ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Maestría En Sistemas De Información Gerencial

"IMPLEMENTACIÓN DE UNA PLATAFORMA DE MANEJO DE EVENTOS ASINCRÓNICOS PARA EL PROCESAMIENTO MASIVO DE TRANSACCIONES PARA UNA EMPRESA DE TELECOMUNICACIONES"

EXAMEN DE GRADO (COMPLEXIVO)

Previo a la obtención del título de:

MAGÍSTER EN SISTEMAS DE INFORMACIÓN GERENCIAL

GALO FERNANDO SOLÍS VARGAS

GUAYAQUIL – ECUADOR

AÑO: 2015

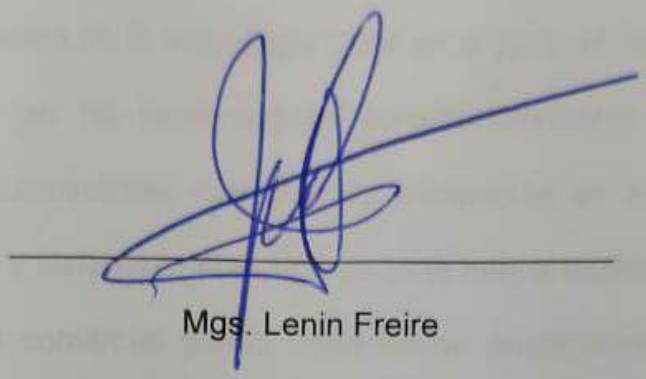
AGRADECIMIENTO

Mi agradecimiento a mi esposa y su constante apoyo para que alcance las metas que nos hemos trazado.

DEDICATORIA

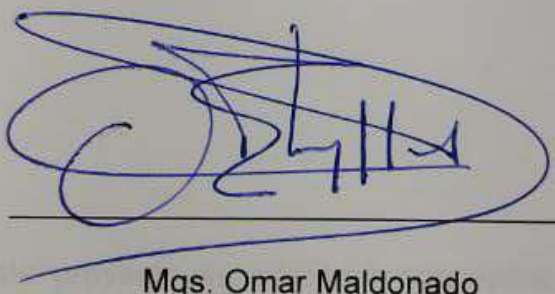
El presente proyecto lo dedico a mis padres Galo y Mary.

TRIBUNAL DE SUSTENTACIÓN



Mgs. Lenin Freire

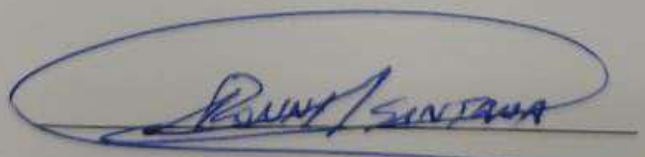
DIRECTOR DEL MSIG



Mgs. Omar Maldonado

PROFESOR DELEGADO

POR LA UNIDAD ACADÉMICA



Mgs. Ronny Santana

PROFESOR DELEGADO

POR LA UNIDAD ACADÉMICA

RESUMEN

Desde la implementación de la tecnología GSM en el país, el número de abonados de telefonía celular se ha incrementado considerablemente, haciendo que la empresa de telecomunicaciones con mayor participación en el mercado del país mejore sus procesos y sistemas para soportar esta nueva realidad. De esta forma, para que el área de comercial pueda implementar promociones y campañas en corto tiempo, se creó una plataforma de manejo de eventos asincrónicos para el procesamiento masivo de transacciones.

Para implementar este proyecto se utilizaron conceptos como manejo de colas, procesamiento asincrónico en multihilos, bitacorización, monitoreo y alarmas, entre otros. Al finalizar el proyecto la empresa explota esta herramienta para constantemente evolucionar su oferta de nuevos servicios y promociones, consiguiendo el TIME-TO-MARKET esperado con la calidad que corresponde.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ABREVIATURAS

APN	Access Point Name
CRM	Customer Relationship Management
ETL	Extract, Transform, Load
GSM	Global System for Mobile communication
MOM	Message Oriented Middleware

INTRODUCCIÓN

En la empresa de telecomunicaciones con mayor participación del mercado del país, la evolución es una constante, no pasa un trimestre que no aparezca una nueva plataforma, un nuevo producto o un nuevo servicio, esto propone un reto, el cual es enfrentar este cambio de forma ordenada y en corto tiempo.

Cada nueva solución, producto o servicio, trae consigo el componente tecnológico que puede diferir dependiendo del proveedor o fabricante, lo que normalmente obligaba al área de sistemas a integrarse a todas estas soluciones, con todas las complejidades que esto refería, como incompatibilidades, falta de estandarización y duplicar activos tecnológicos que ya existían en nuestro portafolio de servicios en el área de sistemas.

Para evitar esta complejidad y enfocar los esfuerzos en temas que den valor a la empresa, se implementó el proyecto que creó una plataforma de manejo de eventos asincrónicos para el procesamiento masivo, utilizando las mejores prácticas respecto a manejo de colas, procesamiento asincrónico en multihilos, bitacorización, monitoreo y alarmas, entre otros.

