SISTEMAS DISTRIBUIDOS

MIDDLEWARE ORIENTADO A MENSAJES (MOM)

Comunicación orientada a mensajes

EQUIPO #1

PROGRAMA

Middleware Orientado a Mensajes (MOM)

- Operación básica MOM y funcionamiento.
- Paradigmas de comunicación
  - Punto a punto
  - Publicador/suscriptor.
    - Centralizada con intermediarios.
    - Centralizada multi-intermediarios
    - Descentralizada multi-intermediarios.
    - Descentralizada sin intermediarios.
- Servicio publicador/suscriptor (Pub-Sub Service), broker.
- Protocolo AMQP, Tecnología MSMQ, API JMS, especificación DDS.
- Servicio RabbitMQ. API o librerías de MOM en Javascrip (amqp.node).

MOM

Operación básica y funcionamiento

Permite que los **componentes de las aplicaciones que utilizan diferentes protocolos de mensajería se comuniquen para intercambiar mensajes**. Además de **traducir** (o transformar) mensajes entre aplicaciones, **gestiona el enrutamiento de los mensajes** para que siempre lleguen a los componentes adecuados en el orden correcto. Entre algunos ejemplos se incluyen las colas de mensajes e intermediarios de mensajes.

MOM

Tipos

Dentro de este tipo de middleware existen dos tipos de paradigmas de comunicación:

- Punto a punto
- Publicador/suscriptor.
  - Centralizada con intermediarios.
  - Centralizada multi-intermediarios
  - Descentralizada multi-intermediarios.
  - Descentralizada sin intermediarios.

PUNTO A PUNTO

Este paradigma cuenta con solo **dos nodos en la comunicación**, uno que **envía** el mensaje, y otro que lo **recibe**.

Este modelo **asegura la llegada del mensaje**, ya que si el receptor no está disponible para aceptar el mensaje o atenderlo, **se le remite igualmente**, **enviándose a una cola para que luego pueda ser recibido cuando se conecte**.

PUBLICADOR/SUSCRIPTOR

En este tipo de paradigma **pueden existir diferentes nodos de comunicación que pueden ejercer los papeles de publicador, suscriptor, o ambos al mismo tiempo**.

La información es generada o consumida en un mensaje, evento o topic (dependiendo de la tecnología utilizada) que **sirve como nexo de unión entre publicadores y suscriptores**.

Este modelo de comunicación **ofrece una potente abstracción para multidifusión y comunicación en grupo**. Los publicadores pueden difundir una determinada información a un grupo de suscriptores que están a la escucha de esta difusión de mensajes.

