Facultad de Informática
Universidad Nacional de La Plata

Monitoreo de procesos y construcción de un tablero de control usando Portlets

Tesina de grado para Licenciatura en Sistemas

Alumno: Emiliano Losso
Directora: Dr. Patricia Bazán
Asesor Profesional: Lic. José Nicolás Martínez Garro

Índice General

Agradecimientos:

En primer lugar, quiero agradecer a mi familia por el gran apoyo y la confianza que depositaron en mí desde el inicio de la carrera. Sin su ayuda, traducida en motivación, comprensión y afecto, no sé si hubiera llegado a esta instancia o hubiera sido todo mucho más difícil. De modo que les dedico especialmente el presente trabajo a mis padres, mi hermana, mi cuñado y mis sobrinos; un particular agradecimiento a mi madre, quien ha estado presente en todo momento y a quien la culminación de la carrera le genera una enorme satisfacción.

También quiero agradecer a mi directora Patricia Bazán por darme la posibilidad de realizar este trabajo bajo su acompañamiento. Han sido de gran utilidad sus consejos y su conocimiento transmitido para el desarrollo del trabajo.
Del mismo modo, le agradezco a José Martínez Garro quien ha estado siempre presente brindando su experiencia para ayudar en lo que necesitaba.
Me resultó muy grato haber tenido la influencia de ambos; fue muy importante y por otra parte me parece admirable la dedicación, la paciencia, y la motivación puestas para que llegara a completar los objetivos.

Quiero dar un especial agradecimiento al Ingeniero Alfredo Prats, quien a distancia, ha tenido una gran predisposición para ayudar en los aspectos más técnicos del desarrollo.

Agradezco a mis compañeros y amigos de la carrera, cuya humildad y solidaridad han sido muy importantes para poder avanzar y sortear las dificultades que se iban presentando.

1

1. Introducción

En los últimos años se ha acrecentado el uso de la tecnología BPM (*Business Process Management o Gestión de procesos de negocio*) por parte de las organizaciones. Esto se debe, en parte, a su capacidad para gestionar y mejorar el rendimiento de un negocio optimizando sus procesos a través de la modelización, ejecución y medida de rendimiento dentro de un ciclo de mejora continua.
Los Sistemas de gestión de procesos de negocios (en inglés *Business Process Management Systems - BPMS*) son herramientas para representar y coordinar las actividades involucradas en un proceso de negocio. El ciclo de vida de los procesos de negocio considera las fases de diseño, medición, monitoreo, y análisis, contemplando en todos sus casos un ciclo de mejora continua [1].
Existen diversas implementaciones de BPMS y muchas de ellas poseen parcialmente las componentes necesarias para abordar el ciclo de vida de los procesos de negocio de manera completa.
En particular Bonita Open Solution es un BPMS open source robusto y muy utilizado el cual en su versión community o libre, carece de un componente de monitoreo de procesos.
Por consiguiente, resulta de gran utilidad la construcción de un componente que permita abordar de forma eficiente la fase de monitoreo de los procesos de negocio dentro de dicho BPMS.
El desarrollo de un tablero de control que permita aplicar los principios de una herramienta BAM (*Business Activity Monitoring o Monitoreo de Actividades de Negocio*), posibilitará realizar el seguimiento y monitoreo de los procesos para la optimización de los mismos.

1.1. Motivación

Las organizaciones han encontrado en la tecnología BPM una forma de conectar a sus usuarios con la información necesaria, de forma eficiente y en los plazos establecidos.
Como resultado del incremento en el volumen de datos a gestionar y en la necesidad de tomar decisiones rápidas a lo largo de múltiples unidades del negocio, los procesos se han vuelto más complejos por lo que se torna de gran utilidad la capacidad de realizar el monitoreo de los mismos y de su información pertinente; esto último se efectúa con el objetivo de poder tomar decisiones oportunas en el caso de situaciones de emergencia en las que los datos no se encuentran dentro de los parámetros normales establecidos.
La mayoría de las herramientas actuales cuentan con alguna funcionalidad para monitorear los procesos, sin embargo estas o bien son limitadas o bien se encuentran disponibles sólo para las versiones propietarias, tal como ocurre con el BMPS Bonita OS.
El hecho de contar con un componente que permita realizar de forma eficiente la fase de monitoreo de los procesos de negocio, permitirá tener un mayor conocimiento de la situación actual de la organización y optimizar así el rendimiento de los procesos. Tal componente usualmente se trata de un tablero de control que mediante un conjunto de indicadores mostrará el rendimiento actual de los procesos del negocio. De esta manera, se podrán tomar decisiones en tiempo real en el caso de situaciones de alerta marcadas previamente por los indicadores cuando los parámetros se encuentran en niveles anormales.
Por lo tanto, desde el punto de vista de la investigación, la motivación es el estudio, aprendizaje, e integración de nuevas tecnologías que brinden las herramientas para implementar una solución BAM. En cuanto al desarrollo de la aplicación, la motivación de la misma es mejorar el rendimiento

de los procesos de negocio dentro de una organización a partir del monitoreo de los mismos, realizando esta acción mediante un tablero de control que muestre la información pertinente durante su ejecución, principalmente en tiempo real.

1.2. Objetivo

El objetivo del siguiente trabajo es investigar e integrar tecnologías que permitan construir un tablero de control para monitorear los procesos que se ejecutan en el BPMS Bonita OS, utilizando portlets para su visualización.
Para esto se espera aplicar los principios de BAM en cuanto a recibir información en forma de eventos, analizarla mediante consultas y operaciones sobre la misma, y por último visualizarla con un tablero de control que provea dicha información en tiempo real y alertas para la toma rápida de decisiones.
Por otro lado, se requiere la elaboración de los KPI´s (*Key Performance Indicators o Indicadores clave de rendimiento*) determinando la información a mostrar y que resulte de utilidad para mejorar la performance de los procesos, así como el diseño de métricas y alertas.
Además es necesario implementar una arquitectura que permita el procesamiento de múltiples eventos en tiempo real y el almacenamiento de una gran cantidad de datos (*Big Data*).
Otra de las metas del trabajo es generar un impacto del tablero en el usuario final, de manera de que se pueda mostrar gran cantidad de información en un espacio reducido, que la misma sea clara, y que se adviertan de forma precisa los indicadores y las alertas para poder cambiar el rumbo de la organización en el caso de registrarse inconvenientes.

Los aportes de este trabajo incluyen:

- Integrar nuevas tecnologías acopladas en una arquitectura que sea capaz de procesar la información proveniente de los procesos de negocio de manera eficiente.
- Desplegar información que resulte útil en un tablero de control implementado mediante componentes denominados *portlets*, teniendo cada uno una función específica y pudiéndoselos agregar en el portal de Liferay.
  Dicho portal, se trata de un sitio web que ofrece al usuario, de forma fácil e integrada, el acceso a una serie de recursos e información relacionados a un mismo tema. Los portlets son componentes modulares de las interfaces de usuario gestionadas y visualizadas en el portal web.

RESULTADOS ESPERADOS

- Investigar tecnologías idóneas para la elaboración de un tablero de control que lleve a cabo el monitoreo de los procesos de Bonita OS, de manera tal de procesar un alto volumen de información relevante de los procesos en forma de eventos, y visualizarla en tiempo real mediante el tablero.
- Diseñar KPI´s que resulten útiles para determinar el rendimiento y la mejora de los procesos en general que se ejecutan en Bonita OS.
- Desarrollar una aplicación que permita integrar las tecnologías estudiadas y que tenga la capacidad de llevar a cabo el monitoreo de los procesos que hacen uso de la misma. Es importante que muestre de forma clara y precisa los KPI´s estipulados.
- Implementar un caso de estudio que permita aplicar en forma concreta todo el contenido investigado.

1.3 Organización del documento:

El presente documento se organiza de la siguiente manera:

Comenzando el marco teórico, en el Capítulo 2 se presentan los conceptos principales de BAM abarcando definiciones y detalles sobre los métodos que utiliza esta tecnología para realizar el monitoreo de las actividades de negocio. Se explica el procesamiento de eventos complejos (CEP) para la recepción de la información, y las distintas métricas utilizadas para realizar mediciones sobre esta. Además se describe el modelo y la arquitectura posible de una solución BAM.

En el Capítulo 3 se explican las características de los Tableros de Control, que constituyen el mecanismo visual idóneo para mostrar toda la información vinculada al monitoreo de procesos de negocio. Se analizan los conceptos más importantes que deben tener los tableros de control para resultar eficaces a los usuarios finales, y además se describe una forma conceptual para llevar a cabo su diseño.

En el Capítulo 4 se explican las tecnologías que se consideraron adecuadas para implantar el tablero de control: los portales web y los portlets. Se describen las particularidades que tienen los portales que resultan propicias para emular al tablero de control tanto visualmente como para que resulte efectiva la presentación de la información. Aquí concluye el marco teórico que da soporte conceptual a la implementación.

En el Capítulo 5 se muestra en detalle cómo se desarrolló el Tablero de Control. En primer lugar se describen las tecnologías o productos específicos que se escogieron e integraron para llevar a cabo la implementación. Luego se expone el caso de estudio que ejemplifica la funcionalidad lograda.

2

2. MONITOREO DE ACTIVIDADES DE NEGOCIO

En este capítulo se describirán los conceptos y las tecnologías que darán soporte al desarrollo del tablero de control que ejercerá el monitoreo de los procesos de negocio. Esto último se hará partiendo desde el procesamiento de eventos complejos (CEP), capaz de realizar la captura y filtrado de la información a partir de eventos en tiempo real. Luego, se prosigue con la explicación de los conceptos que involucran el monitoreo de actividades de negocio (BAM), tales como las métricas para medir la información de los procesos, los indicadores claves de rendimiento (KPIs) que determinan los umbrales críticos, las formas de visualizar la información procesada y las alertas como los tableros de control, así como una posible arquitectura y modelo de monitoreo para culminar describiendo un prototipo de solución BAM. Acto seguido se detallan tres formas de convergencia entre las tecnologías BAM y BPM que se implantarán en el caso de estudio posterior, culminando con la descripción de los conceptos de la Inteligencia de Negocio (BI), y la relación de esta última con el monitoreo de actividades de negocio.

2.1. Complex Event Processing (CEP)

En los últimos años ha surgido CEP (*Complex Event Processing o Procesamiento de Eventos Complejos*), un nuevo paradigma para las aplicaciones dirigidas por eventos que permite procesar gran cantidad de estos últimos, así como detectar nuevos de una mayor complejidad semántica, así como inferir conocimiento valioso para los usuarios finales.

Las arquitecturas orientadas a servicios o *Service-Oriented Architectures* (SOA) han surgido como una solución eficiente para la implantación de sistemas en los que la modularidad y las comunicaciones entre terceros son un factor clave, desarrollándose así aplicaciones distribuidas compuestas de componentes (servicios) reutilizables y compartibles. Estos componentes disponen de interfaces bien definidas independientes de sus implementaciones, lo que permite que estos sistemas puedan adaptarse fácil y rápidamente a las condiciones cambiantes del negocio. Sin embargo, estas arquitecturas no son adecuadas para aquellos entornos en los que hay necesidad de analizar continuamente toda la información que fluye por el sistema para detectar cuanto antes y de forma automática las situaciones que son críticas para los procesos de negocio.

Esta limitación puede suplirse con el uso añadido del procesamiento de eventos complejos (CEP) [2], una tecnología que proporciona un conjunto de técnicas que ayudan a hacer un uso eficiente de las arquitecturas dirigidas por eventos o *Event-Driven Architectures* (EDA).

CEP permite procesar y analizar grandes cantidades de eventos, así como correlacionarlos para detectar y responder en tiempo real a situaciones críticas del negocio.

Para ello, se utilizan patrones de eventos que inferirán nuevos eventos más complejos y con un mayor significado semántico, los cuales ayudarán a tomar decisiones ante las situaciones acontecidas.

Así pues, combinando el uso de SOA con CEP, podremos detectar eventos relevantes en sistemas complejos y heterogéneos. En la actualidad, la integración de EDA y SOA se conoce como SOA dirigida por eventos (ED-SOA) o SOA 2.0 [3], una extensión de SOA para responder a los eventos que ocurren como resultado de los procesos de negocio. Este nuevo enfoque hará posible que los

servicios no intercambien únicamente mensajes entre ellos, sino que también puedan publicar eventos y recibir notificaciones de eventos de distintos servicios.
Para lograrlo, se requerirá de un bus de servicios empresariales o *Enterprise Service Bus* (ESB) que actuará como capa de integración para la transformación, enriquecimiento y encaminamiento de mensajes entre servicios de diferentes aplicaciones.
El procesamiento de eventos es importante para entender el monitoreo de actividades de negocio, pues la salida de un evento es la fuente de información de una solución BAM.

2.1.1 Definición y características:

CEP [2] es una tecnología que proporciona un conjunto de técnicas que favorecen el uso eficiente de las arquitecturas EDA. Se basa en el filtrado de eventos irrelevantes y en el reconocimiento de patrones de los eventos que sí son relevantes. Además, hace posible la detección de nuevos eventos más complejos, conocidos como "situaciones"; es decir, CEP permite la captura y correlación de cualquier tipo de evento con el fin de detectar situaciones de una mayor complejidad semántica e inferir conocimiento valioso para los usuarios finales.
La característica principal de estos eventos complejos procesados mediante tecnología CEP es que pueden ser identificados e informados en tiempo real, reduciendo la latencia en la toma de decisiones, a diferencia del software tradicional de análisis de eventos, que no funciona en tiempo real. En este contexto, siendo que el tiempo es un recurso crítico, cualquier reducción en la latencia resulta determinante.
Esta tecnología es novedosa e innovadora y se afirma que necesitaremos, cada vez más, sistemas CEP por varios motivos [4]. Por un lado, las organizaciones necesitan analizar eventos en tiempo real, debido a que la compañía que analice más rápido su flujo de datos y responda a los resultados obtenidos será la más competitiva. Por otro lado, el software tradicional de análisis de eventos no funciona en tiempo real, por tanto, se pierde tiempo en analizar eventos complejos anteriores, y las bases de datos no están optimizadas para llevar a cabo esta labor.
Así pues, CEP es una tecnología fundamental para aplicaciones que deban responder a situaciones que cambien rápidamente y de forma asíncrona (donde las interacciones no tienen que ser transacciones), gestionar excepciones o reaccionar rápidamente a situaciones inusuales; y que requieran adaptabilidad y bajo acoplamiento.
CEP tiene algunas similitudes y diferencias con SOA. En cuanto a las similitudes, ambos enfoques presentan modularidad, bajo acoplamiento y flexibilidad.
Algunas de las diferencias principales son las siguientes: por un lado, en SOA la interacción está basada en servicios (un usuario debe conocer cuál es el productor del servicio y la interfaz para enviarle las peticiones), mientras que CEP aplicado a EDA es reactivo y más desacoplado, ya que los eventos son generados por los productores de eventos y serán los consumidores los que se encarguen de interceptarlos y procesarlos. Por otro lado, mientras que los procesos SOA utilizan eventos para dirigir el flujo de control [5] (estos procesos pueden tanto enviar como recibir eventos), los motores CEP analizan y correlacionan continuamente estos eventos para evaluar si se cumplen las condiciones definidas en alguno de los patrones de eventos almacenados en estos motores.

2.1.2 Clasificación de los sistemas CEP:

Los métodos para detectar eventos en sistemas CEP pueden ser divididos en tres grupos, según [6]:

1) Enfoques que proporcionan sus propios lenguajes de consulta para analizar los flujos de eventos.
2) Técnicas que proponen definir los patrones de eventos utilizando XML.

3) Métodos que permiten la definición de eventos compuestos y reglas a partir de una interfaz de usuario gráfica.

Los autores afirman que los dos primeros enfoques requieren de expertos que tengan un gran conocimiento en estos lenguajes.
Por otro lado, [7] realiza una clasificación de los lenguajes de procesamiento de eventos atendiendo al estilo del lenguaje:

- Lenguajes orientados al flujo de eventos.
- Lenguajes orientados a reglas; estos a su vez pueden clasificarse en:
    - Reglas de Evento-Condición-Acción (ECA).
    - Reglas de inferencia.
    - Programación lógica.
- Lenguajes imperativos.

Actualmente, la mayoría de estos lenguajes son propietarios y no existe ninguno estándar.
Por otro lado, [8] en lugar de realizar la clasificación atendiendo al lenguaje de procesamiento, la hacen en función de las herramientas CEP disponibles en la actualidad, clasificadas en dos categorías: herramientas comerciales, y herramientas académicas y libres.

2.1.3 Diseño de patrones de eventos complejos:

A partir de la lectura del trabajo de varios autores, se infiere que [19] construye una plataforma SOA utilizando únicamente tecnologías de código abierto. Desarrolla cómo integrar las tecnologías ESB (*Enterprise Service Bus*), que permiten la conexión entre eventos y servicios, y CEP, entre otras. Gracias a que la tecnología CEP puede ser integrada en la implementación de los servicios, la visibilidad en tiempo real en los sistemas es una realidad. [20] Explica cómo usar, diseñar y construir aplicaciones de procesamiento de eventos. También presentan un modelo independiente de la plataforma para facilitar el diseño de estas aplicaciones, así como un editor gráfico para poderlo aplicar. Además, se muestra la construcción de una aplicación de ejemplo que utiliza procesamiento de eventos utilizando diferentes estilos de programación.

Estos autores clasifican las aplicaciones CEP en 5 categorías, que no son excluyentes entre sí:

1) Observación: monitorizan un sistema o proceso buscando comportamientos excepcionales y generando alarmas cuando dicho comportamiento ocurra.
2) Diseminación de la información: permiten entregar la información correcta al consumidor correcto en el tiempo correcto.
3) Comportamiento operacional dinámico: reaccionan a los eventos como parte de las transacciones del negocio, logrando decisiones de baja latencia y reacciones rápidas a las amenazas y oportunidades.
4) Diagnósticos activos: diagnostican problemas basados en síntomas y los resuelven.
5) Procesamiento predictivo: mitigan o eliminan eventos antes de que ocurran

Existen algunos trabajos que tratan sobre el diseño de patrones de eventos complejos, a saber: [9] propone un esquema multidimensional de clasificación de patrones CEP. La primera categoría distingue entre patrones CEP exitosos ante un problema que ocurre frecuentemente y antipatrones, que son similares a los patrones pero su uso produce consecuencias negativas; la segunda categoría engloba los patrones atendiendo al nivel de abstracción; la tercera categoría clasifica a los patrones

CEP según sus objetivos; y, por último, los patrones clasificados según el nivel de gestión (estratégicos, tácticos y operacionales).
Mientras que Paschke define la plantilla de un patrón de eventos con tan sólo los elementos nombre, problema, solución, consecuencias y ejemplos, [10] también determina, entre otros, el contexto en el que el patrón puede aplicarse, una especificación detallada de sus aspectos estructurales, así como una recomendación para su implementación.

2.1.4 Event Processing Query Languajes:

Existen varios lenguajes de consulta para el procesamiento de eventos, para los cuales resulta conveniente tener noción de los patrones básicos de los lenguajes SQL. Sin embargo, en el procesamiento de eventos se requieren expresiones más detalladas. Además, muchas veces se usa una ventana de tiempo, y se observa la concurrencia de eventos durante el período de tiempo definido por la ventana. Abajo se mencionan los lenguajes más importantes:

- EPL (*Event Processing Language*): CEP ofrece un lenguaje específico de dominio para su procesamiento. Dicho lenguaje es EPL; es decir, Lenguaje de Procesamiento de Eventos, siendo el mismo un lenguaje declarativo utilizado para tratar con los datos de eventos que suceden en tiempo real. Las consultas CEP permiten agregar o filtrar datos de entrada en base a reglas preestablecidas. La figura 2.1 muestra cómo fluyen los datos en una aplicación CEP a partir de dos flujos de entrada y se agregan o se filtran en función de los de la aplicación. A continuación, los datos resultantes pasan por un proceso de unión donde, efectivamente, se habrán convertido en un nuevo conjunto de datos que estarán listos para realizar las operaciones que se precisen.

Figura 2.1. *Arquitectura de flujo de datos en una aplicación CEP*

Además, este lenguaje, no sólo permite la consulta y el procesamiento de eventos sino que ofrece la posibilidad de añadir una "dimensión temporal". EPL es un lenguaje muy similar al lenguaje de consulta de bases de datos relacionales SQL (de hecho comparte alguna de sus mismas cláusulas: *SELECT, FROM, WHERE, GROUP BY, HAVING y ORDER BY*). Incluso puede decirse que EPL amplía y extiende a SQL en la definición de reglas temporales y en la aplicación de reglas no lineales para el procesamiento de eventos.

- CQL (*Cassandra Query Language*): Apache Cassandra es una base de datos no relacional distribuida y basada en un modelo de almacenamiento de clave-valor, escrita en Java, que permite manejar grandes volúmenes de datos en forma distribuida.

Cassandra tiene su propio lenguaje para realizar consultas sobre los datos: Cassandra Query Language. Para hacerlo lo más usable posible, CQL es prácticamente igual que un lenguaje SQL, incluso en el concepto de tablas con filas y columnas, a pesar de que el modelo de datos de Cassandra sea tan diferente. La principal diferencia es que CQL no permite realizar consultas con *joins* ni subconsultas.

2.2 Business Activity Monitoring (BAM)

El concepto de monitoreo de actividades de negocio o *Business Activity Monitoring* (BAM) fue propuesto originalmente por investigadores de la empresa de investigación y consultoría Gartner Inc. La idea central es la recopilación, organización, análisis y visualización de datos obtenidos en tiempo real, acerca de las actividades ejecutadas en un proceso de negocio. En palabras de [11], BAM proporciona acceso en tiempo real a indicadores críticos del desempeño del negocio para mejorar la velocidad y la efectividad de las operaciones. También beneficia a las organizaciones en la medida en que permite tomar de forma rápida decisiones con mejor información sobre el desempeño de los procesos de negocio y, más rápidamente, identificar y resolver problemas durante la ejecución del proceso [12].
En la práctica, el monitoreo de las actividades del negocio se centra en tomar mediciones de una gran variedad de métricas sobre la ejecución del proceso de negocio, como volumen de instancias, duración, costos, errores y condiciones especiales. Las herramientas BAM mejoran el desempeño de las organizaciones porque aseguran que los procesos de negocio operan como se esperaba, que cumplen con los compromisos y propósitos de la organización, y reducen la latencia entre el momento en que se realiza la medición del desempeño y el momento en que se toman las decisiones [13]. Esencialmente, las herramientas BAM intentan proveer a las organizaciones lo que los instrumentos de aeronavegación proporcionan a los pilotos de las aeronaves: datos en tiempo real sobre el comportamiento de las operaciones.

2.2.1 Definición y características:

El Monitoreo de Actividades de Negocio (BAM) es un enfoque emergente de un conjunto de tecnologías que estudia los eventos del negocio con el objetivo de que los gerentes puedan reaccionar de una manera más rápida a los problemas y oportunidades [14].
BAM es un término de Gartner que define el concepto de proveer acceso a indicadores de desempeño críticos del negocio para mejorar la velocidad y efectividad de las operaciones del negocio. En su nivel más amplio, BAM es la convergencia de la inteligencia operacional del negocio integrada en una aplicación de tiempo real para lograr las metas del negocio, pero habilitada a través de los avances en las tecnologías de la información. BAM no es un monitoreo a nivel de aplicación, sino que es un monitoreo complejo de la mezcla de aplicaciones y sus interacciones [15].
A diferencia del monitoreo en tiempo real, BAM obtiene su información desde múltiples aplicaciones de sistemas y de otras fuentes internas y externas, habilitando una perspectiva más amplia y rica sobre las actividades del negocio. Cuando ciertas condiciones se presentan, la solución genera alertas. Las empresas que se están iniciando con BAM han encontrado que la tecnología cambia la forma en que su organización es operada [14].
El potencial real de BAM se presenta a un nivel de negocios: habilitar nuevas estrategias de negocio, reduciendo los costos de operación, mejorando el desempeño de los procesos y otras áreas tangibles de interés para la administración [15].

En su nivel más amplio, BAM es la convergencia de diferentes clases de tecnologías, incluyendo:

- Inteligencia de Negocios Operacional.
- Integración de aplicaciones en tiempo real.

- Administración del negocio.

Las soluciones de BAM se conforman esencialmente por cuatro componentes:

- Eventos de negocio, es decir, eventos que son importantes en el contexto de la ejecución de una actividad del negocio.
- Actividades del negocio, es decir, actividades incluidas en un proceso de negocio para brindar un servicio o funcionalidad requerida para cumplir con una necesidad específica de la organización.
- Métricas, es decir, mediciones sobre propiedades específicas de una actividad del negocio que puede utilizarse para monitorear el desempeño de la operación.
- Indicadores clave de desempeño (KPI), que son métricas utilizadas para medir el progreso de un objetivo organizacional a nivel estratégico.

Al revisar la oferta de herramientas de BAM, se encuentra que las funcionalidades comunes que estas ofrecen son:

- Captura instantánea de datos sobre KPI's en diferentes niveles del proceso.
- Presentación sintética de los datos de desempeño en tableros de control (*dashboard*) configurables por el usuario y con diferentes tipos de visualización de los datos.
- Notificaciones automáticas sobre problemas o posibles violaciones a políticas y niveles de desempeño establecidos para el proceso de negocio.
- Generación de reportes sobre el comportamiento y tendencias de los indicadores de desempeño del proceso de negocio.

Estas funcionalidades parecen similares a las ofrecidas por herramientas de Inteligencia de Negocio (*Business Intelligence* – BI). Sin embargo, es necesario hacer una distinción entre las dos, pues, aunque estas están ubicadas en la misma categoría de herramientas de análisis del negocio, tienen una diferencia fundamental sobre el origen de los datos analizados. Las herramientas de Inteligencia de Negocio también tienen que ver con el entendimiento y la optimización del desempeño del negocio, pero la diferencia es que estas se enfocan en el análisis de datos históricos que son utilizados para explorar, utilizando técnicas de minería de datos, el comportamiento que se ha tenido en períodos anteriores y así poder identificar tendencias. Por otra parte, las herramientas de BAM tratan con datos representativos de lo que está sucediendo actualmente en el negocio, es decir, en tiempo real.

Optimización de los procesos de negocio: es una actividad que debe ser realizada de forma iterativa durante el tiempo de vida de los procesos. Esta actividad debe realizarse luego de que la solución sea implantada, como así también luego de ocurrir un cambio significativo en la estructura del proceso.
La optimización de procesos de negocio es una tarea difícil y requiere un buen conocimiento de los mismos dentro de la organización. Sin embargo, incluso con la utilización de las herramientas más potentes, la eficiencia de la optimización depende en cierto modo de las habilidades de la persona responsable.
BAM proporciona información en tiempo de ejecución sobre el negocio, permite el análisis en tiempo real de los procesos, muestra cuellos de botella y tareas poco fiables, mide el tiempo de cada tarea y proporciona herramientas para visualizar toda esa información.
En cuanto a la optimización de los procesos hay una aproximación que es más un "deseo" que una realidad. Se puede involucrar principios de inteligencia artificial en la optimización del negocio, pues una gran cantidad de algoritmos de esta disciplina puede ser útil en las tareas de optimización.

Sin embargo esto hoy es más bien una visión a futuro y en la actualidad sólo podemos ver algunos casos de su aplicación real.

2.2.2 Métricas de proceso de negocio:

Para ejercer el monitoreo de los procesos de negocio y por ende llevar a cabo la optimización de los mismos, es necesario tener mediciones de dichos procesos las cuales permitan determinar qué elementos de los mismos resultan malos e ineficientes. BAM ofrece diversas técnicas que permiten medir varios aspectos como el tiempo, costo, performance y muchos otros.

Términos básicos de medición:

- **Instancia de monitoreo**: generalmente constituye una instancia de proceso. Sin embargo, existen ocasiones en que se puede monitorear un proceso que consta de varios pasos que son identificables por los eventos entregados al monitor y también incluyen las secuencias realizadas fuera del motor de ejecución, por ejemplo uno implementado en BPEL. BPEL (*Business Process Execution Language*) constituye un lenguaje estándar para la integración y automatización de procesos; los procesos de negocio programados con BPEL serán capaces de ejecutarse en diferentes plataformas que cumplan dicho estándar, ofreciendo a los clientes una mayor libertad de elección.
  Un ejemplo de esa instancia de monitoreo puede ser el "proceso de pedido" de un artículo a la venta: varios procesos de pedidos se puede iniciar al mismo tiempo y se desea monitorear su estado.
- **Métrica**: es la medida de un atributo de la instancia del proceso, determinando el aspecto del mismo que se desea monitorear: duración, coste, nombre, etc. Constituye un componente elemental que, una vez definido, permite la composición de estructuras más complejas como los KPIs.
- **KPIs** (Indicadores Claves de Rendimiento): son mediciones cuantificables, pactadas de antemano, que reflejan los factores críticos de éxito (de la empresa, departamento, proyecto).
  Desde el punto de vista del negocio, las KPIs deben concretarse durante el período de definición de objetivos de la entidad, ya que estas unidades permitirán considerar su alcance o no. Esto es tarea del analista de negocio, quien define en documentos las estrategias y objetivos de la organización.
  Desde el punto de vista de la implementación, una KPI es una función de agregación sobre las métricas de todas o algunas instancias de monitoreo elegidas. Ejemplos de KPI pueden ser:
  - el importe medio de las órdenes enviadas por departamento, por mes
  - cuántas ordenes se tramitan en una hora
  - costo de producir una pieza
  - Tiempo que tarda en completar una actividad

  Para mayor detalle ver el siguiente inciso (2.2.3).
- **Trigger/Alarma**: es una pareja de regla y acción. La regla define la condición que genera el disparo del trigger. La acción define la actividad que se realiza cuando el trigger es disparado. Una vez definido el trigger puede ser usado por las métricas y los KPIs. Ejemplo:
  - Una vez que "el tiempo de cumplimiento de la orden" excede el tiempo definido, se dispara el trigger “tiempo excedido” el cual iniciará un proceso que continúa luego con el orden particular y envía una advertencia a la persona responsable.

- **Temporizador**: funciona de manera similar a un cronómetro. Es iniciado por un trigger y mide el tiempo bajo ciertas condiciones. El temporizador también puede desactivar un trigger cuando se alcanzó un intervalo.
- **Contador**: es un simple mecanismo que cuenta eventos iterativos. Cada vez que ocurre un evento el contador se incrementa en uno.
- **Alcance de monitoreo**: son diferentes alcances de monitoreo sobre una misma solución y fuente de eventos. Cada alcance define sus propias métricas, KPIs y triggers. Es un concepto a emplear cuando existen distintos departamentos interesados en monitorear un mismo recurso.
- **Situaciones de negocio**: uno de los objetivos de BAM es permitir a los procesos poder reaccionar bajo ciertas condiciones del negocio. Por lo general, cuando una métrica o KPI alcanza un determinado valor, se pretende iniciar un proceso que va a reaccionar ante la situación de una manera adecuada.

2.2.3 Indicadores Claves de Rendimiento (KPIs):

Uno de los factores trascendentales para el desempeño de una organización es la medición. Los KPIs reflejan y miden las guías estratégicas del negocio, representando las actividades que garantizan el éxito futuro. Estos indicadores de valor mueven la organización en la dirección correcta, para alcanzar sus metas financieras y organizacionales previamente establecidas. Una KPI es una parte clave de un objetivo medible, por ejemplo: "Incrementar el promedio de utilidad por cliente de $10 a $15 para finales del año 2017". En este caso, “promedio de utilidad por cliente” es el KPI.

Los indicadores se usan para calcular:

- Tiempo que se utiliza en mejorar los niveles de servicio en un proyecto dado.
- Nivel de la satisfacción del cliente.
- Tiempo de mejoras de asuntos relacionados con los niveles de servicio.
- Impacto de la calidad de los recursos financieros adicionales necesarios para realizar el nivel de servicio definido.

Es necesario que una organización pueda identificar sus propios KPIs. La clave para esto es:

- Tener predefinido de antemano un proceso de negocio.
- Tener claros los objetivos/rendimiento requeridos en el proceso de negocio.
- Tener una medida cuantitativa/cualitativa de los resultados y que sea posible su comparación con los objetivos.
- Investigar variaciones y ajustar procesos o recursos para alcanzar metas a corto plazo.

Cuando se definen KPIs se suele aplicar el acrónimo SMART, ya que los KPIs tienen que presentar las características que se muestra en la Figura 2.2.

- Específicos (Specific)
- Medibles (Measurable)
- Alcanzables (Achievable)
- Realistas (Realistic)
- a Tiempo (Timely)

Figura 2.2. *Diagrama de cualidades de los KPIs*

Los componentes de un indicador son:

- Medida: grandeza cualitativa y cuantitativa que permite clasificar las características, atributos, resultados y consecuencias de los productos, procesos o sistemas.
- Índice: valor de un indicador en cierto momento.
- Estándar de comparación: es un índice arbitrario dado como aceptable para un indicador y un estándar comparativo para evaluación de cumplimiento.
- Meta: índice orientado por un indicador en relación al estándar de comparación para ser alcanzado durante un cierto tiempo.

En la identificación y selección de un indicador es importante considerar un conjunto de criterios de calidad básicos, que garanticen su posterior operación. Los criterios centrales de calidad de un indicador son:

- La selectividad o importancia: proporciona información acerca de las variables claves del producto, proceso o sistema.
- Simplicidad y claridad o comprensibilidad: los indicadores deberían ser tan simples como comprensibles, tanto como sea posible, a pesar de llevar el mensaje y el significado deseado. Los nombres y expresiones deberían ser fáciles de entender y conocidas por todos los usuarios.
- Representatividad: la capacidad de demostrar, sobre una amplia visión, las más importantes y críticas etapas de un proceso. Los datos innecesarios no deberían recolectarse. Por otra parte, los datos importantes deberían ser precisos, respondiendo a los objetivos y recolectados en la fuente de datos correcta.
- Indagación: los datos deben ser fáciles de investigar, sean estos para registrar o mantener la información.
- Comparabilidad: los indicadores deberían ser fácilmente comparables con las propias referencias, o con referencias externas.
- Estabilidad: procedimientos generados sistemática y constantemente.
- Relación costo-efectividad: diseñados para ser económicamente efectivos. El beneficio en relación al costo, debería satisfacer los niveles de aspiración. Existen algunos indicadores que son muy difíciles de operar para justificar su desarrollo.

Para asegurar la constitución correcta de un indicador, o KPI, es imprescindible que sea:

- Sencillo y claro: permitir a todos la identificación rápida de las informaciones claves.

- Fácil y rápido de elaborar: para asegurar su publicación regular.
- Pertinente: contener los datos relativos a las actividades estudiadas.
- Adaptado a cada función, oficio, proyecto o más generalmente a cada necesidad.

Formulación e implantación de indicadores.

