Exercício 7: RabbitMQ (PC)

Nilo Bemfica Mineiro Campos Drumond - Pedro Didier Maranhão

Implementação - Rust

Foi usada a biblioteca **amiquip** como cliente RabbitMQ para Rust.

O consumidor recebe 40 * 10.000 mensagens e calcula a média de tempo entre o envio e o seu recebimento.

O RabbitMQ estava com as configurações padrões.

Consumer

```rust
main() -> Result<()> {
 let mut connection = Connection::insecure_open("amqp://guest:guest@localhost:5672")?;
 let channel = connection.open_channel(None)?; (channel_id) Channel
 let queue = channel.queue_declare("test", QueueDeclareOptions::default())?; (queue, c
 let consumer = queue.consume(ConsumerOptions::default())?; (options) Consumer

 let mut durations: Vec<i64> = Vec::new();
```

```rust
for (i, message) in consumer.receiver().iter().enumerate() {
    match message {
        amiquip::ConsumerMessage::Delivery(delivery) => { Del
            let data = parse_data(&delivery.body[..8]); (raw)
            let send_time = i64::from_be_bytes(data); i64
            let now = offset::Utc::now().timestamp_millis();
            let time_diff = now - send_time; i64

            durations.push(time_diff);

            consumer.ack(delivery)?;
```

Cliente

```rust
let mut connection = Connection::insecure_open("amqp://guest:guest@localhost:5672")?;
let channel = connection.open_channel(None)?; (channel_id) Channel
let exchange = Exchange::direct(&channel); Exchange

for _ in 0..10_000 {
    let now = offset::Utc::now(); DateTime<Utc>
    let now = now.timestamp_millis(); i64
    let mut now = Vec::from(now.to_be_bytes()); Vec<u8>
    now.resize(size, 0); (new_len, value)

    exchange.publish(Publish::new(&now, "test"))?; (body, routing_key)
}

connection.close()
```

Mudança

Consumer:

```
let channel = connection.open_channel(None)?;
channel.qos(0, prefetch_count, false)?; (pref
```

Tempo Médio de Publicação (ms)