



**ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO**

*Taller de patrones de arquitectura ESB*

**PRESENTADO POR:**

NICOLÁS OSORIO ARIAS

23 DE NOVIEMBRE DE 2018

BOGOTÁ D.C.

# Introducción

En ese justo momento en el que un objeto interactúa con otro, por más mínimo o insignificante que parezca, crean un sistema, con tantas posibilidades como la complejidad de sus elementos lo permita, sin embargo hay ciertas características que siempre tendrán en común sin importar cual sea su magnitud; si hablamos del sistema solar y a todo lo que refiere al mundo físico, los elementos interactúan por medio de un estado de materia con sus respectivas leyes físicas; pero cuando abstraemos esta idea a las relaciones humanas, la interacción se ejecuta por medio de comunicación con un lenguaje en específico, al mismo tiempo que las gestualidades y la corporalidad brindan un apoyo al contexto de esta comunicación.

Las sociedades son grandes sistemas de personas con diversas funcionalidades y necesidades que se comunican por medio de esta interacción humana, no obstante, con el avance tecnológico hemos llegado a construir grandes y complejos sistemas en los que no sólo necesitamos comunicarnos entre nosotros, sino también comunicar a nuestros múltiples sistemas, evidentemente esto se vuelve incontrolable cuando expandimos esta idea a un mundo empresarial, donde necesitamos que múltiples servicios que se componen de otros subsistemas, interactúen con el fin de conseguir el objetivo de la empresa.

Como vemos, realmente cualquier objeto que idealicemos puede ser tomado como un sistema, pero hay algo que sin importar la abstracción o la complejidad que tenga siempre estará presente, la existencia de un medio por el cual logran interactuar sus componentes. Es por esta razón que si queremos realmente hacer que un sistema se adapte a nuestro tiempo, tenemos que brindarle una correcta interacción de sus componentes, y mucho más si lo queremos incorporar en el mundo administrativo, donde es necesario brindar una arquitectura que permita una correcta comunicación de los servicios.

## **Arquitectura Enterprise Service Bus**

Cuando se desarrollan sistemas modernos es necesario contar con una buena comunicación entre los diferentes componentes que tenemos en nuestra empresa, para esto necesitamos una arquitectura que permita que estos se puedan comunicar, sin embargo, no podemos entrar a fijar reglas para cada uno de los subsistemas puesto que sería agregar una complejidad en aumento que con el tiempo se convertiría en algo incontrolable.

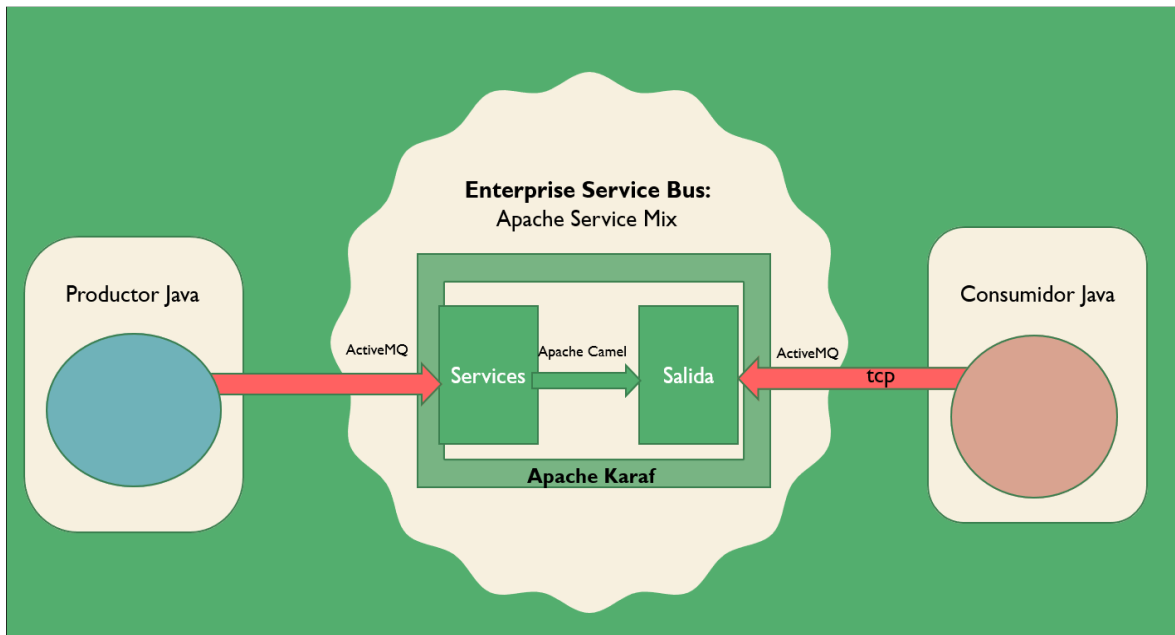
Con estos problemas empresariales es que nace la idea de una arquitectura orientada a servicios, no obstante, esto solo trae la necesidad de pensar en un canal centralizado que gestione la comunicación de los diferentes servicios, para que se puedan comunicar de una forma ordenada, controlada y sencilla. Este canal es llamado Enterprise Service Bus, es el encargado de actuar como intermediario de mensajes entre los distintos servicios que requiera el interesado.