La formulación de indicadores requiere un conjunto de pasos necesarios para asegurar los principios de calidad del sistema de medición de desempeño y su puesta en servicio al interior de la organización.
Estos pasos son:

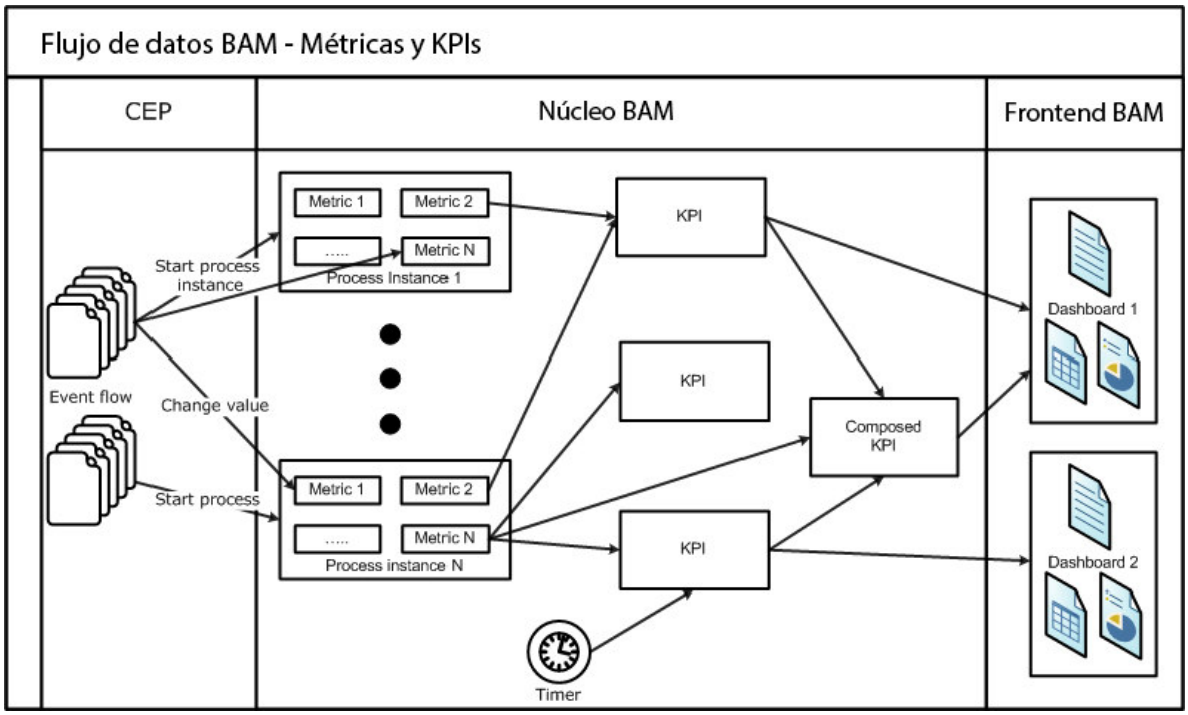
1) Identificar el proceso: en la identificación de los procesos se debe tener claro que lo que se desea medir es muy importante. Generalmente hay muchos procesos y funciones y cada uno necesita potencialmente medición de desempeño. Si hay múltiples procesos, se debe considerar aquellos que tienen impacto en los objetivos estratégicos de la organización y a su vez de estos, seleccionar aquellos que son más importantes para el cliente, tanto interno como externo. Estos procesos pasan a denominarse "procesos claves".
2) Identificar actividades críticas a medir: es importante elegir sólo las actividades críticas para ser medidas. La medición de estas actividades permite controlarlas. El control, o mantener las cosas en curso, no es algo que se haga de manera abstracta. El control es aplicado a una actividad crítica específica. Cuando se seleccione la actividad, el enfoque de la medición debe orientarse más al área crítica que a las personas. Las actividades críticas son aquellas que impactan significativamente en las categorías de indicadores identificadas en el punto anterior. En todos los niveles de administración, las actividades críticas impactan en las prioridades administrativas, las metas organizacionales y las metas para los clientes externos.
3) Establecer metas de desempeño o estándares: estas son necesarias porque de otra manera no habría una base lógica para elegir qué medir, qué decisión tomar, o qué acción ejecutar. Las metas pueden ser una directriz de la administración superior, o pueden establecerse en respuesta a las necesidades o quejas del cliente. Para cada actividad crítica seleccionada para medición, es necesario establecer una meta de desempeño o estándar, con el fin de disponer de una referencia para evaluar e interpretar las lecturas de los indicadores de estas actividades.
4) Establecer medición de desempeño: este paso involucra realizar varias actividades que continuarán construyendo el sistema de medición de desempeño. Cada medida de desempeño consiste en una unidad de medida definida, un censor para medir o grabar los datos primarios, y una frecuencia con la cual la medición es hecha. Para desarrollar una medida, el equipo deberá traducir "lo que se desea conocer", al interior de la medición de desempeño; deberá identificar los datos primarios que generarán la medición de desempeño, determinar dónde localizar los datos primarios,identificar el censor o instrumento de medición para coleccionar los datos y por último determinar que tan a menudo debe hacerse la medición.
5) Identificar las partes responsables: los anteriores pasos son primariamente actividades de equipo. Para continuar el proceso de medición de desempeño, debe identificarse al funcionario responsable de la medición y de la toma de decisiones en las acciones de control y mejoramiento del proceso. En muchos casos una persona es responsable de todo el sistema. El responsable deberá coleccionar los datos, analizar y reportar el actual desempeño, comparar el actual desempeño con las metas y/o estándares, determinar si las acciones correctivas son necesarias y por último hacer cambios.
6) Coleccionar los datos: la determinación de la conformidad depende de la significancia y validez de los datos. Antes de comenzar a coleccionar una cantidad de nuevos datos, es oportuno revisar los ya existentes para estar seguro de que se ha extraído toda la información

que se puede obtener. Los datos son un conjunto de hechos presentados en forma cuantitativa o descriptiva. Obviamente los datos deben ser bastante específicos para proporcionar información relevante. Hay dos tipos de datos; los datos medidos, o variables del proceso; y los datos contados, o atributos del proceso.

7) Analizar y reportar el desempeño actual: antes de perfilar una conclusión sobre los datos, se debería verificar que el proceso de colección de los mismos satisface los requerimientos de responder a las interrogantes originales, que evidencia la inexistencia de sesgos en el proceso de recolección, que se ajusta a la cantidad de observaciones del proceso especificadas y que además dispone de suficientes datos para bosquejar conclusiones significativas.
8) Comparar el actual desempeño con las metas o estándares: dentro de su intervalo de control, los funcionarios responsables comparan el actual desempeño con la meta o estándar. Si hay variación, se autorizan las acciones y se hace un informe al responsable de la toma de decisiones. Una vez que es establecida inicialmente una comparación en contra de las metas o estándares, existen varias alternativas disponibles para acciones posibles. Una es olvidarlo, si la variación no es significativa; la otra es tomar acciones correctivas, si las variaciones son significativas; y la última, es establecer nuevos desafíos si existen condiciones para hacerlo.
9) Determinar si las acciones correctivas son necesarias: este es un paso de decisión. Se puede cambiar el proceso, o cambiar la meta. Si la variación es grande, puede haber un problema con el proceso, y se necesitará hacer correcciones para recuperar el desempeño y llevarlo hacia la meta deseada. Para direccionar este potencial problema, se puede formar un equipo de mejoramiento de calidad o desarrollar un análisis de causa efecto. También se debería considerar analizar si la meta era realista. Si la variación es pequeña, el proceso probablemente está en buena forma, pero se debería considerar re-evaluar las metas para hacerlas más desafiantes. Por lo tanto, si se hacen cambios al proceso, se debería re-evaluar las metas para asegurar que todavía son viables.
10) Hacer cambios para traer el proceso hacia las metas o estándares: este es el primer paso en cerrar el lazo de la retroalimentación. Los cambios comprenden un número de acciones llevadas a cabo para lograr uno o más de los objetivos de corrección, siendo estos: remoción de los defectos, remoción de las causas de los defectos, lograr un nuevo estado de desempeño del proceso y mantener o destacar la eficiencia y la efectividad del proceso.
11) Determinar si nuevas metas o nuevas medidas son necesarias: esta decisión dependerá de tres factores fundamentales: el grado de éxito en el logro de objetivos anteriores; la extensión de cualquier cambio al alcance del proceso de trabajo; y las bondades de las medidas actuales para comunicar el estado de mejoramiento relativo a los procesos críticos de trabajo.

2.2.4 Modelo de monitoreo:

Las buenas decisiones antes y durante el desarrollo del modelo de monitoreo son claves para monitorear eficientemente. Antes de empezar se debe especificar con claridad lo que se quiere monitorear. Todo el desarrollo está principalmente hecho con la metodología *TopDown* (de lo general a lo particular, es decir cada parte del sistema se refina diseñando con mayor detalle). La entrada para el desarrollo del modelo de monitoreo son definiciones de KPIs vinculados con las definiciones de los objetivos de negocio. Se observa el flujo de datos BAM en la figura 2.3.



Definiciones de KPIs: las entradas del modelo de monitoreo deben ser los objetivos del negocio y los KPIs. Normalmente se definen antes de que se desarrolle la solución completa del proceso de monitoreo. Estas entradas derivan de documentos de análisis de objetivos. Cada documento de análisis de objetivos bien hecho debería contener una definición exacta de los indicadores clave de rendimiento. Los KPIs generalmente contienen el valor del objetivo y los rangos esperados de ese valor. Se espera que el valor del objetivo sea igual o aproximado al rango o valor esperado.

Definiciones de Métricas: una vez definidos los KPIs, se procede a analizar y definir las métricas que se utilizarán. De modo que se establece una lista de métricas, contadores y temporizadores que se requieren para calcular los KPIs. Es posible agregar métricas adicionales que no se usan para el cálculo de KPIs, pero que pueden ser interesantes y de utilidad.

Interacciones y situaciones de negocio: una vez definidas todas las métricas, se definen las interacciones entre ellas, los triggers y las acciones que estos disparan. Las acciones constituyen una reacción ante una determinada situación de negocio. Las reacciones son usualmente un envío de advertencia a un canal particular, o el inicio de otro proceso que reacciona ante la situación.

2.2.5 Visualización de resultados:

Esta sección describe la parte *front-end* de BAM que se debe mostrar al usuario final. En el campo de Inteligencia de Negocio hay muchas técnicas de visualización tales como los tableros de control que contienen medidores, gráficos, figuras, entre otros componentes. También existen técnicas más avanzadas como las vistas multidimensionales OLAP (*On-Line Analytical Processing o Procesamiento Analítico en Línea*). Las técnicas de visualización más relevantes son:

- Dashboards (tableros de control): tal como el tablero de control de un avión, representa una vista que permite controlar un conjunto de mediciones en un área pequeña. Hoy en día, los tableros de control (Figura 2.4) se usan con frecuencia en varias herramientas de Inteligencia de Negocio que conforman un superconjunto de herramientas BAM. Los tableros de control son configurables y

presentan un gran número de componentes de visualización y diseño que podemos usar de acuerdo a nuestras necesidades.

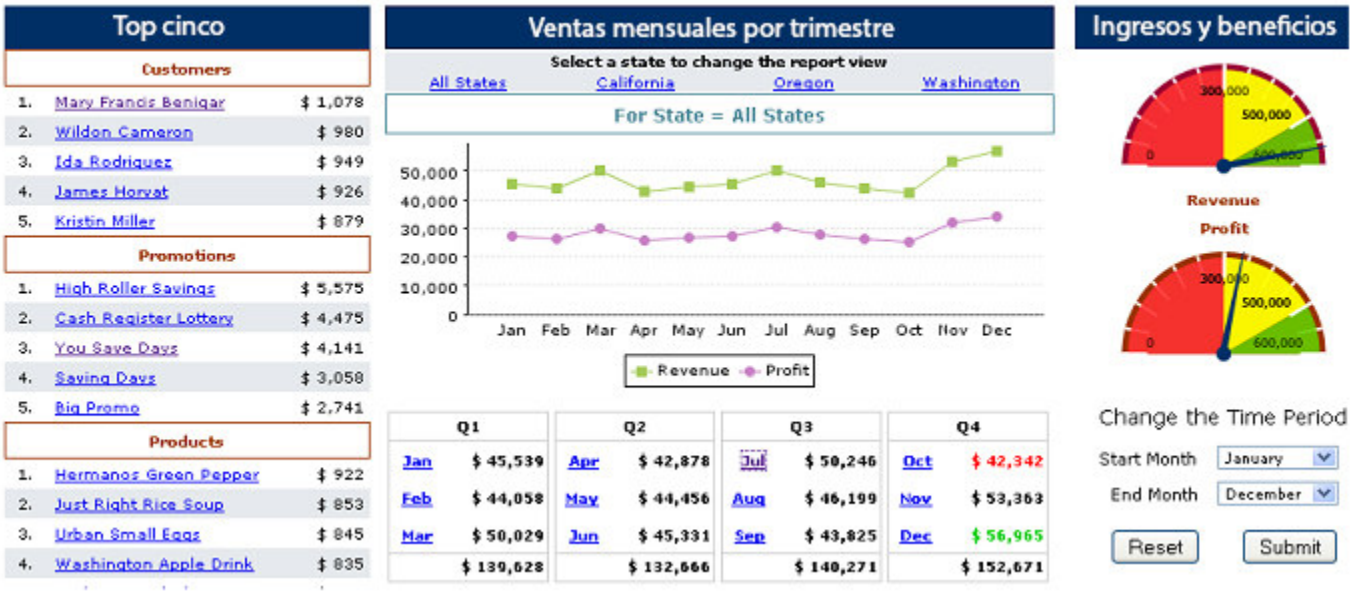


Figura 2.4. *Ejemplo de Tablero de Control*

En el ámbito de BAM los elementos básicos que aparecen en los tableros de control son:

- **Tablas de información**: una de las funciones elementales que se espera de una interfaz BAM en un tablero de control es información básica sobre las instancias de proceso. Aquí se desea ver los valores de métricas para cada instancia en una vista general, que normalmente se presenta en una tabla de información.
- **Medidores** (*gauges*): son elementos de visualización simple generalmente utilizados para expresar el valor de un KPI. El Objetivo y los rangos esperados suelen ser resaltados con un color diferente.

- Análisis OLAP: el procesamiento analítico en línea es una técnica ampliamente utilizada para analizar datos multidimensionales. Tiene sus raíces en el análisis puro de los datos y las bases de datos, y hoy en día es ampliamente utilizado en la Inteligencia de Negocio, informes financieros, pronósticos y muchos otros campos. OLAP permite consultar datos multidimensionales y visualizar los resultados en tablas (Figura 2.5) o en gráficos (Figura 2.6)

Dimensiones				Producto		
				Todos los Productos		
				Food	Non-Consumable	Drink
				Measures	Measures	Measures
Store	Store Country	Store State	Store City	Store Sales	Store Sales	Store Sales
All Stores	USA	CA	Beverly Hills	3,635.77	931.04	498.19
			Los Angeles	2,867.72	760.61	289.16
			San Diego	3,071.47	831.25	393.35
			San Francisco	258.79	36.46	32.08
		OR	Portland	2,676.25	765.55	369.03
			Salem	3,276.66	934.77	323.73
		WA	Bellingham	273.03	88.93	36.82
			Bremerton	3,043.13	748.85	443.62
			Seattle	3,227.56	840.34	376.16
			Spokane	2,838.97	856.93	413.32
			Tacoma	3,881.21	1,038.24	456.77
			Walla Walla	249.28	81.64	27.46
			Yakima	1,447.37	331.90	224.84

Figura 2.5. *Ejemplo de Tabla BAM*

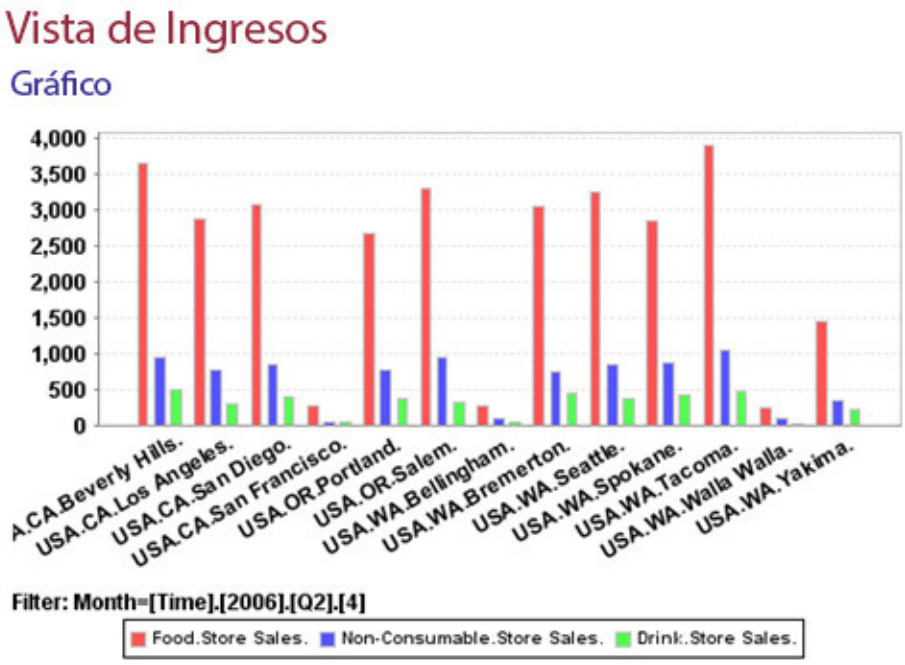


Figura 2.6. *Ejemplo de Gráfico de Barras BAM*

2.2.6 Arquitectura:

En este apartado se muestra un ejemplo de arquitectura de una solución BAM extraído de [16]. Antes de describir la arquitectura en detalle, se deben aclarar los flujos de documentos en la solución BAM.

- Flujo de Documentos: El diagrama de flujo de documentos de la Figura 2.7 muestra cómo los artefactos tales como documentos y datos se transforman durante el desarrollo.

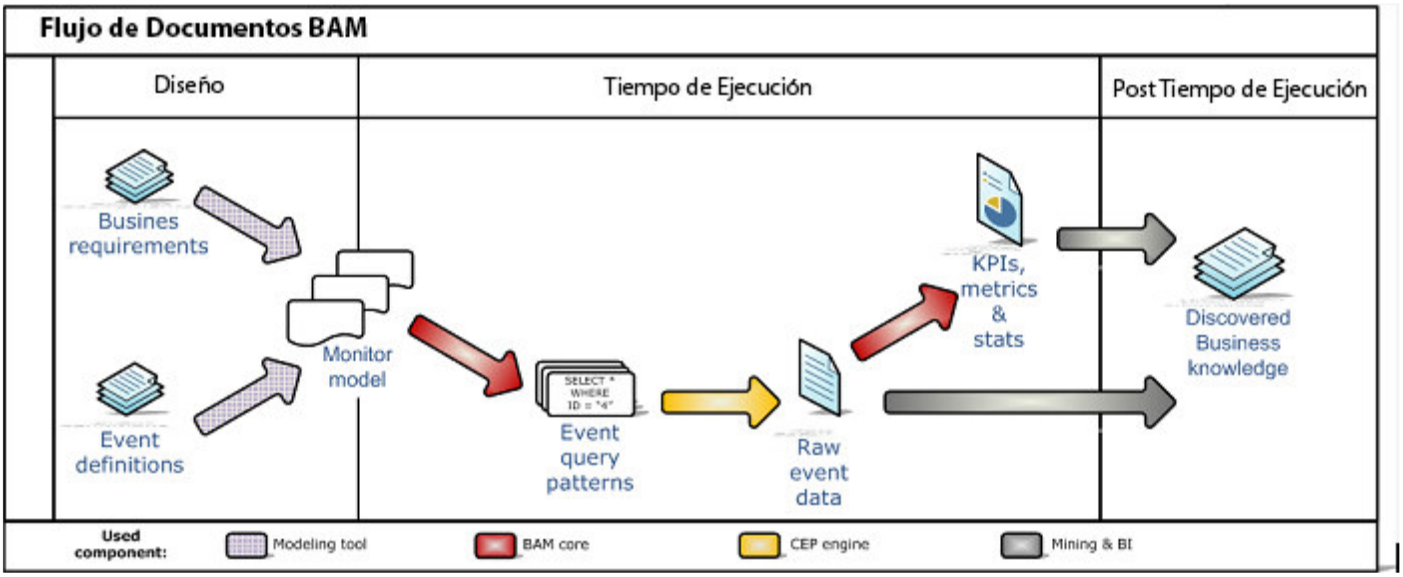


Figura 2.7. *Flujo de documentos BAM*

Se divide en fases de tiempo de diseño, tiempo de ejecución y tiempo de post-ejecución.

- **Tiempo de diseño**: en esta fase se empieza con el monitoreo del proceso, se define el modelo del monitor y se despliega en el motor BAM.
  - Entrada: primero están los requerimientos de negocio, usualmente descritos en lenguaje natural o de alguna manera un poco más formal. Luego, están las definiciones de eventos, usualmente escritas en algún lenguaje de definición de datos. Una opción recomendable puede ser el uso de XSD como estándar de XML. El formato del evento es usualmente retornado por el modelo del proceso.
  - Salida: la salida de esta fase es el modelo de monitoreo que contiene la descripción de las KPIs, las métricas, los triggers, entre otros elementos. El modelo de monitoreo define la información general que luego se presentará en el frontend. De acuerdo con este modelo, el componente de monitoreo será desarrollado y desplegado.
- **Tiempo de ejecución**: durante esta fase el modelo del monitoreo es ejecutado por el núcleo de BAM (*BAM-core*) y este componente inicia el monitoreo. Los requerimientos del modelo de monitoreo son transformados en consultas que son desplegadas en el motor *CEP (CEP engine)*. El motor CEP filtra los eventos y los envía en un formato definido por las definiciones de eventos al motor BAM. Allí, los eventos son procesados y almacenados en la base de datos del motor BAM. Entonces, el frontend puede tomar los datos del monitoreo desde el núcleo BAM y presentarlos al usuario.
  - Entrada: modelo de monitor compilado.
  - Salida: información del monitoreo presentada por un frontend particular. Otra salida son las llamadas del núcleo de BAM de los procesos en el motor BPEL. Estas llamadas son disparadas cuando se produce cierta situación de negocio.

- **Tiempo de post-ejecución**: aquí, la información del monitoreo puede ser procesada. Esto no se hace con tanta frecuencia hoy en día, pero como se mencionó anteriormente, es uno de los caminos esperados en los que se basará la Inteligencia de Negocio en el futuro. Los resultados de este proceso deben servir a la auto-optimización de los procesos.
  - Entrada: información producto del monitoreo.
  - Salida: conocimientos adquiridos y la auto-optimización de la información.

- Descripción general de la Arquitectura: el diagrama arquitectónico de la figura 2.8 muestra la arquitectura general de una solución BAM. A continuación se explica cada uno de los componentes que intervienen en la arquitectura y la función de cada uno de ellos.

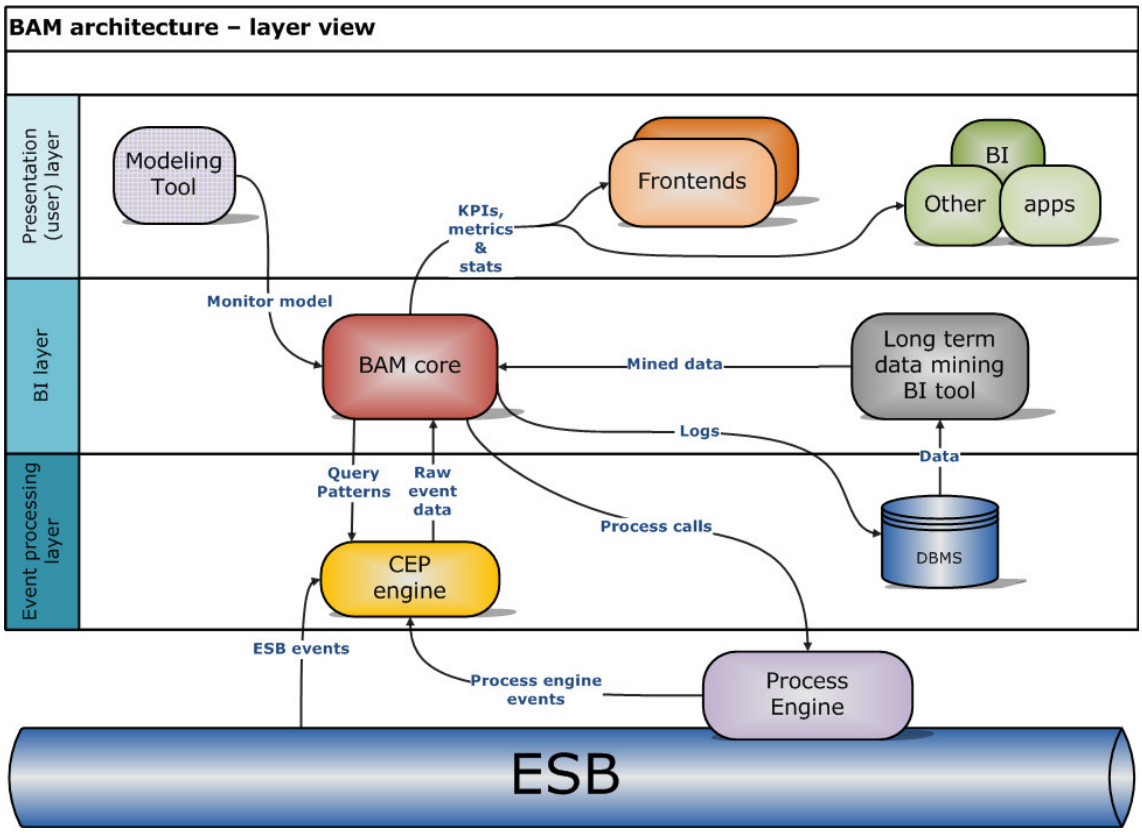


Figura 2.8. *Arquitectura BAM en capas*

- **Herramienta de modelado**: se define el modelo de monitoreo mediante una herramienta GUI (*Graphical user interface* o Interfaz gráfica de usuario). El modelo contiene la definición de eventos que pueden ser capturados, las métricas, los KPIs y los triggers. Una vez terminado el modelo, se lo despliega y el monitoreo puede comenzar.
- **Núcleo BAM**: es el principal componente de una solución BAM. Dicho núcleo genera patrones (consultas) de eventos acordes al modelo de monitoreo y los despliega en el motor CEP. Más tarde, se procesa los datos del evento recibido y se transforman los mismos a la forma deseada. El motor BAM también se puede comunicar con el motor

de procesos (*process engine*), iniciar y detener los procesos, generar eventos, entre otras tareas, de acuerdo a los triggers y alarmas definidos en el modelo de monitoreo.

- **Motor CEP**: es el componente de bajo nivel responsable para filtrar los eventos. Se encarga de recibir las consultas provenientes del núcleo BAM.
- **Frontend/Interfaz**: recibe datos ya procesados, con KPIs calculados y se hace cargo de su visualización. Toda solución BAM debe ofrecer posibilidades de visualización, con componentes como los descriptos anteriormente. Una de las características avanzadas de algunos frontend es proporcionar la visualización en tiempo real de los procesos por medio de gráficos. Resulta común que algún frontend esté basado en la web. Esto se realiza generalmente con los scripts de AJAX (*Asynchronous JavaScript And XML*). Esta técnica de desarrollo web se convirtió hoy en día en el estándar para herramientas de BI basadas en la web, y además tiene el soporte de Jasper Dashboards, uno de los candidatos más fuertes para el código abierto BAM frontend.
- **Herramientas de BI a largo plazo** (*Long term data mining BI tools*): componente opcional que puede analizar los datos desde una perspectiva a largo plazo, realizando minería de datos y KDP (*Knowledge Discovery proceses, Procesos de Descubrimiento de datos*). La información obtenida se entrega nuevamente al núcleo BAM. En estos casos se utilizan varias herramientas externas hoy en día, pero se prevee que en el futuro dichas herramientas sean parte directa de la solución BAM.

2.2.7 Prototipo de solución BAM:

En esta sección se procederá a describir un ejemplo de prototipo de una solución BAM basado en la arquitectura propuesta previamente.
Para este propósito, el prototipo proporciona: una interfaz sencilla de usuario para el desarrollo del modelo de monitoreo basado en la lógica "evento-métricas-KPI", un emisor de eventos que puede simular eventos fuente (por ejemplo, el motor BPEL) y una interfaz simple que muestra los datos del monitoreo.

Plataforma: el prototipo está desarrollado en una aplicación JAVA. Usa EJB3 (*Enterprise Java Beans*) para la lógica del núcleo de BAM core y la persistencia. Y JSF y servlets para la interfaz de usuario web.

Casos de uso:

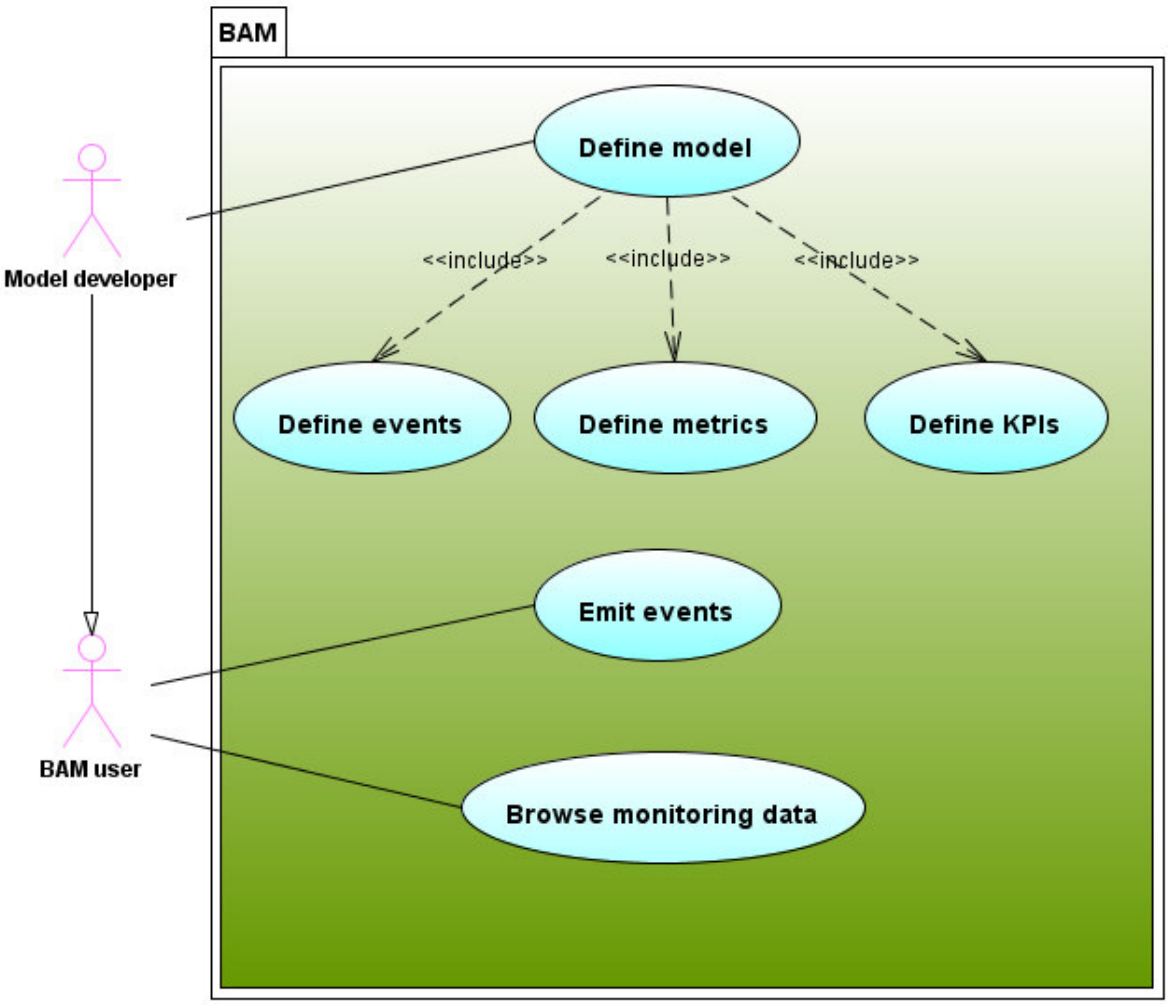


Figura 2.9. *Casos de uso del Prototipo*

- *Define model*: aquí el usuario puede definir el modelo de monitoreo con la ayuda de una interfaz de usuario web. Esto está acompañado con lo siguiente:
  - *Define events*: definir eventos entrantes y datos a ser extraídos para el monitoreo.
  - *Define metrics*: definir métricas que serán medidas y una situación en la que el valor se cambia con la ayuda del mecanismo "*trigger-event handler*".
  - *Define KPIs*: definir KPIs sobre las métricas.
- *Emit events*: el usuario puede emitir eventos definidos en el archivo de texto.
- *Browse monitoring data*: el usuario puede navegar por los datos obtenidos del monitoreo a partir de la emisión de eventos.

Componentes:

- *BAM core*: el motor del núcleo está implementado como una aplicación EJB3, usando "*entity beans*" para la persistencia y algunos "*session beans*" para la lógica. Las entidades son mostradas en el siguiente diagrama de clases:

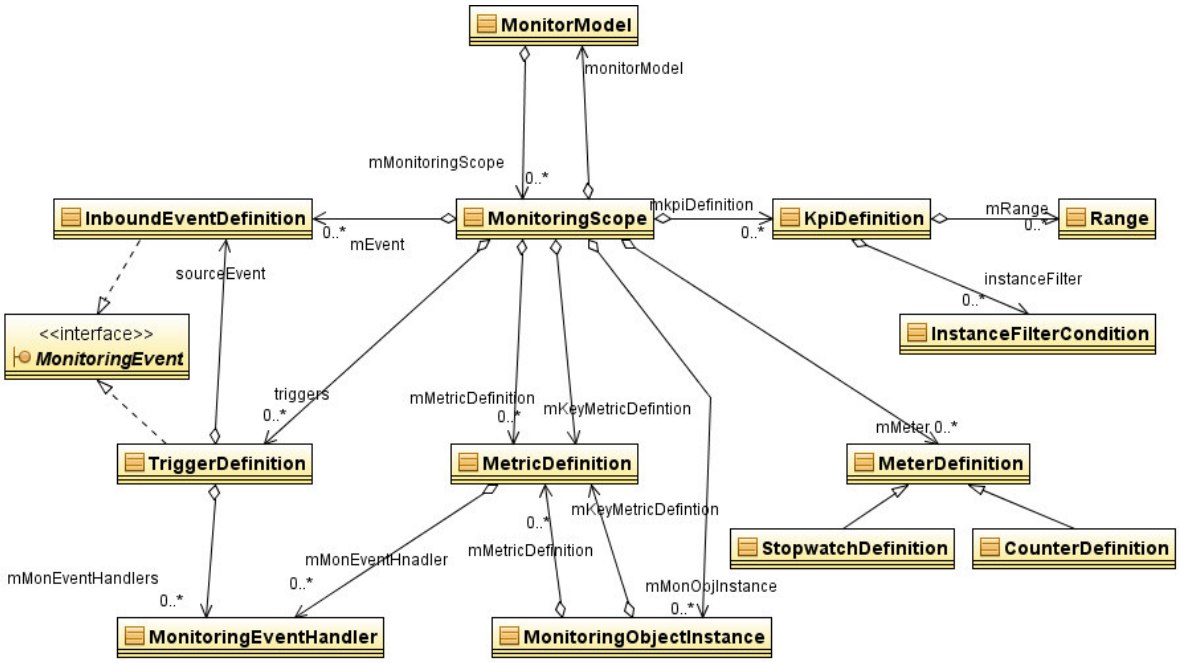


Figura 2.10. *Diagrama de clases del Prototipo*

- *Web frontend*: ambas interfaces de usuario (UI para el desarrollo del modelo de monitoreo, y BAM frontend) son aplicaciones web desarrolladas con JSF y contienen la funcionalidad estándar CRUD.
- *Event emitter*: el emisor de eventos está implementado como un *Web-Service* con un simple servlet basado en el frontend, que permite subir un archivo con eventos que son emitidos.

2.3 BPM + BAM

La gestión de procesos de negocio (BPM) y el monitoreo de actividades de negocio (BAM) convergen de tres maneras significativas. Los planificadores de la Tecnología de la Información (TI) deben revisar los proyectos BAM y BPM y alinear las actividades de planificación. Siempre es mejor realizar una planificación temprana para no incurrir en errores futuros [17].
Los planificadores de TI que desarrollan estrategias de monitoreo de actividades de negocio deben considerar el papel potencial de BPM; asimismo, la planificación aislada de esta última puede conducir a desafíos en BAM. Las implementaciones de BPM serán parte de BAM en algunos casos, e incluso suplantarla en otros. Los arquitectos de las empresas consumidoras deben asegurarse de que los planes BAM y BPM estén coordinados.
La gestión de procesos de negocio (BPM) y el monitoreo de actividades de negocio (BAM) tienen vidas separadas y diferentes patrones de implantación. Como distintas tecnologías de tiempo real para la organización, son herramientas disímiles en el mundo de TI. Sin embargo:

- Son altamente complementarios y a su vez parcialmente convergentes.
- La empresa tendrá muchas herramientas BAM y BPM
- BAM y BPM generalmente se desplegarán juntos para resolver problemas a nivel de negocio.
- Algunos requisitos funcionales de BAM serán satisfechos por las funciones de monitoreo de BPM, en lugar de una herramienta "BAM clásica".

BPM y BAM tienen tres áreas principales de convergencia (y conflicto potencial):

- BPM que actúa como "BPM + BAM"
- BPM que actúa como mecanismo de respuesta o destinatario de BAM
- BPM - o análisis de procesos de negocio (BPA) - que sirve como un modelo analítico / visualizador pasivo para BAM

Los siguientes tres casos representan una aproximación a un escenario donde BPM y BAM convergen. Cabe recordar que BAM es una aplicación múltiple, correlacionando múltiples fuentes de datos independientes.
En cada uno de estos escenarios de convergencia, se asume que BAM está trabajando con más de una aplicación subyacente.