Al implementar este proyecto, los objetivos estratégicos que se han obtenido son: mejorar el time-to-market, estandarizar los activos tecnológicos, homogenizar procesos, facilitar el monitoreo, acelerar tiempos de desarrollo.

CAPÍTULO 1

GENERALIDADES

En este capítulo de forma general se describe el problema que motivó la implementación de la solución y se nombran principales características de la plataforma de manejo de eventos asincrónicos para el procesamiento masivo de transacciones.

1.1 Descripción del problema

En el sector de las telecomunicaciones las empresas del país deben reaccionar rápidamente sobre nuevas tecnologías, productos, servicios, promociones, comportamiento de mercado, entre otros; esto compromete al área de sistemas estar siempre atentos a buscar como acelerar los tiempos de entrega de nueva soluciones de software que soporten estos cambios, sin que esto signifique poner en riesgo la estabilidad de su operación habitual.

Cada solución de software implementada comparte patrones comunes de comportamiento, sin embargo, se corre el riesgo de que cada proyecto de desarrollo de estas solución repita esfuerzo en creación de componentes que otra solución ha creado previamente, evitando el reúso de infraestructura, complicando la administración y monitoreo de estos recursos, exponiendo puntos de fallo o distintas vulnerabilidades dependiendo del diseño de cada implementación realizada.

De esta forma, toda solución de software por lo general tiene por ejemplo: tablas que implementan colas de transacciones, despachadores de procesos, bitacorización, manejo de reintentos, alarmas, monitoreo, depuración; aumentando en cada implementación los activos que posteriormente hay que administrar y dar mantenimiento, llenándonos finalmente de un inventario de componentes que cumplen el mismo propósito pero desarrollados de distinta manera.

1.2 Solución propuesta

Se propuso la abstracción del comportamiento de este tipo de soluciones de software, identificando patrones comunes que se han repetido en cada implementación; y generar una plataforma con distintos componentes reusables, homogéneos y agnósticos, para el manejo de eventos asincrónicos y su procesamiento masivo de transacciones.

De tal forma que cada vez que el negocio requiere crear nuevos procesos, productos, promociones o servicios, estos se adaptan a la plataforma de

manejo de eventos, reusando todos sus componentes, dejando solamente la implementación de la particularidad del proceso, consiguiendo que, los proyectos de desarrollo se concentren en su totalidad en implementar la nueva lógica de negocio y se despreocupen de temas reusables, tales como, asincronismo, bitacorización, colas, despachadores, monitoreo, alarmas, procesamiento multihilos, entre otros.

Dado que los proyectos se orientarían en el desarrollo de las particularidades de la nueva lógica de negocio, se consigue mejorar el TIME TO MARKET, y se evita poner en riesgo la estabilidad de la operación actual, ya que se reusarían componentes probados y estables.

CAPÍTULO 2

CONCEPTOS APLICADOS

En el capítulo 2 vamos a revisar los principales conceptos utilizados en la creación del manejador de eventos asincrónico para procesamiento masivo de transacciones, y mencionar la justificación de su uso dentro de la solución.

2.1 Procesos y Multithreading

Desde el punto de vista de negocio, un proceso se refiere al conjunto de tareas y comportamientos relacionados en forma lógica, que las organizaciones desarrollan con el tiempo para producir resultados de negocios específicos y la forma única en que se organizan y coordinan estas actividades [1]. Al momento que abstraemos el comportamiento en común de los procesos de negocios, todos comparten las siguientes características, tiene datos de entrada, lógica aplicada y una salida esperada.

Figura 2.1 Abstracción de un proceso de negocio

De esta manera, generalizamos que todo dato de entrada que sería procesado por la lógica de negocio es considerado un **evento**, siendo un evento un conjunto de metadata y data que describen un hecho suscitado. El proceso que se ejecutará usando como dato de entrada el evento, es considerado **lógica de negocio**; sin embargo esta lógica de negocio se construye sobre una lógica básica que considera características no funcionales como bitacorización general, trazado de transacciones, generación de indicadores, entre otros. Por último tenemos el resultante del proceso, que se considera como **retorno**, que se conforma de metadata, siendo esta un código que permite representar el resultado del proceso y un mensaje de salida que permite expresar en una narrativa entendible por el negocio el resultado de haber aplicado la lógica sobre el evento.

De esta forma podemos tener un conjunto de procesos que modelan el comportamiento del negocio, sin embargo para conseguir un alto rendimiento y explotando las características de los servidores con procesadores de múltiples núcleos de la actualidad, se requiere utilizar el concepto de **multithreading**, que es la posibilidad de manejar distintas hebras o hilos de ejecución de manera simultánea con el objetivo de paralelizar el código e incrementar el rendimiento de aplicaciones [2].