Uno de los más populares es llamado Apache ServiceMix, el cual es un ESB open Source, que unifica características y funcionalidades de anteriores herramientas de apache, brindando una plataforma completa, que permite crear cualquier solución, ofreciendo un medio por el cual los diversos servicios que uno desee conectar no tengan un alto nivel de complejidad, ni limitaciones.

La arquitectura de Apache ServiceMix como uso de un ESB principalmente se compone de tres elementos que hacen que su funcionamiento sea sencillo y eficiente, primero, una interfaz de mensajes con la que se pueda comunicar con cualquier servicio o cliente que necesitemos, esto es elaborado con Apache ActiveMQ, el cual es un servidor de mensajería y de patrones de integración, permitiendo conectarnos desde cualquier punto que requiramos. Segundo una plataforma de administración y gestión de archivos llamada Apache Karaf, con esta herramienta tendremos un servidor corriendo e impulsado por OSGi.

Finalmente, el punto importante de cualquier ESB, la forma en la cual ServiceMix ejecuta la comunicación entre los servicios, está implementada por Apache Camel,

el cual es una herramienta que permite definir reglas de enrutamiento por medio de archivos de configuración XML, logrando así una comunicación de la data que necesitemos para cada caso, de una manera mucho más organizada.



El diagrama anterior muestra de forma resumida la arquitectura que tiene Apache ServiceMix, lo cual podremos resumir en tres abstracciones, la primera como una interfaz amplia a los diversos servicios de los que necesitemos conectarnos, la segunda como plataforma de gestión y de ejecución constante, y finalmente la transferencia de mensajes o archivos basados en una configuración XML. Para lograr entender esto de una mejor manera, es necesario poner en práctica la arquitectura del diagrama, necesitamos un productor que genere algún mensaje, ubicándolo en una cola de mensajes llamada Services y un consumidor que sea capaz de recibir los mensajes ubicados en la cola Salida, entre estos dos, debe estar ubicado ServiceMix, el cual tendrá que tener una configuración por medio de Blueprints en XML, para que todos los mensajes que lleguen a Services sean redireccionados a la cola de Salida, con esto el consumidor lograría recibir el mensaje de forma correcta.

## Ejemplo en funcionamiento

Para lograr un ejercicio mucho más interesante se procederá a utilizar ServiceMix como ESB desde una maquina EC2 de Amazon Web Services, con esto se logrará comprobar que realmente se está haciendo una transferencia de mensajes completa y correcta, con una maquina externa a la que tenemos localmente.