Convergencia 1 - BPM Actuando como BPM + BAM: muchas empresas de software vinculadas a BPM están incluyendo elementos BAM en los productos que ofrecen. Normalmente, estos proveedores simplemente venden las capacidades actuales de monitoreo de los productos BPM, como una funcionalidad BAM. De modo que sólo emulando una simple forma de BAM, el monitoreo de estos programas de BPM pretende que sean considerados e implantados como una "solución" BAM. En estos casos, la función de monitoreo de los BPM constituye una etapa de la gestión de procesos (modelar el flujo, ejecutarlo y monitorearlo), pero los proveedores reconocen que pueden usar dicha función de monitoreo de los procesos para competir con los productos de BAM puros y con los productos de BAM que se integran como complemento a otras herramientas de BPM.
El enfoque de monitoreo de BPM + BAM también añade un elemento adicional de retorno de la inversión (ROI – Return of investment) que no se encuentra en muchas implementaciones de BAM puras; con BPM no sólo se realiza el monitoreo del proceso, sino que también se realiza su ejecución. Aún así, existen algunos inconvenientes: muchas (pero ciertamente no todas) herramientas de BPM solamente realizarán el monitoreo de lo que modelan y ejecutan. Los productos BPM tienen menos experiencia con el monitoreo de los datos que se encuentran fuera del dominio del negocio (por ejemplo, datos de eventos del sistema de IT o monitoreo del sistema técnico) que los productos BAM puros; aunque muchos de los productos BAM también están limitados en esa área. Muchos productos BPM no están tan a la altura de los requisitos de BAM como los productos de BAM puros que se dedican exclusivamente a realizar el monitoreo de los procesos de negocio. Sin embargo, para un propietario de procesos de negocio, estos puntos pueden significar matices que no empañan el uso de un BPM que haga el monitoreo de procesos tal como lo hace un producto BAM exclusivo [17].

Convergencia 2 - BPM sirviendo como mecanismo de respuesta o destinatario de BAM: en un modelo BAM ROI, las alertas en tiempo real deben ser procesadas ya que en otro caso se pierde el valor del sistema BAM. No sólo alcanza con "advertir", se debe reaccionar también. Cuando BAM emite una alerta, esta puede estar dirigida a varios destinatarios: alguien o algo que pueda reaccionar ante tal situación. Uno de los destinatarios del sistema puede ser una herramienta de BPM que utiliza la alerta como un disparador para una serie de procesos de negocio que necesitan ser activados para reaccionar a la alerta.

Por ejemplo, si una herramienta BAM de un sistema financiero emitió una alerta a partir de que el volumen de stock sobrepasó o disminuyó el nivel del umbral establecido, además de notificar a un grupo de comerciantes, la herramienta BAM podría activar un componente BPM que desencadenaría un flujo BPM impulsado a:

- Solicitar una investigación sobre el movimiento de stock.
- Activar una llamada de un operador para seleccionar a los principales inversores institucionales.
- Comprar o vender una cantidad predeterminada de elementos del stock.

La herramienta BPM ejecutará un conjunto seleccionado de procesos (o un único proceso) que representa la reacción a la alerta BAM. Como alternativa, la herramienta BPM podría utilizar la alerta de BAM para cambiar o adaptar un proceso de negocio ya en ejecución. Hay un enorme énfasis puesto en las alertas generadas por BAM. Las alertas son poderosas, pero son sólo la mitad del ciclo de monitoreo. Se debe reaccionar a una alerta, incluso si la reacción es "obtener más datos". El control del ciclo de reacción será un papel crítico de BPM [17].

Convergencia 3 - BPM como modelo pasivo de análisis/visualización para BAM: los dos escenarios anteriores se vinculan en que ambos representan una futura utilización de BPM en un contexto BAM. El siguiente escenario de visualización tiene un enfoque particular en el que se utiliza BPM para rastrear y monitorear los procesos pero sin ejecutarlos. Un mercado separado, BPA (*Business Process Analysis*), puede ser más apropiado para este escenario (ya que la función principal de BPA es modelar el flujo y escenarios complejos). Este escenario de visualización se presenta gráficamente en la Figura 2.11.

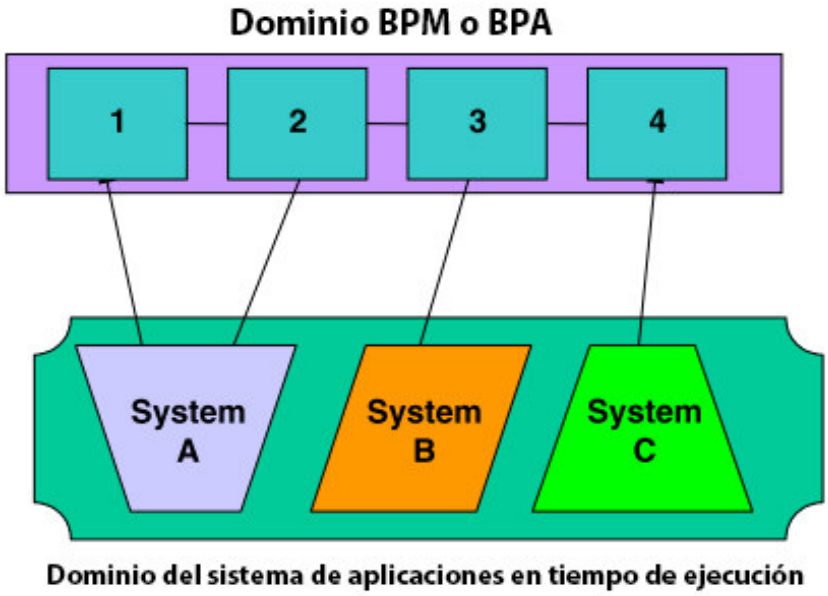


Figura 2.11. *Modelo de visualización usando BPA o BPM*

Los dos escenarios anteriores se centraron en la capacidad de BPM para ejecutar los procesos de negocio.
La convergencia 3 se centra en la capacidad de BPM para representar visualmente los procesos de negocio que se ejecutan utilizando modelos gráficos, en los que no está involucrada ninguna ejecución basada en BPM. Es técnicamente posible que una herramienta de BPM vincule las aplicaciones subyacentes de forma lógica mediante un modelo de proceso con el único propósito de monitorear y mostrar el progreso visual del proceso de ejecución subyacente. La Figura 2.11 muestra un proceso de cuatro pasos (dominio BPM / BPA).
El modelo de visualización no ejecuta el proceso de negocio, sino que sólo sirve como un modelo gráfico de lo que está sucediendo en los sistemas subyacentes en ejecución. A través de una combinación de agentes, adaptadores y demás, el modelo de visualización sirve para presentar una imagen en tiempo real del flujo de un proceso complejo que tiene lugar a través de una cadena de aplicaciones. Con la parametrización apropiada y los umbrales establecidos adecuadamente, el modelo de visualización se convierte en un sistema BAM: el rendimiento del proceso se puede rastrear como en la Convergencia 1, a pesar de que no se lleva a cabo la ejecución de la herramienta de seguimiento. El mercado de monitoreo de los sistemas de TI entiende muy bien este escenario; es hora de que la parte comercial vinculada a BPM considere esta opción. Este modelo sin ejecución es un uso inusual y visionario de BPM [17].

El caso de estudio de este trabajo se basa en algunos conceptos de las dos primeras convergencias para el desarrollo de una solución BAM para BPM. Pues se trata de un componente BAM independiente desplegado integrando varias herramientas BAM puras, y conectándose al BPM Bonita OS. Se trata de un enfoque BPM + BAM pero la herramienta de monitoreo no está incluida dentro del mismo BPM sino que es independiente y posee las prácticas propias de BAM. Además se espera que el BPM actué como mecanismo de respuesta ante las situaciones de alerta que advierta la aplicación BAM tal como lo promueve la convergencia 2.

2.4 Bussines Intelligence

El hecho de disponer de información no quiere decir que esto resulte de utilidad o que directamente signifique una optimización del negocio, pues se puede tener mucha información y no saber luego qué hacer con ella. La Inteligencia de Negocio *o Business Intelligence* (BI) es la solución a ese problema, dado que por medio de dicha información se puede generar escenarios, pronósticos y reportes que apoyen a la toma de decisiones, lo que se traduce en una ventaja competitiva. La clave para BI es la información y uno de sus mayores beneficios es la posibilidad de utilizarla en la toma de decisiones.

2.4.1 Definición y características:

Se denomina Inteligencia de negocio o BI (del inglés *Business Intelligence*) al conjunto de estrategias y herramientas enfocadas en la administración y creación de conocimiento mediante el análisis de datos existentes en una organización o empresa [18].
Las metodologías de BI utilizan la información para mejorar la gestión de las empresas. Gracias al software de BI, los usuarios pueden acceder y analizar los datos con facilidad, y tomar mejores decisiones.
BI es también una alternativa tecnológica y de administración de negocios, que cubre los aspectos del manejo de información, desde su extracción en los sistemas, depuración, transformación y diseño

de estructuras de datos o modelos especiales para el almacenamiento de datos, hasta la explotación de la información mediante herramientas comerciales de fácil uso para los usuarios. Este concepto es llamado también *DataWarehouse* (DWH). Un DWH es un conjunto de datos orientado a temas, integrado, no volátil, estable y que se usa para el proceso de toma de decisiones; almacena una gran cantidad de información histórica del negocio y aísla los sistemas operacionales de las necesidades de información para la gestión. Un cambio en los sistemas operacionales no debe afectar al DWH.
Mediante BI se busca explorar y analizar información estructurada sobre un área (normalmente almacenada en un DWH), para descubrir tendencias o patrones, a partir de los cuales derivar ideas y extraer conclusiones. El proceso de BI incluye la comunicación de los descubrimientos y efectuar los cambios. Las áreas incluyen clientes, proveedores, productos, servicios y competidores.
BI además constituye una categoría amplia de soluciones de software que permiten a una empresa u organización profundizar en sus operaciones a través de la presentación de informes críticos de las aplicaciones y herramientas de análisis.

Con los conceptos de BI enunciados se concluye que esta disciplina brinda herramientas que ayudan a la toma de decisiones, puesto que presenta la información de manera ordenada, rápida y oportuna, y permite a las empresas profundizar en su análisis, ya que se tiene indicadores claves que ayudan a la gestión y manejo de la misma.
BI permite la recolección de información efectiva, de manera que combinando datos y análisis, se pueda obtener el conocimiento necesario para tomar estrategias de mejora continua, y esto permita apalancar el buen funcionamiento de la planificación para cumplir los objetivos propuestos.

Las principales características de BI son:

- Ayuda en la toma de decisiones: posee herramientas de visualización avanzadas como gráficos, tablas, velocímetros, que ayudan a obtener rápidos tiempos de respuesta, permite una gran navegabilidad, seleccionar y manipular información que le interese al usuario.
- Acceso a la información: brinda datos de calidad, completos, correctos y coherentes. Permite el ingreso a los datos de manera independiente.
- Orientación al usuario final: se busca el manejo de interfaces amigables, que permitan al usuario cierta intuición, sin necesidad de conocimiento técnico para su uso.

Proceso de BI [Figura 2.12]:

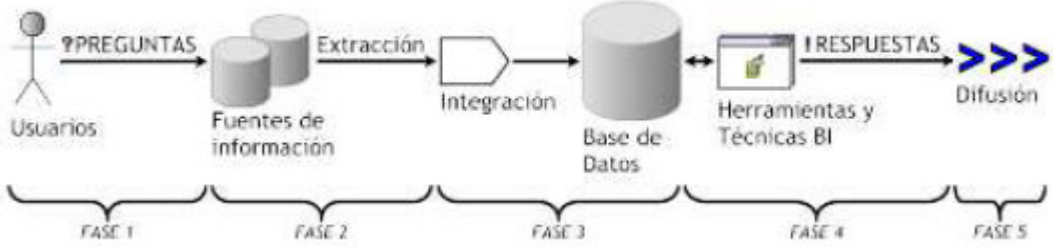


Figura 2.12. *Proceso BI*

1. Dirigir y Planear: fase inicial en la cual se recolectan los requerimientos de la información específicos de los distintos usuarios, de esta manera se comprenderán sus distintas necesidades, que ayuden a generar las distintas preguntas que ayudarán a alcanzar los objetivos.

2. Recolección de Información: se realiza el proceso de extracción de distintas fuentes de información de la empresa. Puede ser de manera interna o externa, esto ayudará a encontrar las respuestas a las preguntas planteadas en el paso anterior.
3. Procesamiento de Datos: se carga e integra los datos en su forma más rústica en un formato que se utiliza para el análisis, creando una base de datos completamente nueva donde se consolida la información.
4. Análisis y Producción: se procede a trabajar sobre datos extraídos e integrados, utilizando las herramientas y técnicas que brinda BI. Con el objetivo de crear inteligencia, el resultado a las preguntas planteadas en un inicio se genera mediante creación de reportes e indicadores, entre otros.
5. Difusión: en la fase final se entrega a los usuarios las herramientas adecuadas que les permitirá interactuar con los datos de manera sencilla y rápida.

La información de BI puede ser presentada en distintas frecuencias, ya sea mensualmente, semanalmente o diariamente, sin embargo es una tecnología que en su nivel más genérico no es de tiempo real. Por lo tanto está organizada para hacer análisis puntuales, históricos, y minería sobre los datos. BI suele presentarse más confiable que otras tecnologías que se basan en el procesamiento de datos en tiempo real, pues se utilizan varios procesos para depurar la información y para algunas aplicaciones la consistencia de datos es más importarte que el tiempo real.

2.4.2 BAM + BI:

La arquitectura lógica de BI en su forma clásica está dirigida a los analistas de negocio, diseñada para el análisis y la *minería de datos* (campo de la estadística y las ciencias de la computación referido al proceso que intenta descubrir patrones en grandes volúmenes de conjuntos de datos), no estando directamente diseñada para el procesamiento en tiempo real. En contraposición, la arquitectura lógica de BAM está dirigida a los administradores, está diseñada para actuar rápidamente y en tiempo real.

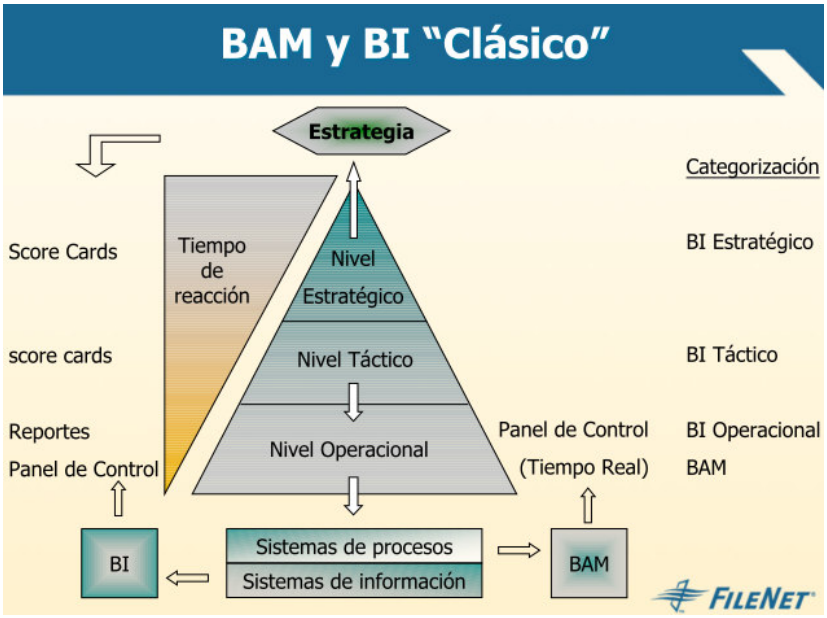


Figura 2.13. *Roles de BAM y BI en una arquitectura lógica de monitoreo*

Las funcionalidades que ofrece BI son similares a las que ya se analizaron en la sección de BAM en cuanto al procesamiento de la información, la toma oportuna de decisiones, y la visualización pertinente de dicha información. Sin embargo, es necesario hacer la distinción entre ambas

tecnologías, pues, aunque están ubicadas en la misma categoría de herramientas de análisis del negocio, tienen una diferencia fundamental sobre el origen de los datos analizados. Las herramientas de BI también tienen que ver con el entendimiento y la optimización del desempeño del negocio, pero la diferencia es que estas se enfocan en el análisis de datos históricos que son utilizados para explorar, a través de técnicas de *minería de datos*, el comportamiento que se ha tenido en períodos anteriores y así poder identificar tendencias. Por otra parte, las herramientas de BAM tratan con los datos de lo que está sucediendo actualmente en el negocio, es decir, en tiempo real.

En conclusión, podría decirse que BAM es BI pero actuando en tiempo real. Como se analizó ambas tecnologías se relacionan, una más inclinada al análisis histórico de la información (BI) y la otra al procesamiento inmediato de la misma (BAM), pero ambas tienen como propósito optimizar el rendimiento de una organización en base al análisis de los datos y mostrar de manera clara lo que está ocurriendo en las actividades de negocio al usuario.

Precisamente, el siguiente capítulo abordará la descripción de la herramienta adecuada para desplegar toda la información que se procesó previamente: el Tablero de Control.

3

3. Tablero de Control

Con el continuo avance en la tecnología, los niveles de información han crecido de una manera significativa, desencadenando un tremendo volumen de datos. Dominar esta situación se ha convertido en el objetivo principal de las industrias de la información.
En muchas organizaciones se ha buscado administrar la información que manejan de la mejor manera posible ya que ello implica tener la visión del negocio y ser cada vez más competitivos en el mercado.
Por esta razón, y como se describió previamente, surgen áreas como Inteligencia de Negocios (BI), la cual se especializa en administrar grandes cantidades de datos que posee la organización de la cual forma parte y que tiene por objetivo principalmente el procesar y almacenar en un repositorio centralizado esta información; además de poseer el talento humano, técnicas y herramientas necesarias para la exploración y consulta de la misma, la cual es base para la toma de decisiones, elemento vital para la supervivencia de cualquier negocio hoy en día. Por esto las organizaciones se han visto en la tarea de investigar e invertir recursos para fortalecer esta importante área de trabajo.
Así se puede decir que el área de Inteligencia de Negocios busca, a partir de todo su trabajo, brindar el conocimiento para tomar las decisiones correctas que den el lineamiento que debe seguir la organización a partir de la información recolectada.
En años recientes ha surgido una herramienta para este propósito que ha adquirido gran fuerza en este campo, son los llamados **Tableros de control (*Dashboards*),** los cuales se utilizan para el análisis y posterior toma de decisiones dentro de las organizaciones; siendo muy generales, un tablero de control es una interfaz computacional con gráficos, reportes, indicadores visuales y mecanismos de alerta que son consolidados en una plataforma de información con el fin de tener una visión clara del negocio.
En este capítulo se abordará en detalle los conceptos inherentes a los Tableros de control, los cuales brindan una visión conveniente para su elaboración. Además se describen las características de los tableros y una forma de diseño (*dashboarding*) de los mismos que servirá como soporte para la construcción del tablero que se expondrá más adelante en el caso práctico.

3.1. Definición

El término “tablero de control” puede asociarse por ejemplo al panel que se encuentra debajo del parabrisas de un vehículo y que contiene varios instrumentos y controles donde se reúnen todos los datos y funciones pertinentes. Esto proporciona un fácil uso y comodidad a lo hora de tomar decisiones al conducir un automóvil. De manera similar ocurre en una aeronave, donde los pilotos calificados, a través de estos elementos en el tablero de control, pueden monitorear, tener una visión de hacia dónde se dirigen y saber si se tiene algún problema durante el vuelo. Estos ejemplos han inspirado el mismo concepto de sistema, es claro el propósito del tablero de control en estos 2 escenarios, “controlar y manejar un sistema complejo e interdependiente”. [21]

Teniendo en cuenta entonces que los pilotos pueden procesar la información de un gran número de indicadores que se les muestran en estos tableros para navegar en sus aeronaves, se podría pensar,

"¿por qué no creer que los ejecutivos también pueden tener elementos similares que les permitan orientar sus organizaciones?".
Las personas que toman las decisiones en las organizaciones deben analizar y controlar la información de cada negocio. Recorrer hojas de cálculo o llamar especialistas de élite de información son labores muy tediosas, por lo que ciertamente los tableros de control cobran suma importancia al suplir ciertas necesidades de los profesionales hoy en día.
Ya bien se conoce que lo que no se puede medir difícilmente se pueda manejar, por lo que los tableros de control deben proveer una clara visión para dirigir a través de las espesas nubes de sobrecarga de datos, llegando a una idea que viene convergiendo desde principios de este siglo y es de tener la información correcta en el momento correcto.
Casi todas las organizaciones han experimentado un crecimiento exponencial en el poder computacional y en volúmenes de datos durante los últimos años. Este crecimiento conduce a que la gestión organizacional cree procesos más ilustrados para la toma de decisiones en un ambiente abundante en información.

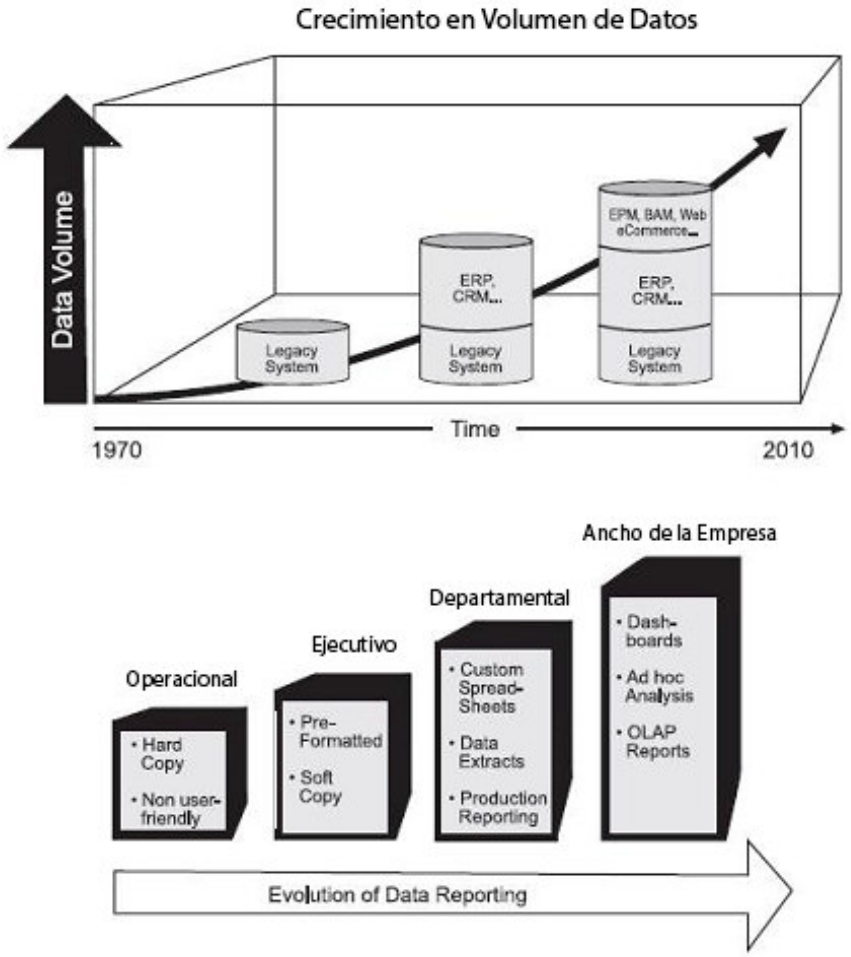


Figura 3.1. *Saltos en la información y presentación de informes. Crecimiento en el volumen de datos*

Retomando la analogía con los tableros de control de una aeronave o automóvil, se tiene entonces un punto de partida para describir las características básicas de un tablero de control o *Dashboard* como se conoce alrededor del mundo. Para empezar se debe tener en cuenta que la gran mayoría de tableros de control para las cabinas normalmente poseen el mismo conjunto de Indicadores Clave de

Rendimiento o KPI, tales como velocidad de la aeronave, altitud, dirección, velocidad del viento, humedad, estado de combustible, temperatura del motor, latitud, longitud, entre otros. Muchos fabricantes de aviones pueden tener diversos diseños para sus paneles, pero en esencia, todos ellos poseen el mismo conjunto de KPIs críticos para el éxito de un vuelo. Similarmente se puede hacer el análisis para los automóviles, concluyendo así que un fabricante, ya sea de autos o aviones, puede replicar con facilidad en una línea de montaje miles de paneles según sea el caso.
En contraste con un avión o automóvil, cada organización tiene un conjunto de KPIs que difiere significativamente de otras, aunque estén en el mismo sector. Cada negocio y administración organizacional han evolucionado de manera diferente y cada división dentro de la organización posee distintos tipos de KPIs relevantes para ellos mismos.
Debido a esta diversidad se hace necesaria la realización exhaustiva e individualizada del análisis de requisitos, con el fin de construir tableros de control eficaces para cada organización. Para hacer las cosas aún más complejas, las fuentes de información que necesitan ser presentadas a través de los tableros de control son muy diversas dentro de las organizaciones. De modo que no hay dos organizaciones que coincidan en la multitud de fuentes de información para el acceso a los tableros de control, haciendo más compleja dicha labor.

3.2. Características

Se puede mencionar algunas características básicas específicas para un tablero de control usando un acrónimo. Como el tema es conocido a nivel mundial, la palabra en inglés se conoce como **SMART** *(Synergetic, Monitor, Accurate, Responsive, Timely*) [21], lo cual se interpreta como lo siguiente:

- Sinérgico: debe ser ergonómico y visualmente efectivo para un usuario, acerca de los diferentes aspectos dentro de una misma pantalla.
- Monitor de KPI: debe mostrar indicadores críticos que sean requeridos para una toma de decisiones efectivas en el dominio al que el tablero de control abastece.
- Exacto: la información que se presenta debe ser precisa, con el fin de obtener la plena confianza del usuario con el tablero. El soporte de datos para el tablero debe estar probado y validado.
- Responsivo: debe responder a los umbrales predefinidos para crear alertas de usuario, en adición a la presentación visual del tablero, por ejemplo alarmas de sonido, correos electrónicos, buscador de personas, entre otros, con el fin de obtener de inmediato la atención del usuario cuando ocurra un asunto crítico.
- Oportuno: debe mostrar la información lo más actualizada posible, la cual debe ser en tiempo real y oportuna.

Sin embargo con estas características no es suficiente para garantizar una efectiva administración organizacional. Otros elementos adicionales importantes para el tablero se capturan en el siguiente acrónimo en inglés: **IMPACT** (*Interactive, More Data history, Personalized, Analytical, Collaborative, Traceability*) [21] que se interpreta de la siguiente manera:

- Interactivo: debe permitir a los usuarios llegar a los detalles de la información. Imaginar cuan provechoso sería para un piloto hacer *click* en el indicador de combustible y ver la tasa de consumo durante la última hora de vuelo, sólo para encontrar que la tasa de consumo se ha

disparado al doble del nivel normal hace 15 minutos, indicando un súbito escape de combustible.

- Más Datos de la Historia: el tablero debe permitir a los usuarios revisar la historia para un determinado indicador. Por ejemplo en las ventas, determinar qué ocurre en un período de tiempo al día de hoy y cómo fueron en ese mismo período el año anterior, para analizar su comportamiento.
- Personalizado: la presentación del tablero debe ser específica para el dominio de responsabilidad, privilegios, restricción de datos y demás que tenga el usuario. Otras opciones de la personalización deberían estar disponibles, tales como el idioma y preferencias visuales de color y estilos de fondo para una mejor experiencia del usuario.
- Analítico: debe permitir a los usuarios navegar por distintos medios para comparar, contrastar y hacer inferencias analíticas de lo que se muestra en el tablero. De esta manera se le facilita una mejor comprensión de las variables del negocio.
- Colaborativo: el tablero debe facilitar la capacidad a los usuarios para el intercambio de notas y la realización de observaciones específicas en relación a sus tableros. Esto también sirve para llevar un control con el flujo de trabajo y procesos. Un buen diseño de colaboración serviría como una plataforma de comunicación en la tarea de gestión y control de cumplimiento.
- Trazabilidad: debe permitir a cada usuario personalizar los parámetros que quieran seguir. Tales parámetros podrían ser la vista por defecto del tablero después de iniciar sesión, por ejemplo.

3.2.1 Buenas Normas de Software:

Luego de haber dado algunas bases para los tableros de control, cabe destacar que el software que se usa para su construcción debe cumplir con ciertas normas que todo buen software debería tener. Algunas de estas normas son:

- Rápida Respuesta: los usuarios no deberían experimentar una excesiva demora en la recuperación de sus tableros e informes.
- Intuitivo: los usuarios finales no necesitarán pasar por un arduo proceso de aprendizaje.
- Basado en Web: los usuarios deberían tener acceso a través de este medio si poseen los permisos indicados para ello.
- Seguro: los administradores del sistema pueden administrar fácilmente la seguridad del sistema reduciendo y monitoreando accesos ilícitos. También debe permitir la encriptación cuando la información es de alto contenido confidencial.
- Escalable: que un gran número de usuarios pueda acceder al sistema sin que se presenten problemas de lentitud. Esta característica va muy de la mano con la calidad del hardware y el ancho de banda.
- Compatibilidad con la Industria: el software debe poder integrarse con bases de datos de diferentes proveedores y trabajar con diferentes normas de servidores (Net, J2EE) y sistemas operativos (Unix, Windows, Linux).
- Tecnología Abierta: el software no debería ser rígido en el sentido de hacer muy complicado el extender su alcance dentro del complejo entorno de tecnologías de información TI. Se debe

trabajar con protocolos de intercambio de información, tales como XML, ODBC, OLE, PP, SMC y servicios web. Cabe aclarar que "tecnología abierta" no es lo mismo que código abierto, lo cual se refiere al software libre con acceso abierto a su código fuente.

Para el caso de estudio de este trabajo se han seleccionado tecnologías de código abierto para el desarrollo del tablero de control. Específicamente la parte de la vista está diseñada mediante aplicaciones JAVA denominadas *portlets,* donde cada una representa un componente particular del tablero y se despliegan en el portal Liferay. Estas componentes toman la información de la base de datos relacional Mysql.

3.2.2 Errores comunes acerca de los Tableros de Control

Existen ciertas percepciones sobre los tableros de control que sencillamente no son las correctas:

Los tableros de control son sólo para altos ejecutivos (¡Incorrecto!): hay un concepto que aún prevalece y es que los tableros de control son para los grandes ejecutivos de las organizaciones, para darles una visión global del desempeño organizacional. Esto se debe replantear ya que hoy en día la tecnología de los tableros de control está diseñada de tal forma que un tablero de control sea una herramienta eficaz dentro de todos los niveles de una organización [21].

Los tableros de control sirven solamente para distribución de Informes (¡Incorrecto!): toda la plataforma para los tableros de control no debe tratarse sólo como algo conveniente para realizar informes o visualizar diferentes KPI. Esto disminuye en gran medida el verdadero valor y eficacia que poseen los tableros de control y de cómo estos pueden mejorar el rendimiento de la organización. Entonces, el propósito central del tablero es advertir al usuario cuando los indicadores que se manejen en el tablero se encuentran fuera de un límite establecido, es decir dar avisos y alertas de cuando esto ocurra para tomar decisiones oportunas y tener control de hacia dónde se dirige la organización [21].

3.3. Dashboarding

En el proceso para desarrollar los tableros de control hay que tener, además de las bases, ciertos elementos para llevar a cabo esta labor. Existen 3 interrogantes que son muy importantes y que al tenerlos claros, se contará con una primera línea a seguir para el desarrollo:

- ¿Qué Información?
- ¿Para Quién?
- ¿Cómo presentar la Información?

3.3.1 ¿Qué Información?

Como primer nivel este proceso requiere dar respuestas a preguntas que sean críticas para el negocio y que necesiten ser contestadas a través de la información que se despliegue del tablero; así entonces se deberá mapear o llevar estas preguntas a indicadores claves de rendimiento (KPIs) del tablero de control, para que se tenga al final las respuestas a dichos interrogantes críticos para el negocio [28].

El primer paso en este proceso es **documentar todos los KPI's** que se hayan capturado ya sea a través de reportes periódicos o por análisis. Normalmente estos indicadores clave están bien documentados y mapeados desde diversas fuentes de datos que hayan sido usadas para su creación. En este punto, sería útil poder determinar algún otro KPI que se desee y que por el momento no se tuviese disponible por alguna u otra razón.
Otra estrategia para **categorizar los KPI's** es distribuirlos en divisiones dentro de la organización. Algunos estándares de los indicadores clave para estas divisiones son:

- KPIs en Ventas: Ingresos Brutos y Netos, Ventas x Unidad, Número de Ordenes, Valor Promedio de las Órdenes, Días desde la última Venta, Ingresos por Empleado.
- KPIs Mercadeo: Porcentaje de respuesta en una promoción, Elasticidad en los precios, Visitas a un sitio web, Venta de productos mixtos.
- KPIs cadenas de suministros: Tiempo de Ciclo, Cantidad de Inventario, Volumen de Ventas Perdidas, Tasa de Devoluciones, Porcentaje en tiempo de Entrega, Giro en Inventario.
- KPIs Servicio al Cliente: Porcentaje de retención, Porcentaje de venta Cruzada, Porcentaje del nivel de servicio cumplido, Tiempo de manipulación, tiempo de llamada, Porcentaje de remisión, Porcentaje si resolver, Asignación de casos.
- KPIs Recursos Humanos: Volumen de negocios por empleado, tiempo del ciclo de contratación, Nivel de habilidades.
- KPIs Financieros: Ingresos, Ganancia Bruta, Porcentaje de Margen Bruto, Ingresos Netos, Cuentas por Cobrar, Flujo de Caja.
- KPIs Manufactura: Trabajo como porcentaje de los costos, Porcentaje de Inactividad, Tiempo desde la orden hasta el envío, tiempo de empaque.

Definiendo Indicadores: el proceso para **definir los KPI's** no es muy diferente a construir una infraestructura de información o un sistema de Inteligencia de Negocios. Para hacer este proceso con rigor y detalle se necesita un analista de información experimentado, el cual debe adquirir un conocimiento muy a fondo de cuáles son las fuentes en donde está dispersa la información dentro de la organización, la infraestructura existente en Inteligencia de Negocios y entender a fondo los procesos del negocio junto con los requisitos de la información. Además de todo esto, debe ser capaz de aprovechar la asesoría de un equipo de expertos en las divisiones de la empresa, informática y análisis de datos, para tener un panorama completo. Cada indicador debe desglosarse en 4 elementos:

- Fuentes de Datos: las fuentes de los datos identifican el alto nivel de información sobre dónde recuperar la información para un determinado indicador. Tal nivel de información incluye identificar la base de datos (como por ejemplo un *data warehouse*), fuentes OLAP, archivos de datos, informes existentes o fuentes de apoyo.
  Aquí es donde surgen los retos para integrar todas esas fuentes de información y que se comuniquen con el software de los tableros de control para presentarla en completa armonía. Aquí es donde entran en juego elementos como son los drivers de las bases de datos, interfaces de programación, agentes, adaptadores y demás.
  Algo que cabe mencionar es que cuando se está en el proceso de identificación de todos los KPIs que necesita el tablero, dentro de la información de la organización se pueden encontrar lagunas o baches, es decir, que al llevarse a cabo todo este análisis de cómo obtener los indicadores se encuentran detalles de falta de estandarización, falta de validación de datos, redundancia de información a lo largo de varias fuentes de datos, entre otros. Es común que cuando se está haciendo este proceso de obtener los indicadores, el mismo traiga consigo un esfuerzo paralelo respecto a estandarización, mejoras del data warehouse y almacenamiento, extracción, transformación y carga de datos.

También es importante mencionar que un solo Indicador KPI puede involucrar muchas fuentes de datos.