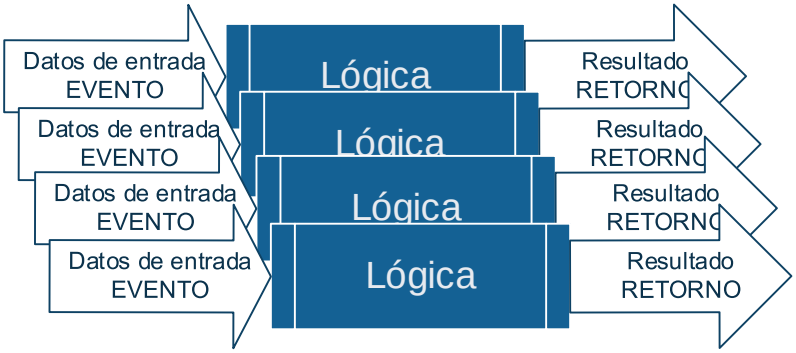


Figura 2.2 Multithreading

Usando multithreading conseguimos procesar de forma masiva y en paralelo una gran cantidad de eventos de entrada, logrando evitar dependencia entre hilos, de tal manera que si el rendimiento de procesamiento de un hilo se ve afectado por alguna irregularidad, este comportamiento no afecte otros hilos de procesamiento, aislando el problema sin impactar el procesamiento en general; otra estrategia es identificar en el flujo de procesamiento aquellos eventos que ralentizan el sistema como un todo y a estos eventos asignarles hilos exclusivos de procesamiento en una especie de afinamiento controlando la cantidad de recursos asignados a estos para que se despachen con mayor celeridad.

El uso de la abstracción de los proceso y multithreading nos permite modelar las distintas soluciones que el negocio requiere implementar, la comunicación entre los procesos los realizamos utilizando mensajes y encolamiento de estos, sin embargo la capacidad de procesamiento de los servidores es limitado, por lo tanto necesitamos controlar la carga que los servidores pueden procesar, esto lo conseguimos con el uso de otro

concepto aplicado que es el asincronismo, mismo, que da paso a la siguiente sección.

2.2 Message Oriented Middleware

La comunicación entre los procesos las realizamos utilizando mensajes asíncronos, de ahí la importancia del concepto Message Oriented Middleware o en adelante MOM, siendo el MOM un instrumento facilitador de la integración funcional de aplicaciones, donde se utilizan los mensajes como método de integración y provee mecanismos para crear, manipular, almacenar y transmitir esos mensajes, que no son más que metadata mas data [3].

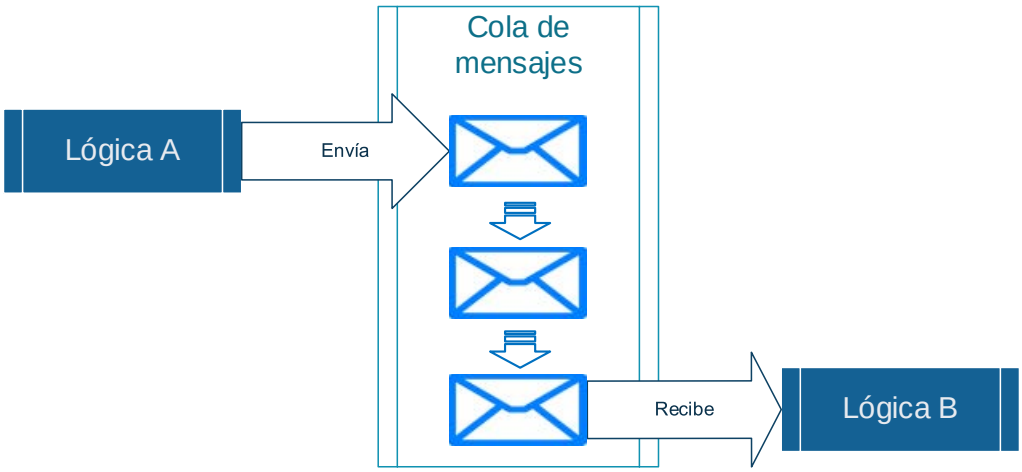


Figura 2.3 Sistema basado en MOM

Como podemos observar en el sistema basado en MOM de la figura 2.3, existen dos procesos que se comunican entre sí utilizando mensajes, sin embargo hacen uso de un intermediador para conseguirlo que es la cola de

mensajes, de tal forma que, el proceso que tiene la lógica A, al ejecutarse retorna como resultado un mensaje que deposita en la cola de mensajes, para que posteriormente el segundo proceso que contiene la lógica B, tome de la cola de mensajes el evento que le corresponde y lo aplica como su entrada para ejecutar su lógica implementada, de esta forma se consigue el asincronismo y la interoperabilidad de los procesos modelados en el manejador de eventos. Existen dos tipos de implementaciones del MOM, el modelo punto a punto y el modelo publicador/suscriptor que es el que usamos en el manejador de eventos, el modelo publicador/suscriptor cuenta con varios clientes, uno que publica eventos y otros distintos que consumen esos eventos, la diferencia con el punto a punto es que en este modelo se tiende a tener más de un consumidor [4].

Por naturaleza el comportamiento de las colas es asincrónico, sin embargo puede también ser sincrónico, que se consigue cuando el consumidor al procesar el mensaje hace un callback a quien lo produjo [5], sin embargo, ese método no es usado en esta solución.

