



PROTOCOLO AMQP E O BROKER RABBITMQ

Victor Samuel dos Santos Lucas - 180028685

João Pedro Silva de Carvalho - 180033743

Detalhes da Apresentação

01

Advanced Message Queueing Protocol - AMQP

Visão geral, características e
funcionalidades

02

Cenários de criação de aplicativos de mensagens usando RabbitMQ

Visão geral, execução, situações
práticas e diferenças


03

Referências



Advanced Message Queueing Protocol (AMQP)

O AMQP é um acrônimo de Advanced Message Queuing Protocol traduzindo é Protocolo avançado de enfileiramento de mensagens. Ele serve para troca de mensagens entre aplicações independente da linguagem, ou seja, é um protocolo de enquadramento e transferência padronizado para transferir mensagens de forma assíncrona, segura e confiável entre duas partes. Ele é o principal protocolo de mensagens do Barramento de Serviço e dos Hubs de Eventos do Azure. Fazendo uma analogia podemos dizer que o AMQP é como o HTTP é para aplicações web convencionais.



Vantagens


- Produzir um padrão aberto para protocolos de mensageria
- Permitir a interoperabilidade entre várias tecnologias e plataformas.





RabbitMQ

O RabbitMQ é uma aplicação de código aberto que suporta o protocolo AMQP para receber mensagens e organizá-las em filas, além de disponibilizá-las para que outras aplicações possam recebê-las. Chamamos o RabbitMQ de um Message Broker (Intermediador de mensagens). Quando trabalhamos com o RabbitMQ a nossa aplicação envia as mensagens para uma Exchange e ele se encarrega de organizar as mensagens nas filas.



Conceitos

- **Exchange**
 - Uma Exchange é quem vai definir para qual fila as mensagens recebidas vão. O RabbitMQ possui quatro tipos de Exchange, são elas direct, fanout, headers e topic e uma Exchange coringa chamada AMQP default.
- **Filas (Queue)**
 - Uma fila basicamente é a onde as mensagens enviadas vão ser armazenadas para serem consumidas por outras aplicações. No RabbitMQ as mensagens não podem ser ordenadas se estamos falando de uma fila, a primeira que entra é a primeira a sair.
- **Mensagem**
 - Uma mensagem é nada mais que os dados enviados pela sua aplicação. Fazendo uma analogia a mensagem é como uma requisição HTTP.

Conceitos

- **Produtor**

- O Produtor é aquele que produz as mensagens e/ou tarefas a serem inseridas nas estruturas de fila. Pode ser que ele poste as mensagens ou pode depender de outra estrutura.

- **Consumidor**

- O consumidor é aquele que recebe as mensagens diretamente das filas às quais está conectado.



Aplicação

Dentre as aplicabilidades do RabbitMQ estão possibilitar a garantia de assincronicidade entre aplicações, diminuir o acoplamento entre aplicações, distribuir alertas, controlar filas de trabalhos em background.



Exemplos de aplicação

Hello World

Exemplo simples de aplicação

Filas de trabalho

Distribuindo tarefas entre os trabalhadores (o padrão de consumidores concorrentes)

Publicar/Assinar

Enviando mensagens para muitos consumidores de uma só vez

Roteamento

Recebendo mensagens seletivamente

Tópicos

Recebendo mensagens com base em um padrão (tópicos)

RPC

Exemplo de padrão de solicitação/resposta

Confirmação do editor

Publicação confiável com confirmação

Hello World

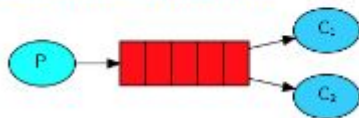
The simplest thing that does
something



É uma estrutura simples com produtor
consumidor em que o primeiro envia a
mensagem para uma fila e o segundo recebe
as mensagens vindo dessa fila .

Filas de trabalho

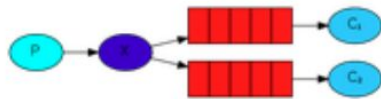
Distributing tasks among
workers (the competing
consumers pattern)



Aqui temos o conceito de tarefa encapsulada. Essas tarefas são colocadas dentro de mensagens, que por sua vez são colocadas dentro da mensageria. Assim é possível ter vários consumidores, distribuindo a execução das tarefas entre eles. Cada consumidor recebe apenas uma tarefa, que não se repete entre os demais

Publish/Subscribe

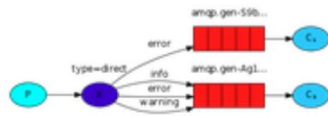
Sending messages to many consumers at once



Esse cenário tem uma proposta diferente do anterior. Enquanto o anterior tinha a proposta de que cada consumidor possuísse unicamente uma mensagem, aqui o objetivo é enviar uma mensagem a todos os consumidores.

Routing

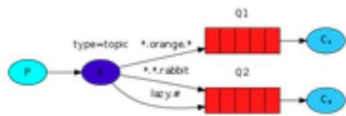
Receiving messages
selectively



Esse cenário se baseia em várias filas, as quais cada consumidor pode “escolher” qual fila de mensagens se inscrever. Diferentemente do Publish/Subscribe, aqui as filas possuem conteúdos diferentes, o que é possibilitado pelo roteamento das mensagens do produtor entre as filas.

Topics

Receiving messages based
on a pattern (topics)

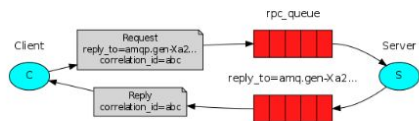


Nesse cenário existem duas filas que possuem “interesses” diferentes, mas ao contrário do Routing, nesse cenário, cada fila pode ter vários filtros de mensagens, recebendo mensagens que passam por qualquer um dos filtros.

RPC

Request/reply pattern

example




Execução de funções de um programa em hosts remotos usando a interface de filas. Semelhante às filas de trabalho, mas aqui não são tarefas e sim execução de funções.



Publish confirms

Nesse cenário, o broker confirma ao produtor que a mensagem foi entregue com sucesso, conferindo assim a uma maior confiabilidade.





Bibliografía

- ◀ <https://www.rabbitmq.com/>
 - ◀ <https://diogobemfica.com.br/Introducao-ao-rabbitmq#:~:text=O%20AMQP%20%C3%A9%20um%20acronimo,%C3%A9%20para%20aplica%C3%A7%C3%B5es%20web%20convencionais.>
 - ◀ <https://blog.cedrotech.com/rabbitmq-o-que-e-e-como-utilizar>
 - ◀ <https://www.devmedia.com.br/introducao-ao-amqp-com-rabbitmq/33036>
 - ◀ <https://docs.microsoft.com/pt-br/azure/service-bus-messaging/service-bus-amqp-protocol-guide>
 - ◀ <https://github.com/rabbitmq/rabbitmq-tutorials/tree/master/javascript-nodejs>
- 