- Granularidad: la granularidad establece los distintos niveles de cálculos requeridos para cada uno de los KPI. Cada indicador puede tener diferentes "Granos" a través de tres dimensiones: Tiempo, Geografía y Producto, por lo que las combinaciones únicas y factibles de estas tres dimensiones determinan los diferentes "granos" para un solo KPI.
    - Tiempo: determina los atributos de tiempo para un determinado KPI, como son por ejemplo por hora, diariamente, semanal, mensual, trimestral, anual, del año hasta la fecha, y así sucesivamente.
    - Geografía: determina los atributos de zona geográfica como, mundo, región, país, estado, sección, territorio, ciudad, código postal y así sucesivamente.
    - Producto: determina la agrupación de los atributos del producto, como total compañía, división, categoría del producto, marca, tema del grupo, artículo, Código de Producto Universal (UPC), entre otros datos.
- Cálculos: el cálculo se refiere sencillamente a cualquier operación matemática requerida para un determinado KPI. La mayoría de las veces las operaciones usadas para llegar a un indicador son suma, promedio y porcentaje. Sin embargo dependiendo de la situación también se usan el mínimo, máximo, media entre otros. Todo esto ocurre cuando los datos se extraen de una única fuente, donde los cálculos no involucran operaciones complejas. Sin embargo cuando ya un KPI se extrae de dos o más fuentes entrarían a jugar operaciones de un nivel mayor de complejidad para realizar ese cruce de información. Algunos ejemplos de operaciones con algunos indicadores:
    - Ingresos: Suma de ingresos
    - Beneficio bruto (Perdida) %: (Suma de Ingresos – Suma de Costos) X 100 /Suma de Ingresos
- Varianza: es la diferencia que establece la comparación de referencia para un indicador. Tiene dos requisitos: 1. Base para el cambio y 2. El cálculo del cambio. La mayoría de veces esto se usa para hacer comparaciones periódicas, como por ejemplo: hace un año, hace un trimestre, hace un mes y así sucesivamente. Y los valores que se analizan en esas diferencias básicamente son porcentajes, cifras y valores los cuales son el objeto de estudio para la comparación y dan una visión de cómo va el negocio.

Definiendo los Umbrales de los Indicadores: los umbrales de los KPIs son parámetros establecidos por la organización con el fin de evaluar el rendimiento y tomar acciones en torno a organizar el flujo de información. Definir los umbrales de los KPIs complementa la imagen y ayuda a la organización a obtener vitales respuestas cualitativas para su negocio.
Los umbrales pueden variar para cada grano KPI (como se mencionó anteriormente), y son establecidos por los administradores o conocedores del negocio en sí para cada uno de los "granos KPI". Para ejemplificar este concepto utilizamos la escala de porcentajes para servicio al cliente y vemos que una organización podría definir algunos umbrales para diferentes grupos de productos:

- Cuentas para Empresas:
    - Excelente: Más del 90%
    - Bueno: 80% al 90%
    - Normal: 75% al 80%
    - Pobre: Por debajo del 75%
- Cuentas de Consumo:
    - Excelente: Más del 80%
    - Bueno: 70% al 80%
    - Normal: 60% al 70%

- Pobre: Por debajo del 60%

Como se observa, para dos tipos de indicadores se usa la misma escala o resolución de porcentajes, sin embargo tienen distintos valores o medidas de un indicador al otro, dándole a la organización poder para dar prioridad. También se podría aplicar para diferencias en los ingresos de una compañía, por ejemplo ver las diferencias entre los ingresos del año anterior:

- Excelente: Más de 100 Millones
- Bueno: Entre 50 y 100 Millones
- Normal: De 0 a 50 Millones
- Pobre: Cualquier disminución.

Un formato gráfico muy popular para visualizar los umbrales de los indicadores y el rendimiento es el velocímetro. En él se pueden ver los distintos niveles del umbral, la aguja representa el valor real de los datos.

Figura 3.2. *Velocímetro mostrando diferentes umbrales*

Definiendo las Alertas: las alertas de los KPIs son sencillamente acciones que se toman una vez que el umbral del indicador es alcanzado. Sin embargo no son definidas para cada límite del umbral. Normalmente sirven como un sistema de alertas cuando el KPI muestra un comportamiento de bajo rendimiento o tendencia indeseable. Desde el envío de correos electrónicos, hasta indicadores visuales, como un parpadeo o alguna animación en el tablero, son posibles elementos que estarían acompañando a las alertas. La otra variable de las alertas es el receptor que puede ser uno o más destinatarios para cada alerta.

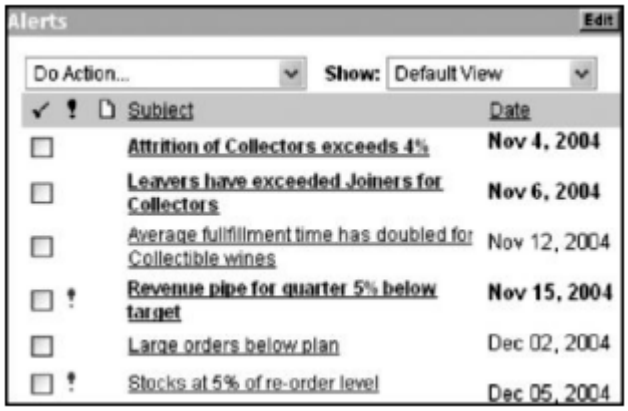


Figura 3.3. *Alertas que muestran cuando un umbral excede los parámetros definidos*

Además de informar la dispersión de la información a la persona correcta en el momento indicado, también se pueden usar las alertas como mecanismo de control. Por ejemplo, si el nivel de inventario está por debajo del límite establecido, se puede activar una alerta que haga una orden de compra en el sistema del proveedor para dicho elemento.

Definiendo las Jerarquías: las jerarquías de las organizaciones son estructuras específicas para la gestión organizativa. Cada una de las tres dimensiones de granularidad (Tiempo, Geografía y Producto) tiene su propia jerarquía.
Ejemplos de jerarquían se reflejan en la Figura 3.4.

Figura 3.4. *Pirámide jerárquica para cada una de las tres dimensiones básicas*

3.3.2 ¿Para quién?: hasta el momento se vio visto de manera general una de las preguntas importantes respecto a los tableros de control, la cual es ¿Qué información? Ahora se analizará el siguiente interrogante ¿Para quién?
El público es un factor clave en el diseño de la arquitectura de un tablero de control. Determinar los perfiles de usuario y sus dominios de privilegios contribuye a la creación de una experiencia del tablero personalizada. Un tablero efectivo debe mostrar sólo los KPIs que son relevantes para un usuario determinado y dentro de sus privilegios, para ello se necesita definir tres marcos de trabajo:

- Grupos de Usuarios y Jerarquías: el objetivo principal para el desarrollo de grupos de usuarios y jerarquías es evitar la repetitiva tarea de estar asignando grupos de privilegios y acceso a contenidos para cada usuario individualmente. El hecho de establecer tales grupos de usuarios permite la asignación de un conjunto de privilegios a todos los usuarios dentro de un grupo con una sola instrucción de software. Similarmente a través de la herencia, a múltiples grupos de usuarios dentro de una jerarquía, se les puede asignar un conjunto de privilegios con un solo click. Para todo esto hay que tener claro cuáles son las funciones del negocio y sus diferentes necesidades de información. Es importante saber la jerarquía de la organización y cuál es su nivel de dominio para cada nivel de la jerarquía, correspondiente a sus necesidades de información.
- Privilegios de Dominio (rol de usuario, rol de administrador): los privilegios son básicamente funciones que se pueden realizar dentro de alguna aplicación de software. En el ámbito de los tableros de control, aquí se listan algunas funciones básicas para ello:
  - Crear un tablero sólo para consulta.
  - Modificar un tablero.
  - Poner mensajes en un tablero existente.
  - Crear elementos subyacentes para un tablero, tales como KPIs, alertas, informes, gráficos, plantillas para tableros.
  - Crear categorías o pestañas para la navegación
  - Cambiar privilegios.
  - Definir grupos de usuarios y jerarquías.

- Definir una nueva base de datos o fuente de datos.

Con frecuencia las funciones anteriores pueden ser agrupadas a un rol específico, es decir:

- Rol de usuario del Negocio: crear un tablero sólo para consulta o poner mensajes en un tablero existente.
- Rol de Analista o Usuario Avanzado: crear un tablero sólo para consulta, modificar un tablero, crear elementos subyacentes para un tablero, tales como KPI, alertas, informes, gráficos, plantillas para tableros e incluso crear categorías o pestañas para la navegación
- Rol Administrador: todas las funciones que permita realizar la aplicación para los tableros.

Para una eficiente asignación de roles, es importante tener claro el organigrama de funciones en la organización de los usuarios que reciban acceso a los tableros. Esta información la manejan principalmente los jefes de área que en última instancia son los responsables de aprovechar el poder de los tableros en su campo de trabajo.

- Contenido de Dominio (seguridad, relevancia): la gestión del contenido del dominio implica dos importantes aspectos: Seguridad y Relevancia. Seguridad se refiere a las restricciones para la entrega de información a aquellos que cuenten con el privilegio de accederla. Cada organización tiene sus límites con respecto a quién y qué tipo de información puede ver, ya que es confidencial y hay que tener cierto recelo respecto a ello. La seguridad debe ser establecida durante el desarrollo de los tableros de control, es decir definir los permisos y restricciones del dominio de contenido para cada usuario. Relevancia se refiere al filtro del contenido de la información con lo más relevante o necesario para un usuario del tablero. El tablero debe ser lo suficientemente flexible como para mostrar dicha información relevante, pero con la capacidad de permitirle al usuario poder acceder a más información, dependiendo de su rol, según sea necesario.

3.3.3 ¿Cómo presentar la información?: a continuación se analizará cómo es el diseño de un tablero de control, así como qué elementos y recomendaciones se deben tener en cuenta a la hora de desarrollarlos.

Storyboarding: el término se deriva del campo del diseño multimedia en donde la animación se conceptualiza primero a través de un tablero de bocetos para perfilar la secuencia de escenas y qué cambios surgen a través de ellas. En otras palabras el storyboard es el proceso de contar una historia para la animación a través de imágenes estáticas.
Similar a este concepto de multimedia que puede involucrar la definición de la interacción del usuario y su correspondiente respuesta: un buen tablero de control está repleto de esas interacciones de usuario y respuestas del tablero a esas acciones.
El storyboarding reúne las áreas claves del proceso dashboarding, es decir, la meta-información (información acerca de la información), el público, la presentación y las alarmas. Los siguientes pasos pueden seguirse como el storyboard de un tablero de control:

1. Identificar los grupos de usuarios clave.
2. Identificar las agrupaciones de tableros clave.
3. Determinar la matriz de privilegios: Grupos de usuarios y Grupos de tableros.
4. Boceto del esquema del tablero para cada grupo de tableros.
5. Boceto de la secuencia de navegación para cada componente del tablero.

Planificación del proyecto: la implementación de tableros de control exige la puesta en común de un conjunto de habilidades de diferentes áreas y niveles de especialización. Todo este proceso

requiere una buena coordinación y planificación de proyectos para garantizar un buen uso del tiempo de todos los integrantes del equipo.
El siguiente es un ejemplo de los recursos y el conocimiento necesarios para un equipo de dashboarding: experto en software de tableros de control, experto en inteligencia de negocio, analista del negocio, administrador de bases de datos, director de tecnologías de información, director de proyecto, entre otros.
Además de todo esto es crítico contar desde un comienzo con la ayuda de una muestra representativa de usuarios finales o algún grupo de prueba que evalúe el alcance y la información.
También cuando el proceso se encuentre en una etapa más avanzada, puede incluso formar parte del equipo, un diseñador gráfico que ayude en la parte estética para que esta sea más agradable al usuario, pudiendo también dar opciones de usar logos, colores, fondos, gráficos y símbolos de la compañía, todo para que sea un proyecto de gran impacto en la organización.

Gestión de proyectos: los buenos principios de la gestión de proyectos pueden aplicarse al proceso *dashboarding* al mismo grado que en cualquier otro tipo de proyecto. Metodologías conocidas como el modelo en cascada, RAD (Desarrollo Rápido de Aplicaciones), CMM (Modelo de Capacidad de Madurez) para software, entre otras, son ejemplos de ello. Cualquiera que sea el enfoque que se tome, los factores clave de éxito para un proyecto de tableros de control son los siguientes:

- Rápida participación de los usuarios.
- Una adecuada composición del equipo de trabajo.
- Colaboración y trabajo en equipo.
- Oportuna disponibilidad de los prerrequisitos del formato de los datos
- Selección adecuada del software para tableros de control.

Diseño del tablero: un tablero de control bien diseñado debe tener un atractivo estético y un despliegue poderoso de visualización, para poder transmitir gran cantidad de información dentro de un espacio limitado. Algunos elementos clave para el diseño del tablero son los siguientes:

- Gráficos de pantalla y colores**:** son muy importantes en el marco de construcción de un tablero de control, a diferencia de los reportes, el usuario espera mucho más atractivo visual de los tableros.
- Adecuada selección de los tipos de gráficos**:** la selección de los gráficos requiere una buena mezcla entre pensamiento analítico y representación artística. Dependiendo de la información que vaya a ser representada, algunos tipos de gráficos son los más apropiados. Por ejemplo, si se quisiera ver qué participación en el mercado posee la organización, un gráfico de torta o "*pie*" es indiscutiblemente bueno para representar esta información. Si se quisiera ver la tendencia o comportamiento de algún tema en específico, una línea gráfica funciona bien. Si se quiere comparar dos cifras, una columna doble o un gráfico en barras es una buena opción. Pero en muchas ocasiones la elección del grafico no es tan obvia, requiriendo así cierto grado de flexibilidad y creatividad a la hora de implementar los tableros. Algunos gráficos populares incluyen semáforos, velocímetros, termómetros *donuts* y gráficos de burbuja, pero todo esto debe tener en cuenta un elemento importante y es el espacio que existe en el tablero de control, por lo que la elección del gráfico no debe descuidar esta parte.
  Por ejemplo, si en la distribución de los elementos, el área que se tiene para el gráfico es estrecha pero tiene buena altura, un gráfico de termómetro funcionaría mejor que por ejemplo un velocímetro o un gráfico en barras que requieren más área cuadrada. Por último, hay que tener buenas prácticas en cuanto al color usado para estos gráficos, ya que si el tablero es para una cantidad considerable de usuarios, es bueno asesorarse de un diseñador gráfico sobre los colores que son buenos para ello, ya que como muchas cosas, el color también tiene su ciencia y es útil tener dicho conocimiento y aplicarlo.

- Aprovechar el poder de la percepción visual: el ojo y la corteza visual del cerebro humano, forman un procesador que proporciona el canal de mayor ancho de banda en los centros cognitivos humanos. En niveles superiores de procesamiento, la percepción y la cognición están estrechamente relacionadas entre sí. Sin embargo el sistema visual tiene sus reglas, ya que se puede ver qué patrones presentados de ciertas maneras pueden ser fácilmente visualizados, pero pueden resultar invisibles de otras. Por lo que si se logra entender cómo trabaja la percepción, el conocimiento puede traducirse en reglas para la visualización de la información y permitir presentar datos de tal manera que los patrones de la información importante se resalten [22].
- Entender los límites de la memoria a corto plazo: es el lugar donde la información se guarda durante el proceso consciente. Los elementos más importantes que hay que saber de la memoria a corto plazo son:
  - Es temporal
  - Una parte de ella es dedicada a la información visual
  - Tiene limitada capacidad de almacenamiento

  Lo que constituye un fragmento de la información visual varía en función de la naturaleza de los objetos que vemos, los aspectos del diseño y de qué tan familiarizados estemos con ellos. Por ello, cuando se almacenan números en un tablero de control, no generan tanto impacto como un patrón de líneas en una gráfica que puede representar mucha información en un fragmento. Esta es una de las grandes ventajas de mostrar información de manera gráfica, por esto los tableros de control deben ser diseñados de modo que la información que se quiere transmitir sea percibida y entendida de manera eficiente.

  La limitada capacidad de la memoria a corto plazo es también una razón del por qué la información no debe ser desplegada en muchos tableros de control. El navegar demasiado implica que la persona que está viendo la información tenga que ir a diferentes lugares para poder ver todo lo que se quiere mostrar, ocasionando que en ese ir y venir olvide elementos que ya haya entendido.
- La codificación visual de datos para una rápida asimilación: el proceso de pre-atención es la fase inicial de la percepción visual y ocurre a un nivel más bajo que el consciente. Además está sintonizado para detectar un conjunto específico de atributos visuales. Con ciertos parámetros y características se puede analizar cierta información más rápida de acuerdo al modo en que se nos presenta.

  Los atributos de pre-atención que tienen injerencia para una asimilación más rápida de la información se pueden organizar en 4 categorías:
  - Color
  - Forma
  - Orientación Espacial
  - Movimiento.
- Principios *Gestalt* de la percepción visual: en 1912, la escuela de Psicología *Gestalt* iniciaba sus esfuerzos fructíferos para comprender como percibimos patrones, formas y organización de lo que vemos. El término alemán *Gestalt* significa patrón. Estos investigadores reconocieron que nosotros organizamos lo que vemos de forma particular en un esfuerzo de darle sentido. Su trabajo concluyó con una serie de principios *Gestalt* de percepción que revela aquellas características visuales que nos inclinan a agrupar objetos. Estos principios aún hoy en día siguen vigentes y ofrecen algunas ideas útiles para aplicar en el diseño de los tableros. [22]
  - Principio de proximidad: percibimos los objetos que están localizados unos cerca de los otros como en un mismo grupo.

Figura 3.5. *Principio de proximidad explica por qué vemos 3 grupos en vez de 10 puntos en la imagen*

Esta es la manera más simple de enlazar datos que se quieren ver juntos. Un espacio en blanco es todo lo que se necesita para hacer esta separación. También este principio puede ser para que el espectador explore los datos en una dirección particular del tablero, ya sea de izquierda a derecha o de arriba a abajo. Colocar secciones de datos horizontalmente alienta a los ojos del observador a que la exploración de datos sea de izquierda a derecha. Sucede muy similarmente cuando se ubican verticalmente, lo cual la exploración será de arriba hacia abajo.

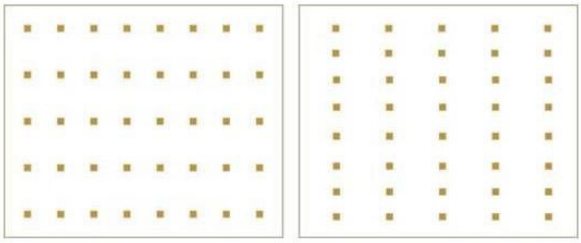


Figura 3.5. *Principio de proximidad puede ser usado para un análisis de datos horizontal o vertical*

- Principio de cercado o encerramiento: se perciben objetos como un grupo, cuando están rodeados por un borde visual (una línea por ejemplo o un color de fondo en una zona específica). Este "encierro" hace que los objetos parezcan separados en una región que se distingue del resto de elementos que se ven.

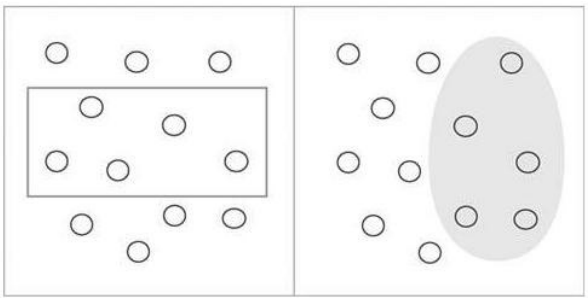


Figura 3.6. *Principio de cercado o encerramiento*

- Principio de cierre: esto se puede aplicar en los tableros, especialmente en los gráficos. Por ejemplo, se pueden agrupar objetos (puntos, líneas, barras) en donde se difieran las regiones que ponemos en el tablero sin la necesidad de sobrecargarlo con fronteras o fondos por completo. Esto debido a la necesidad de mostrar una gran cantidad de información en un espacio relativamente limitado, por lo que se necesita eliminar todo lo que no sea estrictamente necesario. En la Figura 3.7 vemos un ejemplo práctico delimitando un área con barras sólo trazando los ejes **x** e **y.**

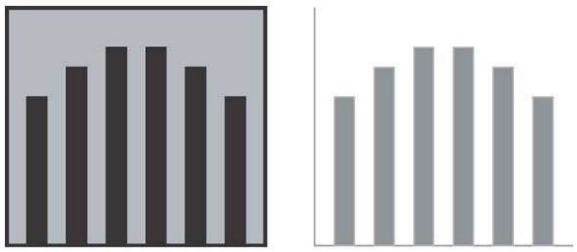


Figura 3.7. *Gráfico a partir de dos ejes que encierren el conjunto de Objetos*

- Principio de Continuidad: en esta imagen es claro qué elementos son nombres de divisiones, cuáles nombres de departamentos, basados sólo en la forma en que están alineados. Viendo la Figura 3.8 podemos determinar qué pertenece a cada grupo sin necesidad de líneas verticales que los separe. Esto se puede utilizar en los tableros para separar secciones de información y no sobrecargarlo con elementos que no son necesarios aprovechando este principio.

División/Departamento	Personal
G&A	
Finance	15
Purchasing	5
Information Systems	17
Sales	
Field Sales	47
Sales Operations	10
Engineering	
Product Development	22
Product Marketing	5

Figura 3.8. *Principio de continuidad explica cómo las sangrías en los textos funciona como agrupación de información*

- Principio de Conexión: la percepción de agrupación producida por la conexión es más fuerte que la que se produce por la proximidad o similitud de color, forma o tamaño. Únicamente es superado por el principio de encerramiento o cercado [Figura 3.9]. El principio de conexión es especialmente útil para unir datos no cuantitativos, por ejemplo cuando se quiere representar las relaciones entre los pasos de un proceso o entre los empleados de una organización.

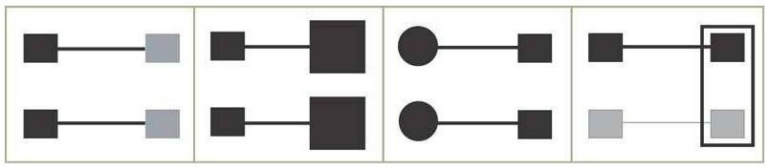


Figura 3.9. *Ejemplos de agrupamiento, sólo superado por el encerramiento*

Cuando se diseñan tableros, sólo se debe incluir la información que es absolutamente necesaria. Ella debe condensarse de forma que no se pierda significado y se debe mostrar por medio de mecanismos que, aunque sean pequeños, permitan leer y entender. Los tableros de control entregan la información:

- Excepcionalmente bien organizada.

- Condensada, principalmente en forma de resúmenes y excepciones.
- Específica y personalizada para un público.
- Despliega usando medios concisos que comuniquen el mensaje de la forma más clara posible.

Además, otra regla de oro que se debe tener presente siempre para el diseño de un tablero de control, es la simplicidad: mostrar datos de la manera más clara y simple posible y evitar distracciones y decoraciones innecesarias.

Tableros de control para monitoreo de procesos o actividades (*Process/Activity Monitoring Dashboard*): los tableros de control pueden ser desarrollados para monitorear procesos específicos del negocio o actividades generalizadas, tales como patrones del clima, seguridad nacional, control de enfermedades, entre otras.
Un tablero de control visualmente efectivo y analítico ayuda a evitar problemas antes de que se vuelvan realidad y que alcancen proporciones de crisis; en síntesis ellos ayudan a hacer un seguimiento de todas las actividades que se llevan a cabo en la organización y ver qué ocurre con ellos previendo problemas futuros y otorgando la capacidad de cambiar el rumbo de la organización ante la detección de alguna situación de alerta.
Precisamente este constituye el tipo de tablero de control que se expondrá en el caso de estudio más adelante [28].

Concluyendo, los tableros de control les dicen a las personas lo que está pasando y les ayuda a reconocer inmediatamente qué necesita de su atención, tal cual como el tablero de un automóvil, para el monitoreo de velocidad, combustible, nivel de aceite, gasolina, problemas en la maquinaria, entre otros.
Para poder llevar a cabo todas estas tareas de monitoreo en un contexto específico, se requieren los medios adecuados como para comunicar dicha información. Por ello un tablero debe estar construido utilizando adecuadamente estos medios.

El siguiente capítulo describirá los medios o tecnologías que se consideraron convenientes para conformar la visualización del tablero de control a desarrollar. Estas tecnologías son: **Portales y Portlets**.

4

4. Portales y Portlets

Cada día es mayor la necesidad que tienen las organizaciones de motivar a sus usuarios, concentrando en un único sitio una serie de servicios y contenidos que se hallan dispersos. La tendencia actual se enfoca en el desarrollo de portales Web que permitan mejorar la experiencia del usuario, ofreciéndole la información adecuada para satisfacer sus necesidades. Como una alternativa en la construcción de portales se destaca el empleo de los portlets. Estos componentes son colocados en el portal y pueden mostrar extractos de sitios Web, ejecutar búsquedas o acceder a colecciones de información.
Tras lo mencionado y las características que ofrecen, se eligió a estos elementos como las tecnologías a utilizar para la conformación visual del tablero de control.
En este capítulo se describirá en detalle las ventajas y las cualidades de estas tecnologías.

4.1. Portales

En los últimos años han empezado a aparecer en las empresas aplicaciones saturadas de datos y contenidos, que invaden con grandes cantidades de información a las organizaciones. Es evidente la necesidad de hallar una solución que integre las aplicaciones y unifique los contenidos. Las empresas han comenzado a apostar por el empleo de aplicaciones que recuperen gran cantidad de información expuesta en distintos puntos, haciéndola disponible desde una interfaz única. Este tipo de aplicación recibe el nombre de aplicación compuesta [23].
Hoy en día la tendencia a seguir para el desarrollo de estas aplicaciones son los portales web. Al referirse al concepto de portal en [24] se plantea que es una aplicación que integra información provista por múltiples sistemas y presenta los resultados en una interfaz cómoda para el usuario. El mismo responde a las necesidades de los usuarios permitiendo la interacción con información diversa, procesos de negocio y personas.
El objetivo principal de los portales consiste en lograr que los usuarios los utilicen como la página de inicio para navegar [25]. Además de ofrecer de forma fácil e integrada, el acceso a una serie de recursos y servicios para resolver sus necesidades específicas.
Un portal es un modelo que inspira y ordena la construcción en la red. Su estructura implica una fuerte interacción entre las diversas aplicaciones, que permiten analizar interrelaciones que serían realmente complejas y lentas si no se contara con ellas [26]. Los portales mantienen a los usuarios actualizados, a partir de ofrecer una serie de enlaces de interés. La mayoría permiten configurar la apariencia de las páginas, adaptándose a las preferencias del usuario. Sin duda, son útiles a la hora de ganar tiempo y organizar un poco el desorden de la red, además de facilitar la entrada a un conjunto de servicios, comercio e información. Su aceptación está dada por el potencial que poseen para desarrollar nuevas formas de comunicación.

4.1.1 Definición y características:

El portal es un sitio web que provee un único punto de interacción con aplicaciones, información, personas y procesos, personalizados a las necesidades y responsabilidades del usuario. Constituyen la próxima generación de escritorios de trabajo, permitiendo la ejecución de aplicaciones a través de Internet utilizando no sólo una PC, sino además otros dispositivos como PDA (*Personal Digital Assistant* o *asistente digital personal*) y teléfonos celulares.
Un portal brinda de una manera integrada contenidos y aplicaciones, con el agregado de un espacio de trabajo unificado y colaborativo, con el objeto de proveer al usuario de toda la información relevante que necesita para poder tomar decisiones de manera acertada, acorde a sus necesidades y responsabilidades, en cualquier lugar y a cualquier hora.
Los portales permiten entonces el acceso a una serie de recursos y de servicios relacionados con un mismo tema. Incluyen: enlaces, buscadores, foros, documentos, aplicaciones, compra electrónica, entre otras funcionalidades. Principalmente un portal en Internet está dirigido a resolver necesidades de información específica de un tema en particular.
Además de eso, proveen otras funciones de valor agregado como seguridad, *single sign-on*, publicación y búsqueda de contenidos, herramientas de colaboración y *workflow*. Cuentan con una serie de funcionalidades que permiten a usuarios personalizar y organizar su propia vista del portal, manejar sus perfiles y publicar y compartir documentos entre ellos.

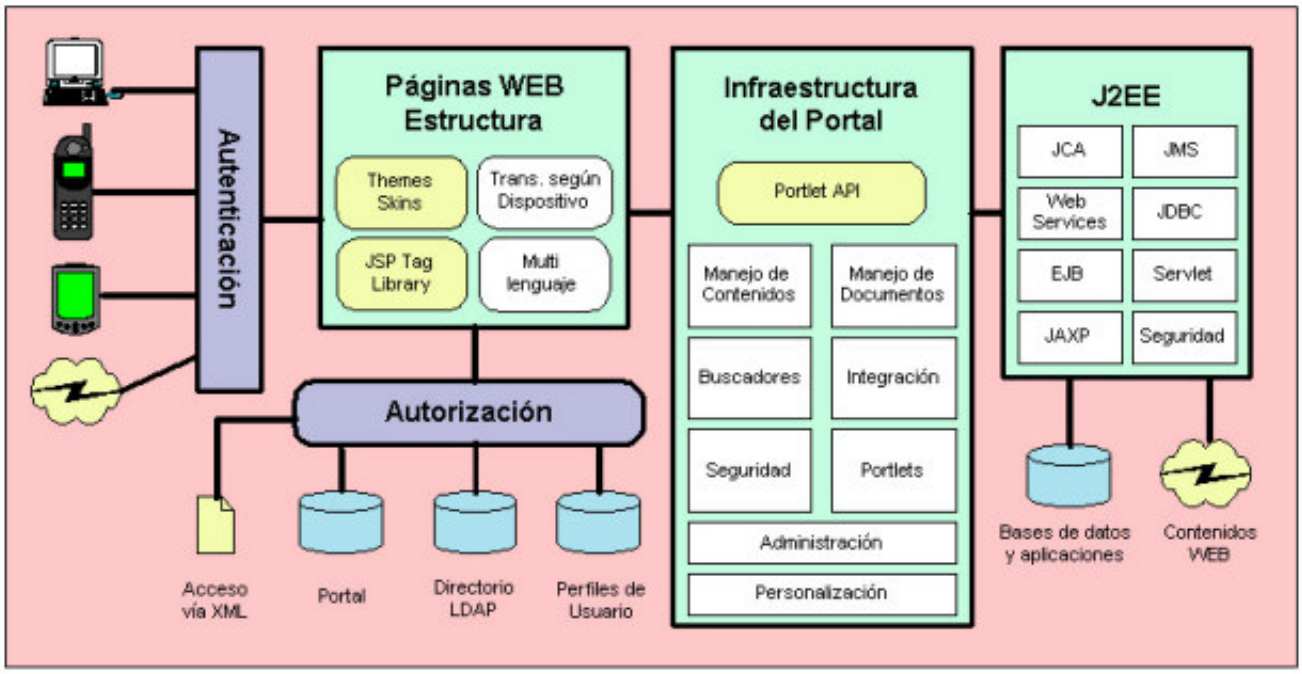


Figura 4.2. *Tecnologías J2EE involucradas en el portal*

Portales Móviles: un portal no sólo puede ser accesible a través del navegador de una PC; también es posible acceder al mismo utilizando dispositivos móviles, soportando su acceso mediante la generación de páginas en tres lenguajes: HTML, WML para dispositivos WAP (teléfonos móviles) y HTML para dispositivos móviles.
Los usuarios pueden personalizar una página de inicio para cada dispositivo, seleccionando el contenido y las aplicaciones más útiles para el mismo. Cuando esta página es requerida, el portal determina el dispositivo del cual proviene el requerimiento y ensambla los contenidos de la misma utilizando el lenguaje acorde al dispositivo.

El portal es considerado un intermediario de información que tiene como fuente de ingreso la de tener una forma simple de acceder a toda y no sólo a una parte de la información referida al tema del mismo.

4.2 Portlets

Actualmente, las grandes compañías invierten en desarrollar tecnologías que faciliten la creación de portales. Entre ellas los portlets se destacan como los componentes Web de interfaz de usuario, capaces de procesar peticiones de los clientes y generar contenido dinámico [27].

La página de un portal se visualiza como una colección de ventanas de portlet que no se solapan. Por lo tanto una colección de portlets se encuentra hospedada dentro del portal, aportando una vista sencilla de información que se origina desde múltiples fuentes. Ejemplos típicos de portlets pueden ser de noticias provenientes de un Sistema para la Gestión de Contenido (CMS), correo electrónico, foros, encuestas, formularios, servicios Web, integración de aplicaciones, herramientas de análisis, herramientas de trabajo en grupo, entre otros.

4.2.1 Definición y características:

Los portlets son componentes modulares de las interfaces de usuario gestionadas y visualizadas en un portal web, constituyendo estos un entorno de ejecución para portlets. Generalmente, se invocan muchos portlets en una única solicitud de una página al portal. Cada portlet produce un fragmento de la salida, que es combinado con la salida de otros portlets, todo junto con la salida de una página del portal.
Los portlets hacen referencia a una pequeña aplicación del portal, usualmente diagramados como un componente dentro de la página web. Cualquier portlet puede ser desarrollado, publicado y ejecutado independientemente de otro portlet, existiendo una serie de portlets ya disponibles para las funcionalidades básicas del portal (e-mail, calendario, chat, entre otros.), los cuales sólo es preciso configurarlos para su implementación.
Los portlets no son sólo simples vistas de contenido web, son aplicaciones completas, que resuelven una problemática determinada. Para ello cuentan con sus propios datos y pantallas, con la posibilidad de integrarse a otros portlets ya publicados en el portal. Cuentan con un entorno de ejecución (*portlet container*) provisto por la infraestructura del portal, que es utilizada además para acceder a otros servicios como seguridad, manejo de ventanas, personalización, acceso a contenido remoto, entre otros. Este entorno de ejecución está basado en una API (*Application Programming Interface* o Interface de Programación de aplicaciones) que provee las clases e interfaces necesarias para programar los portlets, lo que posibilita abstraerse de los mecanismos utilizados a bajo nivel por el portal para su ejecución.
Generalmente una aplicación que se ejecute en el portal, va a estar formada por una serie de portlets, y serán los administradores del portal los encargados de configurarlos en un entorno de producción, sin necesidad de detener el portal.

Funcionalidades: los portlets poseen las siguientes funcionalidades.

- Almacenamiento persistente de las preferencias
- Procesamiento de solicitudes
- Modos de los Portlets
- Estado de la Ventana

Modalidades de visualización: los portlets proveen diferentes interfaces con el usuario, dependiendo de la tarea a realizar, entre ellos los modos de visualización, edición, ayuda y configuración, todos invocados desde la barra superior de la ventana del portlet a través de iconos.
El modo de visualización es el modo normal de ejecución; el de ayuda provee al usuario asistencia referida al uso del portlet; el modo de edición permite al usuario cambiar seteos de presentación y el modo de configuración. Si está soportado por el portlet, provee una página de configuración, cuya configuración será compartida por todos los usuarios.
Cada ventana donde se visualiza el portlet se puede maximizar y minimizar; cuando un portlet es maximizado, este ocupa toda la ventana del portal, cubriendo las vistas de otros portlets de la página; y cuando es minimizado sólo se visualiza la barra superior de la ventana con el título.

Facilidades de integración: los portlets incluyen la funcionalidad de *Click to Action* (Clickear para accionar). Esta característica se refiere a la capacidad de transmitir la información de un portlet a otro, en tiempo real. Esta facilidad es la que permite a los usuarios acceder a información integrada aunque los recursos que utilicen se encuentren materialmente distribuidos en diferentes aplicaciones y bases de datos.

Estándares: los portlets en cualquier aplicación de negocio constituyen la suma de un conjunto de funciones que se despliegan en una ventana de interacción con el usuario. Cuantos más potentes sean los portlets más potente será la aplicación de negocio. En este contexto resulta crucial cumplir con estándares, porque permite aprovechar de modo inmediato el trabajo hecho para portales de otros fabricantes.
En particular, la especificación JSR-168 describe la API estándar de **Java** para el desarrollo de portlets que permite la interoperabilidad de los mismos entre diferentes portales Web. Dicha especificación pone orden en el mundo de los portales permitiendo que los portlets puedan ser compatibles entre aquellas plataformas que cumplan la especificación, lo que supone en la práctica que dichos componente pueden ser instalados en cualquier portal sin sufrir modificaciones.
Para desarrollar portlets basados en JSR168 o portlets estándar como también se les conoce, se puede utilizar un simple editor de texto o utilizar entornos de programación comúnmente conocidos como IDE, entre ellos:

- Eclipse
- Sun Java Studio Creator
- Oracle JDeveloper
- Jbuilder
- IntelliJ
- Rational Application Developer for WebSphere Software

Otro estándar para la incorporación de contenido lo constituye el protocolo de Servicios Web para Portlets Remotos (WSRP). Este protocolo permite acceder y mostrar contenido que se halla en un servidor remoto, así como el desarrollo de portlets con cualquier tecnología, ya sea J2EE, .NET o cualquier servicio accesible mediante SOAP. WSRP separa portlets de portales, e introduce los conceptos de productor (proveedor de servicio) y de consumidor (cliente del servicio).
El empleo de los portlets en la construcción de portales reporta un conjunto amplio de ventajas, pero su primacía radica en el hecho de que son componentes Web estándar y reutilizables, que pueden ser consumidos en cualquier portal que contenga un productor y que siga la especificación JSR 168. A ello se suma que pueden ser añadidos, quitados y recolocados de manera dinámica dentro del portal, sin necesidad de reprogramar nada.
Con los estándares se procura que el desarrollador de software conciba portlets que puedan ser utilizados en cualquier portal que soporte dichos estándares.

