**Plan de Trabajo Final de la Licenciatura en Ciencias de la Computación**

### Título

Streaming - Cola de mensajería basada en contenido

### Autor

Jorge Federico Arito

### Asesor

Dr. Daniel Riesco

### CoAsesor

Ms. Ariel Leiva

**Introducción**

Este proyecto de tesis comienza con una introducción a conceptos de la Ingeniería de Software [1] [2] como así también de las arquitecturas y patrones, para luego continuar con conceptos básicos que ayudarán, en conjunto con el entendimiento de distintos tipos de arquitecturas, a formular la tesis presentada como Streaming - Cola de mensajería basada en contenido.

La ingeniería del software, según la definición de la IEEE en 1993 [3], es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software. La ingeniería del software ofrece métodos o técnicas para desarrollar y mantener software de calidad que resuelven problemas de todo tipo (o lo intentan), y trata áreas muy diversas de la informática y de las ciencias computacionales.

Con el transcurso de los años se han desarrollado recursos que conforman la ingeniería del software, es decir, herramientas y técnicas de especificación, diseño e implementación del software: la programación estructurada, la programación orientada a objetos, las herramientas CASE, la documentación, los estándares, CORBA, los servicios web, el lenguaje UML, etc.

La ingeniería del software comprende:

* Proceso de desarrollo de software (especificación, implementación y diseño, etc…).
* Metodologías para el desarrollo de software (RUP, patrones, framework…).
* Herramientas de desarrollo de software.

Dentro de la ingeniería de software se abre un tema que es de vital importancia: la arquitectura del software.

La arquitectura de software por otra parte consiste en el diseño de componentes de una aplicación (entidades del negocio) [4], generalmente utilizando patrones de arquitectura. El diseño arquitectónico debe permitir visualizar la interacción entre las entidades del negocio y además poder ser validado, por ejemplo por medio de diagramas de secuencia. Un diseño arquitectónico describe en general el cómo se construirá una aplicación de software. Para ello se documenta utilizando diagramas, por ejemplo:

* Diagrama de clases
* Diagrama de base de datos
* Diagrama de despliegue
* Diagrama de secuencia
* Diagrama de componentes

Dentro de las arquitecturas de software tenemos un vertical donde se va a enmarcar el contenido de esta tesis que es el Streaming de datos de gran magnitud.

Arquitecturas de software para Streaming de datos, son arquitecturas pensadas especialmente para transmisión masiva de datos. Una cantidad tan excesiva de datos que las aplicaciones de software que tradicionalmente usamos no son capaces de computar, tratar y poner en valor en un tiempo razonable [6].

Streaming - Cola de mensajería basada en contenido surge como necesidad para resolver un problema puntual que es la comunicación de mensajes de diferentes tipos (topic queue) entre sistemas distribuidos permitiendo a los sistemas enviarlos (publicar) y a los interesados (suscriptores) recibirlos y además permitiendo a estos filtrar solo los que les pueden resultar relevantes (Content-Based) y todo mediante una interfaz REST.

Que es un sistema de streaming ?

La retransmisión (en inglés streaming) es la distribución digital de contenido multimedia o texto a través de una red de computadoras, se refiere frecuentemente a una corriente continua que fluye sin interrupción, y habitualmente a la difusión de audio y video, en este sistema usado para transmitir mensajes en formato JSON (explicado más adelante).

Que sería un sistema de streaming basado en contenido?

Un sistema de colas basado en contenidos es un sistema igual al descrito en el punto anterior, sumando la posibilidad de suscribirse a un tema particular o de interés, donde se pueda especificar ciertas condiciones que son las que a este le interesa que los mensajes cumplan. En cuanto un mensaje cumple las condiciones será enviado al sistema interesado, en caso de haber mensajes que no cumplen con las condiciones especificadas ese mensaje se descarta para el sistema en cuestión [5].

Esta mejora al sistema de colas tradicional, permite a los sistemas suscriptores evitar tener una arquitectura de software muy compleja dada que la carga que van a recibir es un porcentaje de la totalidad de mensajes publicados en un tópico particular. Otra mejora es que de esta manera al no enviar la totalidad de mensajes a todos los suscriptores, se reduce la carga que se le da a los recursos de networking [7].

**Objetivos**

El objetivo principal del desarrollo de esta tesis está directamente relacionado con la motivación, al final del camino se quiere poder contar con un diseño de arquitectura de software que permita:

* Especificar un conjunto de tópicos donde se podrán publicar mensajes, estos tópicos serán colas de mensajes que deberán funcionar similar al modelo conocido como colas FIFO (primero en entrar, primero en salir). Y además,
* Permitir a las aplicaciones interesadas en un tema participar, suscribirse para empezar a recibir mensajes, con el agregado de poder especificar condiciones sobre ellos, que deberán cumplir para ser recibidos por el suscriptor.
* De los dos puntos anteriores se desprenden las acciones básicas que el sistema deberá permitir, que tiene que ver con la administración tanto de los tópicos como de los suscriptores, que son las acciones de : crear, borrar, consultar o modificar.

