DESARROLLO DE SOFTWARE BASADO EN COMPONENTES

ENUNCIADO DE PRÁCTICA 2

Fecha de entrega de enunciado: 2016 05 07

Fecha límite de entrega de implementación: 2016 05 21 a las 23:59

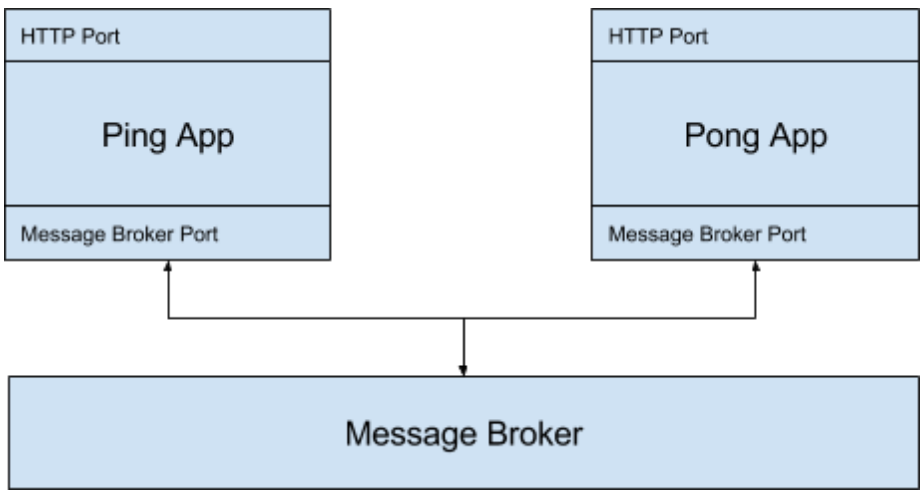
Modo de entrega: deben hacer un video en el cual muestren tanto el código como el software funcionando y una explicación hablada de estos dos frentes. Deben subir el código a un repo git y el video a un servidor de videos bien sea youtube.com vimeo.com o el que ustedes prefieran.

MESSAGE DRIVEN PING PONG

Debe implementar dos aplicaciones muy simples. Una aplicación Ping y una aplicación Pong. Un cliente dará un comando a la aplicación Ping para que emita un mensaje PING_MESSAGE. Dicho comando lo recibirá la aplicación Ping por HTTP. Al recibir este comando, la aplicación Ping emitirá el mensaje PING_MESSAGE a un Message Broker. La aplicación Pong recibirá mensajes PING_MESSAGE, simulará un retraso de 2 segundos de cómputo y emitirá un mensaje PONG_MESSAGE al Message Broker. La aplicación Ping al recibir un mensaje PONG_MESSAGE completará el request HTTP que ocasionó esa respuesta por parte de la aplicación Pong y dará respuesta al cliente HTTP.

Finalmente, la aplicacion PONG debe tener la capacidad de responder consultas por HTTP acerca de cuántos mensajes PING_MESSAGE ha recibido y contestado.

La siguiente figura muestra los diferentes componentes de alto nivel de la implementación:



Observaciones importantes

- Así como en la práctica #1 la tecnología en la que implemente la práctica es la que usted prefiera que le ayude a cumplir los objetivos en el tiempo que tiene para implementarla.
- El Message Broker puede ser el que usted desee. Puede ser cualquier servidor AMQP (RabbitMQ, ActiveMQ, ZeroMQ, JBoss Fuse, Apache Kafka etc).
- La comunicación entre la aplicación Ping y la aplicación Pong debe ser asíncrona.
- Para cumplir con la restricción de comunicación asíncrona entre las dos aplicaciones debe utilizar algo que le ayude a hacer IO asíncrono para el manejo del ciclo request-response HTTP. Esto significa que el mismo hilo que atiende el HTTP request no debe ser el mismo hilo que ira a estar esperando por el mensaje PONG_MESSAGE de vuelta. Ya existen en las tecnologías de mayor uso librerías que le ayudan a hacer esto (incluso tecnologías que van en camino a la obsolescencia como Java implementan esto http://docs.oracle.com/javaee/7/api/javax/ws/rs/container/AsyncResponse.html)
- No es necesario que la implementación del puerto HTTP para la aplicación PONG presente los recursos en una GUI. Es suficiente con que retorne un JSON con las estadísticas de uso de la aplicación y que se pueda ver por algún cliente HTTP (curl, o un web browser si lo prefiere).

Puntos adicionales interesantes

- Si entrega pruebas de carga que identifiquen el throughput máximo al que es capaz de ser sometida la aplicación PING se le reconocerá en la calificación de esta práctica o en el parcial #2 que aún falta por presentar. Si decide hacer esto debe entregar el código de las pruebas de carga y sustentarlo en el video de entrega. Para hacer pruebas de carga puede utilizar herramientas como Gatling, JMeter o relacionadas.
- Si implementa el patrón de Presión Contratia (Back Pressure) en la aplicación PING y sustenta tanto su implementación como su uso se le reconocerá bien en la calificación de esta implementación o en el parcial #2. Si le interesa conocer este patrón y como ayuda para esta implementación puede referirse a la siguiente explicación de Martin Thompson.

http://mechanical-sympathy.blogspot.com.co/2012/05/apply-back-pressure-when-overloaded.html