Integración de aplicaciones:

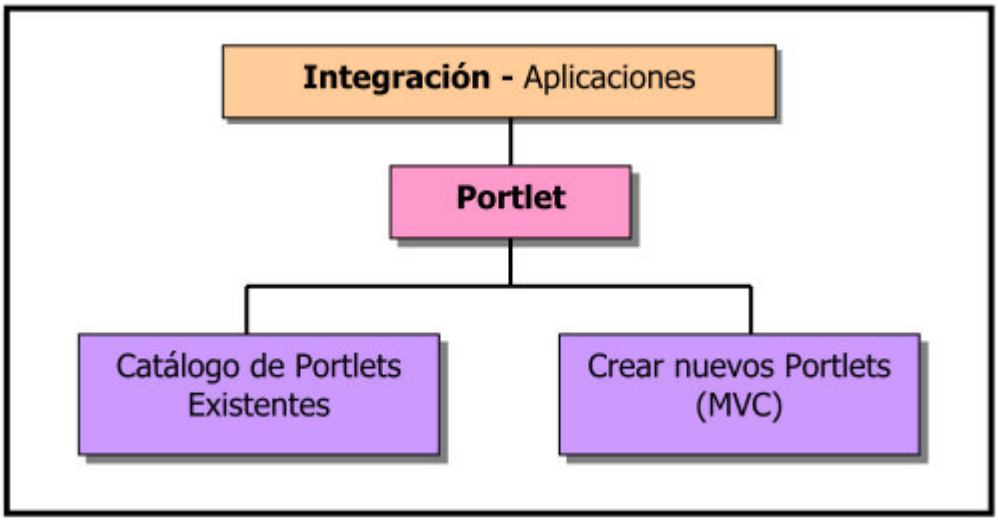


Figura 4.3. *Integración de Aplicaciones*

El mecanismo de acceso a aplicaciones existentes dentro del portal es mediante un portlet.
Para ello existen dos alternativas:

- Configurar y personalizar portlets ya existentes.
- Desarrollar nuevos portlets siguiendo la arquitectura MVC (*Model – View – Controller* – Modelo – Vista - Controlador).

El desarrollo de nuevas aplicaciones mediante el uso de portlets, tiene como objetivo brindar soluciones que sean fáciles de utilizar por los usuarios del portal, y ayuden a trabajar de manera eficiente, combinando todo lo que ellos necesitan en un solo lugar.

Contenedor de portlets: ejecuta los portlets, les provee el ambiente de ejecución requerido y maneja el ciclo de vida de los mismos.

- Este es el que provee el almacenamiento persistente para las preferencias de los portlets.
- Recibe pedidos desde el portal para ejecutar pedidos en los portlets que aloja. El contenedor invoca a los portlets que se encargarán de generar el fragmento, el cual es manejado por el contenedor para ser pasado al servidor del portal que estará a cargo de agregarlo a la página del portal.

Portlets vs. Servlets: un servlet es una clase en el lenguaje de programación Java, utilizada para ampliar las capacidades de un servidor. Aunque los servlets pueden responder a cualquier tipo de solicitudes, éstos son utilizados comúnmente para extender las aplicaciones alojadas por servidores web, de tal manera que pueden ser vistos como applets de Java que se ejecutan en servidores en vez de navegadores web. Existen varios puntos en común con respecto a los portlets, a continuación se enumeran las similitudes y diferencias.

Los portlets son similares a los servlets en los siguientes puntos:

- Los portlets son manejados por un contenedor especializado.
- Los portlets generan contenido dinámicamente.
- El ciclo de vida de los portlets es controlado por el contenedor.

- Los portlets interactúan con el cliente Web mediante el uso del paradigma *request/response* (requerimiento - respuesta).

Los portlets son diferentes a los servlets en los siguientes puntos:

- Los portlets son únicamente generados como fragmentos etiquetados y no como documentos completos.
- Los portlets son accesibles directamente por una URL.
- Los portlets no pueden generar contenidos arbitrarios, ya que el contenido de los mismos va a estar incluido la página del portal. Si un servidor de un portal está solicitando html/text, entonces todos los portlets deben ser generados en html/text. Por otro lado si el servidor del portal está solicitando contenido WML, entonces cada portal deberá ser generado dicho estándar

Tipos de peticiones: existen dos tipos de peticiones "*action request*" y "*render request*":

- Peticiones de acción (**action request**): son peticiones que modifican el estado del portlet o que causan una redirección. Se procesan redefiniendo el método *processAction*.
- Petición de renderización (**render request**): son peticiones que solicitan el markup (<<visualización>>) del portlet. Se invoca al método *render(...)*

Una petición de acción sobre un portlet siempre va seguida de una petición de renderización sobre ese portlet, y sobre el resto de portlets de esa página (cuyo contenido no esté almacenado en caché).

Formas de compartir datos: algunas maneras de compartir datos entre portlets son:

- Portlet Session: cada WAR (archivo de aplicación web) contiene su propia sesión y no es compartida con otros WAR. Esto quiere decir que portlets del mismo WAR deberían poder comunicarse a través de la sesión.
- Mecanismos IPC:
  - Public Render Parameters: en JSR-168 setear parámetros render está habilitado sólo para renders del mismo portlet. Con los parámetros render públicos, los parámetros seteados en un *processAction* pueden estar habilitados en render de otros portlets
  - Event: los portlet events son disparados por el portlet que está configurado para enviar el evento y sólo los portlets que están configurados para escuchar el evento lo reciben. De modo que el ciclo de vida pasa a ser: *actionRequest - event – renderRequest*

Con este tema concluye el marco teórico que da soporte conceptual a la aplicación que se expondrá en la siguiente sección. El portal elegido para el desarrollo del tablero de control del caso de estudio es Liferay. Sus particularidades serán detalladas en el siguiente capítulo, junto con cada una de las tecnologías o productos específicos que ponen en práctica los conceptos que se vieron hasta ahora, de manera tal de posibilitar la implementación del tablero de control para monitorear los procesos de negocio.

5

5. Desarrollo del Tablero de Control

Luego de haber analizado los conceptos de BAM, CEP, de Tableros de Control y por último de Portales y Portlets, se posee el marco teórico necesario para aproximarse a la construcción del Tablero de Control que lleve a cabo el monitoreo de los procesos de negocio.
Para esto se seleccionaron varias aplicaciones *open source* que aplican los conceptos requeridos y que a su vez permiten su integración para llevar a cabo la implementación del tablero. La integración mencionada de tecnologías novedosas está sustentada por una arquitectura de base, lo cual es parte del aporte de este trabajo.
En el capítulo siguiente se describirán las herramientas seleccionadas, junto con la arquitectura a la que se llegó integrando tales tecnologías con el propósito de realizar el monitoreo efectivo de los procesos de negocio. Además, se comentará el aporte tecnológico del desarrollo en general. Esta sección finalizará con la presentación de un caso de estudio que muestre un ejemplo del funcionamiento del Tablero de Control desarrollado.

5.1 Herramientas utilizadas

En esta sección se describen las aplicaciones utilizadas para el desarrollo. Estas son: Bonita OS, WSO2 BAM, WSO2 CEP y Liferay. En base a estas descripciones, se explican brevemente los motivos por los cuales fueron elegidas dichas herramientas, junto con la forma en que funcionan en la aplicación subyacente.

5.1.1. Bonita OS:

La gestión de procesos de negocios (Business Process Management: BPM) consiste en la metodología corporativa, que tiene como objetivo mejorar la eficiencia dentro de las organizaciones por medio de la gestión de procesos de negocio, que se deben modelar, organizar, documentar y optimizar de forma continua [29].
El gestor de procesos de negocios incluye un conjunto de recursos y actividades interrelacionadas que transforman elementos de entrada en elementos de salida. Bonita Soft es un gestor de procesos de negocios en software libre, también visto como un conjunto de aplicaciones de ofimática para la gestión de procesos de negocio. Es de código abierto, y puede ser descargado bajo el licenciamiento GPL v2.
Uno de los objetivos de BonitaSoft es democratizar BPM con una solución fácil e intuitiva que permita minimizar el costo de implantación. Esta aspira a convertirse en el líder mundial en soluciones de gestión de procesos empresariales de código abierto (*Open Source Business Process*

Management - BPM), proporcionando soluciones de BPM flexibles y potentes para las organizaciones.
Existen 2 versiones dentro del producto Bonita BPM:

- Bonita BPM Community edition: esta versión es de descarga gratuita, a través de la página http://www.bonitasoft.com/, y no necesita licencia para su utilización.
- Bonita BPM Subscription Pack versión: Ésta versión requiere una licencia. Se proporcionan diferentes licencias de acuerdo con el número de características, el entorno de despliegue, el número de núcleos, entre otras características.

Como ya ha sido mencionado, la versión libre de Bonita OS carece de un componente de monitoreo de procesos. Esta es una de las causas que refuerza el desarrollo de un componente BAM que sea capaz de ser compatible con este BPMS y de forma independiente enriquecerlo.

5.1.2. WSO2 BAM:

WSO2 ofrece una completa suite de productos Open Source SOA y son contribuidores de muchos de los productos de Apache, como Apache Axis, Apache Rampart, Apache Synapse, todos ellos distribuidos bajo licencia Apache.
Una de sus características es que la versión libre descargada de sus productos también es única y se trata de la versión "productiva", ya que no manejan una versión alternativa "community", como otros vendedores [30].

Por otro lado, el concepto *Business Active Monitoring* (BAM) describe los procesos y tecnologías que permiten el análisis de indicadores de negocio basados en datos en tiempo real. BAM es usado para mejorar la velocidad y efectividad de operaciones de negocio, manteniendo la pista de qué es lo que está pasando y haciendo visibles los posibles problemas. El concepto BAM puede ser implementado con muchos tipos de herramientas de software [31].

WSO2 ofrece un producto BAM de propósito general, es decir, uno que no sólo es para monitorizar indicadores de procesos de negocio, sino casi cualquier fuente de datos que se conecten usando su API.

Uno de sus usos es el caso de *Big Data*, un conjunto de datos muy extenso y diverso difícil de procesar usando herramientas de base de datos o aplicaciones tradicionales de procesamiento de datos. En estos casos, BAM utiliza productos como Hadoop para procesar tal cantidad de datos en poco tiempo, incluyendo las capacidades para capturar, subsanar, almacenar, buscar, transferir, analizar y visualizar estos datos. Además, incluye una función *out-of-the-box*, muy sencilla y simple de configurar, para monitorear otros productos WSO2.

Precisamente este es el uso que se le va a dar a esta aplicación en el caso de estudio.
Capacidad *multitenancy* [Figura 5.1] (*clientes múltiples* corresponde a un principio de arquitectura de software en la cual una sola instancia de la aplicación se ejecuta en el servidor, pero sirviendo a múltiples clientes o instancias): en una organización grande, es muy común tener departamentos y dominios diferentes de datos, lo cual va unido a la necesidad de tener que separar totalmente la administración, la configuración y los datos que usa el producto.
Gracias a esta capacidad, esto se puede lograr sin necesidad de crear otra instancia de este.

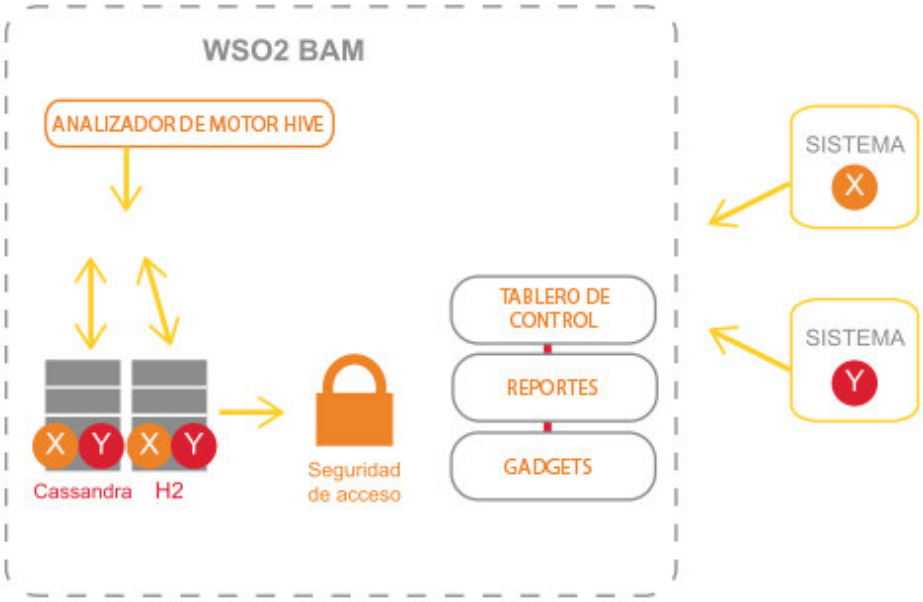


Figura 5.1. *Capacidad "multi-instancia" de WSO2 BAM*

Capacidad de visualización: el input de los componentes de visualización es, por defecto, la base de datos relacional H2.
Para esta capacidad se tiene tres componentes:

- Dashboard: para indicadores en tiempo real en una interfaz web, no consumible desde otra aplicación.
- Reports: para tener reportes impresos de JasperReports
- Gadgets: es el componente más interesante de los tres. Los gadgets son pequeñas porciones consumibles, como portlets, que siguen las especificaciones de Google Gadgets; y que se pueden desplegar en cualquier servidor, como JIRA o Confluence, siempre y cuando soporte las especificaciones de Google. Este servidor, recibe los datos que llegan de un servicio REST implementado con Jaggery; y se encarga de formatearlos, usando sólo CSS, HTML5 y Javascript. Los gadgets pueden ser construidos desde cero o basándonos en uno generado con Gadget Generator Tool.

Estas constituyen formas de visualización que ya tiene incorporadas el producto WSO2 BAM, aunque las mismas carecen de personalización. Es por eso que se consideró conveniente abstraerse de estas herramientas, y generar un apartado visual independiente, como es el caso del uso del portal Liferay y los portlets embebidos. Quizás la herramienta más similar a las mencionadas es el uso de Gadgets, mas por cuestiones de performance y de capacidad de personalización se decidió por los Portlets.

Capacidad de escalamiento y alta disponibilidad [Figura 5.2]: WSO2 BAM puede crecer horizontalmente, ya que cada uno de sus componentes se puede duplicar y ser parte de un cluster; pero también puede ser desplegado en alta disponibilidad. Los *Data Receivers* se pueden configurar y crecer en número, evitando posibles problemas de cuello de botella cuando hay alto tráfico de datos.
La base de datos Cassandra puede formar un cluster usando varias máquinas, ya que WSO2 BAM reconoce esta configuración y la usa. Los procesos de análisis Hadoop se pueden hacer en varias máquinas, pudiendo sacar de ello un procesamiento intensivo y rápido.
Finalmente, la aplicación web de visualización (Dashboard, Reports y Gadgets u otra independiente) también se puede clusterizar, ya que se ejecutaría sobre un servidor de aplicación WSO2 Carbon.

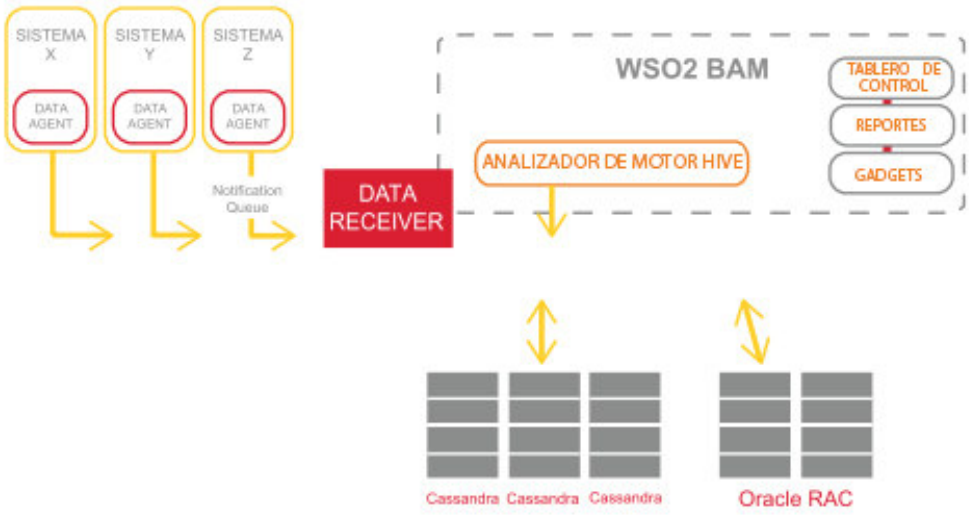


Figura 5.2. *Capacidad de escalamiento y alta disponibilidad de WSO2 BAM*

¿Cómo funciona WSO2 BAM?

- Paso 1: los datos llegan de sistemas externos a través del **API Data Agent**, quien puede hacerlo de forma sincronizada o no. Es decir, en este primer instante, se posee una API REST para enviar los datos al servidor, de manera que para ejecutar los datos del cliente se puede utilizar cualquier lenguaje. En el caso de utilizar Java, se dispondría de librerías para hacerlo de forma más fácil.
  En el caso de estudio se utilizaron dichas librerías de Java.

- Paso 2: cuando se genera un evento (*data publisher*), los datos son recibidos por el componente **Data Receiver**, quien los carga en **Cassandra**, la base de datos en donde se almacenan los que no han sido procesados.
  Es posible tener 1 o n *data receivers* incluso en alta disponibilidad. En el caso de que sea necesario procesar más eventos y no alcance con los *receivers* actuales, se pueden añadir más de estos.
  Cassandra es una *key database*, *nosql*, almacena pares de datos. Escalable y consume muy pocos recursos. No se guía por el concepto relacional. Del mismo modo, si se requiere un procesamiento mayor, se puede añadir un nuevo nodo al cluster de Cassandra.

  Apache Cassandra es una base de datos no relacional distribuida y basada en un modelo de almacenamiento de clave-valor, escrita en Java. Permite grandes volúmenes de datos en forma distribuida. Por ejemplo, lo usa Twitter para su plataforma. Su objetivo principal es la escalabilidad lineal y la disponibilidad. La arquitectura distribuida de Cassandra está basada en una serie de nodos iguales que se comunican con un protocolo P2P con lo que la redundancia es máxima.
  Cassandra es desarrollada por la Apache Software Foundation.
  En las versiones iniciales este producto utilizaba una API propia para poder acceder a la base de datos. En los últimos tiempos están apostando por un lenguaje denominado CQL (Cassandra Query Language) que posee una sintaxis similar a SQL aunque con muchas menos funcionalidades. Esto hace que iniciarse en el uso de la misma sea más sencillo. Permite acceder en Java desde JDBC.

- Paso 3: para analizar los datos, se usa **Apache Hive**, encargado de recoger los datos de Cassandra, analizarlos, e insertarlos en la base de datos relacional (por defecto H2, pero también puede ser Oracle, MySQL, u otra). En el caso de estudio se utilizó Mysql.
  En el caso de usar Apache Hive, ayudará a abstraerse de la fuente y destino de los datos; lo que permitiría centrarse solamente en implementar los Querys Hive, los cuales son SQL "friendly", y analizan los datos. Estos Querys se pueden programar para ejecutarse cada X cantidad de tiempo.
  Apache Hive usa, por debajo, procesos Hadoop para ejecutar la función analítica en paralelo, usando MapReduce.

  - Hive procesa sobre el cluster Hadoop, utilizando la técnica **MapReduce** *(divide y vencerás)*. 1 o n nodos en el cluster de Hadoop permiten aplicar MapReduce sobre la BD Cassandra.
  - Como su propio nombre indica, este modelo se basa en dos funciones, map() y reduce() que se aplican sobre datos estructurados en tuplas del tipo (clave, valor) en dos fases. En la primera, se reparten los datos entre varios procesadores que trabajan en paralelo (Clusters); y en la segunda fase (reduce), el resultado de la primera se reduce y se integra.

    Por ejemplo, resultaría imposible realizar la operación *group by* sobre Mysql teniendo millones de registros, en términos de eficiencia. Por eso suelen utilizarse técnicas como las mencionadas

- Paso 4: los datos ya analizados, son insertados en la base de datos relacional, y están listos para ser explotados por las herramientas de visualización. En el caso de este trabajo los portlets en el portal Liferay.

- Paso 5: a continuación, se extraen de la base de datos relacional los datos procesados, para poder ser presentados.

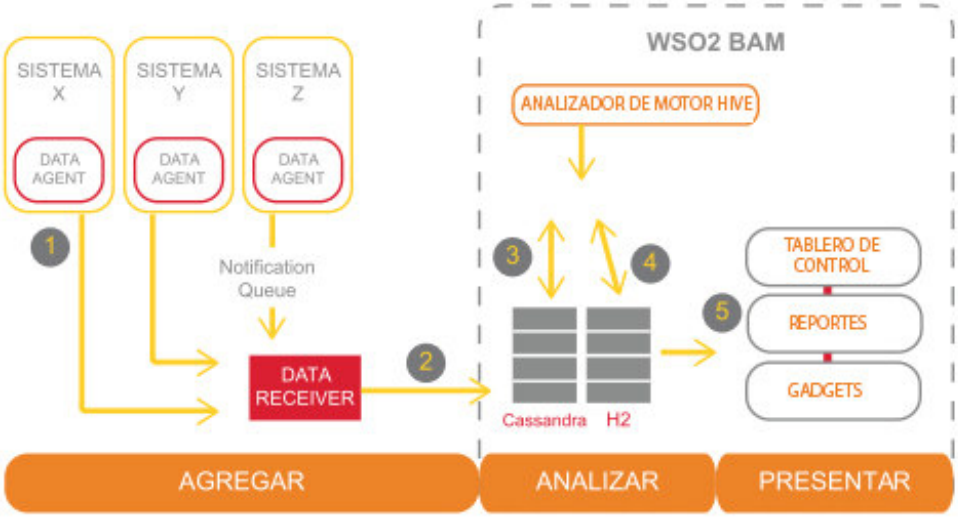


Figura 5.3. *Arquitectura de WSO2 BAM*

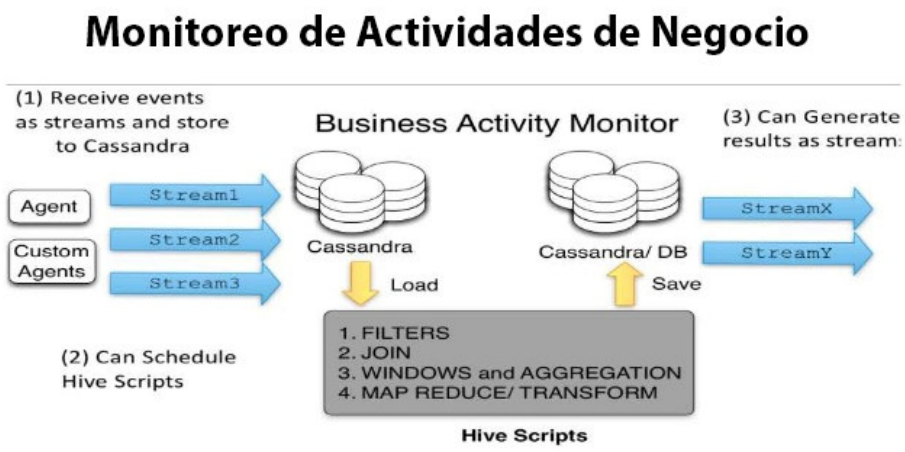


Figura 5.4. *Funcionamiento de WSO2 BAM*

En el caso de estudio, se utilizó **WSO2 BAM** para el procesamiento de un alto volumen de datos (*Big Data*). Los datos se procesan cada cierto tiempo, por lo que se usará WSO2 BAM para información histórica, NO de tiempo real.
Esto ocurre debido a las capacidades de esta aplicación previamente descriptas.

5.1.3. WSO2 CEP:

Todas las empresas para sus operaciones y transacciones diarias deben generar un sinnúmero de información a ser intercambiada entre emisores y receptores. Estas acciones son automatizadas en sistemas informáticos pues se tendrá un gran flujo de información a través de las redes de la empresa y hacia y desde el exterior. Este flujo de información consiste en un flujo constante de eventos que se generan y aquellas empresas que son capaces de capturarlos y analizarlos en **tiempo real** tienen una ventaja mayor con respecto a sus competidores. Esto se debe a que pueden saber qué pasa en todo momento en su negocio y reaccionar rápidamente ante eventos o patrones incorrectos.
Claro que implementar una solución que capture los eventos, los procese y genere respuestas en dependencia de los patrones que ocurran no es tarea fácil, y es ahí donde WSO2 sirve de gran ayuda [32].

El CEP o *Complex Event Processor* de WSO2, que en español vendría a ser "Procesador de Eventos Complejos" es una herramienta que recibe eventos generados por el resto de las herramientas de la suite, y por cualquier otra solución que se configure para ello, siendo capaz de determinar en tiempo real si determinados patrones de eventos están ocurriendo, y reaccionar ante ellos consecuentemente. Está de más decir que la herramienta está bajo licencia apache v2, y que tiene un alto rendimiento y escala de lo mejor.

Cuando se comentó que la solución recibe eventos de las herramientas de la suite y de cualquier otra herramienta que se configure para que lo haga, significa que en el caso de las herramientas de la suite es necesario ajustarlas para que todos los eventos generados por ella sean enviados al servidor del CEP, y en el caso de otros tipos de herramientas o soluciones informáticas pues igualmente hay que implementar para que hagan lo mismo. Afortunadamente es muy similar a cómo se hace con el BAM.

Arquitectura y funcionamiento de WSO2 CEP: la arquitectura del CEP [Figura 5.5] puede verse como un poco compleja al inicio, pero es bastante fácil de entender.

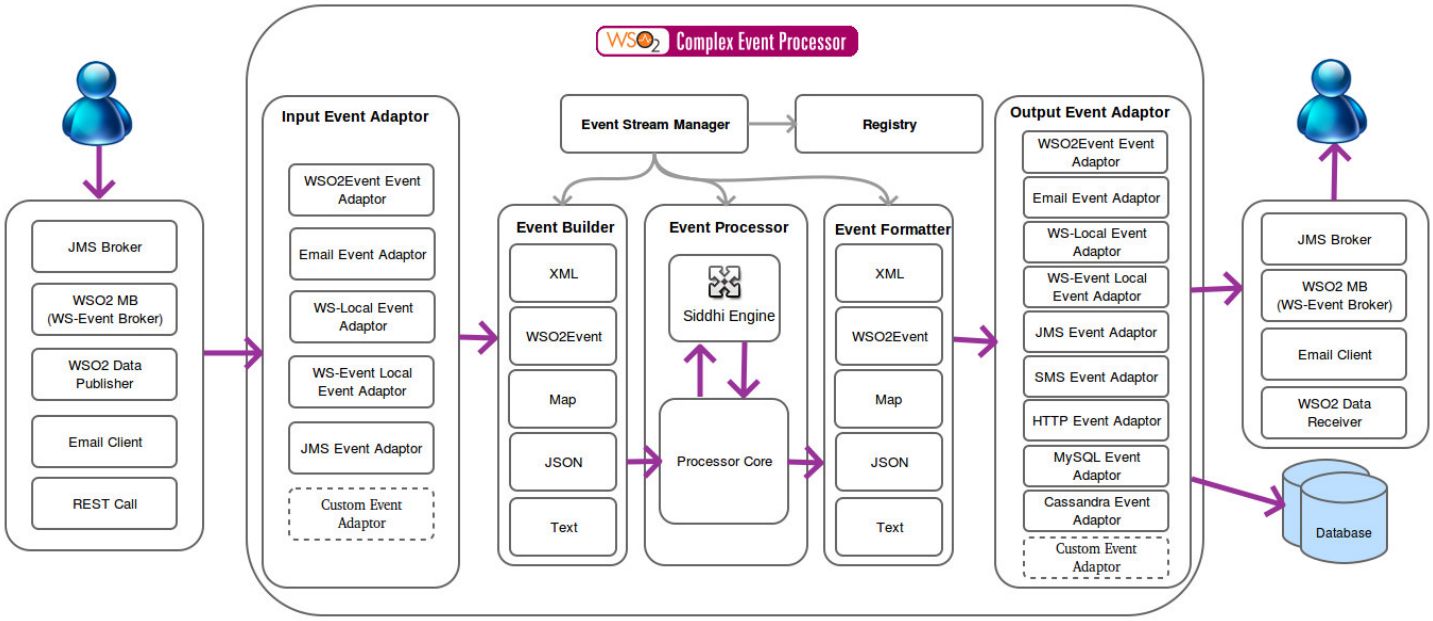


Figura 5.5. *Arquitectura de WSO2 CEP*

Como se ve, los eventos son generados de diferentes maneras, bien sea a través de *brokers* de mensajería, a través de los publicadores de datos de WSO2, vía email o por llamadas REST.

Cuando estos eventos llegan al CEP son recibidos por un "*Input Event Adaptor*" (Adaptador de eventos de entrada) que se corresponde a la forma en que se mandó el evento recibido. Lo interesante es que si se ha implementado una aplicación que manda los eventos al CEP de una manera no registrada aquí, es posible implementar un adaptador propio e incluirlo dentro del CEP.
Luego los eventos son pasados al "*Event Builder*" que se encarga de convertir el formato de los eventos a un formato estándar "*WSO2Event*" y de esta manera llegan los eventos al *core* o núcleo del CEP, el denominado "*Event Processor*".

En el "*Event Processor*" se manejan diferentes planes de ejecución con la ayuda de *Siddhi*, que es el framework base usado por el CEP para el manejo de eventos. Aquí es donde ocurre lo interesante del procesamiento.

Luego los eventos son nuevamente convertidos de "*WSO2Event*" a su formato de destino en el "*Event Formatter*".

La publicación de los eventos ocurre entonces en el "*Output Event Adaptor*" encargado de mandar los eventos publicados a sus destinatarios.

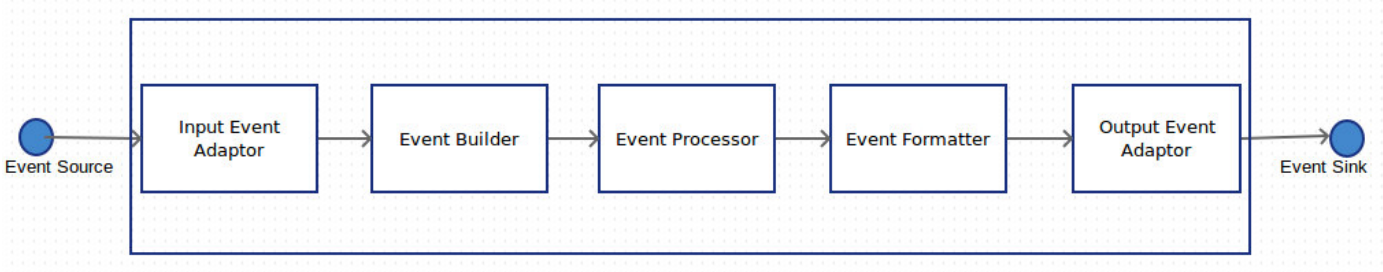


Figura 5.6. *Módulos de WSO2 CEP*

Para entender la utilidad del CEP algunas cosas que se pueden hacer:

- Se puede implementar un plan de ejecución que detecte en tiempo real cuando el precio de determinadas acciones en el mercado han cambiado más de un 2% en menos de 1 minuto, y notificar a los usuarios subscritos a este evento.
- Se puede implementar un plan de ejecución que detecte en tiempo real cuando un vuelo viene retrasado y notifique a los usuarios subscritos a este evento, bien sea vía correo electrónico, sms, o a través de una aplicación web.
- Se puede implementar un plan de ejecución que detecte múltiples intentos de autenticación erróneos usando una misma cuenta, por ejemplo desde diferentes IP, o con un criterio de más de 5 intentos en un segundo, y notificar al dueño de la cuenta.

Con este criterio cualquier evento que se quiera detectar se puede implementar usando el CEP de WSO2, y lo más interesante es que será en tiempo real.

En el caso de estudio, se utilizó **WSO2 CEP** para el procesamiento de eventos complejos en tiempo real. Esto es debido a las capacidades de esta aplicación, tales como tener capacidad de escalamiento y alta disponibilidad, proveer un lenguaje de consultas potente y extensible para el procesamiento del streaming de eventos temporal, y hacer uso de "*execution plans*" que se ejecutan cada pocos segundos, y se encargan de cargar la información proveniente de los eventos, en "*event tables*" temporales los cuales se pueden persistir en una BD relacional (en este caso MYSQL). El hecho de que todo este procesamiento de información pueda hacerse en tiempo real es de vital importancia para el uso de esta aplicación y para el desarrollo de los indicadores dinámicos del tablero de control. Además como se mencionó anteriormente el funcionamiento y la performance de esta herramienta es excelente, como se puede observar en la Figura 5.7.

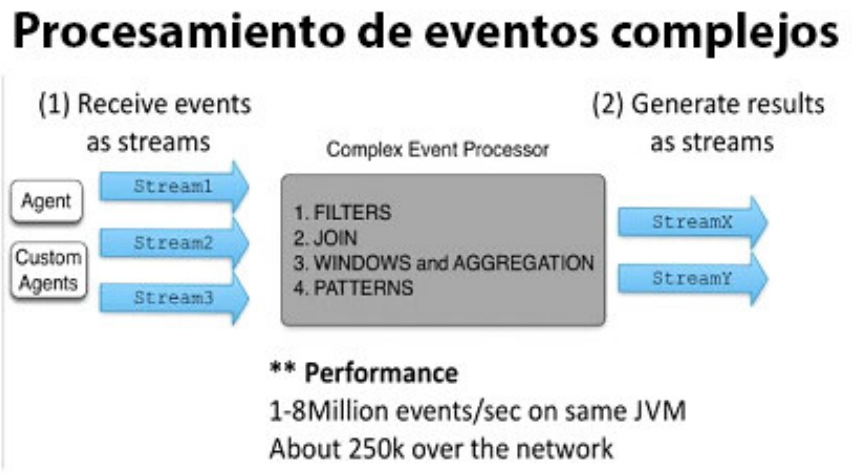


Figura 5.7. *Funcionamiento de WSO2 CEP*

5.1.4. Liferay:

Liferay Portal es una plataforma web que permite construir soluciones empresariales de manera rápida y sencilla. Esta cuenta con las siguientes características [33]:

- Facilita el diseño de interfaces de usuario
- Framework de integración de aplicaciones
- Herramientas incluidas en el producto
- Soporte de Single Sign On (SSO)
- Soporte de campos personalizados
- Integración de motores de reglas
- Grupos de usuarios, Organizaciones y Sitios
- Plataforma SOA
- Personalización de usuarios
- Publicación de contenidos basada en roles
- Administración mediante Arrastrar y Soltar
- Auditoría y monitorización de rendimiento
- Búsqueda y Tagging
- Soporte Multi-Idioma

Para facilitar el desarrollo de aplicaciones, Liferay proporciona una herramienta o IDE.
Liferay IDE es una extensión para Eclipse IDE que apoya el desarrollo de proyectos de plugins para la plataforma Liferay Portal. Es posible instalar Liferay IDE como un conjunto de plugins de Eclipse desde un sitio de actualización. La última versión de Liferay IDE soporta el desarrollo de *portlets, hooks, layout templates, themes y Ext plugins*. Para usar Liferay IDE, se requiere el paquete para desarrolladores de Eclipse Java EE utilizando Indigo o una versión posterior.
Con Liferay IDE se cuenta con todo las herramientas necesarias para desarrollar su portal.

Portlet en Liferay: para la creación de portlets en Liferay, además de los archivos de configuración genéricos de toda aplicación de portlet como son *web.xml* y *portlet.xml*, es necesario contar con otros archivos como:

- *liferay-display.xml*: en este archivo se especifica la categoría de un portlet; esta será visualizada en el Portal de Liferay a la hora de administrar los mismos.
- *liferay-portlet.xml*: describe mejoras específicas de Liferay para los portlets instalados en un servidor Liferay portal.