Además se quiere que el sistema final cuente con ciertas características de calidad, descritas a continuación:

* Agnóstico: es decir que el sistema no sepa qué mensajes o tópicos se transfieren a través de él, si que cumpla que el mensaje que llega se distribuya.
* Consistente: tanto desde el punto de vista de cantidad de mensajes como así también del contenido de los mensajes. De esta manera el sistema solo tiene que ser un “pasamano” de mensajes, y asegurando siempre que el mensaje que llega, es un mensaje que se transmite a los suscriptores interesados.
* Alta Disponibilidad: tiene que ser de alta disponibilidad, este es el pilar fundamental del sistema ya que el sistema en sí puede formar parte del camino crítico de las aplicaciones que lo utilizan.
* Alta performance: el sistema debe cumplir con los objetivos anteriores, pero debe hacerlo en tiempos aceptables, esto es para evitar que la mensajería no llegue a tiempo a destino.
* Agilidad: capacidad de responder rápidamente a un entorno en constante cambio. Dado que los componentes del sistema son de un solo propósito y están completamente desacoplados de otros componentes del sistema, los cambios generalmente están aislados para uno o varios procesadores de eventos y pueden realizarse rápidamente sin afectar a otros componentes.
* Facilidad de despliegue: esto tiene que ver con que por más complejo que termine siendo el sistema, este deberá ser de fácil administración en entornos productivos, seguramente esta característica va a influir mucho en las decisiones de arquitectura que se tomen.
* Testeable: Si bien las pruebas de unidades individuales no son demasiado difíciles, sí requieren algún tipo de cliente de prueba especializado o herramienta de prueba para generar eventos.
* Escalable: junto con el objetivo de alta disponibilidad, es otro objetivo fundamental, a futuro el sistema debe comportarse de la misma manera tanto para grandes cargas de tráfico como para gran cantidad de tópicos y suscriptores. Seguramente también esta característica determinará gran porcentaje de las decisiones de arquitectura que se tomarán.
* Web: que el sistema sea pensado desde el momento cero para ser soportado en un entorno de webservices.

### Metodología

El trabajo propuesto surge de una necesidad concreta planteada como problema en MercadoLibre S.A., la de propagar información entre microservicios de manera escalable y consistente, para llegar a la solución planteada en este trabajo, se tuvo una gran parte de investigación de las herramientas apropiadas para resolver de la mejor manera posible el problema, lo que permitió al final del camino contar con una herramienta versatil, facil de entender su funcionamiento y que cumple con los objetivos planteados.

Durante el desarrollo de este trabajo se agregarán nuevas características/funcionalidades al sistema resultante, tales como interfaz para administrar los recursos del mismo, y diagramas que describirán esas funcionalidades desde el punto de vista de la arquitectura de software. Se harán pruebas de carga para entender cuales son las prestaciones y performance del sistema, para determinar si cumple con los objetivos planteados al inicio. Finalmente se redactará el informe correspondiente.  
  
Implementar el sistema permitirá que otros desarrolladores puedan resolver problemas similares cumpliendo con estándares de compañías de software cuyos niveles de procesamiento de información es muy elevado.

### Plan y cronograma de tareas

Parte de la investigación y experimentación de las herramientas para montar la arquitectura del entorno de desarrollo remoto, ya han sido realizadas por el autor de este trabajo y se encuentran productivas. Además se detallan a continuación las tareas restantes que se desarrollarán durante la realización de este trabajo, para así cumplir con los objetivos propuestos.

Tareas:

* Implementación del sistema: 9 meses. Ejecutada.
  + Investigación de soluciones similares en el mercado.
  + Comparación de distintas tecnologías.
  + Diseño del sistema: interfaces, arquitecturas, etc.
  + Desarrollo de las funcionalidades del sistema.
* Despliegue, puesta en producción del sistema: 2 meses. Ejecutada.
* Redacción del informe: 3 meses. En ejecución.

Duración estimada total: 14 meses.

### Recursos

Todos los recursos utilizados, desde laptop para análisis, desarrollo del sistema, pruebas integrales y de carga, puesta en producción en entornos virtualizados y herramientas de monitoreo son provistos por MercadoLibre S.A.

### Referencias

[1] https://histinf.blogs.upv.es/2010/12/28/ingenieria-del-software/

[2] http://pegasus.javeriana.edu.co/~mad/Generalidades%20de%20las%20Arquitecturas%20de%20SW.pdf

[3] Institute of Electrical and Electronics Engineers (IEEE), IEEE Software Engineering

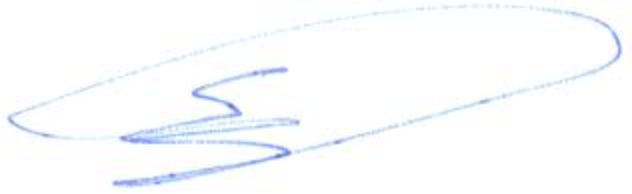
Standards Collection. IEEE Standard 610.12-1990, 1993.

[4] Pattern-Oriented Software Architecture - Volume 1 - Frank Bushman, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal - Wiley - 1996

[5] Content-based Publish/Subscribe Systems - Haiying Shen - 2010 - http://www.cs.virginia.edu/~hs6ms/publishedPaper/bookChapter/2009/sub-pub-Shen.pdf

[6] Fundamentals of Software Architecture: An Engineering Approach - Mark Richards, Neal Ford - O'Reilly - 2020

[7] Pattern-Oriented Software Architecture - Volume 1 - Frank Bushman, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal - Wiley - 1996



Jorge Federico Arito Dr Daniel Riesco