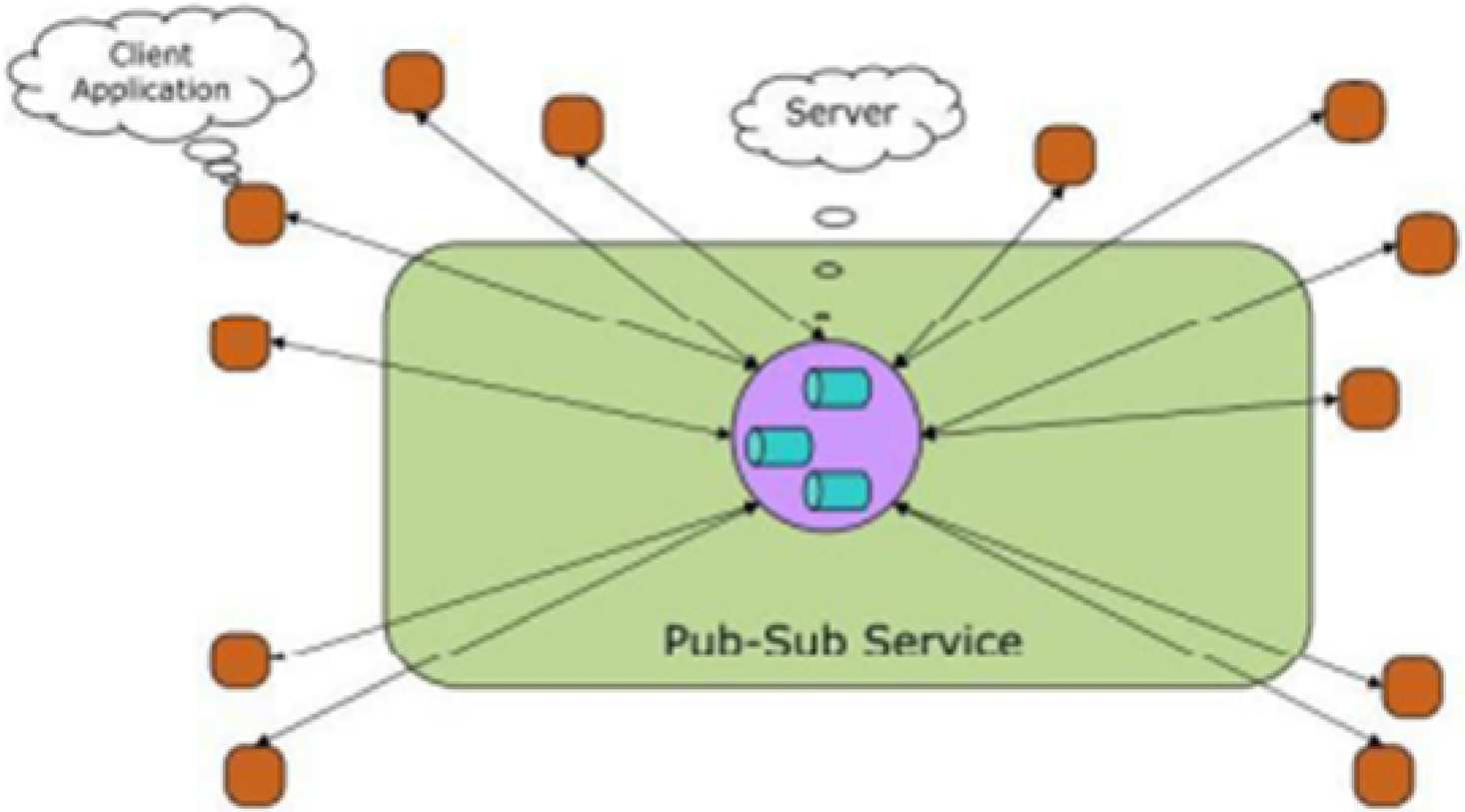
PUBLICADOR/SUSCRIPTOR

Centralizada con intermediarios

- También conocido como cliente-servidor.
- Se divide en dos partes, el cliente y el servidor.
- El cliente es la parte de la aplicación que se ejecuta en la máquina del usuario.
- El servidor es la parte que se ejecuta en una maquina remota.
- El servidor central sirve como punto de enlace entre todas las entidades.
- Todo el tráfico para por el servidor.
- Los intermediarios son componentes que se ubican entre el cliente y el servidor para facilitar la comunicación (Balanceador de carga, servidor cache).

