

**Pontificia Universidad Católica Madre y Maestra**

**Facultad de Ciencias de La Ingeniería**

**Escuela de Ingeniería en Computación y Telecomunicaciones**

**Estudiantes**

2004-1265 Roselin Sosa

2014-0795 Jean Lemoine

**Práctica**

Implementando sistemas basados en Broker de Mensajería

**Asignatura**

Programación Web Avanzada (ISC-517)

**Profesor**

Ing. Carlos Camacho

**Fecha de Entrega**

7 de Julio del 2020

Introducción

Esta práctica consiste en Implementar Broker de Mensajería, crear aplicación basadas en intercambio de mensajes asíncronos, implementar colas de suscripción y publicación, también el uso de Docker Compose. Las tecnologías o herramientas para usar son las siguientes:

* Gradle es una herramienta de automatización de compilación de código abierto que está diseñada para ser lo suficientemente flexible como para construir casi cualquier tipo de software.
* HTML o HyperText Markup Language es un lenguaje de marcado que se utiliza para el desarrollo de páginas web, por el cual consiste en una serie de códigos escritos en un archivo de texto por el desarrollador del sitio, ya sea utilizando etiquetas correctamente para crear el sitio web, y permite al usuario crear y estructurar secciones, párrafos encabezados, enlaces y citas en bloque para aplicaciones web.
* Freemarker es un motor de plantillas o una biblioteca de Java del lado del servidor para entornos web y autónomos, y consiste en aportar plantillas a su flujo de trabajo de desarrollo, especialmente con HTML que se puede mostrar en los distintos navegadores y también funciona como prototipos estáticos, lo cual permite una colaboración más concreta en las herramientas de desarrollo.
* H2: es un manejador de base de datos de código abierto, puede integrarse en aplicaciones del lenguaje Java o ejecutarse en modo cliente-servidor.
* JPA o Java Persistence API es una colección de clases y métodos para almacenar persistentemente las grandes cantidades de datos en una base de datos. JPA permite una fácil interacción con la instancia de la base de datos para reducir la carga de escribir códigos para la gestión de objetos relacionales.
* Docker: es un software de código abierto que permite facilitar la creación, implementación y ejecución de aplicaciones mediante el uso de contenedores de por el cual permiten a un desarrollador empaquetar una aplicación con todas las librerías y dependencias que necesita.
* Docker Hub: es un servicio proporcionado por Docker para encontrar y compartir imágenes de contenedores con su equipo, clientes o la comunidad de Docker en general.

Desarrollo

Link del repositorio:

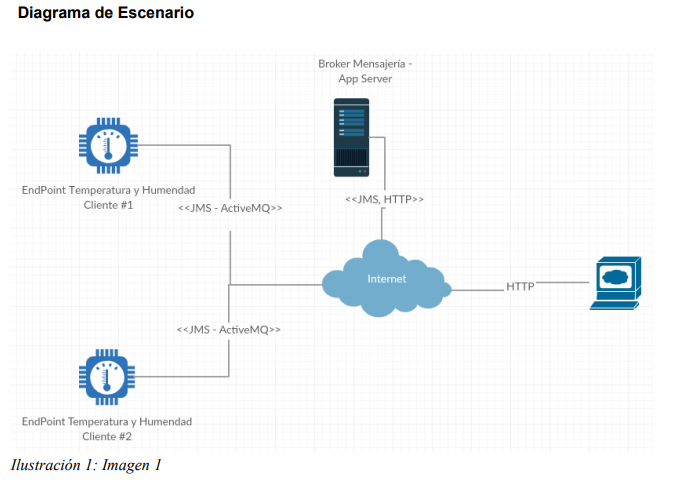
<https://github.com/RSDarcy/Practiva-JMS-WebAvanzada.git>

Una empresa especialista en el desarrollo de sensores electrónicos para la toma de temperatura y humedad, requiere de nuestros servicios para comunicar de forma eficiente cada dispositivo final (EndPoint) con el servidor y dicha información pueda ser visualizada en una aplicación web. Cada Endpoint implementa una trama JSON con la siguiente estructura:

| **Campo** | **Tipo** |
| --- | --- |
| fechaGeneración | String formato DD/MM/YYYY HH:mm:ss |
| IdDispositivo | Int |
| temperatura | Number |
| humedad | Number |

Para enviar la información entre los dispositivos se estará utilizando un servidor que implemente el estándar JMS y algún protocolo orientado a mensaje (OpenWire, MQTT, AMQP) utilizando una cola del tipo de publicación/subscripción, llamada notificacion\_sensores. Cada dispositivo procesa información cada minuto y la envía a la cola de distribución, el servidor toma el mensaje y lo persiste en la base de datos. Una aplicación debe graficar en tiempo real la información que es recibida. Dado el siguiente planteamiento y escenario realice:

1. Desarrolle un cliente que simule los Endpoint, que generen de forma aleatoria valores de temperatura y humedad y envíe la trama en el formato JSON definido a la cola indicada. Debe instanciar dos clientes para la prueba.
2. Implemente un servidor que soporte JMS y algún protocolo orientado a mensaje que inicialice la cola de distribución del tipo publicación/subscripción. Puede ser una aplicación con el protocolo de forma embebida o una servidor diseñado para esos fines (Apache ActiveMQ, RabbitMQ, Mosquitto, entre otros).
3. Realicen una aplicación web que visualice los datos procesados mediantes gráficos del tipo linea en tiempo real, donde se visualice la información de la temperatura y la humedad vs el tiempo (separar los gráficos), para cada uno de los sensores conectados. La comunicación entre la aplicación web y cliente puede ser utilizando el protocolo HTTP o WebSocket.
4. El escenario mostrado en la imagen #1, debe estar implementado utilizando imágenes Docker y el arranque vía Docker Compose.



Conclusión

En esta práctica se cumplieron con la mayoría de los objetivos propuestos. Nuestro mayor reto fue el requerimiento de lograr que la práctica arrancara con docker compose y lograr realizar las gráficas en tiempo real.

Nos enfrentamos a varios errores de configuración, los cuales entendemos fueron influenciados por la falta de experiencia, sin embargo logramos conseguir la mayor parte de los objetivos así que nos sentimos satisfechos.

Bibliografía

* Gradle Inc.. (2020). What is Gradle?. 2020, de Gradle Sitio web: <https://docs.gradle.org/current/userguide/what_is_gradle.html>
* Domantas G.. (2019). What is HTML? The Basics of Hypertext Markup Language Explained. 25 de Noviembre del 2019, de Hostinguer Sitio web: <https://www.hostinger.com/tutorials/what-is-html>
* Tutorials Point. (2020). H2 Database - Introduction. 2020, de Tutorials Point Sitio web: <https://www.tutorialspoint.com/h2_database/h2_database_introduction.htm>