Entrega Final 5
Grupo Los Sixteen

Resultados cuantitativos y cualitativos de la experimentación

Escenario de calidad de escalabilidad

Este escenario fue el más complejo de todos. Consistía en poder medir los mensajes en el broker y poder escalar la cantidad de consumidores dependiendo de la cantidad de mensajes pendientes por procesar en el tópico “comandos-producto”. Para ello fue necesario al montar dicho escenario en k8, realizar una investigación adicional con ayuda de Prometheus, Prometheus adapter + Grafana para visualizar dicha información. Adicionalmente fue necesario generar un nuevo servicio para monitorear el broker y visualizar de alguna manera los mensajes pendientes en dicho tópico y que k8 pudiera interpretar dicha información para aplicar una regla de escalabilidad horizontal (HPA).

Efectivamente se pudo visualizar como dicha escalabilidad ocurre, incluyendo en el consumidor “admin-productos” un pequeño retraso (DELAY) aplicado deliberadamente para poder llenar el broker con el tópico “comandos-productos” con los mensajes necesarios para escalar dicho pod (El criterio que se utilizo fue de 8 segundos en el delay, y una cantidad de más 5 mensajes para que el pod escalara.)

A continuación algunas gráficas. Ver experimentación

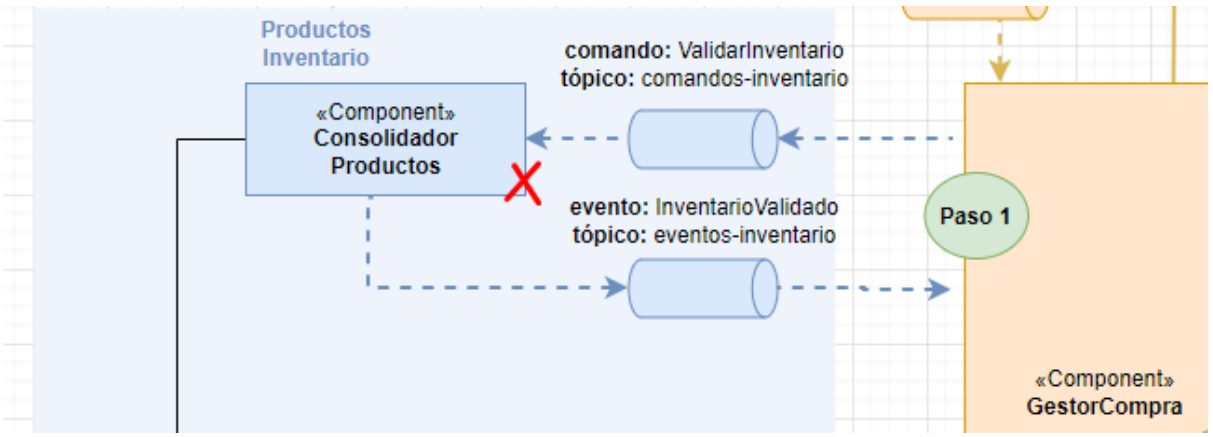
Escenario de calidad de desempeño

Para este escenario, se implementó la táctica priorización de eventos en el componente ***GestorCompra***. Allí mediante el BFF, el usuario envía unos parámetros definidos.
El componente fue configurado para procesar primeramente los eventos marcados como “Domicilio: ***True***” y después los marcados como “Domicilio: ***False***”

Evidencia del ejercicio mediante BFF:

Escenario de calidad de disponibilidad

Para este escenario se validó haciendo que el servicio “consolidador” se mantuviera fuera de operación por un tiempo. Sin embargo, el usuario no percibe errores aún cuando el servicio esté caído (es decir, se enmascaró la falla), pues los mensajes siguen llegando al broker sobre el tópico “comandos-inventario”, y cuando el servicio consolidador se encuentre nuevamente disponible (ya sea que el grupo responsable haya encontrada y solucionado dicha falla), el servicio consolidador volverá a consumir los servicios pendientes del broker. En ese sentido, tener un sistema que funciones de manera asíncrono (con un sistema manejado por eventos), junto con una arquitectura de microservicios es posible desacoplar los diferentes contextos manteniendo una alta disponibilidad aún si unos de los microservicios falla.



Conclusiones

- ## Sobre escenario de calidad de escalabilidad

  En el desarrollo del escenario, se pudo evidenciar satisfactoriamente como el sistema escala en relación al incremento de comandos en cola.
  Gracias a la implementación de monitoreo en la cola de comandos, esto nos funcionó para tomar la decisión de escalar el sistema.

- ## Sobre escenario de calidad de desempeño

  Con la implementación de la táctica de priorización de eventos, el sistema logró clasificar los eventos generados por el usuario para así mismo dar prioridad y atender de manera eficiente.

- ## Sobre escenario de calidad de disponibilidad

  Cuando el microservicio fallaba, se logró probar como esta se enmascaraba y el proceso de validación se mantenía hasta que el microservicio se recuperaba en máximo 1 minuto. De cara al usuario, el sistema siempre estuvo disponible en esta ventana de tiempo.

- ## Sobre la implementación de SAGA

  La implementación de un coordinador saga como orquestador no pudo ser terminada, por lo cual se optó por experimentar y comprobar las hipótesis de los escenarios de calidad sin esta herramienta.

  El diseño planteado para el coordinador de la orquestación consiste en un módulo dentro del componente gestor compra, utilizando las herramientas y código base sugeridos en clase (aiopulsar principalmente). El proceso iniciaría con el comando generado desde el BFF y este le indicaría la SAGA iniciar el proceso respectivo.

Los inconvenientes que llevaron a que no se completara exitosamente la implementación del coordinador fueron:

- Limitantes de tiempo por razones directamente relacionadas con la maestría, lo cual exigía priorizar la asignación de este recurso a otras tareas académicas.
- Desconocimiento y poco entendimiento del proceso de implementación, a pesar de tener muy claro el concepto base del patrón orquestador SAGA, esto por contar con experiencia en desarrollos previos (aunque en otros lenguajes y con otras técnicas).
- Al encontrar fallos y blockers con aiopulsar y sus dependencias, se evidenció que el soporte y documentación de estas herramientas son muy limitados, lo cual generó retrasos en las pruebas iniciales.

Las evidencias de los avances en la implementación de saga se encuentran en el módulo sagas del microservicio gestor-compra.