Universidad EAFIT
ST0263: Tópicos Especiales en Telemática
Proyecto 1: Diseño e Implementación de un Middleware que Implemente un Servicio de Mensajería Asincrónica entre Aplicaciones

2025-1

Fecha de entrega: Abril 13 de 2025

1. Introducción

Un middleware se entiende como un componente de software que implementa una funcionalidad compleja y ABSTRAE a las aplicaciones usuarias de la complejidad y detalles internos del sistemas.

ver:

- Middleware - Wikipedia, la enciclopedia libre
- Qué es middleware: definición y ejemplos | Microsoft Azure
- ¿Qué es el middleware? (redhat.com)
- https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjzpPbT8eLuAhWNjlkKHcEtAzUQwqsBMAx6BAgiEAM&url=https%3A%2F%2Fes.coursera.org%2Flecture%2Faplicaciones-web%2Fvideo-1-que-es-el-middleware-gf7iN&usg=AOvVaw2bZsRaZsVXw2X5y8s6PQ7X
- ¿QUÉ ES MIDDLEWARE? HISTORIA, APLICACIONES, Y MÁS (tecnoinformatic.com)

El objetivo de este proyecto 1 es diseñar e implementar un MIDDLEWARE ORIENTADO A MENSAJES (MOM) que permita a un conjunto de CLIENTES enviar y recibir mensajes de datos. Esto permitirá a los alumnos evidenciar, conocer y aplicar, muchas de las características subyacentes a los sistemas distribuidos (ej: heterogeneidad, transparencia, seguridad, escalabilidad, entre otros) que deben implementar las aplicaciones o los subsistemas base (sistema operativo, middlewares, frameworks, apis, etc). En este caso, dicha complejidad y características del sistema distribuido serán diseñadas e implementadas en un MOM, de tal manera que para las aplicaciones usuarias (CLIENTES) sea transparente y seguro su uso.

El MOM debe implementar las siguientes funcionalidades y servicios:

1. El MOM debe ser diseñado e implementado con un acercamiento en clúster, es decir n nodos MOM implementándole el sistema.
2. Conexión y desconexión al servidor (para el envío o recepción de mensajes, en forma permanente - con estado - o en sin conexión constante - sin estado -
3. Ciclo de vida de tópicos (los canales tienen nombres únicos):
    a. Crear un tópico
    b. Borrar un tópico
    c. Listar los tópicos
4. Ciclo de vida de colas (las colas tienen nombres únicos):
    a. Crear una cola

    b. Borrar una cola
    c. Listar las colas
5. Envío de un mensaje a un tópico
6. Envío de un mensaje a una cola
7. Recepción de un Mensaje de un tópico
8. Recepción de un Mensaje de una cola

2. Objetivos

- **Objetivo General:** Diseñar e implementar un middleware MOM en cluster que permina a un conjunto de aplicaciones comunicarse por colas o tópicos.

- **Objetivos Específicos:**

    - Diseñar e implementar un API entre Cliente y MOM para la gestión de colas y tópicos, así como el envio y recepción de mensajes.
    - Implementar RPC basados en API REST y gRPC entre Cliente y MOM, y MOM a MOM.
    - Diseñar e implementar un mecanismo de particionamiento y replicación en el clúster MOM.
    - Diseñar e implementar unas aplicaciones cliente sencillas para probar las funcionalidades del MOM.