Si se utiliza Liferay IDE estos archivos son creados automáticamente.
Para desarrollar los portlets se puede usar el framework de portlets provisto por Liferay, llamado *MVCPortlet*, como cualquier otro framework, que puede ser *JSF, Struts*, u otro producto. También se puede usar directamente la definición de Portlet de Java, sin la necesidad de utilizar un framework.

Para la creación de Portlets, se ha probado el uso de *MVCPortlet*, *JSF* y *GenericPortlet* como estructura.

- *MVCPortlet:* por defecto Liferay IDE usa MVCPortlet, el cual es un framework que esconde parte de la complejidad de los portlets y facilita las operaciones más comunes. Por defecto

usa páginas JSP separadas para cada modo del portlet; cada modo de portlet registrado tiene una correspondiente JSP con el mismo nombre del modo (*view.jsp, edit.jsp, help.jsp*).
En la definición del portlet en el archivo portlet.xml se especifica la clase MVCPortlet; si uno quiere hacer alguna otra tarea como definir acciones, es necesario heredar de la misma y cambiar la clase en portlet.xml.

- *JSF (Java Server Faces):* facilita el manejo de la vista de un portlet. Un portlet que use JSF debe definir en *portlet.xml* la clase *GenericFacesPortlet*. Con JSF se definen en los parámetros de inicio del portlet la vista para los modos registrados del portlet, entonces cada modo tendrá asociado un *xhtml*. Si un portlet debe responder a una acción es necesario crear un *Bean* con un método que luego será llamado por la vista correspondiente.

Liferay Themes and Layout Templates (temas y plantillas): si se desea cambiar la apariencia de Liferay o la disposición de los portlets en el portal, esto se puede realizar en Liferay por medio de la creación de *Themes* y *Layout Templates*.

- Themes (temas): permiten modificar completamente la interfaz de usuario. Con estos plugins se puede definir temas, los cuales se especifican en el archivo *liferay-look-and-feel.xml*. En este se definen los temas deseados y las propiedades que pueden ser editadas luego desde la administración del portal.
  Las modificaciones se aplican en los archivos css, las imágenes, los javascripts y los html. Para que los cambios en estos archivos se tomen, es necesario guardarlos con los mismos nombres en una carpeta específica ("*_diffs*"). Luego, una vez que el proyecto es subido al servidor, a los temas creados se le asignan todos los archivos que trae por defecto Liferay y se sobrescriben los correspondientes archivos según lo que se haya definido en la carpeta *_diffs*.
- Layout Templates (plantillas): en caso de que no convenzan las maneras de embeber los portlets que provee Liferay, es posible crear un *Liferay Template Layout* el cual va a permitir crear una nueva disposición de los portlets.
  Para hacer esto se crea un *Layout Template Project*. Los layouts que se crean, se deben especificar en *liferay-layout-templates.xml*, indicando el HTML por defecto, el HTML para dispositivos móviles y la imagen de presentación.
  Para realizar la nueva disposición de los portlets es necesario completar el archivo *.tpl* el cual contiene el código HTML correspondiente. En este, se especifica cómo se desea desplegar los portlets, por ejemplo, 1 en la primera fila, 4 en la segunda y 1 en la tercera.

Hooks y Ext-plugins:

- Hooks: son el mejor plugin para la personalización de las características fundamentales de Liferay. Estos deberían usarse siempre que se pueda para modificar la funcionalidad principal de Liferay. También es posible usar Ext plugins para muchas de las mismas tareas, pero los hooks son más convenientes debido a que son *hot-deployable* (pueden implantarse automáticamente) y tienen una mayor compatibilidad con versiones.
  Algunas de las cosas que se pueden hacer con hooks son:
  - Sobreescribir recursos web.
  - Customizar páginas JSP extendiendo la original.

    - Realizar una acción personalizada, entre otras
- Ext plugins: son herramientas poderosas para extender Liferay. Debido a que aumentan la complejidad de la instancia de Liferay, sólo se debe usar un plugin ext si se está seguro de que no se puede lograr el objetivo usando una herramienta diferente.
  Los plugins Ext permiten el uso de las API internas o incluso sobrescribir archivos del núcleo Liferay. Cuando se actualiza a una nueva versión de Liferay se debe revisar todos los cambios y modificar manualmente el Ext Plugin para fusionar los cambios con Liferay. Además, los plugins no son *hot-deployable* y para implementarlos se debe reiniciar el servidor.

5.2. Integrando las tecnologías

Las aplicaciones descriptas fueron elegidas de forma premeditada debido a su posible convivencia o integración en un ambiente cooperativo que resulte eficiente.
De esta manera las aplicaciones WSO2 BAM y WSO2 CEP tienen la capacidad de recibir los eventos disparados por el gestor de procesos de negocio Bonita OS. Esto se logra mediante la construcción de varios conectores del BPMS los cuales permiten extraer la información de los procesos que se van ejecutando mediante la API Rest de Bonita OS. Dichos conectores mediante la importación de clases JAVA provistas por los productos WSO2, pueden enviarle a estas aplicaciones la información de los procesos en forma de eventos.
Como se mencionó previamente ambas aplicaciones van a tener un uso diferente debido a sus potenciales características. WSO2 BAM se va a utilizar para recibir una gran cantidad de información cada cierto período prolongado de tiempo, por lo cual se usará para un monitoreo histórico de datos. WSO2 CEP por el contrario se encargará de recibir la información inmediata de los procesos en ejecución por lo cual procesará eventos en tiempo real. Para que convivan ambas tecnologías es necesario plasmar una arquitectura que permita tener capas de procesamiento de datos tanto para información histórica como para información en tiempo real. Esta arquitectura se llama Arquitectura Lambda y se explicará más en detalle en la siguiente sección.
Una vez los datos puedan ser procesados por ambas capas, estarán disponibles para su visualización. Los productos WSO2 tienen la capacidad de persistir los datos relevantes en una base de datos relacional, en este caso MYSQL, por lo que la aplicación de visualización puede acceder de forma sencilla a los datos para mostrar. De esta forma los portlets que conforman el tablero de control en el portal Liferay acceden a estos datos para mostrárselos al usuario cuando este los requiera.

5.2.1. Arquitectura Lambda

La Arquitectura Lambda [Figura 5.8] une el procesamiento de datos *por lotes* y el procesamiento de datos en *tiempo real* en un mismo modelo.

De esta manera, la información entrante es enviada a ambas capas, donde la "**batch layer**" pre-calcula una visión histórica del sistema, mientras que la "**speed layer**" calcula la visión más reciente del sistema.
La "**serving layer**" combina ambas capas para satisfacer las consultas dadas.

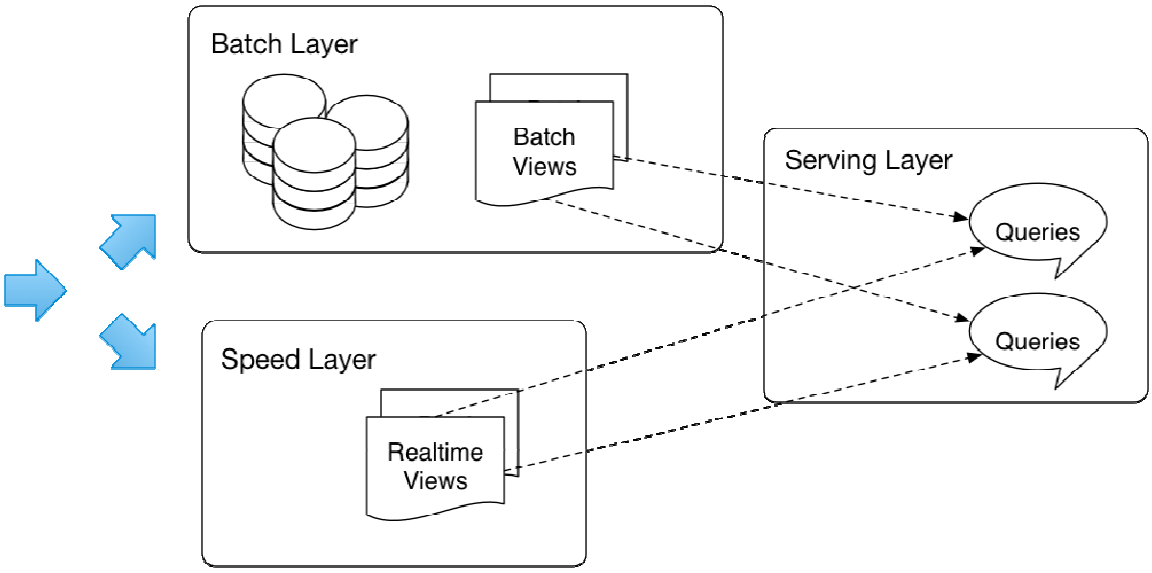


Figura 5.8. *Arquitectura Lambda*

Puede usarse **WSO2BAM** para implementar la "batch layer" y **WSO2CEP** para implementar la "speed layer" como se observa en la Figura 5.9. Ambos utilizan un transporte de datos de alto rendimiento denominado "data bridge", que puede alcanzar hasta 300.000 eventos/segundo.

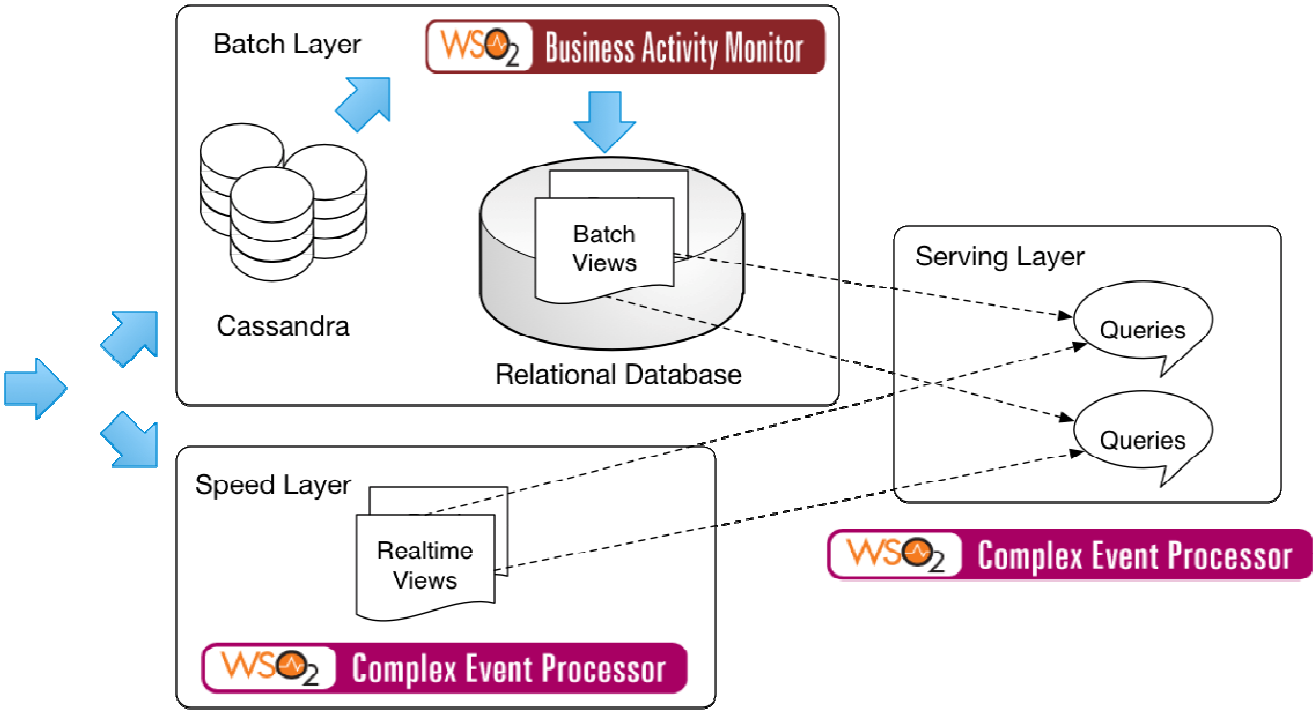


Figura 5.9. *Arquitectura Lambda simulada con los productos WSO2*

5.3. Arquitectura

A continuación se observa la arquitectura correspondiente al Tablero de Control desarrollado [Figura 5.10], que constituye el caso de estudio de este trabajo.

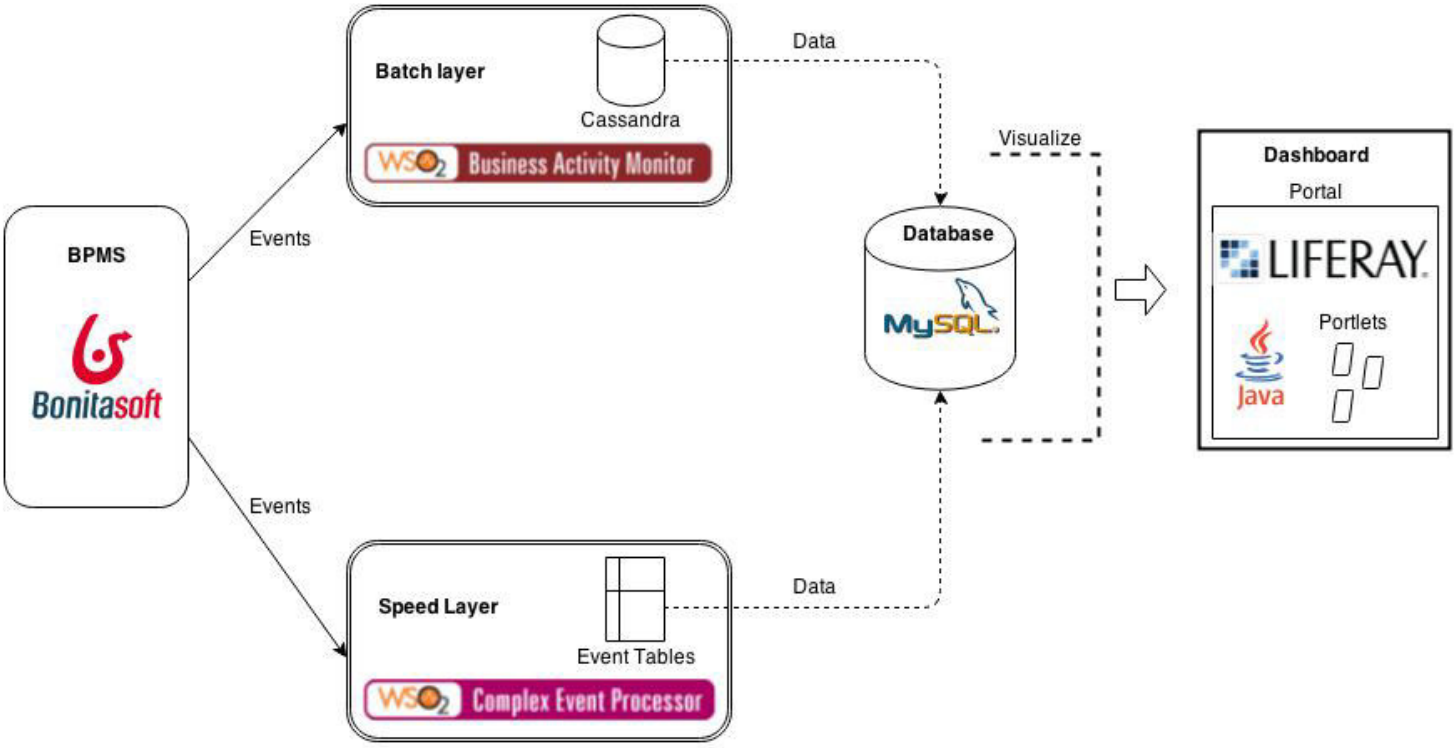


Figura 5.10. *Arquitectura del Tablero de Control*

Se observa cómo las aplicaciones de la capa histórica (WSO2 BAM), y las aplicaciones de la capa de tiempo real (WSO2 CEP) reciben la información de los procesos de Bonita OS en forma de eventos. Ambas aplicaciones de procesamiento de datos hacen su trabajo y persisten la información relevante en la base de datos relacional MYSQL. Luego la capa de visualización puede acceder a dichos datos, de modo que los portlets hacen uso de esta característica. De esta manera el Tablero de Control toma vida cuando los usuarios despliegan los portlets que deseen, cada uno con una funcionalidad para monitoreo específica, y estas aplicaciones modulares de Java visualicen los datos que se van procesando tanto en los motores BAM como CEP.
La lógica del negocio, es decir el tratamiento de los KPIs, y la visualización de las alertas, está desarrollada en los portlets. Por consiguiente, cuando determinados datos estén fuera de cierto rango o umbral preestablecido, los portlets entenderán esta situación y accionarán al respecto para que el usuario pueda reaccionar rápidamente ante esa situación indeseada.

5.4. Caso de estudio

En esta sección se expondrá el caso de estudio sobre el desarrollo del Tablero de Control que ejerce el monitoreo de los procesos de negocio. Se explicará los objetivos de la aplicación y se mostrará un ejemplo de la aplicación funcionando.

5.4.1. Propuesta

Idea:

La propuesta se basa en la elaboración de un **tablero de control** que realice el *monitoreo* de la ejecución de los procesos de una organización. Es decir, que el tablero realizará el seguimiento de

los procesos asignados al tablero (por medio de un conector), y mostrará la información pertinente durante su ejecución en tiempo real, con el fin de que el usuario pueda tomar una decisión rápida ante alguna situación de alerta.
La idea es desarrollar una serie de **conectores** para Bonita BPM, que permitan asignárselos a un proceso y que se conecten con la aplicación BAM (en este caso **WSO2 BAM y WSO2CEP**), enviándole la información proveniente de dicho proceso.
El tablero se encargará de visualizar la información que resulte útil procedente de los procesos y de ciertos indicadores (**KPIs**) que sirvan para controlar el rendimiento de los procesos en ejecución.

Portada:

La presentación visual del tablero se realiza mediante el uso de componentes denominados **portlets**, los cuales tendrán cada uno una función específica y se los podrá agregar y desplegar en el portal de **Liferay**.

Figura 5.11. *Portada de Liferay mostrando el despliegue de portlets*

Indicadores estáticos:

Por un lado se tiene información estática, resultante del registro de datos en un lapso de tiempo considerable. De modo que se puede tener una serie de portlets, cada uno mostrando una estadística *histórica* general determinada.
La información histórica se almacena a partir de un *streaming* de datos, los cuales son recibidos por la capa de procesamiento por lotes mediante la aplicación WSO2 BAM, siendo lanzados en forma de *eventos*.

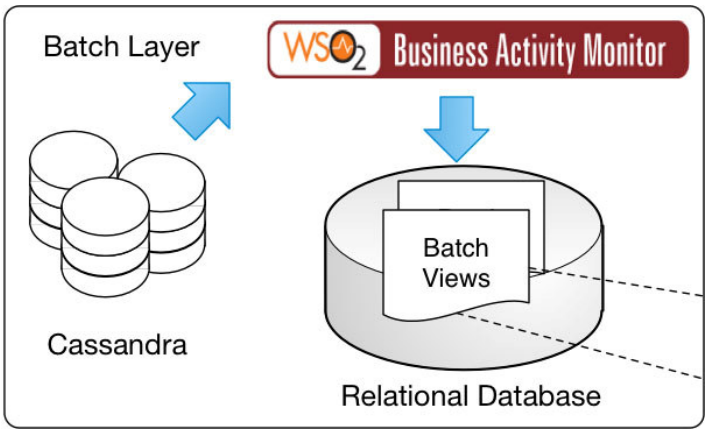
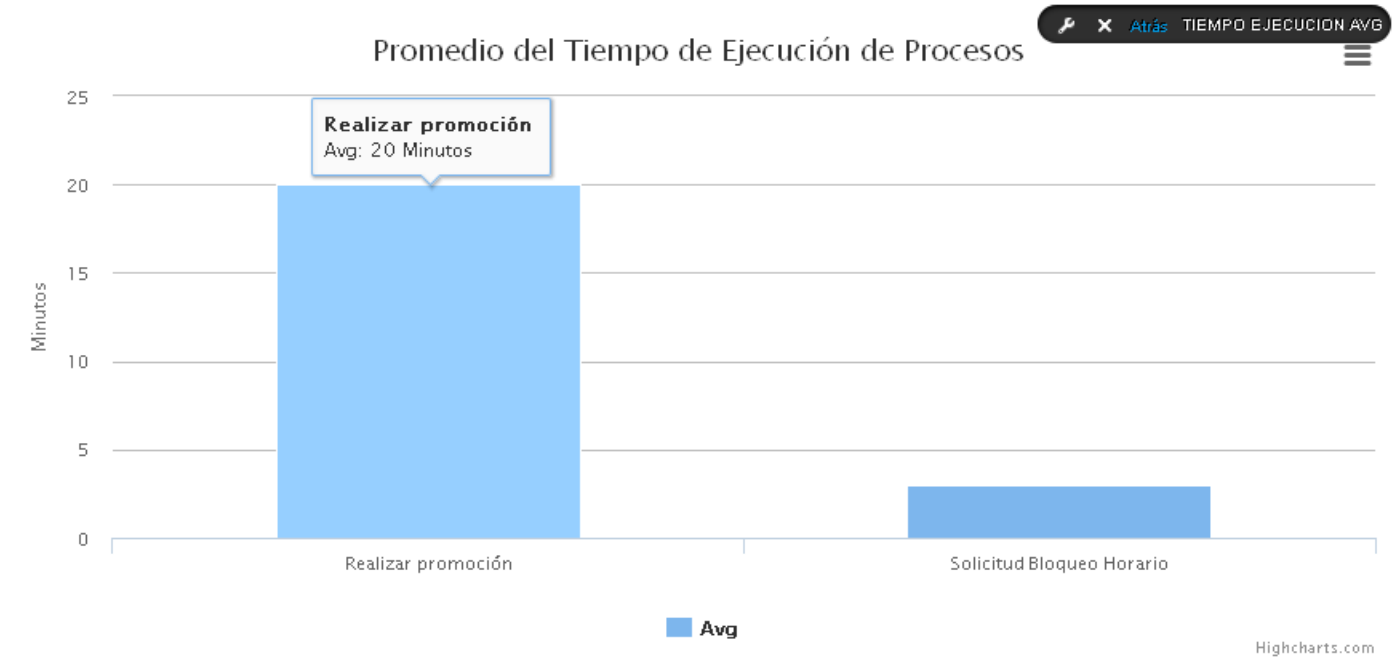


Figura 5.12. *Capa WSO2 BAM para indicadores estáticos*

El portlet desarrollado como indicador estático para ejercer este tipo de monitoreo es:

- **Promedio del tiempo de ejecución:** calcula el promedio de la duración de todas las instancias de un proceso que han finalizado.



	Avg
Realizar promoción	20
Solicitud Bloqueo Horario	3

Figura 5.13. *Portlet que muestra el promedio del tiempo de ejecución de un proceso determinado*

Indicadores dinámicos:

Por otro lado la información más interesante, y que va a proveer el control de los procesos en *tiempo real*, es la información dinámica de lo que está ocurriendo en el momento y que se asocia con los **Indicadores claves de rendimiento** (**KPIs**).

La información en tiempo real se almacena a partir de un *streaming* de datos, los cuales son recibidos por la **capa de procesamiento veloz** [Figura 5.14] y mediante la aplicación **WSO2 CEP**, siendo lanzados como *eventos*.

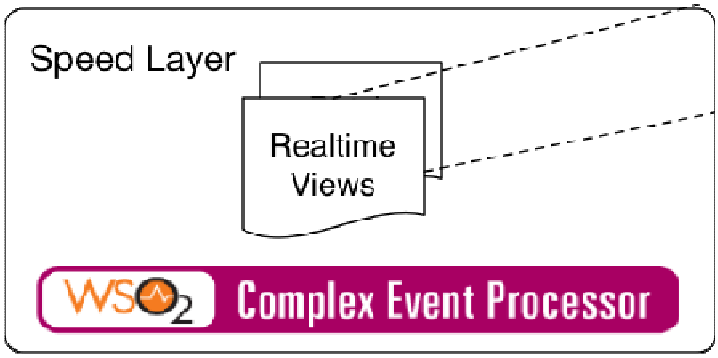


Figura 5.14. *Capa WSO2 CEP para indicadores dinámicos*

Los portlets desarrollados como indicadores dinámicos para ejercer este tipo de monitoreo son:

- **Tiempo de ejecución de los procesos activos:** se muestra el tiempo transcurrido y el tiempo restante derivado de un indicador límite (*deadline*), el cual si se sobrepasa se informa un alerta.

Figura 5.15. *Portlet que muestra el tiempo transcurrido y el tiempo restante de un proceso en tiempo real*

- **Control de tareas fallidas** [Figura 5.16]: se muestra en tiempo real, las tareas que van fallando y su información respectiva (proceso, caso, id, tipo, entre otros atributos). Sirve a su vez para determinar si un proceso está activo o no, dado que una falla en una tarea cambia el estado del proceso a consideración del tablero.

Figura 5.16. *Portlet que muestra en tiempo real las tareas que van fallando y su información*

5.4.2. Ejemplo de uso

En esta sección se describirá un ejemplo práctico de la aplicación funcionando. Se explicará en detalle cada uno de los componentes que forman parte del Tablero de Control tanto en cuestiones de funcionamiento como de implementación. Cabe recalcar que este es un ejemplo ilustrativo, pues para realizar una prueba estrictamente con BigData se requiere una arquitectura adecuada como por ejemplo bases de cómputo para recibir grandes volúmenes de eventos e información. Ese escenario es en el que realmente se vuelven muy eficientes las aplicaciones utilizadas. Sin embargo este ejemplo sirve como para tener una certera noción de cómo se pueden utilizar estas aplicaciones y como cooperan en conjunto para los propósitos de este trabajo.

Tiempo de ejecución de los procesos activos

A continuación se explicará el indicador dinámico que calcula el tiempo transcurrido y el tiempo restante derivado de un indicador límite (deadline), el cual si se sobrepasa se informa un alerta. Para esto primero se explicará cómo se da la comunicación entre Bonita OS y la aplicación CEP para registrar y persistir en tiempo real los datos del proceso cuando se inicia. Luego se explicará cómo se utiliza Liferay por medio de portlets para tomar la información, procesarla y visualizarla en el portal.

Bonita OS

Partiendo del proceso llamado “Realizar Promoción” diseñado en Bonita OS [Figura 5.17], el mismo tiene un conector llamado “registroProceso” el cual se ejecuta cuando inicia el proceso y se le envía como parámetro el identificador de la instancia del proceso.

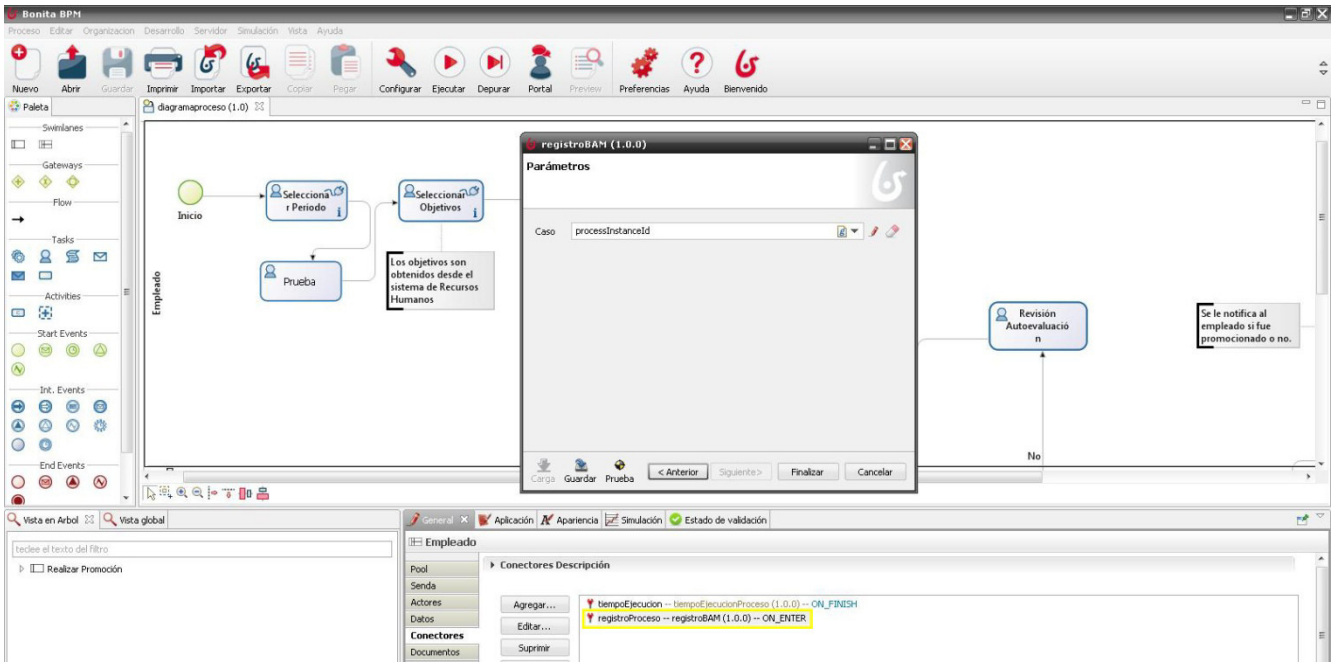


Figura 5.17. *Proceso "Realizar Promoción" con el conector "registroProceso"*

El conector "registroProceso" en su implementación debe importar las librerías de WSO2 CEP [Figura 5.18] para poder utilizar los métodos y conectarse con dicha aplicación.

```
4  package org.wso2bam.connectorInit;
5
6  import java.io.IOException;
7  import java.net.MalformedURLException;
8  import java.text.SimpleDateFormat;
9  import java.util.Calendar;
10 import java.util.Date;
11 import java.util.Map;
12
13 import org.bonitasoft.engine.connector.ConnectorException;
14 import org.json.simple.JSONObject;
15 import org.json.simple.parser.JSONParser;
16 import org.wso2.carbon.databridge.agent.thrift.exception.AgentException;
17 import org.wso2.carbon.databridge.commons.exception.AuthenticationException;
18 import org.wso2.carbon.databridge.commons.exception.DifferentStreamDefinitionAlreadyDefinedException;
19 import org.wso2.carbon.databridge.commons.exception.MalformedStreamDefinitionException;
20 import org.wso2.carbon.databridge.commons.exception.StreamDefinitionException;
21 import org.wso2.carbon.databridge.commons.exception.TransportException;
22 import org.wso2.carbon.databridge.commons.exception.UndefinedEventTypeException;
23 import org.wso2bam.connector.BonitaProxy;
```

Figura 5.18. *Librerías WSO2 CEP en la implementación del conector* "registroProceso"

La implementación del conector "registroProceso" [Figura 5.19] se basa en traer los datos requeridos de la instancia del proceso mediante el uso de la API Rest de Bonita OS, y setearlos en las variables de instancia del conector; esto lo realiza el método "conectarApi()" y se utiliza el ID de la instancia del proceso que es recibido como parámetro por el conector.
Luego se envían dichos datos a la aplicación WSO2 CEP mediante el método "testSendingEvent()" de la clase "EnviarDatosBAM".

```
@Override
protected void executeBusinessLogic() throws ConnectorException{
    //conecto a la api rest y seteo en las variables de proceso los datos del caso traidos desde la api
    try {
        this.conectarApi();
    } catch (IOException e) {
        // TODO Bloque catch generado automáticamente
        e.printStackTrace();
    }

    EnviarDatosBAM bam = new EnviarDatosBAM();

    //calculo la fecha actual
    String DATE_FORMAT_NOW = "dd-MM-yyyy HH:mm:ss";
    Calendar cal = Calendar.getInstance();
    SimpleDateFormat sdf = new SimpleDateFormat(DATE_FORMAT_NOW);
    String fechaini = sdf.format(cal.getTime());

    //envío el evento
    try {
        bam.testSendingEvent(nombreprocesoapi, iddelproceso, processId, descripcion, fechaini, usuario, version);
    } catch (MalformedURLException e) {
        // TODO Bloque catch generado automáticamente
        e.printStackTrace();
    } catch (AuthenticationException e) {
        // TODO Bloque catch generado automáticamente
        e.printStackTrace();
```

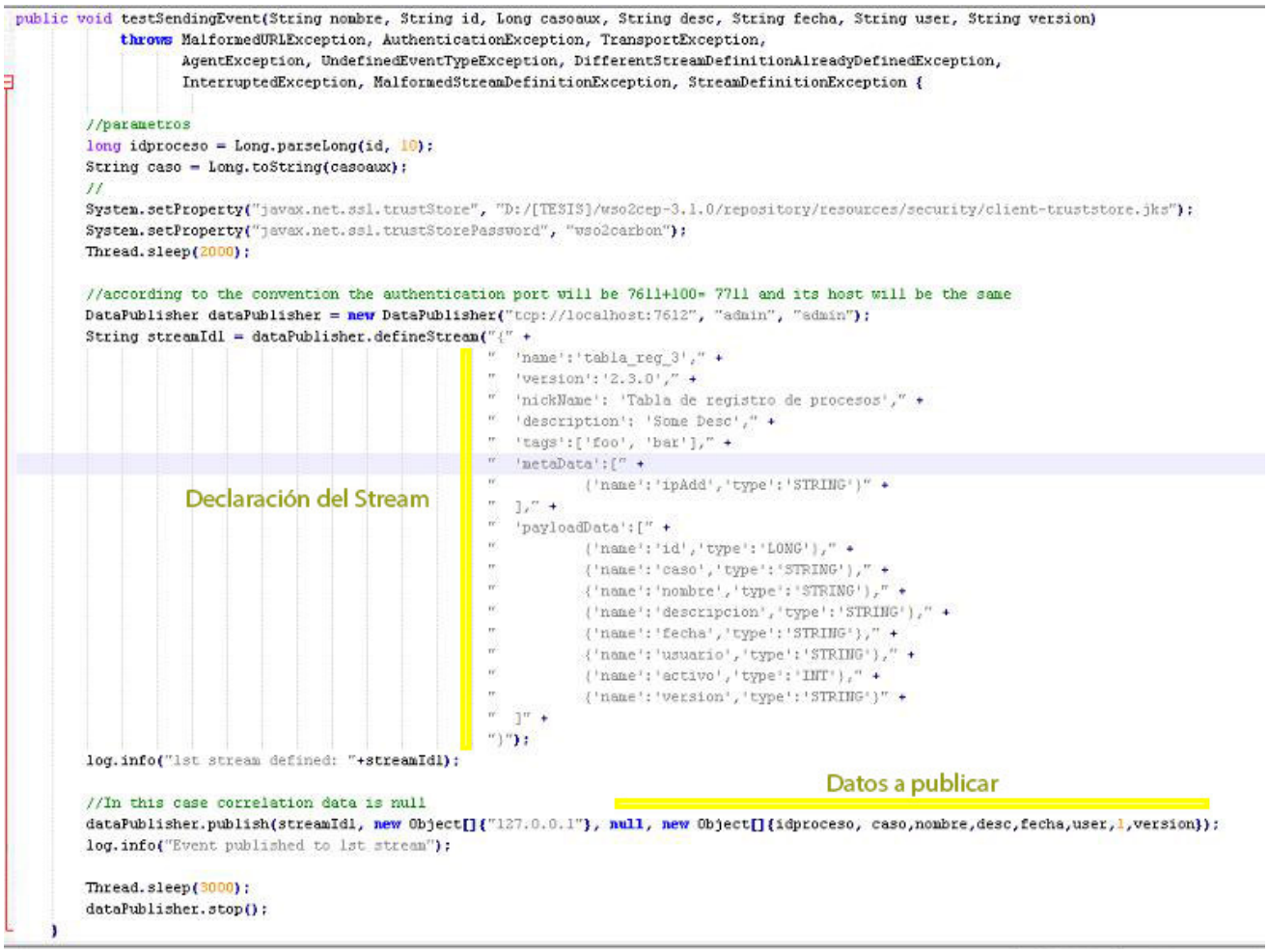
Figura 5.19. *Implementación del conector "registroProceso"*

El método "conectarApi() hace uso de la clase "BonitaProxy" la cual mediante el método "enviarPeticion()" se encarga de autentificarse en Bonita con los datos del usuario que iniciará el caso, y luego se encarga de hacer una llamada a la API Rest de Bonita, recibir el InputStream resultante, transformarlo a string y devolverlo. Después, como se puede observar en la Figura 5.20, se parsea el string resultante a un archivo json y se lo recorre según los datos que se necesite. En este caso, la sentencia "json.get('start')" retorna el tiempo en que se inició el caso; así sucesivamente se obtienen los datos restantes requeridos.

```
    private void conectarApi() throws IOException{
//      public Long idUsuario = (long)1;
        String idUsuario = "0";

        Long caso1 = (long)0;
        if (processId != null)
            caso1 = (long) processId;

        BonitaProxy proxy= new BonitaProxy();

        //Pido el tiempo en que inicio el caso
        String url= "http://localhost:8083/bonita/API/bpm/case/"+caso1;
        String metodo= "GET";
        String resultado= proxy.enviarPeticion(url, metodo, null, null);
            //Recorremos el archivo JSON
        try {
            //Creo un objeto JSON a traves del resultado string
            JSONParser par= new JSONParser();
            JSONObject json= (JSONObject) par.parse(resultado);
            //Desde aca consigo cada etiqueta por separado
            String res= (String) json.get("start");
            fechainicioapi= res;
        }
        catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

Figura 5.20. *Implementación del método "conectarApi()"*

El método "testSendingEvent()" [Figura 5.21] declara un stream con los datos a publicar en el motor CEP que corresponden a la instancia del proceso que se está ejecutando, y luego se envían dichos datos a la aplicación WSO2 CEP. El nombre del stream enviado es "tabla_reg_3", y la versión es "2.3.0".
Nótese que el campo "activo" del proceso en cuestión se lo setea en 1.




