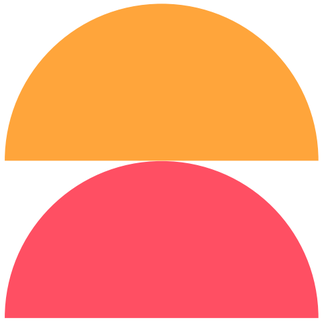
- Recebem as mensagens dos clientes

Encaminhamento de mensagens

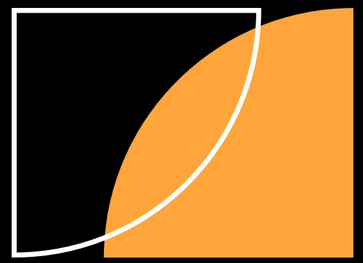
- Encaminham para as filas
- Regras determinadas pelos bindings

Os diferentes tipos

- Direct
- Fanout
- Topic
- Headers



Exchanges Importância e seus tipos



Um pouco mais sobre o broker RabbitMQ e suas implementações

Um serviço de mensageria utilizando o protocolo AMQP

Uma simples fila

Apenas um produtor conversando com um consumidor

```python
connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
canal = connection.channel()

canal.queue_declare(queue='exemplo1')

def callback(canal, metodo, propriedades, corpo):
    print(f"[x] Recebido: {corpo}")

canal.basic_consume(
    queue='exemplo1',
    auto_ack=True,
    on_message_callback=callback
)

print('[*] Esperando mensagens. CTRL+C para sair')
canal.start_consuming()
```

```python
canal.basic_publish(
    exchange='',
    routing_key='exemplo1',
    body='Ola mundo'
)

print("[x] Mensagem enviada")
connection.close()
```

Customizando as filas

- Dividindo tarefas
  - Round-Robin
  - Divisão justa
- Durabilidade da mensagem
  - Acknowledgment (ACK)
  - Salvando as filas

```
canal.queue_declare(queue='exemplo2', durable=True)

mensagem = ' '.join(sys.argv[1:]) or "Ola mundo"
canal.basic_publish(
    exchange='',
    routing_key='exemplo1',
    body=mensagem,
    properties=pika.BasicProperties(
        delivery_mode=pika.spec.PERSISTENT_DELIVERY_MODE
    )
)
```

```
canal.queue_declare(queue='exemplo2', durable=True)

def callback(canal, metodo, propriedades, corpo):
    print(f"[x] Recebido: {corpo.decode()}")
    time.sleep(corpo.count(b'.'))
    print("[x] Feito")
    canal.basic_ack(delivery_tag=metodo.delivery_tag)

canal.basic_qos(prefetch_count=1)
canal.basic_consume(
    queue='exemplo1',
    on_message_callback=callback
)
```

Padrão publish/subscribe

Uma mensagem para vários consumidores usando exchanges

```python
canal.exchange_declare(exchange='logs', exchange_type='fanout')

result = canal.queue_declare(queue='', exclusive=True)
nome_fila = result.method.queue

canal.queue_bind(exchange='logs', queue=nome_fila)

def callback(canal, metodo, propriedades, corpo):
    print(f"[x] Recebido: {corpo.decode()}")

canal.basic_consume(
    queue=nome_fila,
    on_message_callback=callback,
    auto_ack=True
)
```

```python
canal.exchange_declare(exchange='logs', exchange_type='fanout')

mensagem = ' '.join(sys.argv[1:]) or "Ola mundo"
canal.basic_publish(
    exchange='logs',
    routing_key='',
    body=mensagem,
)
```

Direcionando mensagens

Realizando a inscrição dos clientes em apenas algumas mensagens

```python
canal.exchange_declare(exchange='logs_diretos', exchange_type='direct')

gravidade = sys.argv[1] if len(sys.argv) > 1 else 'info'
mensagem = ' '.join(sys.argv[2:]) or "Ola mundo"

canal.basic_publish(
    exchange='logs_diretos',
    routing_key=gravidade,
    body=mensagem,
)
```

```python
canal.exchange_declare(exchange='logs_diretos', exchange_type='direct')

result = canal.queue_declare(queue='', exclusive=True)
nome_fila = result.method.queue

gravidades = sys.argv[1:]
for gravidade in gravidades:
    canal.queue_bind(
        exchange='logs_diretos',
        queue=nome_fila,
        routing_key=gravidade
    )
```

Usando tópicos

Direcionando mensagens com vários tópicos

```
canal.exchange_declare(exchange='logs_topico', exchange_type='topic')

result = canal.queue_declare(queue='', exclusive=True)
nome_fila = result.method.queue

gravidades = sys.argv[1:]
for gravidade in gravidades:
    canal.queue_bind(
        exchange='logs_topico',
        queue=nome_fila,
        routing_key=gravidade
    )
```

Criando um RPC com RabbitMQ

```python
def on_response(self, canal, metodo, propriedades, corpo):
    if self.corr_id == propriedades.correlation_id:
        self.resposta = corpo

def call(self, n):
    self.resposta = None
    self.corr_id = str(uuid.uuid4())
    self.canal.basic_publish(
        exchange='',
        routing_key='fila_rpc',
        properties=pika.BasicProperties(
            reply_to=self.nome_fila,
            correlation_id=self.corr_id,
        ),
        body=str(n)
    )

    while self.resposta is None:
        self.connection.process_data_events()

    return int(self.resposta)
```

```python
def fib(n):
    if n in [0, 1]:
        return n
    else:
        return fib(n-1) + fib(n-2)

def on_request(canal, metodo, propriedades, corpo):
    n = int(corpo)
    response = fib(n)

    canal.basic_publish(
        exchange='', routing_key=propriedades.reply_to,
        properties=pika.BasicProperties(
            correlation_id=propriedades.correlation_id),
        body=str(response)
    )
```

Referências

AMQP – Protocolo de Comunicação para IoT. Medicci T. S. 2018. Disponível em: <https://www.embarcados.com.br/amqp-protocolo-de-comunicacao-para-iot/>. Acesso em: 16 de fev. de 2022.

AMQP. Disponível em: <https://www.amqp.org/>. Acesso em: 16 de fev. de 2022.

RabbitMQ Tutorials. Disponível em: <https://www.rabbitmq.com/getstarted.html>. Acesso em: 16 de fev. de 2022.