3. Requerimientos del Proyecto.

- La conexión / desconexión, debe ser con usuarios autenticados
- Solo puede borrar canales o colas de los usuarios que los crearon.
    - ¿Qué pasaría con los mensajes existentes en un canal o una cola?
- El envío y recepción de mensajes debe identificar los usuarios.
- Todos estos servicios deben ser expuestos como un API REST hacia los Clientes.
- El transporte de los mensajes debería ser encriptada así como el servicio de autenticación (opcional)
- Definir el mecanismo de recepción de mensajes en modo pull o push/eventos
- ¿Qué mecanismos de persistencia de datos debería tener este middleware?
- ¿Qué implementaría en tolerancia a fallos?
    - En servidor? tener varios?
    - En mensajes?
- Definir la arquitectura más adecuada.
- Debe aplicar los conceptos vistos sobre Replicación, y/o Particionamiento de las colas y tópicos, inspirarse en sistemas de colas/tópicos como Apache Kafka.
- Desde el punto de vista del sistema distribuido y teniendo en cuenta el modelo/middleware a diseñar e implementar, considere:

    - Interacción sincrónica/asincrónica.
    - Interacción simétrica/asimétrica.
    - Manejo o no de sesión y estado.
    - Modelo de manejo de fallos.
    - Modelo de seguridad.
    - Niveles de transparencia.
    - Multiusuario
    - Particionamiento
    - Replicación
- Consideraciones de escalabilidad, extensibilidad y otros criterios de diseño de la arquitectura de la aplicación.

Para probar este middleware realizará una aplicación sencilla la cual elegirá entre una de estas dos:

La aplicación ejemplo en el github de la materia para probar las funcionalidades de RabbitMQ.

Resumen de Requerimientos:

1. Realice el análisis, diseño (arquitectura y detallado) e implementación de un Middleware de Mensajería.
2. Realice una aplicación ejemplo que utilice dicho middleware en una app como la presente en el github de la materia.
3. Realice la implementación en el lenguaje de programación de su preferencia. No se distraiga con la interfaz gráfica.
4. utilizar API REST y gRPC entre los clientes y el MOM, si va a comunicar entre servidores la conexión debe ser por gRPC. (usuario a MOM con API REST y MOM-MOM con gRPC)
5. Todas las especificaciones de análisis, diseño y detalles de implementación deben ser documentadas.
6. Realizar la gestión del código fuente en GITHUB, donde muestre el aporte de cada uno de los integrantes y sea entregado al profesor.
7. Realizar el despliegue del Middleware y la Aplicación en máquinas virtuales en Amazon AWS Academy, cuya IP y archivo de credencial .pem sea compartido con el profesor para su verificación.
8. Realizar en no más de 30 minutos, una exposición del proyecto, donde sintetice: Requerimientos (análisis), diseño, implementación Y QUE NO SE ALCANZÓ A REALIZAR, tanto del middleware como de la app. (en el momento de sustentación del proyecto1)

4. Implementación y entragables

4.1. Fases de Desarrollo

1. Definición de los requerimientos funcionales y no funcionales del sistema.
2. Diseño de la arquitectura y definición de APIs.
3. Implementación del cliente y APIs.
4. Implementación del particionamiento y replicación del MOM, y requerimientos de tolerancia a fallos.
5. Pruebas e integración de todos los componentes.
6. Evaluación del rendimiento y documentación del sistema.

4.2. Entregables

- Código fuente de la implementación.
- Documentación técnica detallada de la arquitectura y componentes.
- Presentación final del proyecto.

5. Evaluación

- Diseño e implementación de las APIs Cliente-MOM y MOM-MOM tanto en APIREST como en gRPC (30%)
- Diseño e Implementación del cluster MOM con particionamiento, replicación y tolerancia a fallos. (40%)
- Pruebas, integración y análisis del funcionamiento completo de la solución, aplicaciones ejemplo (15%)
- Documentación y presentación final (15%)

Fecha de entrega:

- Abril 13 de 2025, enviando un email por Interactiva virtual (buzón de entrega), donde nuevamente anuncie el repositorio github, la IP y adjuntar las credenciales de acceso en AWS, integrantes,etc.
    - El repositorio github debe tener claramente identificados a los integrantes + emails
    - El repositorio github debe contener todo el código fuente y documentación (archivos .md y en especial el README.md template compartido por el profesor), donde de detalles de requerimientos, análisis, diseño, implementación y uso/aplicación.