```
public void testSendingEvent(String nombre, String id, Long casoaux, String desc, String fecha, String user, String version)
        throws MalformedURLException, AuthenticationException, TransportException,
                AgentException, UndefinedEventTypeException, DifferentStreamDefinitionAlreadyDefinedException,
                InterruptedException, MalformedStreamDefinitionException, StreamDefinitionException {

    //parametros
    long idproceso = Long.parseLong(id, 10);
    String caso = Long.toString(casoaux);
    //
    System.setProperty("javax.net.ssl.trustStore", "D:/[TESIS]/wso2cep-3.1.0/repository/resources/security/client-truststore.jks");
    System.setProperty("javax.net.ssl.trustStorePassword", "wso2carbon");
    Thread.sleep(2000);

    //according to the convention the authentication port will be 7611+100= 7711 and its host will be the same
    DataPublisher dataPublisher = new DataPublisher("tcp://localhost:7612", "admin", "admin");
    String streamId1 = dataPublisher.defineStream("{" +
                                                  "  'name':'tabla_reg_3'," +
                                                  "  'version':'2.3.0'," +
                                                  "  'nickName': 'Tabla de registro de procesos'," +
                                                  "  'description': 'Some Desc'," +
                                                  "  'tags':['foo', 'bar']," +
                                                  "  'metaData':[" +
                                                  "          {'name':'ipAdd','type':'STRING'}" +
                                                  "  ]," +
                                                  "  'payloadData':[" +
                                                  "          {'name':'id','type':'LONG'}," +
                                                  "          {'name':'caso','type':'STRING'}," +
                                                  "          {'name':'nombre','type':'STRING'}," +
                                                  "          {'name':'descripcion','type':'STRING'}," +
                                                  "          {'name':'fecha','type':'STRING'}," +
                                                  "          {'name':'usuario','type':'STRING'}," +
                                                  "          {'name':'activo','type':'INT'}," +
                                                  "          {'name':'version','type':'STRING'}" +
                                                  "  ]" +
                                                  "}");
    log.info("1st stream defined: "+streamId1);

    //In this case correlation data is null
    dataPublisher.publish(streamId1, new Object[]{"127.0.0.1"}, null, new Object[]{idproceso, caso,nombre,desc,fecha,user,1,version});
    log.info("Event published to 1st stream");

    Thread.sleep(3000);
    dataPublisher.stop();
}
```

Figura 5.21. *Implementación del método "testSendingEvent()"*

WSO2 CEP

En la presente sección se procede a ver cómo la aplicación WSO2 CEP captura los streams de eventos y los procesa.
Primero se levanta el servidor CEP en su URL correspondiente (https://192.168.0.100:9448/carbon/) como lo refleja la Figura 5.22.

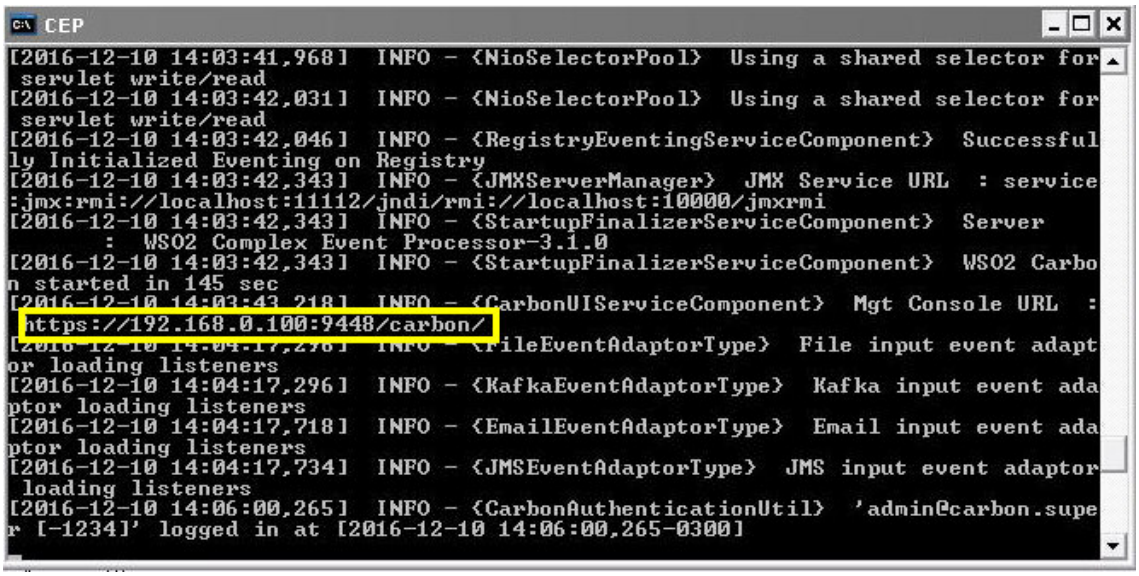


Figura 5.22. *Consola de WSO2 CEP iniciándose*

Luego de ingresar a la portada de CEP en la dirección anterior, se genera un Event Stream [Figura 5.23] para recibir el evento de Bonita OS, que se debe mapear con el mismo nombre que el disparador del evento en este caso “tabla_reg_3” y además la versión que es la “2.3.0”.
A su vez se generan los mapeos de las variables recibidas por el evento, como para luego enviarlas como campos de una consulta a una base de datos relacional MYSQL.

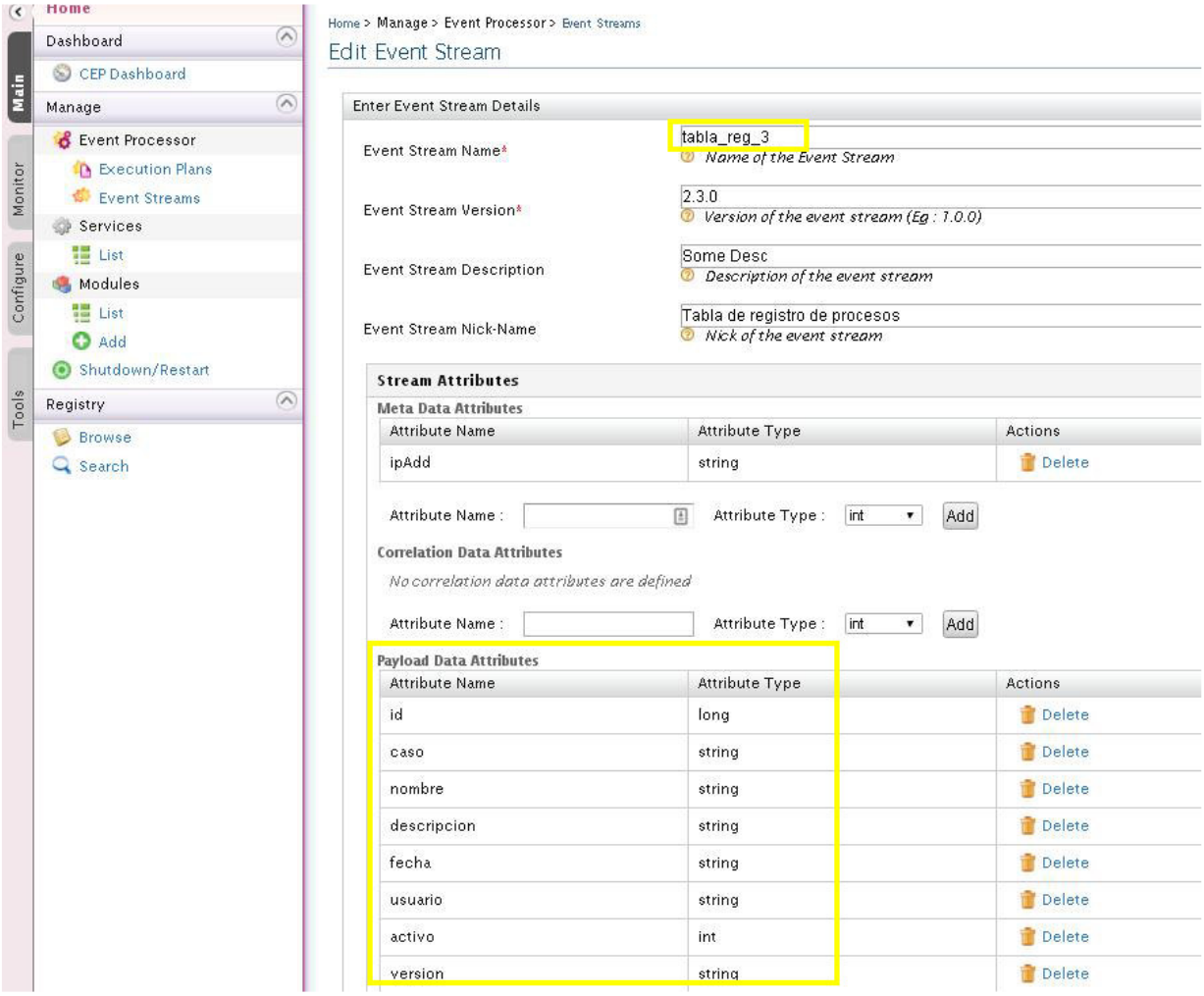


Figura 5.23. *Panel de control CEP, vista de los Event Streams*

Además, se debe crear un nuevo Data Source con los parámetros de la base de datos MYSQL que se va a utilizar [Figura 5.24].

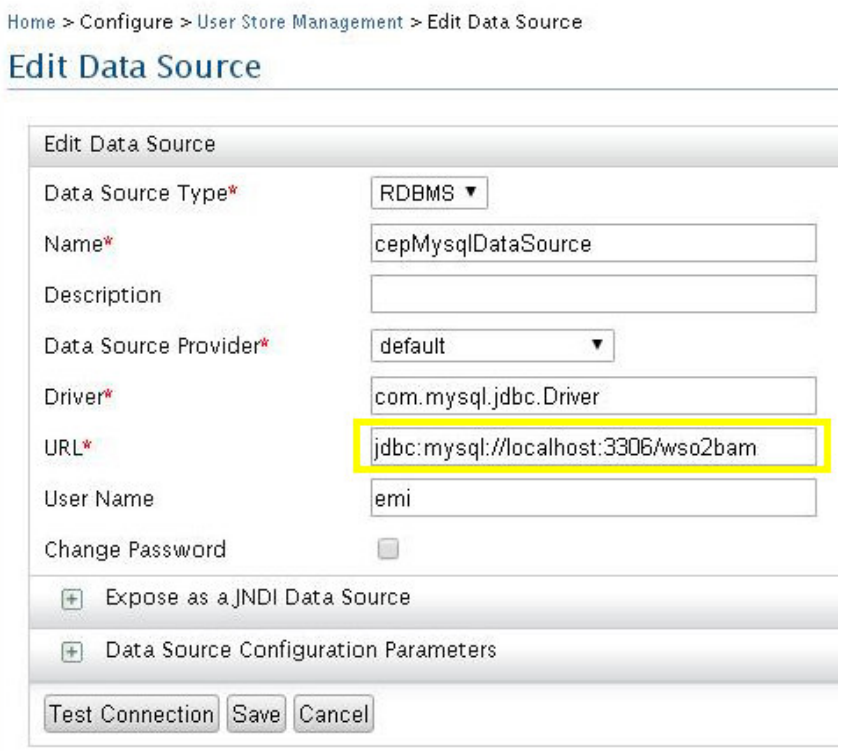


Figura 5.24. *Panel de control CEP, vista de los Data Source*

Luego se crea un *Execution Plan* que tienen la capacidad de ejecutarse en intervalos muy pequeños, en cada segundo prácticamente y de aquí el beneficio de utilizar WSO2 CEP para el procesamiento en tiempo real.
Como se observa en la Figura 5.25, se recibe el evento capturado por el *Event Stream*, y después se realiza el mapeo de los datos de ese *stream* primero a una tabla temporal y luego a la base de datos MYSQL utilizando dicha tabla. Para la persistencia en la base de datos relacional se hace uso del Data Source creado anteriormente.

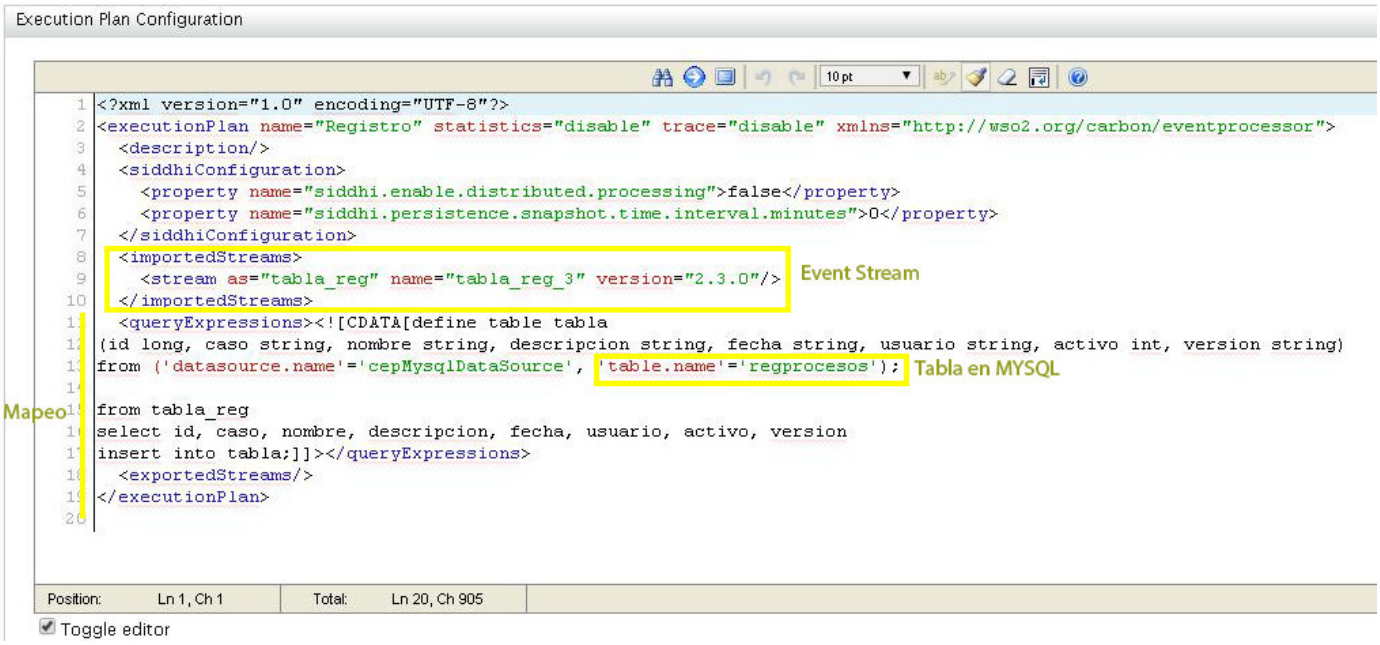
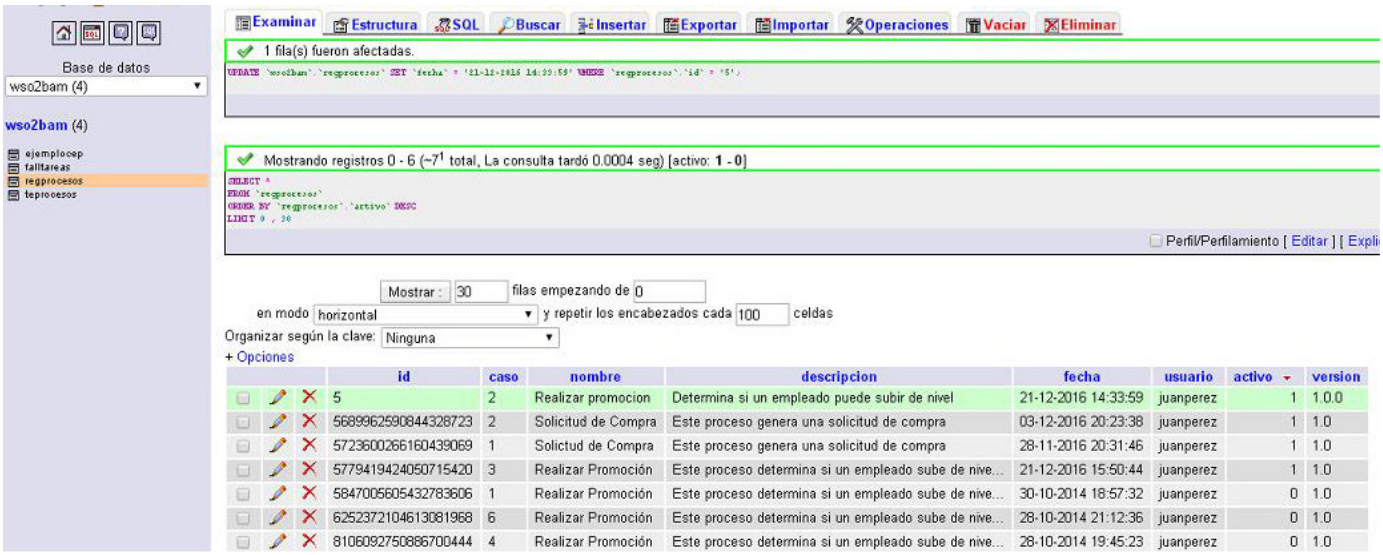


Figura 5.25. *Configuración del Execution Plan*

Una vez interceptado el evento disparado por el conector de Bonita mediante el *event stream* declarado anteriormente, el *execution plan* se realizará de forma continua hasta que perciba dicho *stream* y acto seguido realizará el mapeo de los datos y la consecuente persistencia de los mismos en la base de datos MYSQL.
Como se observa en la Figura 5.26, el procesamiento de CEP tuvo efecto al persistir en la tabla "reg_procesos" los datos del proceso que acabó de iniciarse.



id	caso	nombre	descripcion	fecha	usuario	activo	version
5	2	Realizar promocion	Determina si un empleado puede subir de nivel	21-12-2016 14:33:59	juanperez	1	1.0.0
5689962590844328723	2	Solicitud de Compra	Este proceso genera una solicitud de compra	03-12-2016 20:23:38	juanperez	1	1.0
5723600266160439069	1	Solictud de Compra	Este proceso genera una solicitud de compra	28-11-2016 20:31:46	juanperez	1	1.0
5779419424050715420	3	Realizar Promoción	Este proceso determina si un empleado sube de nive...	21-12-2016 15:50:44	juanperez	1	1.0
5847005605432783606	1	Realizar Promoción	Este proceso determina si un empleado sube de nive...	30-10-2014 18:57:32	juanperez	0	1.0
6252372104613081968	6	Realizar Promoción	Este proceso determina si un empleado sube de nive...	28-10-2014 21:12:36	juanperez	0	1.0
8106092750886700444	4	Realizar Promoción	Este proceso determina si un empleado sube de nive...	28-10-2014 19:45:23	juanperez	0	1.0

Figura 5.26. *Tabla "regprocesos" actualizada*

Con los datos cargados en la base de datos relacional, se explicará cómo Liferay se sirve de los mismos para mostrarlos en una interfaz en forma de tablero de control, amigable, y que los muestre en forma oportuna, en tiempo real.

Liferay

Por un lado se tiene un portlet llamado "Procesos Activos" [Figura 5.27] que muestra aquellas instancias de procesos que se iniciaron y se encuentran activas, es decir que no fallaron ni finalizaron. Recordar que cuando se inició un proceso, se disparó un evento que el CEP se encargó de registrar y como resultado se persistió un campo "activo = 1" para esa instancia.

Figura 5.27. *Portlet de "Proceos Activos"*

De este modo el portlet de procesos activos muestra una lista con el nombre de esos procesos y un link que al accederlo, dispara un evento de Liferay hacia el portlet "Control Procesos" [Figura 5.28] que contiene el identificador de ese proceso que se seleccionó. De esta manera este último portlet que gestiona el tiempo transcurrido y restante de un proceso en particular se activa para ese proceso seleccionado.

Figura 5.28. *Portlet "Control Proceso"*

El funcionamiento se da de esa manera: primero se selecciona un proceso y luego se lo supervisa. El portlet receptor, que hace el control, guarda el identificador del proceso elegido en su sesión, por lo que si se recarga el portal por el uso de los otros portlets siempre mostrará la información referente a la última selección. Cabe destacar que los eventos en Liferay son dinámicos al punto de que cuando un portlet los recibe, automáticamente se ejecuta el método "render" por lo que además de procesar la lógica del propio evento, luego se recarga y ejecuta también su método "render" del modo view. Por lo expuesto, cuando se selecciona un proceso en el portlet de selección automáticamente se muestra ese proceso en el portlet de control.

El portlet "Control Procesos" busca el identificador del proceso elegido en su sesión y mediante ese id busca en la tabla "regprocesos" todos los datos referentes a esa instancia, que se encargó de procesar el CEP. En particular, este portlet realiza varios cálculos para determinar el tiempo transcurrido y el tiempo restante de dicha instancia en base a un "deadline" o tiempo límite que se configura desde el mismo portlet en su modo "edit".

Para esto es necesario ingresar a las preferencias del portlet [Figura 5.30], en el icono de "Options/Preferences".

Figura 5.29. *Opciones del Portlet "Control Proceso"*

Figura 5.30. *Preferencias del Portlet "Control Proceso"*

El "Deadline" debe ser un número que indique la cantidad máxima de minutos que se espera que la instancia de proceso deba durar. Por ejemplo en el caso de la figura anterior, se espera que el proceso tarde en ejecutarse menos de una hora. En caso de que en el momento los números sean normales, se muestra el tiempo transcurrido del proceso y el tiempo restante. Si se da el caso que se sobrepasa ese límite, el portlet mostrará un alerta indicando tal situación y el tiempo en que se excedió el proceso.

En la Figura 5.31 se observa cómo se da la primera situación mencionada: habiéndose configurado un "deadline" de 60 minutos, y ya iniciado el caso 3 del proceso "Realizar Promoción" , el usuario desplegó en el portal de Liferay los portlets "Procesos activos" y "Controlar Proceso" seleccionando luego el proceso actual "Realizar Promoción". Automáticamente el portlet de control muestra la información referente a esa instancia, como se puede apreciar en la figura. Se observa que el proceso lleva activo 15 minutos desde que se lo inició y que faltan 45 minutos restantes hasta llegar al indicador límite de 60 minutos.

Figura 5.31. *Tiempo de ejecución del proceso activo normal*

Pasado el tiempo límite, el portlet de control mostrará un alerta indicando la situación como se puede ver en la Figura 5.32. Evidentemente la instancia todavía se encontraba activa, y se sobrepasó el deadline 31 minutos. El tiempo transcurrido total desde que inició el proceso es de 1 hora 31 minutos.

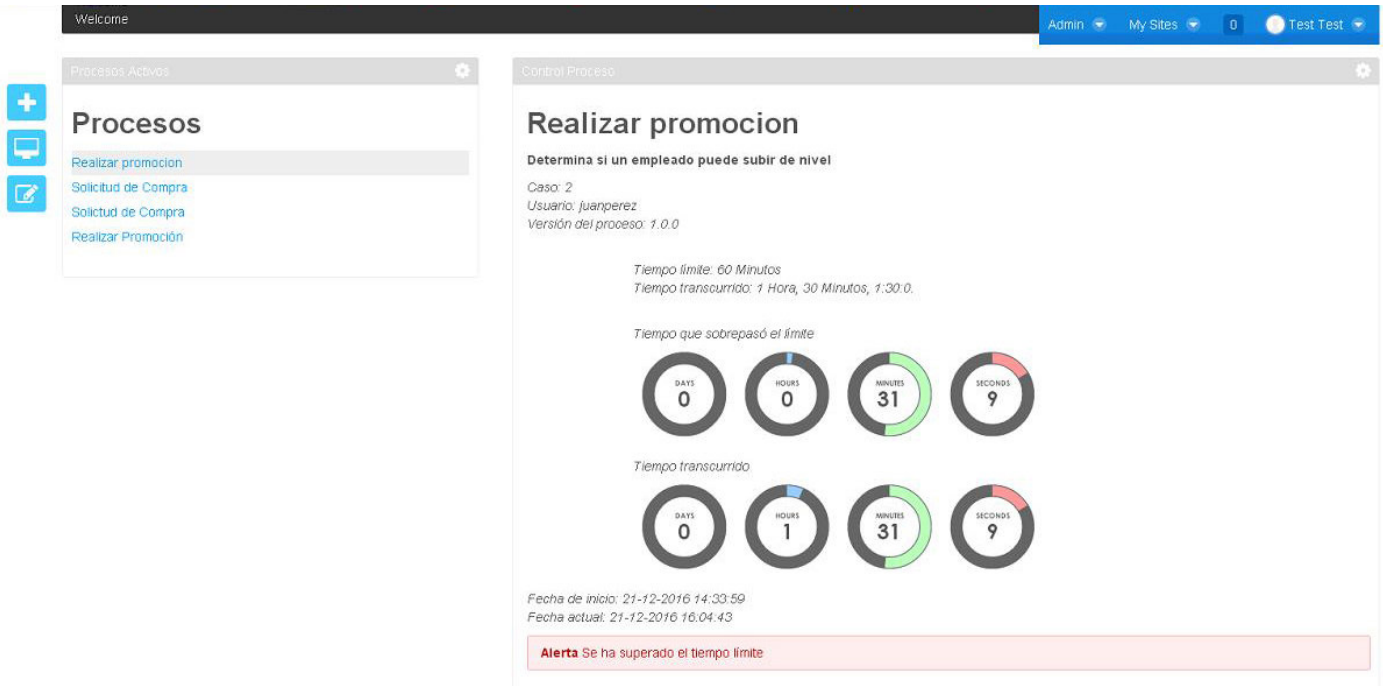


Figura 5.32. *Tiempo de ejecución del proceso activo sobrepasando el deadline*

Promedio del Tiempo de Ejecución:

A continuación se explicará el indicador estático que muestra el promedio del tiempo de ejecución de un proceso en particular a lo largo de su historia. Por lo tanto se va a explicar la interacción entre Bonita OS y la aplicación WSO2 BAM para capturar y almacenar la información de los tiempos de ejecución de los procesos. Luego se explicará la función de Liferay para visualizar la información.

Bonita OS

Como en el caso anterior, se utiliza el mismo proceso de Bonita OS, el cual tiene asignado otro conector llamado "tiempoEjecucion" para este indicador que se ejecuta cuando finaliza el proceso. Nuevamente se le envía como parámetro al conector el identificador de la instancia del proceso, para que en su implementación pueda acceder a los datos de esa instancia. También, en su implementación el conector debe importar las librerías de WSO2 BAM para poder utilizar los métodos y conectarse con la aplicación.

La implementación del conector "tiempoEjecucion" se encarga en primer lugar, de traer los datos de la instancia que comenzó a ejecutarse a través de la API Rest de Bonita. En este caso, a partir del identificador de la instancia del caso iniciado, se consigue el identificador del proceso, el nombre del proceso y el tiempo en que se inició el caso.

Luego se realiza el cálculo del tiempo de ejecución que tardó el proceso [Figura 5.33]. Para esto se utilizan dos fechas: la fecha actual es decir la fecha en que el proceso finalizó (el conector se ejecuta cuando el proceso termina), y la fecha de inicio del proceso que se la obtuvo desde la API Rest. Después de formateadas ambas fechas, se hace uso de la clase "PruebaTime" para realizar el cálculo de la diferencia de fechas mediante el método "calcularDiferencia()" cuyo resultado es el tiempo de ejecución de esa instancia.

```
// calculo el tiempo de ejecución
Long dif;
String time;
    //fecha2 actual
String DATE_FORMAT_NOW = "dd-MM-yyyy HH:mm:ss";
Calendar cal = Calendar.getInstance();
SimpleDateFormat sdf = new SimpleDateFormat(DATE_FORMAT_NOW);
String fecha2 = sdf.format(cal.getTime());
    //fecha1 de inicio traída desde la api
SimpleDateFormat originalFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
SimpleDateFormat targetFormat = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss" );
Date date = null;
  try {
    date = originalFormat.parse(fechainicioapi);
} catch (java.text.ParseException e1) {
    e1.printStackTrace();
}
  String fecha1 = targetFormat.format(date);
   // fecha actual menos fecha de inicio (tiempo que tardó el proceso)
PruebaTime calc = new PruebaTime();
dif = calc.calcularDiferencia(fecha1, fecha2);
time = calc.calcularFecha(dif);
```

Figura 5.33. *Cálculo del tiempo de ejecución en el conector "tiempoEjecución"*

Seguidamente se realizan dos acciones: primero debido a que se ejecuta el conector cuando el proceso finalizó, se lo desactiva mediante el método "setAlertaProceso" cuya implementación consiste en setear el campo activo en 0 de dicho proceso en la base de datos relacional.

La otra acción es enviar los datos obtenidos a la aplicación WSO2 BAM mediante el método "testSendingEvent()" de la clase "TestClientGenerali". Estas acciones se pueden observar en la Figura 5.34.

```
//desactivo el proceso porque terminó
try {
    this.setAlertaProceso(iddelproceso);
} catch (Exception e1) {
    // TODO Bloque catch generado automáticamente
    e1.printStackTrace();
}

//envio el evento
TestClientGenerali t = new TestClientGenerali();
try {
    //processId = caso
    t.testSendingEvent(time, nombreprocesoapi, dif, iddelproceso, processId, fecha2);
} catch (MalformedURLException e) {
    // TODO Bloque catch generado automáticamente
    e.printStackTrace();
} catch (AuthenticationException e) {
    // TODO Bloque catch generado automáticamente
    e.printStackTrace();
```

Figura 5.34. *Implementación del conector "tiempoEjecución"*

El método "testSendingEvent()" [Figura 5.35] es similar al que se utilizó en el indicador descripto anteriormente. Se declaran en este caso dos streams con los datos a publicar en la aplicación WSO2 BAM. El primer stream de datos corresponde al registro del tiempo de ejecución de tal instancia de proceso, cuyo nombre del stream de ejemplo es "tabla_te_3" y la versión "2.3.1".

```
System.setProperty("javax.net.ssl.trustStore", "D:/[TESIS]/wso2bam-2.4.1/repository/resources/security/client-truststore.jks");
System.setProperty("javax.net.ssl.trustStorePassword", "wso2carbon");
Thread.sleep(2000);

//according to the convention the authentication port will be 7611+100= 7711 and its host will be the same
DataPublisher dataPublisher = new DataPublisher("tcp://localhost:7611", "admin", "admin");
String streamId1 = dataPublisher.defineStream("{" +
                                              "  'name':'tabla_te_3'," +
                                              "  'version':'2.3.1'," +
                                              "  'nickName': 'Tabla de tiempo de ejecucion'," +
                                              "  'description': 'Some Desc'," +
                                              "  'tags':['foo', 'bar']," +
                                              "  'metaData':[" +
                                              "          {'name':'ipAdd','type':'STRING'}" +
                                              "  ]," +
                                              "  'payloadData':[" +
                                              "          {'name':'id','type':'LONG'}," +
                                              "          {'name':'caso','type':'STRING'}," +
                                              "          {'name':'nombre del proceso','type':'STRING'}," +
                                              "          {'name':'tiempo de ejecucion','type':'STRING'}," +
                                              "          {'name':'milisegundos','type':'LONG'}," +
                                              "  ]" +
                                              "}");
log.info("1st stream defined: "+streamId1);

//In this case correlation data is null
dataPublisher.publish(streamId1, new Object[]{"127.0.0.1"}, null, new Object[]{idproceso, caso,nombre,te,dif});
log.info("Event published to 1st stream");
```

Declaración del Stream

Datos a publicar

Figura 5.35. *Implementación del método "testSendingEvent()" primer stream*

El segundo stream de datos corresponde al registro de los procesos finalizados, cuya implementación se observa en la Figura 5.36.