Gracias al uso del intermediador, la cola de mensajes, podemos regular la capacidad utilizada de los servidores donde reside la solución, de tal forma que si existe algún problema en la ejecución del motor, los procesos no se ven afectados, más bien lo que sucede es un encolamiento en los repositorios donde persisten los mensajes, los mismos que mantendrán los eventos hasta que el problema se solvente y el sistema vuelva a tener la

capacidad de poder procesarlos, para que continúe con el despacho de mensajes desde el punto donde se detuvo.

CAPÍTULO 3

DESARROLLO DE LA SOLUCIÓN

En el tercer capítulo, presentamos los componentes desarrollados en la solución, cuáles son sus responsabilidades y principales características. Tenemos componentes de gestión como lo son el administrador de colas, el componente de bitacorización, monitoreo y depuración; por otro lado tenemos los componentes de servicio, como son el productor y consumidor.

3.1 Administrador de colas

El administrador de colas es un componente de gestión en la solución, que tiene como objetivo crear colas donde reposarán los eventos antes de ser procesados por la lógica de negocio para posteriormente ir a históricos diarios.

Desde una perspectiva general, todo evento de cualquier tipo llega a una cola principal, cuyo objetivo es redistribuirlo en las distintas colas que se suscriban al evento. De esta forma, podemos con un mismo evento de entrada, como por ejemplo, el alta de una nueva línea celular, enviarlo a distintas colas que necesitan conocer sobre dicho evento para realizar una acción en particular, como por ejemplo enviarle un SMS de bienvenida a la operadora. El modelo mental del manejo de colas es el siguiente:

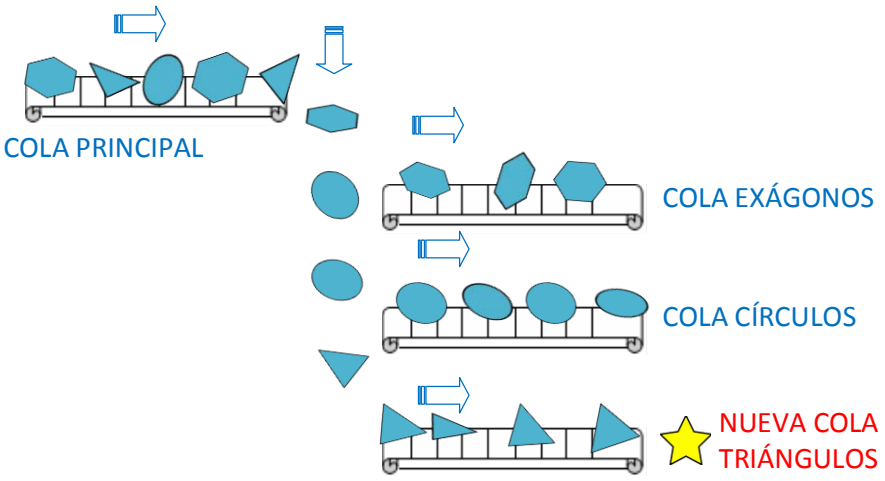


Figura 3.1 Modelo mental de interacción entre colas

El motor por defecto viene con la **cola principal** que redistribuirá los eventos a las colas que se creen, el administrador de colas tiene como principal responsabilidad crear nuevas colas, las nuevas colas tienen las siguientes características:

- Nombre de la cola que identifique su propósito.
- Características de los eventos que espera recibir (filtros)
- Ventana de tiempo en la que espera recibir los eventos.

- Lógica de negocio que se ejecutará por cada evento que reciba
- Ventana de tiempo en la que procesará los eventos
- Edad máxima de los eventos que se mantendrán en históricas
- Cantidad de hilos que procesarán los eventos de la nueva cola.

3.2 Productor

El componente productor tiene como responsabilidad alimentar la solución con eventos, existen dos tipos de productores, el productor PULL y el productor PUSH.

El productor PULL se encarga de extraer información de plataformas, transformarlas al formato del manejador de eventos y de manera masiva cargarlas en la cola principal para su posterior redistribución y procesamiento.

Este productor tiene un esquema no intrusivo, porque no requiere que los proveedores de las plataformas modifiquen sus procesos para alimentar al manejador de eventos, se requiere básicamente programar la transformación entre el evento heterogéneo a un evento que cumpla las características respectivas que permitan a las distintas colas interpretarlo y procesarlo según su necesidad.