PUBLICADOR/SUSCRIPTOR

Centralizada con intermediarios



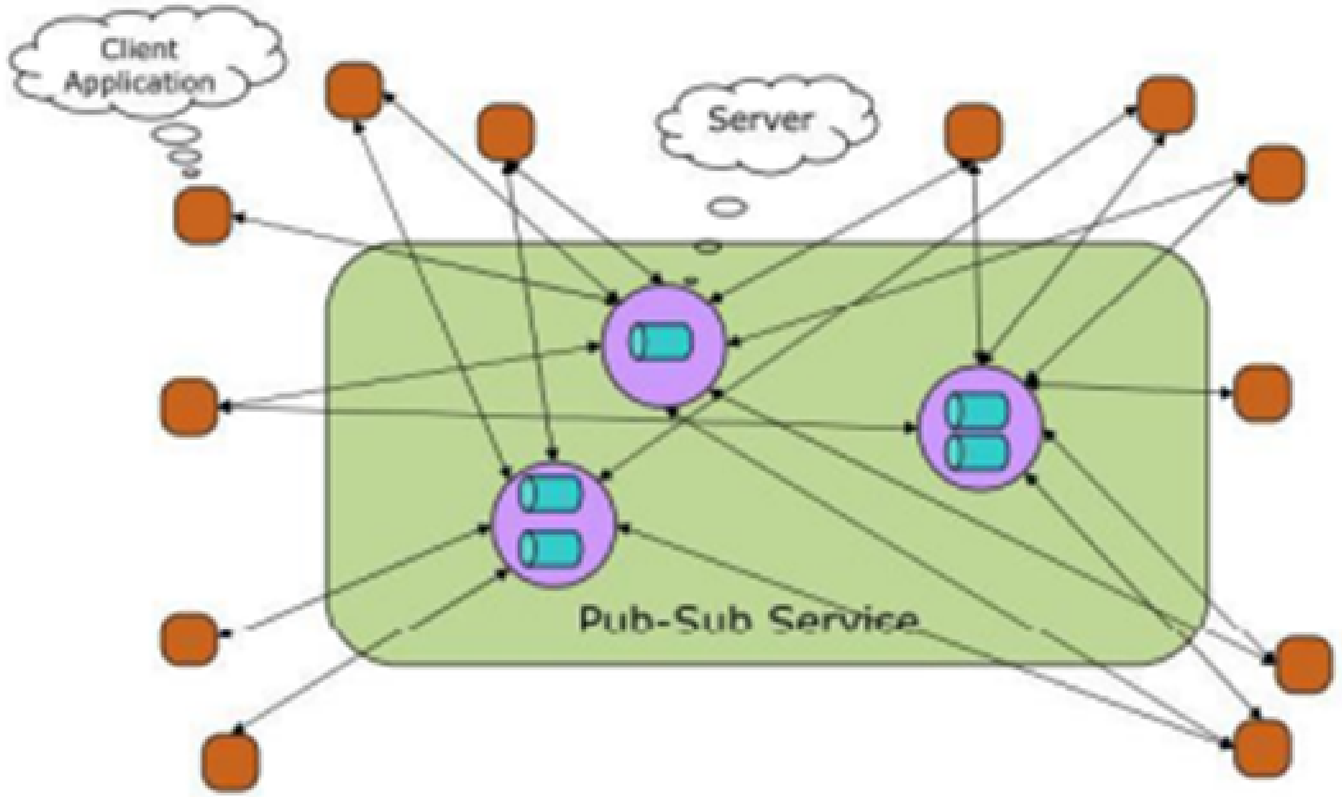
PUBLICADOR/SUSCRIPTOR

Centralizada multi-intermediarios:

- Es una variante del modelo centralizado con intermediario el cual usa múltiples intermediarios, para mejorar la escalabilidad, disponibilidad y seguridad de la aplicación.
- En este modelo los intermediarios se dividen en diferentes capas.
- Cada capa tiene una función especifica. (Autenticación, Autorización, Persistencia de datos).
- Cada cola se encuentra en servidores distintos, interconectados entre ellos.

PUBLICADOR/SUSCRIPTOR

Centralizada multi-intermediarios:



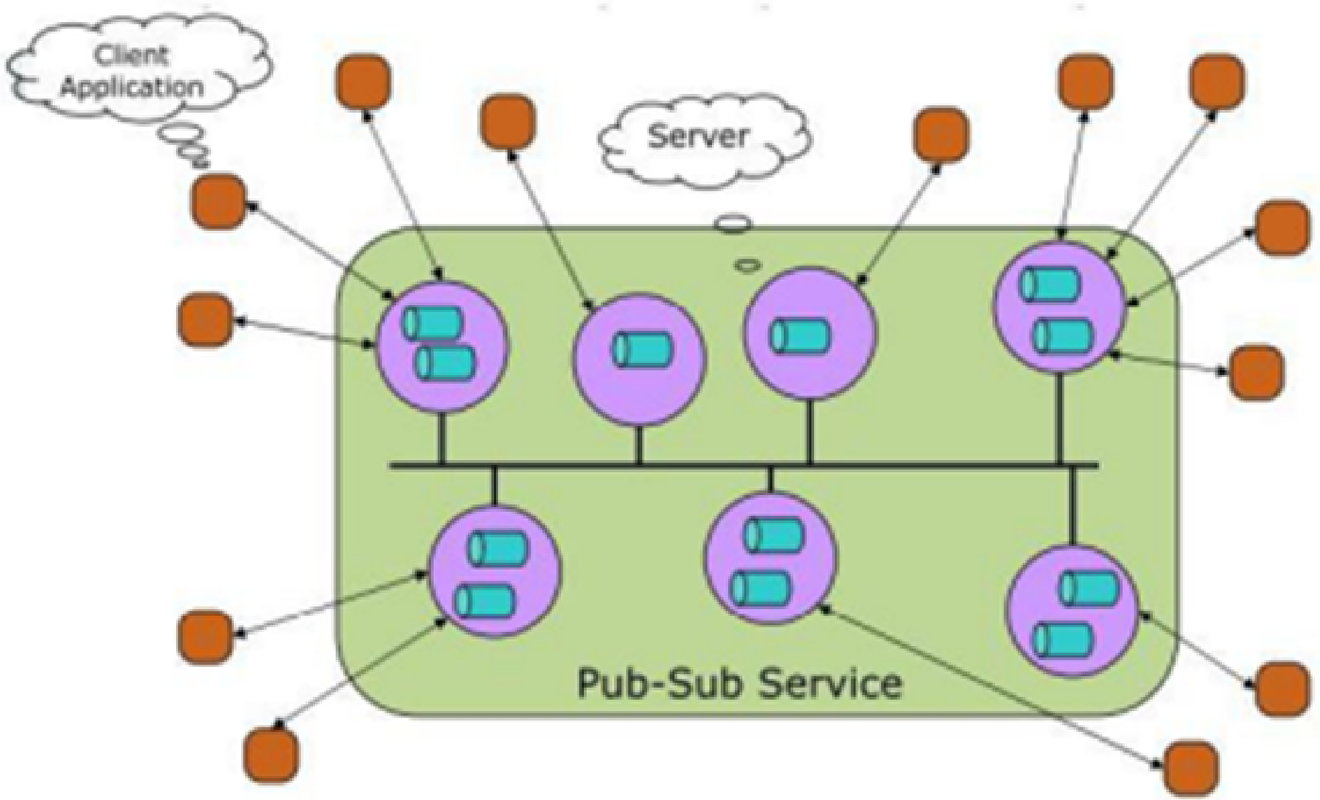
PUBLICADOR/SUSCRIPTOR

Descentralizada multi-intermediarios

- Se caracteriza por tener múltiples nodos que interactúan entre sí en igualdad de condiciones, sin que haya uno que tenga el control de toda la aplicación.
- Cada uno puede tener sus propios intermediarios.
- La comunicación se da de forma distribuida y en red.
- Aquí el servicio publicador/suscriptor distribuye los mensajes internamente entre los puntos de acceso.

PUBLICADOR/SUSCRIPTOR

Descentralizada multi-intermediarios



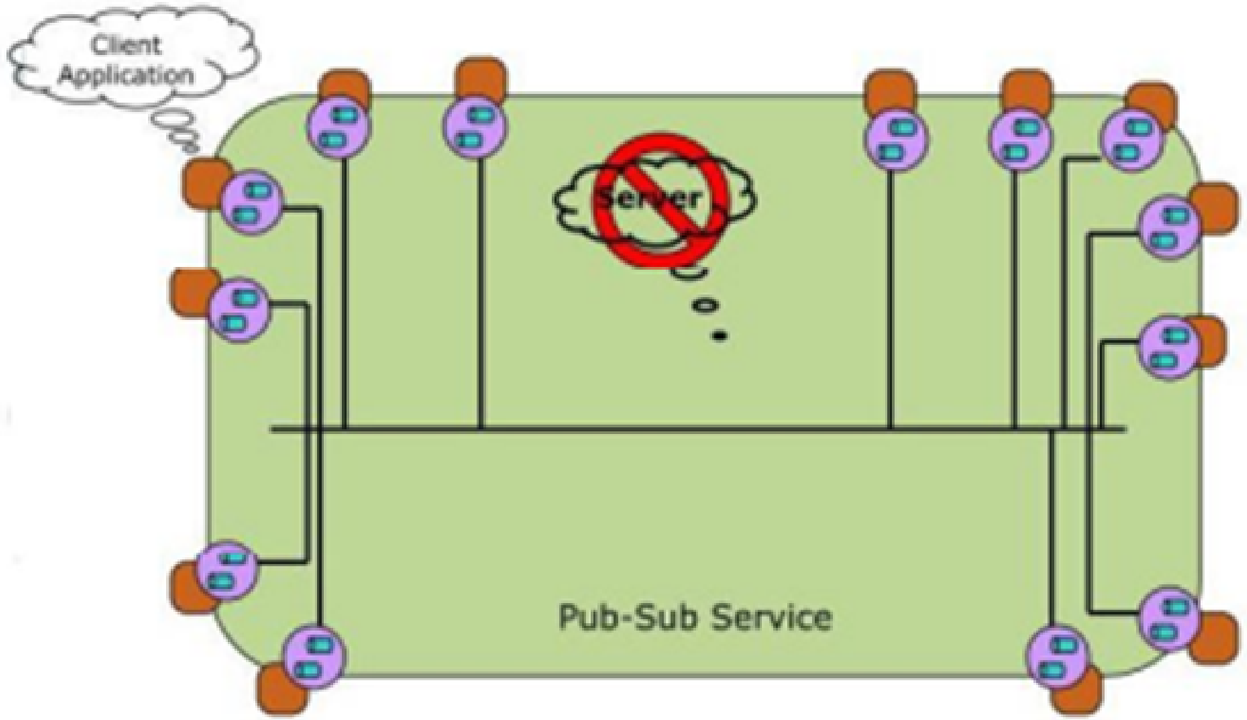
PUBLICADOR/SUSCRIPTOR

Descentralizada sin intermediarios

- Cada nodo se comunica directamente con otro nodo sin la necesidad de un intermediario centralizado.
- No existen servidores.
- Cada nodo es responsable de la autenticación y validación de los datos que recibe, así como la distribución de la información a los demás nodos en la red.