Una vez apache ServiceMix se encuentra en nuestra maquina EC2, se procede a la configuración de las rutas con los correspondientes blueprints que se necesitan para lograr enrutar las colas de Service y Salida. De igual forma es necesario ejecutarlo y ponerlo en marcha para que nuestra herramienta pueda iniciar con su operación.

```
[ec2-user@ip-172-31-26-21 apache-servicemix-7.0.1]$ cd bin/
[ec2-user@ip-172-31-26-21 bin]$ dir
client      instance.bat  servicemix.bat  shell.bat      status.bat
client.bat  karaf         setenv          start          stop
contrib     karaf.bat    setenv.bat      start.bat      stop.bat
instance    servicemix    shell           status
[ec2-user@ip-172-31-26-21 bin]$ ./servicemix
Please wait while Apache ServiceMix is starting...
100% [=====]
Karaf started in 23s. Bundle stats: 222 active, 223 total

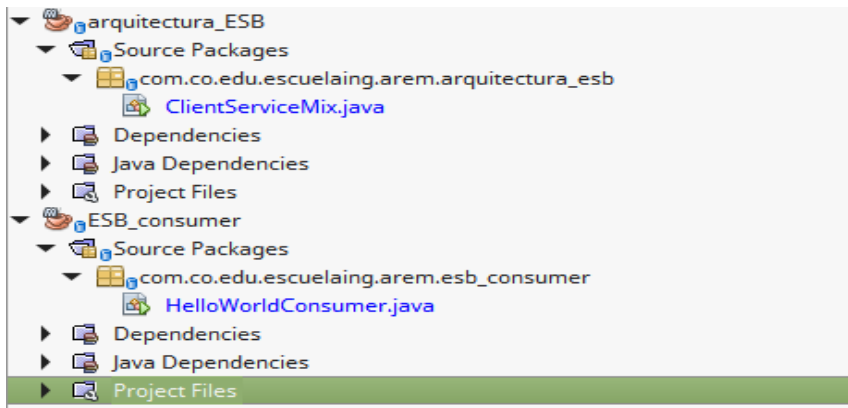
ServiceMix

Apache ServiceMix (7.0.1)

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or 'system:shutdown' to shutdown ServiceMix.

karaf@root>
```

Continuando con el ejercicio, es necesario crear dos distintos proyectos que logren desempeñar el papel de productor y consumidor que tendremos para esta prueba. Lo haremos por medio de dos proyectos Java, los cuales se conectarán con ActiveMQ a la Ip brindada por AWS con un protocolo TCP.



Para ejecutar las pruebas de una forma correcta, primero probaremos que el productor puede completar el envío del mensaje de manera satisfactoria, y que ServiceMix logre recibir en la cola de Events la información.

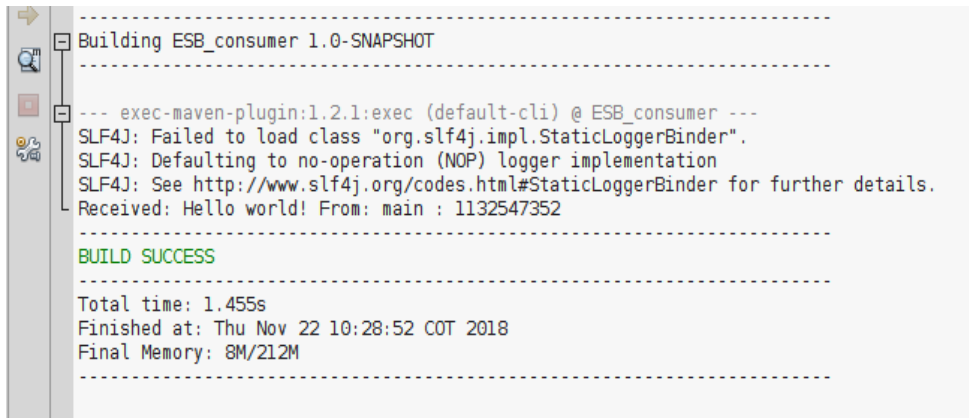
```
Sent message: 1731722639 : main
-----
BUILD SUCCESS
-----
Total time: 1.869s
Finished at: Thu Nov 22 10:15:00 COT 2018
Final Memory: 8M/150M
-----
```