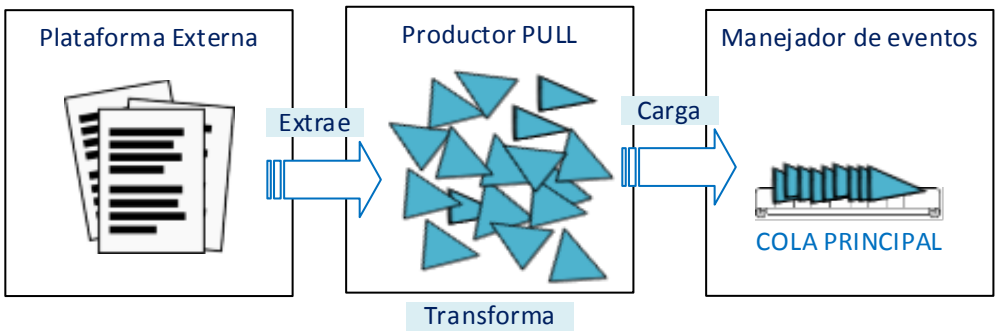


Figura 3.2 Esquema de productor PULL

El productor PUSH es una interfaz que permite a sistemas internos o externos dentro de su propia lógica inyectar uno a uno los eventos al manejador, si bien es cierto usar el productor PUSH es intrusivo y requiere que el proveedor de la plataforma o sistema modifique su transacción, el productor está expuesto en distintas tecnologías estándares que facilita su integración y otra ventaja de usar el productor PUSH es que no hay que esperar que la plataforma externa genere los archivos en batch sino que en línea nos inyecte el evento de tal forma que podemos tomar una acción más pronta con él.

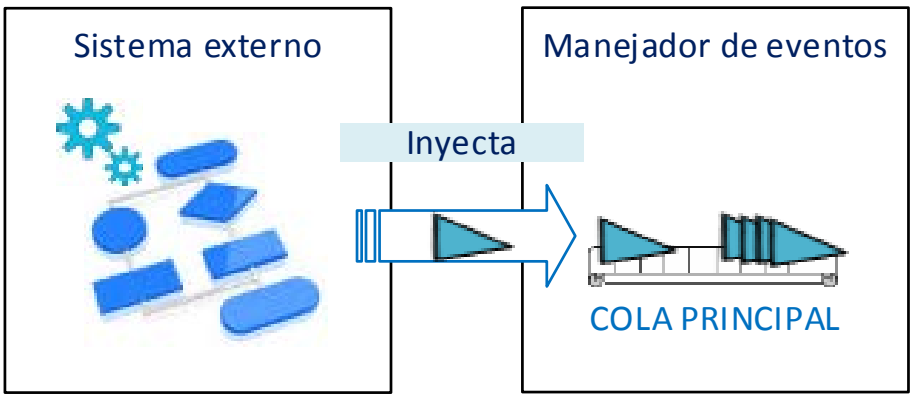


Figura 3.3 Esquema de productor PUSH

3.3 Consumidor

Todas las colas del manejador de eventos deben estar acompañadas por una lógica de negocio que se ejecuta por cada evento que llegue a la cola. Incluso la cola principal que es donde primero se alojan el 100% de eventos, tiene lógica de negocio asociada, que es redistribuir los eventos según la configuración de las otras colas; cada cola puede tener solamente una lógica de negocio asociada, cada lógica de negocio tiene como entrada un solo evento de la cola, y se ejecutan en paralelo dependiendo de cuantos hilos se configuraron para el despacho de la cola.

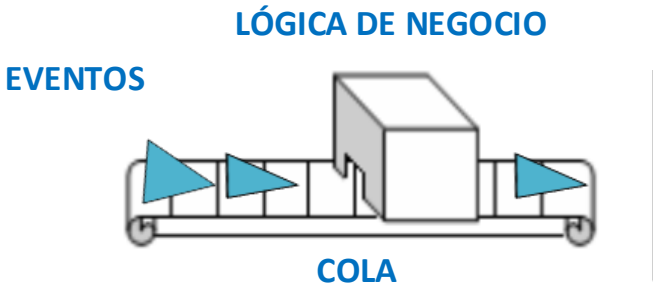


Figura 3.4 Modelo mental de la lógica de negocios

Para que el consumidor se active y comience a despachar los eventos de una cola, en la metada de configuración dicha cola debe estar activa y la fecha del evento debe estar vigente respecto a las fecha inicio y fecha fin de consumo de la cola. La lógica de negocio es lo que implementa cada proyecto, es donde se desarrolla lo que se desea realizar en presencia de cada evento, por ejemplo entregarle un descuento en la factura del cliente

por realizar una llamada de más de 5 minutos, es decir la lógica de negocio está fuera del alcance de la solución y depende exclusivamente de quien utilice el motor. Una característica del proceso consumidor y de la arquitectura en general es que como cada proceso pertenece a una sola cola, si existe un proceso que se demora en despachar, generará encolamiento solamente en dicha cola, las otras colas no se involucran con ese incidente, de tal forma que crear nuevas colas con nuevos procesos de negocio no aumenta riesgo de que afecte a procesos actuales que corren sobre la solución.

Si bien es cierto que la lógica de negocio está fuera del alcance del motor, es necesario que la lógica de negocio retorne la siguiente información: código de salida, mensaje de salida; el mensaje de salida es un dato informativo, pero el código de salida es interpretado por el manejador de evento para tomar una decisión, si el código de salida es cero o positivo indica que la lógica de negocio hizo lo que tenía que hacer con el evento y solo queda depositarlo en la tabla histórica diaria, pero si el código de salida que retorna la lógica de negocio es negativo o es nulo, le está indicando al manejador de eventos, que la lógica de negocio falló y requiere que el evento que acaba de procesar sea reinyectado en la cola para un posterior reintento; el tiempo entre reintentos y el máximo número de reintentos para un mismo evento se parametriza cuando se crea la cola utilizando el administrador de colas, así como la edad máxima del evento en la cola, es decir en la cola por reintentos se mantiene el evento hasta lograr el máximo número de intentos o la expiración del evento.