PUBLICADOR/SUSCRIPTOR

Descentralizada sin intermediarios



PUB-SUB SERVICE, BROKER

Service Broker ayuda a los programadores a crear **aplicaciones asincrónicas de acoplamiento flexible** en las que los componentes independientes funcionan conjuntamente para llevar a cabo una tarea. Estos componentes de aplicación intercambian mensajes que contienen la información necesaria para completar la tarea.

AMQP, MSMQ, API JMS, DDS.

Protocolo AMQP

AMQP (Protocolo Avanzado de Espera de Mensajes) es **un protocolo estándar abierto para la comunicación mediante un middleware orientado a mensajería**, donde su objetivo principal es **alcanzar la interoperatividad entre diferentes servicios**.

A diferencia de otras tecnologías, AMQP no solamente define un API (Interfaz de Programación de Aplicaciones), sino también es un protocolo a nivel de cable, es decir, **define el formato de los datos que son enviados a través de la red** como un flujo de octetos. Por tanto, cualquier aplicación puede crear e interpretar mensajes conforme a este formato de datos, independientemente del lenguaje que se utilice.

AMQP, MSMQ, API JMS, DDS.

Tecnología MSMQ

MSMQ (Cola de Mensajes de Microsoft) **es una tecnología** propietaria de Microsoft que **permite que aplicaciones que están ejecutándose en diferentes lugares puedan comunicarse a través de redes heterogéneas y sistemas que pueden estar temporalmente inactivos**. Las aplicaciones envían mensajes a las colas para que puedan ser leídos por otras aplicaciones.

AMQP, MSMQ, API JMS, DDS.

API JMS

JMS (Servicio de Mensajería de Java) es una API de Java diseñada por Sun y otras compañías en 1998, y cuya última versión (2.1) data del 2002. **Permite a las aplicaciones crear, enviar, recibir y leer mensajes, definiendo un conjunto de interfaces y su semántica asociada, posibilitando a los programas escritos en Java comunicarse con otros mediante mensajes**.

AMQP, MSMQ, API JMS, DDS.

Especificación DDS

DDS (Servicio de Distribución de Datos) **es una especificación** adoptada por la OMG para el **intercambio de datos en sistemas distribuidos en tiempo real**, basándose en el **paradigma de comunicación publicador/suscriptor**.

La OMG es un consorcio internacional sin ánimo de lucro formado por organizaciones que desarrollan estándares de tecnologías orientadas a objetos que se encuentran ampliamente extendidas en la actualidad dentro de la industria informática.

RABBITMQ, API, JAVASCRIP

Servicio RabbitMQ, API o librerías en Javascrip (amqp.node)

RabbitMQ es un broker de mensajería de código abierto, distribuido y escalable, que sirve como intermediario para la comunicación eficiente entre productores y consumidores.

En una forma simplificada, en RabbitMQ **se definen colas que van a almacenar los mensajes que envían los productores hasta que las aplicaciones consumidoras obtienen el mensaje y lo procesan**. Esto nos permite diseñar e implementar sistemas distribuidos, en los cuales un sistema se divide en módulos independientes que se comunican entre sí a través de mensajes.

REFERENCIAS

URL

- https://es.theastrologypage.com/message-oriented-middleware
- https://www.ibm.com/mx-es/topics/middleware
- https://biblus.us.es/bibing/proyectos/abreproy/70308/fichero/3.+ESTADO+DEL+ARTE+2.pdf
- https://learn.microsoft.com/es-es/sql/database-engine/service-broker/what-does-service-broker-do?view=sql-server-ver16
- https://www.gotoiot.com/pages/articles/amqp_intro/index.html
- https://docs.oracle.com/javaee/5/tutorial/doc/bnceh.html
- https://www.pragma.com.co/academia/lecciones/conozcamos-sobre-rabbitmq-sus-componentes-y-beneficios#:~:text=RabbitMQ%20es%20un%20broker%20de,eficiente%20entre%20productores%20y%20consumidores.