Como vemos el mensaje se ha enviado satisfactoriamente, ahora es necesario verificar si nuestro ESB realmente recibió este mensaje enviado.

```
2018-11-22 15:10:47,875 | INFO | FelixStartLevel | BlueprintCamelContext
| 43 - org.apache.camel.camel-core - 2.16.5 | Route: route4 started and co
nsuming from: Endpoint[activemq://events]
2018-11-22 15:10:47,875 | INFO | FelixStartLevel | BlueprintCamelContext
| 43 - org.apache.camel.camel-core - 2.16.5 | Total 1 routes, of which 1 i
s started.
2018-11-22 15:10:47,875 | INFO | FelixStartLevel | BlueprintCamelContext
| 43 - org.apache.camel.camel-core - 2.16.5 | Apache Camel 2.16.5 (CamelCo
ntext: camel-4) started in 0.198 seconds
2018-11-22 15:10:47,895 | INFO | lixDispatchQueue | Main
| Thread: current | Karaf started in 23s. Bundle stats: 222 active, 223 total
2018-11-22 15:15:00,448 | INFO | Consumer[events] | events
| 43 - org.apache.camel.camel-core - 2.16.5 | Exchange[ExchangePattern: In
Only, BodyType: String, Body: Hello world! From: main : 1132547352]
karaf@root>log:display
```

Con este resultado podemos observar que la comunicación se está haciendo de manera satisfactoria, recibiendo el mensaje de forma correcta.

Finalmente, ahora solo necesitamos que el cliente que tenemos logre encontrar la información dentro de la cola de salidas, para que esto salga bien, es necesario que la configuración de las rutas esté elaborada de manera correcta, y que el consumidor pueda extraer esa información de allí.



```
-----  
Building ESB_consumer 1.0-SNAPSHOT  
-----  
--- exec-maven-plugin:1.2.1:exec (default-cli) @ ESB_consumer ---  
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".  
SLF4J: Defaulting to no-operation (NOP) logger implementation  
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.  
Received: Hello world! From: main : 1132547352  
-----  
BUILD SUCCESS  
-----  
Total time: 1.455s  
Finished at: Thu Nov 22 10:28:52 COT 2018  
Final Memory: 8M/212M  
-----
```

Como se puede observar, se obtuvo un resultado satisfactorio en el consumidor, logrando una comunicación completa entre estos dos componentes, brindándonos un resultado positivo en cuanto a la configuración del ESB. Con esta prueba, podemos tener un modelo muy básico de una comunicación de servicios, sin embargo, puede llegar a ser tan escalable como se desee.

## **Conclusiones:**

- Un Enterprise Bus Service es una parte fundamental para la comunicación e interacción de los diversos componentes o servicios que tiene una empresa, este brinda soluciones de organización, administración, y disminución de complejidad de los servicios que necesita cada empresa.
- El ejercicio de prueba fue satisfactorio, en este se pudo poner en marcha un ESB implementado con Apache ServiceMix, en una maquina EC2 de Amazon Web Services, recibiendo los mensajes que le enviaba un productor desde una maquina local, organizándolos en una cola llamada Events, de inmediato la configuración previa de ServiceMix, hace que este mensaje recién puesto, se redireccione a la Cola de Salida, donde un consumidor puede encontrar el mensaje que le fue enviado.
- Los sistemas modernos exigen cada vez más una interacción entre servicios independientes que se relacionan entre sí, obligando cada día más a migrarnos hacia una Arquitectura orientada a servicios, la cual no puede tener un funcionamiento deseado si no tiene en su núcleo un ESB destinado a manejar la mensajería que ocurre allí.
- En un nivel empresarial, donde el tráfico de información crece de forma cada vez más abrupta y los cambios en las estructuras y/o estrategias de las empresas son frecuentes, es de vital importancia tener un buen sistema de comunicación entre los componentes, tanto para la disminución de complejidad, como para tener un mejor control de cada uno de los componentes.