3.4 Bitacorización, monitoreo y depuración

El motor implementa bitacorización a nivel de base de datos, donde se puede obtener información como:

- Fecha y hora de cuando se crea una cola.
- Fecha y hora de cuando se levanta el proceso productor PULL del motor.
- Fecha y hora de cuando utilizando el productor PUSH inyectan eventos los sistemas externos.
- Fecha, hora y duración de lo que toma en despacharse los eventos de cada cola.
- Código y mensaje de salida de cada uno de los eventos procesados por las distintas colas.

Esta información permite a través de otro componente externo a la presente solución alimentar un panel de monitoreo del servicio, que posibilita al área de procesos de manera visual tener indicadores del rendimiento de las distintas colas e identificar incidentes.

Esta información está disponible dependiendo de la naturaleza de cada cola y existen procesos que por cada cola depura la información histórica previamente habiéndola respaldado en cinta, al permitir que la depuración se realice por cola se permite configurar una edad muy joven para colas donde los eventos son voluminosos y una histórica longeva para colas donde no existen muchos eventos, por ejemplo no es lo mismo un día de

historia de contratación de recargas de saldo versus un día de historia de renuncias de clientes, en el primer caso el volumen de información es considerable y requiere una depuración más exhaustiva y frecuente.

CAPÍTULO 4

ANÁLISIS DE RESULTADOS

En el presente capítulo mostraremos los resultados que hasta la fecha hemos obtenido al haber implementado el manejador asincrónico de eventos masivos, presentaremos los principales tipos de proyectos implementados donde se ha utilizado el motor de manera exitosa, la diferencia sustancial en el time to market para el negocio al disminuir el tiempo de los desarrollos al soportar funcionalidad existente, la importancia en el reúso de componentes no solo de vista al negocio sino también a la operación que le da mantenimiento y soporte, y posteriormente mostraremos el análisis de rendimiento del motor, una mirada a la capacidad que en la actualidad tiene la solución para procesar grandes volúmenes de datos y para finalizar estrategias de como al identificar el comportamiento de un proceso se puede con solo configuración modificar el manejador de eventos para aislar el problema y que no impacte a otros procesos que no deberían ser afectados.

4.1 Proyectos tipo implementados

Dentro de los principales proyectos implementados tenemos aquellos que como evento de entrada tienen la recepción de un mensaje corto, el cliente envía un mensaje corto a un destinatario determinado y dicho evento llega al motor, se ejecuta lógica de negocio específica para ese tipo de evento y retorna el resultado, la lógica de negocio utilizada por ejemplo es:

- Trivias de satisfacción al cliente
- Servicio de llamadas de emergencia “quieren llamarte”
- Trivias de contratación de paquetes de datos

Otro proyecto tipo implementado es cuando el cliente realiza un cambio de equipo, la plataforma que identifica el evento alimenta al motor y existen distintas lógicas de negocios asociadas a esta, como por ejemplo:

- Registro del cambio en el CRM,
- Habilitación/Inactivación de APN’s en caso de aplicar.

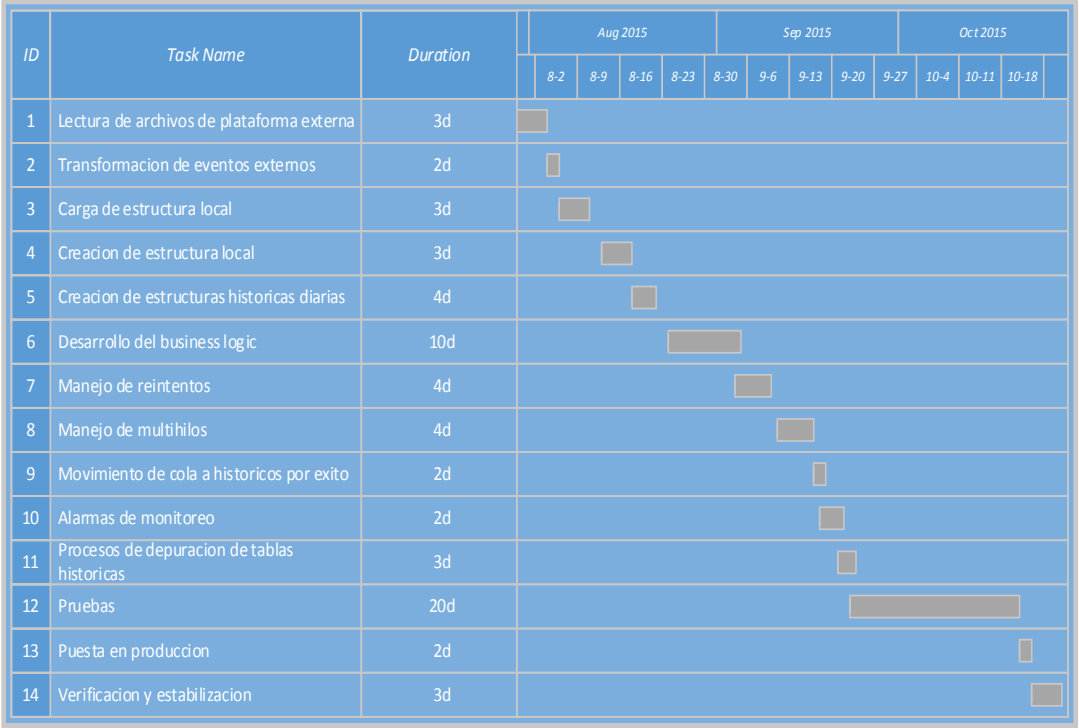
Un escenario donde el manejador de eventos asincrónicos ha sido de alto nivel de reúso es cuando el evento realizado es una recarga de saldo, cada vez que un cliente realiza una recarga sea física o virtual por cualquier canal de activación, las distintas plataformas inyectan este evento al motor consiguiendo implementar lo siguiente por ejemplo:

- Envío de mensaje del regulador con precios oficiales de recargas
- Activación de promociones.

4.2 Análisis del TIME TO MARKET

Para analizar el time to market vamos a considerar varios escenarios, el primer escenario es cuando no existía el manejador de eventos y por cada proyecto había que construir todos los artefactos necesarios para implementar una solución de estas características, sin un mayor nivel de reúso.

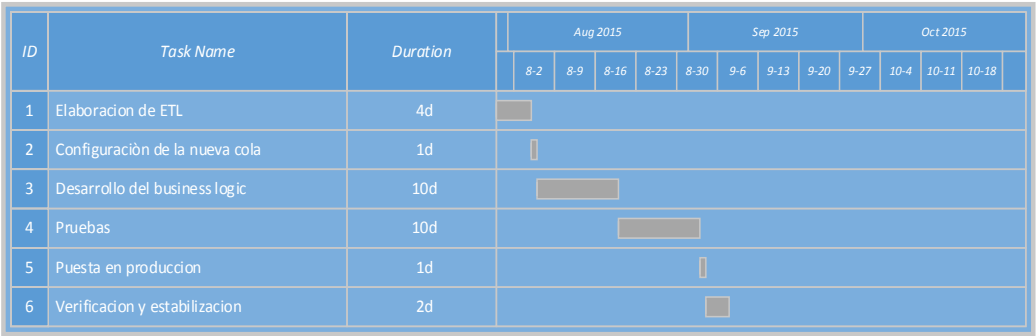
Tabla 1 Gantt de un proyecto sin manejador de eventos



ID	Task Name	Duration
1	Lectura de archivos de plataforma externa	3d
2	Transformacion de eventos externos	2d
3	Carga de estructura local	3d
4	Creacion de estructura local	3d
5	Creacion de estructuras historicas diarias	4d
6	Desarrollo del business logic	10d
7	Manejo de reintentos	4d
8	Manejo de multihilos	4d
9	Movimiento de cola a historicos por exito	2d
10	Alarmas de monitoreo	2d
11	Procesos de depuracion de tablas historicas	3d
12	Pruebas	20d
13	Puesta en produccion	2d
14	Verificacion y estabilizacion	3d

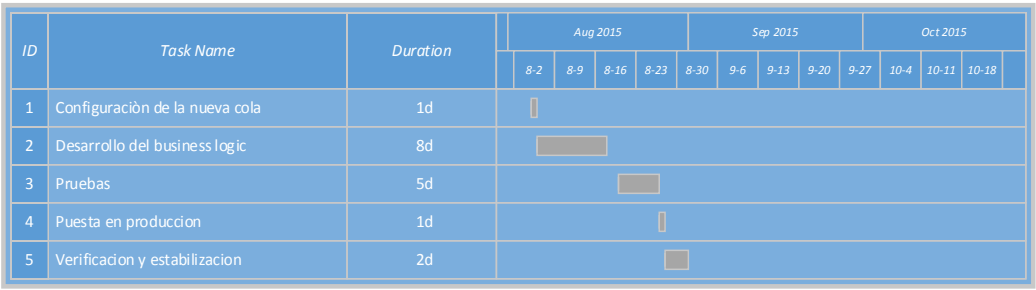
Luego evaluemos un escenario donde ya hemos implementado el manejador de eventos, pero el evento que necesitamos procesar no existe en el manejador aún, para este caso tenemos los siguientes tiempos:

Tabla 2 Gantt de un proyecto con manejador de eventos y ETL



Y para finalizar veamos el escenario donde el evento que deseamos utilizar ya existe registrado en el manejador de eventos, en ese caso los tiempos del proyecto son los siguientes:

Tabla 3 Gantt de un proyecto con manejador de eventos



El éxito del modelo es que las tareas que se repiten entre proyecto y proyecto no se vuelvan a presentar nuevamente, de tal manera que no es necesario volver a realizar pruebas de componentes que se deben encontrar estables en ambiente de producción, de esta manera podemos demostrar la mejora en el time to market que en la práctica ha llegado a ser

de requerimientos atendidos hasta en 1 día desde que al negocio se le presenta una idea hasta que lo tenemos lanzado al mercado comercialmente.

4.3 Reúso de componentes

Al reusar componentes no solamente reusamos al componente como tal, sino también reusamos todo lo que está detrás del mismo, como por ejemplo su etapa de diseño, sus distintas pruebas, sean estas funcionales, de integración, de carga; su monitoreo, sus alarmas, etc. Mientras más componentes de una solución se reúsen más valor dicho activo tecnológico le genera a la empresa, los componentes que se reúsan son los siguientes en esta solución:

Tabla 4 Componentes reusados

Nombre del componente	Descripción	Contenido en
Productor PUSH	Plataformas externas inyectan eventos al manejador	Plataformas externas
Multiplexor de eventos	Redistribuye los eventos a las colas que se hayan suscrito al evento	Lógica de negocio de la cola principal
Asignador de hilos	Asigna un hilo de	Productor PUSH

	ejecución a un evento dentro de una cola, es responsable que los eventos de un mismo teléfono siempre caigan en el mismo hilo de ejecución	Productor PULL Multiplexor de eventos
Manejador de reintentos	Permite reinyectar el evento en la cola respectiva con una fecha de procesamiento futura para manejar reintentos	Despachador
Creación de tablas históricas	Diariamente crea estructuras que almacenaran el resultado del procesamiento de eventos	Despachador
Depuración de tablas históricas	Las tablas históricas que superan el tiempo configurado respecto a su edad en días, se eliminan automáticamente	Despachador

4.4 Análisis de rendimiento

La distribución del comportamiento de generación de eventos en un día promedio habitual tiende a lo siguiente:

Figura 4.1 Cantidad de eventos por hora

El rendimiento se mide tomando el tiempo entre que se recibe un evento y es consumido por la lógica de negocio, para un día habitual tenemos el siguiente rendimiento tanto de un proceso ligero como de una lógica de negocio pesada:

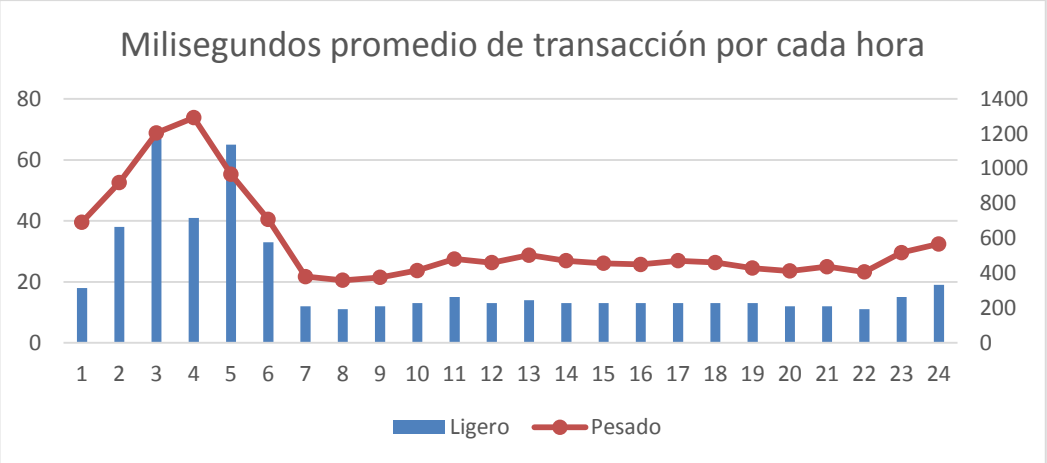


Figura 4.2 Milisegundos promedio de transacción por cada hora

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

1. Se concluye que con la creación del manejador de eventos las distintas áreas de la empresa que en sus proyectos lo han utilizado han observado una mejora en el time to market de sus proyectos.

2. Al conseguir manejar los distintos procesos con el reúso de los mismos componentes, el nivel de complejidad en la administración, soporte y mantenimiento de los mismos decreció para el área de producción.

Recomendaciones

1. La tecnología utilizada para la creación de esta solución no fue orientada a poder crecer horizontalmente, por lo que se recomienda su migración de esta solución a servidores de aplicaciones que permitan esta característica

2. Dado el importante volumen de datos que maneja esta solución se recomienda que la tecnología de hardware y software sea de primer nivel.

BIBLIOGRAFÍA

[1] Kenneth C. Laudon Jane P. Laudon, Sistemas de Información Gerencial, Pearson Decimosegunda edición, 2012.

[2] David Vallejo Fernández, Carlos González Morcillo y Javier Alonso Albusac Jiménez, Programación Concurrente y Tiempo Real, Edlibrix Segunda edición, 2014

[3] Marta Beltrán Pardo, Fernando Sevillano Jaén, Cloud Computing, tecnología y negocio, Ediciones Paraninfo, S.A., 2013.

[4] Java Message Service https://es.wikipedia.org/wiki/Java_Message_Service, fecha de consulta Julio del 2015

[5] Basic JMS API Concepts https://docs.oracle.com/javaee/6/tutorial/doc/bncdx.html, fecha de consulta Julio del 2015