```
//NOTE publico otro stream con los datos de finalización del proceso
String streamId2 = dataPublisher.defineStream("{" +
        "  'name':'finprocesos'," +
        "  'version':'2.3.1'," +
        "  'nickName': 'Tabla de procesos finalizados'," +
        "  'description': 'Some Desc'," +
        "  'tags':['foo', 'bar']," +
        "  'metaData':[" +
        "          {'name':'ipAdd','type':'STRING'}" +
        "  ]," +
        "  'payloadData':[" +
        "          {'name':'id','type':'LONG'}," +
        "          {'name':'nombre','type':'STRING'}," +
        "          {'name':'caso','type':'STRING'}," +
        "          {'name':'fecha_final','type':'STRING'}" +
        "  ]" +
        "}");

dataPublisher.publish(streamId2, new Object[]{"127.0.0.1"}, null, new Object[]{idproceso, nombre, caso, fechaFin});
log.info("Event published to 2nd stream");
```

Figura 5.36. *Implementación del método "testSendingEvent()" segundo stream*

WSO2 BAM

Una vez que se publican los streams de datos, la aplicación WSO2 BAM los recibe y los procesa. Para esto se debe realizar ciertas configuraciones en la aplicación. Primero se levanta el servidor BAM en su url correspondiente (https://192.168.0.45:9446/carbon/) como lo refleja la Figura 5.37.

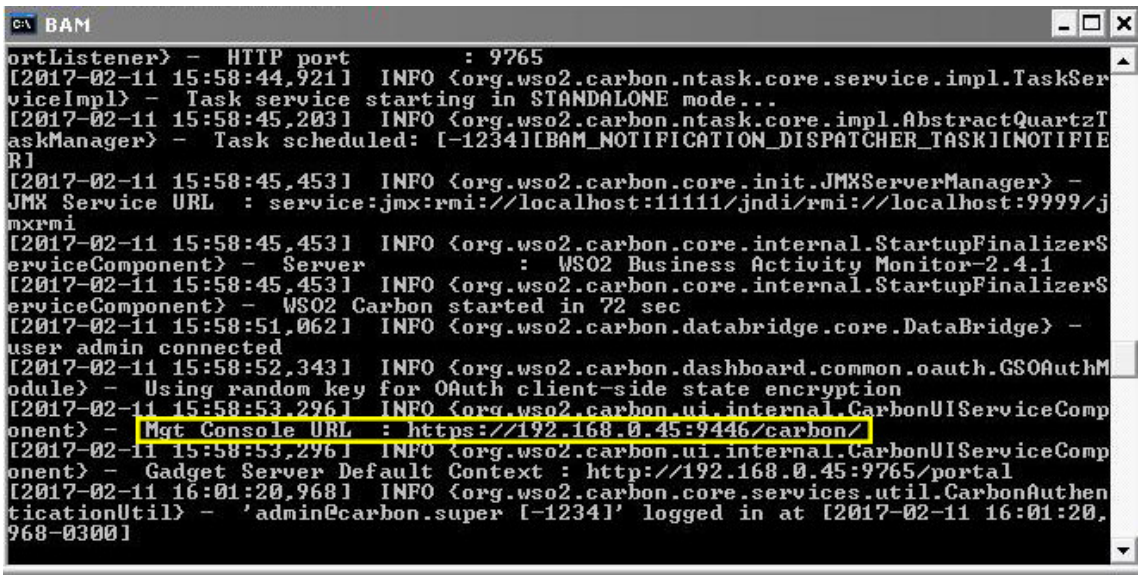


Figura 5.37. *Consola de WSO2 BAM iniciándose*

Luego se generan dos Event Streams para recibir como eventos los dos streams de datos disparados por el conector de Bonita OS. Cabe destacar que el nombre de cada event stream debe coincidir con el nombre del publicador del evento desarrollado en el conector. Para el caso de la Figura 5.38 el nombre del stream de datos es "tabla_te_3" y la versión "2.3.1". Además se realiza el mapeo de las variables recibidas por el evento declaradas en el publicador del conector.

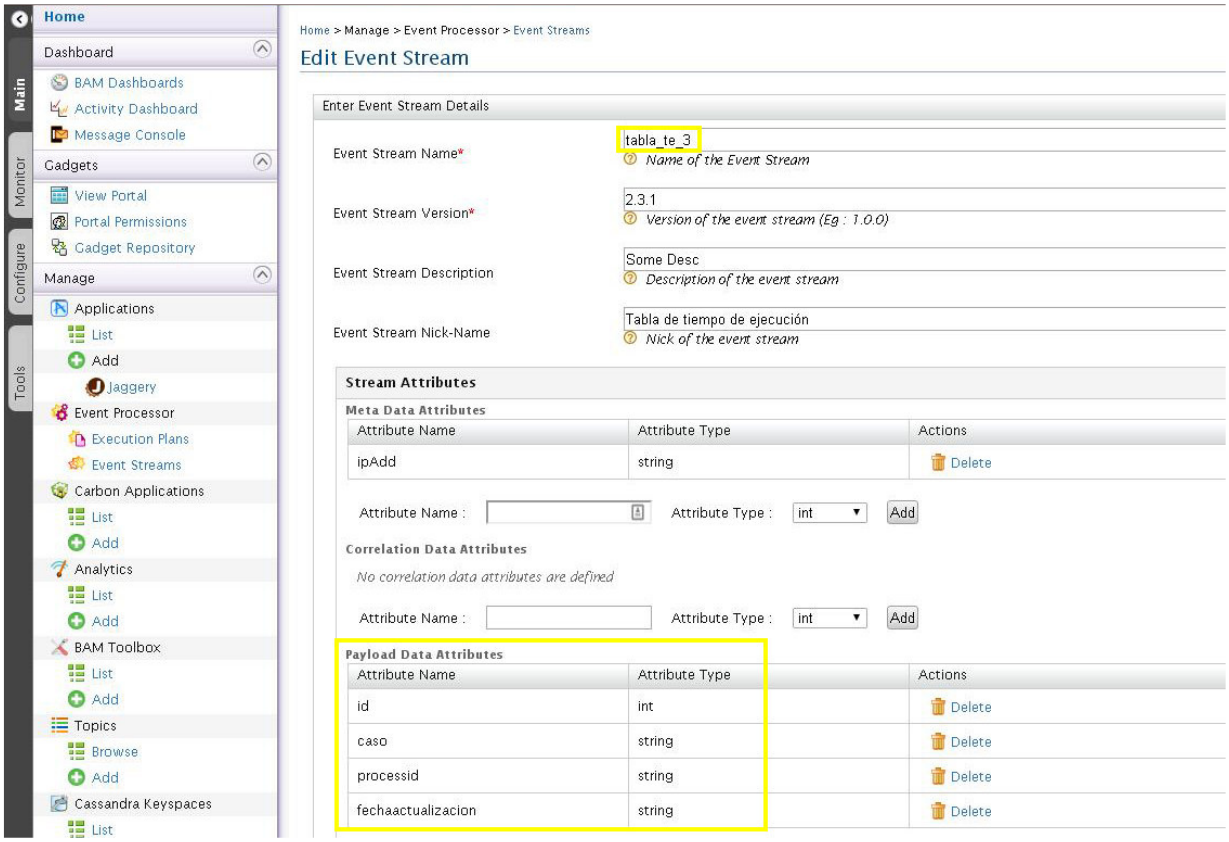


Figura 5.38. *Event stream tiempo de ejecución*

De la misma forma se crea un nuevo *Event Stream* para el stream de datos de los procesos finalizados [Figura 5.39].

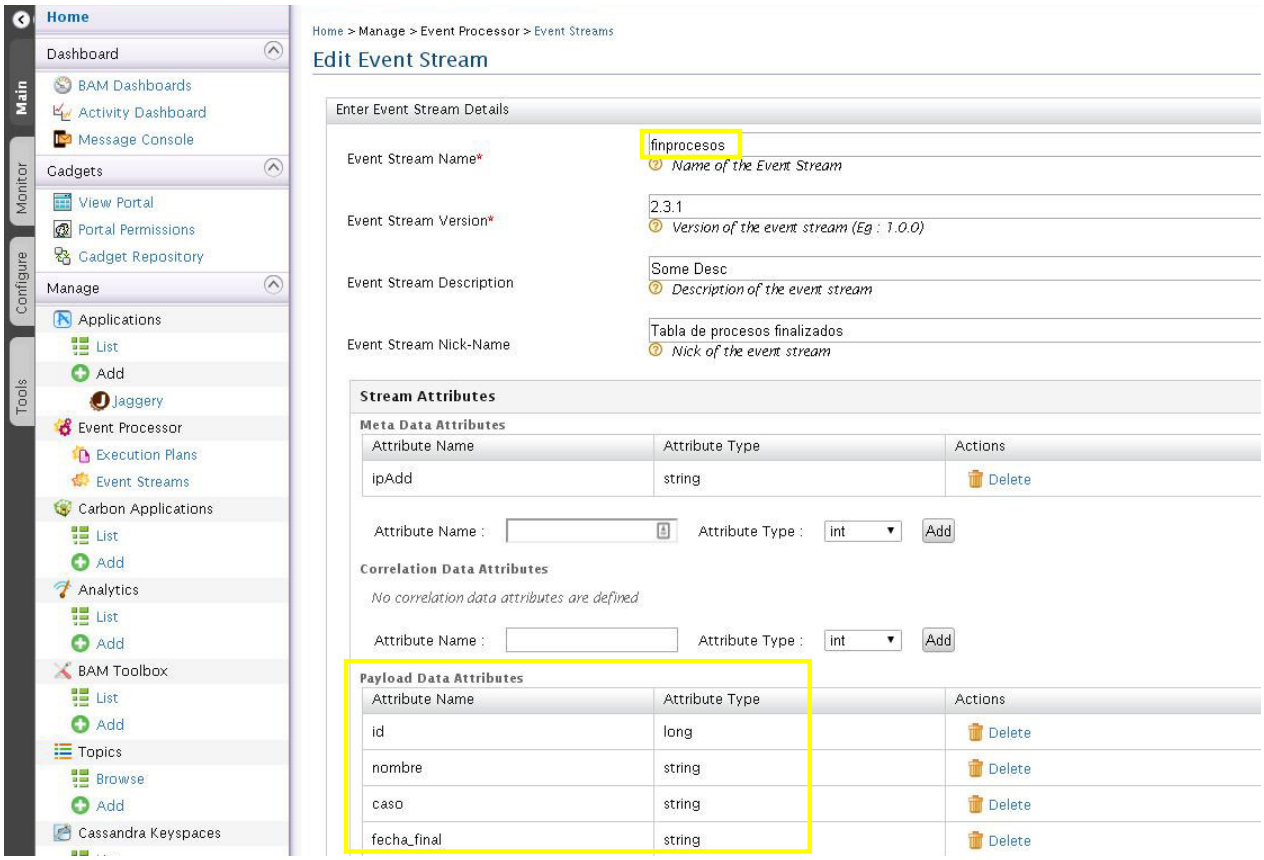


Figura 5.39. *Event stream fin de proceso*

Los datos en este caso se guardan en Apache Cassandra, la base de datos no relacional distribuida que posee WSO2 BAM. Como se mencionó en secciones anteriores, esta base de datos se caracteriza por permitir almacenar grandes volúmenes de datos en forma distribuida consumiendo pocos recursos.

Una vez persistidos los datos en la base de datos Cassandra, mediante el uso de scripts Hive se realiza el mapeo de la información que se necesita para enviarla a la base de datos relacional MYSQL. El hecho de usar Apache Hive, ayuda a abstraerse de la fuente y destino de los datos; lo que permite centrarse solamente en analizar los datos requeridos, e implementar los scripts Hive.
Cabe recordar que Apache Hive usa procesos Hadoop que hacen uso de la técnica "*Map Reduce*" para mejorar la eficiencia en el procesamiento de un gran volumen de datos (BigData).

En la aplicación WSO2 BAM, los scripts Hive se agregan en la sección *Analytics*. A su vez se puede programar los scripts para que se ejecuten cada cierta cantidad de tiempo con la función "*Schedule Script*". El promedio del tiempo de ejecución es un indicador histórico de modo que esta técnica resulta apropiada programando que el script de traspaso de datos se ejecute cada cierto período prolongado de tiempo. Por ejemplo para el caso de estudio se configuró que el script de tiempo de ejecución se ejecute cada 30 días.
El script que transfiere los datos referentes al tiempo de ejecución de los procesos es el que se muestra en la Figura 5.40.

```
DROP TABLE IF EXISTS hive_tiempo1;
CREATE EXTERNAL TABLE IF NOT EXISTS hive_tiempo1(
    logId STRING,idproceso STRING,caso BIGINT,
    nombre STRING,tiempo STRING,milisegundos BIGINT)
    STORED BY 'org.apache.hadoop.hive.cassandra.CassandraStorageHandler'
    WITH SERDEPROPERTIES (
    "wso2.carbon.datasource.name" = "WSO2BAM_CASSANDRA_DATASOURCE" ,
    "cassandra.cf.name" = "tabla_te_3" ,
    "cassandra.cf.validatorType" = "UTF8Type,LongType,LongType,UTF8Type,UTF8Type,LongType",
    "cassandra.columns.mapping" = ":key,payload_id, payload_caso, payload_nombre del proceso,payload_tiempo de ejecucion,payload_milisegundos");

DROP TABLE IF EXISTS teprocesos;
CREATE EXTERNAL TABLE IF NOT EXISTS teprocesos(
    id STRING, nombre STRING, tiempo STRING, milisegundos BIGINT)
    STORED BY 'org.wso2.carbon.hadoop.hive.jdbc.storage.JDBCStorageHandler'
    TBLPROPERTIES (
    'mapred.jdbc.driver.class' = 'com.mysql.jdbc.Driver' ,
    'mapred.jdbc.url' = 'jdbc:mysql://localhost:3306/wso2bam' ,
    'mapred.jdbc.username' = 'emi' ,
    'mapred.jdbc.password' = '123' ,
    'hive.jdbc.update.on.duplicate' = 'true' ,
    'hive.jdbc.primary.key.fields' = 'id' ,
    'hive.jdbc.table.create.query' = 'CREATE TABLE teprocesos (id VARCHAR(100), nombre VARCHAR(100),tiempo VARCHAR(100),
    milisegundos LONG, PRIMARY KEY (id))');

insert overwrite table teprocesos
        select cast(idproceso as STRING)as id, cast(nombre as STRING) as nombre, cast(tiempo as STRING) as tiempo,
        cast(milisegundos as BIGINT) as milisegundos
        from hive_tiempo1
```

Figura 5.40. *Script Hive tiempo de ejecución*

Una vez que transcurrió el período de tiempo establecido en la configuración del script, el mismo se ejecuta generando que los datos se persistan en la base de datos MYSQL. También puede ejecutarse el script manualmente cuando se desee. La base de datos cargada para ser utilizada directamente por la aplicación del Tablero de Control se muestra en la Figura 5.41.

Figura 5.41. *Tabla "teprocesos" actualizada*

Con los datos cargados en la base de datos relacional, se explicará cómo Liferay hace uso de los mismos para visualizarlos en su interfaz.

Liferay

Se genera un portlet denominado "Tiempo de Ejecución Avg" el cual básicamente lo que hace es realizar el cálculo del promedio de tiempo de ejecución de las instancias de un proceso mediante una consulta SQL. Luego de obtener la lista de procesos con sus promedios respectivos, se hace uso de un componente de gráficos de barra que se despliega en el portlet cuando este se ejecuta en Liferay.

La Figura 5.42 muestra la clase principal del portlet llamada “TiempoEjecucionAvg” que setea como atributo a la lista de los procesos con sus promedios, esto derivado del método “avgProcesos()” de la clase “MySQLAccess”; luego se redirige a la vista del portlet.

```
1  package portlets;
2
3  import java.io.IOException;
12
13 /**
14  * Portlet implementation class TiempoEjecucionAvg
15  */
16 public class TiempoEjecucionAvg extends GenericPortlet{
17
18     @RenderMode(name = "VIEW")
19     public void renderForm(RenderRequest request, RenderResponse response) throws PortletException, IOException {
20         MySQLAccess con = new MySQLAccess();
21         try {
22             request.setAttribute("lista",con.avgProcesos());
23         } catch (Exception e) {
24             // TODO Auto-generated catch block
25             e.printStackTrace();
26         }
27
28         getPortletContext().getRequestDispatcher("/jsp/tiempoAvg_view.jsp").include(request, response);
29     }
```

Figura 5.42. *Clase “TiempoEjecucionAvg”*

La consulta se encarga de obtener el nombre y el promedio de cada proceso insertado en la tabla “teprocesos”. Para esto realiza la sumatoria de los tiempos de un determinado proceso agrupando por nombre, dividido por la cantidad de instancias de ese mismo proceso. El método que realiza esta tarea se observa en la Figura 5.43.

```
15
16 public ArrayList<HashMap<String, String>> avgProcesos() throws Exception {
17     try {
18       String cons = "SELECT nombre, SUM(milisegundos)/ COUNT(id) as promedio FROM teprocesos GROUP BY nombre";
19       this.enviarConsulta(cons);
20       OperacionesTiempo op = new OperacionesTiempo();
21       ArrayList<HashMap<String, String>> lista = new ArrayList<HashMap<String, String>>();
22       while (resultSet.next()) {
23           HashMap<String, String> aux = new HashMap<String, String>();
24           String nombre = resultSet.getString("nombre");
25           String promedio = op.calcularMinutos(resultSet.getLong("promedio"));
26           aux.put(nombre, promedio);
27           lista.add(aux);
28         }
29       return lista;
30     } catch (Exception e) {
31       throw e;
32     } finally {
33       connect.close();
34     }
35   }
```

Figura 5.43. *Método “avgProcesos()”*

La vista hace uso de un componente de gráfico de barras de la librería *HighCharts* de JavaScript, el cual se carga de los datos mediante una tabla. Como se observa en la Figura 5.44, se itera en la lista seteada anteriormente como un arreglo clave valor.

```
4
5  <portlet:defineObjects />
6
7  <ui:setProject name="tablero"></ui:setProject>
8  <ui:theme name="tablero"></ui:theme>
9  <ui:javaScript name="tablero"></ui:javaScript>
10 <ui:columnParsed units="Minutos" tableId="datatable" title="Promedio del Tiempo de Ejecución de Procesos"></ui:co
11                                   Componente
12     <table id="datatable">
13         <thead>
14             <tr>
15                 <th></th>
16                 <th>Avg</th>
17             </tr>
18         </thead>
19         <tbody>
20             <c:forEach items="${requestScope.lista}" var="map"> Datos
21                 <c:forEach items="${map}" var="proceso">
22                     <tr>
23                             <th>${proceso.key}</th>
24                             <td>${proceso.value}</td>
25                     </tr>
26                 </c:forEach>
27             </c:forEach>
28         </tbody>
29     </table>
30
```

Figura 5.44. *Vista "tiempoAvg_view.jsp"*

El resultado del despliegue del portlet en el portal de Liferay con los datos cargados en el momento de hacer la prueba [Figura 5.45], muestra que el proceso "Realizar Promoción" tuvo un promedio de 20 minutos acumulado en las instancias que se ejecutaron en el último mes. Mientras que el proceso "Solicitud Bloqueo Horario" tuvo un promedio de 3 minutos.

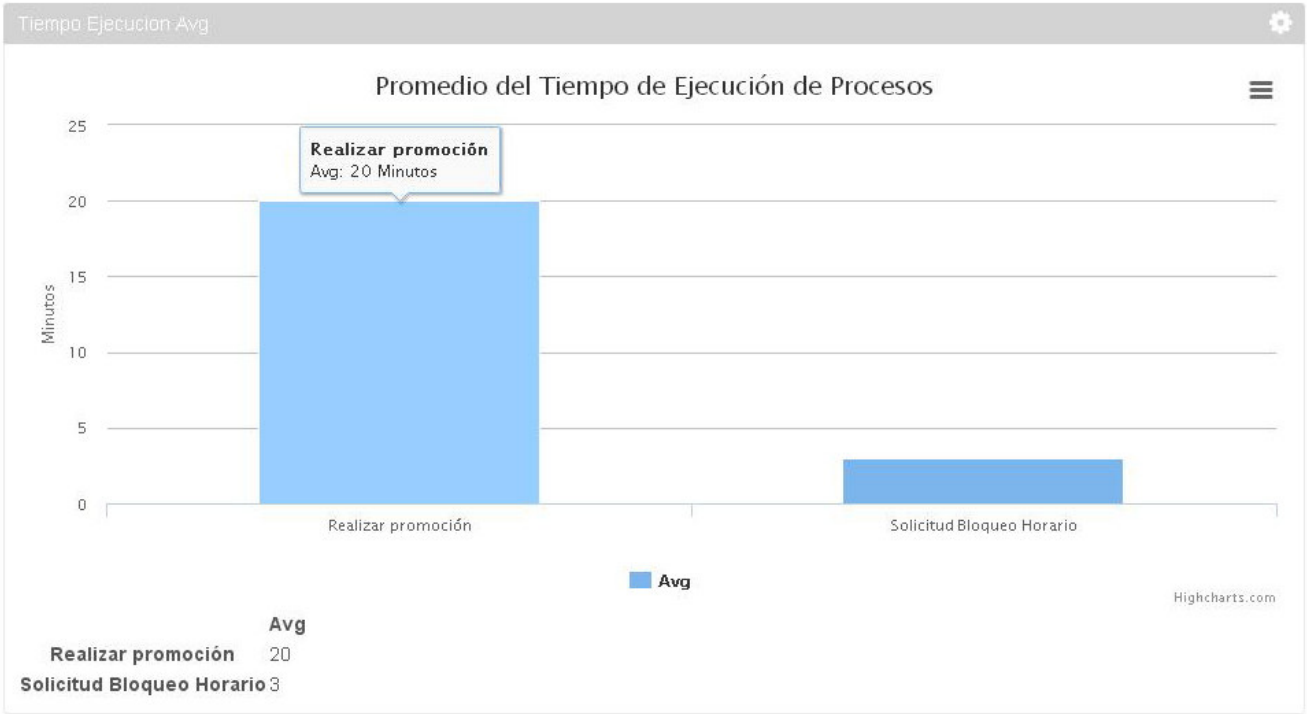


Figura 5.45. *Portlet "Tiempo Ejecucion Avg"*

Control de tareas fallidas:

Este indicador muestra las tareas que van fallando, cada cierto intervalo de tiempo, junto con información relativa al proceso que disparó cada tarea.
Para esto el portlet en su vista [Figura 5.46] tiene una redirección hacia una aplicación externa cuya ejecución se realiza cada cierto intervalo de tiempo. Este se setea con la función "setInterval()" la cual para el ejemplo se establece en 5000 milisegundos.

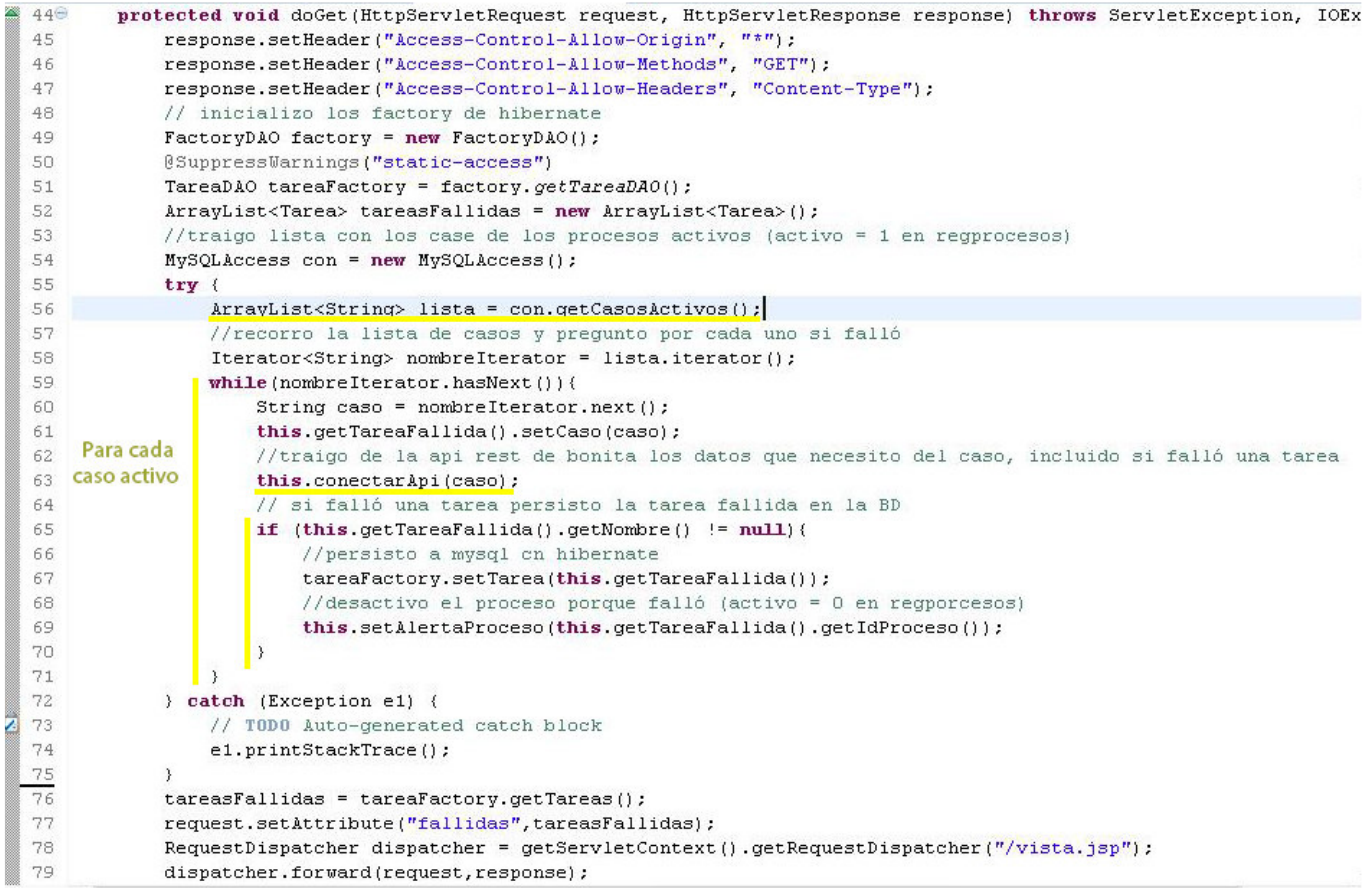
La idea de este indicador surge a partir de que los procesos se encuentran activos hasta, o bien terminan de ejecutarse, o bien fallan. Por lo tanto la forma de descubrir si falla algún proceso, es averiguando si tuvo un fallo en una tarea cada cierto tiempo.

```
<script type="text/javascript">
var url= "http://localhost:8081/ControlErrores/controlarAPI";  Aplicación

function redireccionar(){
    $.ajax({
        type: "GET",
        url: url,
//          data: {id:id},
        dataType: "html",
        success: function (msg) {
            $(".add2").append();
            $(".add").html(msg);
            $(".update").show("slow");
            $('html,body').animate({scrollTop: $(".update").offset().top});
        },
        error: function (msg){
            alert("Error. Vuelva a Intentarlo.");
        }
    });
}
setInterval('redireccionar()',5000); Intervalo
</script>

<h2> Tareas Fallidas</h2>
    <div class="add"></div>
```

Figura 5.46. *Vista del portlet de errores "errores_view.jsp"*

La aplicación externa "controlarAPI" es una aplicación Java que hace uso de los datos ya procesados por la aplicación WSO2 CEP y que persistió la información relevante en la base de datos MYSQL.

Esta aplicación directamente interactúa con MYSQL para traer la lista de las instancias de procesos activos, es decir los que fueron iniciados y persistidos por WO2CEP en la base de datos relacional; información disparada por el conector que registra los procesos que inician. La implementación de dicha aplicación se puede observar en las Figuras 5.47 y 5.48.




```
44    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOEx
45        response.setHeader("Access-Control-Allow-Origin", "*");
46        response.setHeader("Access-Control-Allow-Methods", "GET");
47        response.setHeader("Access-Control-Allow-Headers", "Content-Type");
48        // inicializo los factory de hibernate
49        FactoryDAO factory = new FactoryDAO();
50        @SuppressWarnings("static-access")
51        TareaDAO tareaFactory = factory.getTareaDAO();
52        ArrayList<Tarea> tareasFallidas = new ArrayList<Tarea>();
53        //traigo lista con los case de los procesos activos (activo = 1 en regprocesos)
54        MySQLAccess con = new MySQLAccess();
55        try {
56            ArrayList<String> lista = con.getCasosActivos();
57            //recorro la lista de casos y pregunto por cada uno si falló
58            Iterator<String> nombreIterator = lista.iterator();
59            while(nombreIterator.hasNext()){
60                String caso = nombreIterator.next();
61                this.getTareaFallida().setCaso(caso);
62                //traigo de la api rest de bonita los datos que necesito del caso, incluido si falló una tarea
63                this.conectarApi(caso);
64                // si falló una tarea persisto la tarea fallida en la BD
65                if (this.getTareaFallida().getNombre() != null){
66                    //persisto a mysql cn hibernate
67                    tareaFactory.setTarea(this.getTareaFallida());
68                    //desactivo el proceso porque falló (activo = 0 en regporcesos)
69                    this.setAlertaProceso(this.getTareaFallida().getIdProceso());
70                }
71            }
72        } catch (Exception e1) {
73            // TODO Auto-generated catch block
74            e1.printStackTrace();
75        }
76        tareasFallidas = tareaFactory.getTareas();
77        request.setAttribute("fallidas",tareasFallidas);
78        RequestDispatcher dispatcher = getServletContext().getRequestDispatcher("/vista.jsp");
79        dispatcher.forward(request,response);
```

Figura 5.47. *Implementación del servlet "controlarAPI.java"*

Luego se recorre la lista de procesos activos y se llama al método "conectarApi()" que se encarga de determinar si el caso analizado tuvo una tarea fallida, como así también de recolectar toda la información relevante de ese caso. Todo esto lo realiza por medio de la Api Rest de Bonita, de modo que mediante el id del caso de cada proceso activo se conecta a la Api para determinar si cada caso ha sido abortado por el fallo de una tarea, y de ser así la información de dicha tarea fallida.

```
104
105         //Pido la tarea fallida si es que tiene, para un determinado processid
106         String iddelproceso = this.getTareaFallida().getIdProceso();
107         url= "http://localhost:8083/bonita/API/bpm/task?p=0&f=state%3dfailed&f=processId%3d"+iddelproceso;
108         metodo= "GET";
109         resultado= proxy.enviarPeticion(url, metodo, null, null);
110             //Recorremos el archivo JSON
111         try {
112             //Creo un objeto JSON a traves del resultado string
113             JSONParser par= new JSONParser();
114             JSONArray array= (JSONArray) par.parse(resultado);
115             //Desde aca consigo cada etiqueta por separado
116             JSONObject json = (JSONObject) array.get(0);
117             String name = (String) json.get("name");
118             String id = (String) json.get("id");
119             String type = (String) json.get("type");
120             String fechaf = (String) json.get("reached_state_date");
121             idusuario = (String) json.get("assigned_id");
122             idactor = (String) json.get("actorId");
123             this.getTareaFallida().setNombre(name);
124             this.getTareaFallida().setId(id);
125             this.getTareaFallida().setTipo(type);
126             this.getTareaFallida().setFechaFallo(fechaf);
127
128         }
129         catch (Exception e) {
130             // TODO Auto-generated catch block
131             e.printStackTrace();
132         }
```

Datos de la tarea fallida o null

Figura 5.48. *Implementación del servlet "controlarAPI.java"*

La recolección de los datos mencionados, tuvo como efecto la instanciación y seteo de variables de instancia de un objeto de clase TareaFallida, pues para hacer la persistencia de la lista de tareas fallidas a la base de datos MYSQL se utilizó el framework ORM (mapeo objeto relacional) Hibernate.

Si es que el objeto es nulo, significa que ese caso no falló; de manera opuesta se persiste la tarea fallida en la base de datos relacional para luego usarse en la vista, y además se desactiva el proceso a asociado nivel base de datos porque falló; de esta manera se registra que tal proceso ya no está más activo.

Luego la vista HTML [Figura 5.49] toma la lista de tareas fallidas recientemente persistida, para incrustarla en la vista de Liferay.

```
50
51 <table class="table-hover">
52     <tr>
53         <th> Id</th>
54         <th> Caso</th>
55         <th> Nombre</th>
56         <th> Nombre Proceso</th>
57     </tr>
58     <c:forEach items="${fallidas}" var="tarea">
59         <!-- HIDDEN / POP-UP DIV -->
60         <div class="pop-up" id="${tarea.id}">
61           <div class="tit">Información adicional</div>
62           <p>
63                 - Tipo: <c:out value="${tarea.tipo}"></c:out> <br>
64                 - Actor: <c:out value="${tarea.actor}"></c:out> <br>
65                 - Usuario: <c:out value="${tarea.usuario}"></c:out> <br>
66                 - IdProceso: <c:out value="${tarea.idProceso}"></c:out> <br>
67                 - FechaFallo: <c:out value="${tarea.fechaFallo}"></c:out> <br>
68           </p>
69         </div>
70         <tr id="trigger">
71             <td> <c:out value="${tarea.id}"></c:out> </td>
72             <td> <c:out value="${tarea.caso}"></c:out> </td>
73             <td> <c:out value="${tarea.nombre}"></c:out> </td>
74             <td> <c:out value="${tarea.nombreProceso}"></c:out> </td>
75         </tr>
76     </c:forEach>
77 </table>
```

Figura 5.49. *Vista la aplicación ControlErrores "vista.jsp"*

Realizando pruebas en las que de forma intencionada se hicieron fallar a determinadas tareas, el portlet de las tareas fallidas resultante se visualiza en Liferay como muestra la Figura 5.50.

Figura 5.50. *Portlet "Control Errores"*

Todos los portlets desarrollados conforman en conjunto el Tablero de Control. Para el uso del tablero de control completo se requiere, mientras está corriendo un proceso de Bonita, que estén levantados el servidor de WSO2 BAM, el servidor de WSO2 CEP, el servidor de la base de datos MYSQL, el servidor de Liferay y para el caso del portlet de control de errores, el servidor Tomcat que aloca la aplicación externa que usa dicho portlet. La Figura 5.51 muestra todos los portlets desplegados conformando el Tablero de Control.

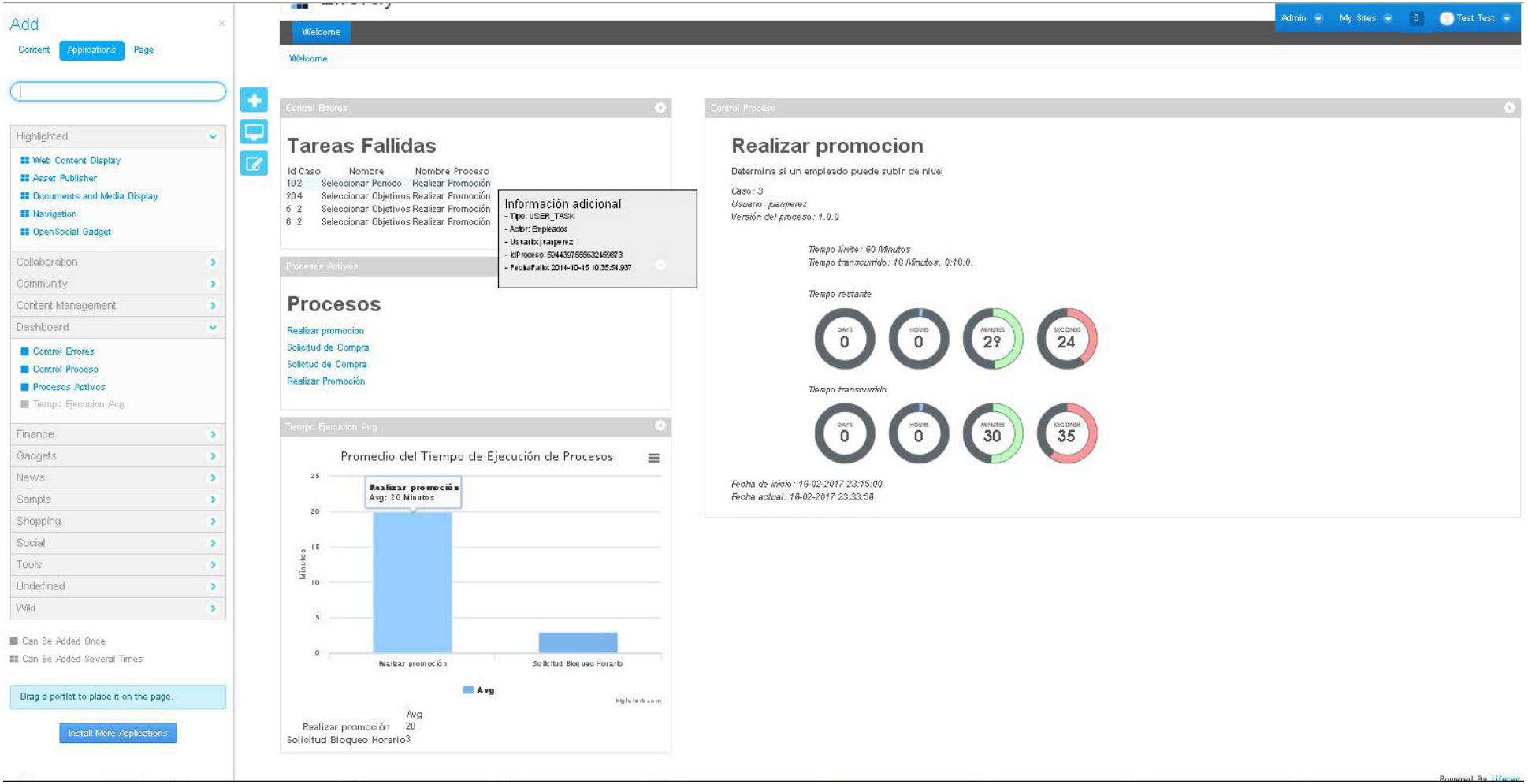


Figura 5.51. *Tablero de Control*

6

6. Conclusiones y trabajos futuros

Partiendo de la premisa de construir un tablero de control que monitoree los procesos de un BPMS, en primera medida se buscó investigar cada uno de los tópicos generales para este desarrollo desde un punto de vista teórico. La parte de la tecnología BAM fue fundamental para tener noción de los aspectos importantes a tener en cuenta para medir los procesos, y que estas mediciones sugieran mejoras en el rendimiento de los mismos.

Para esto resultó importante el concepto de **tiempo real**, de manera de poder manejar la información en el instante que ocurre y por consiguiente poder tomar decisiones rápidas ante situaciones no deseadas. Estas situaciones están determinadas por los **indicadores clave de rendimiento**, otro concepto significativo para establecer qué datos del proceso conviene medir y cuándo se alcanzan valores indeseados. Por otro lado, otro aspecto a considerar fue la parte visual, donde se muestra la información en forma de **tablero de control**, de modo que se hizo gran hincapié en cómo construir el tablero para que se muestren los indicadores y las alertas de forma precisa e interactiva.

Una vez entendidos estos aspectos generales vinculados a los objetivos del trabajo, se buscó en primera medida diseñar los indicadores clave de rendimiento, es decir qué información recolectar de los procesos para mostrar en el tablero que resulte útil para monitorear. Debido a la experiencia tenida utilizando el BPMS **Bonita OS** se prosiguió en el uso de dicha herramienta para gestionar y ejecutar los procesos de negocio.
Se pensó en indicadores que sirvan para todos los procesos en general, que no sean sólo adecuados para un único dominio. Al mismo tiempo, y en cuanto a la forma de recolectar dicha información se optó por el uso de **conectores** los cuales pueden asignarse a un proceso y luego conectarse a la API REST de Bonita para obtener la información del mismo.

Más allá de la perspectiva de tiempo real, se consideró que resultaría útil que el tablero contase con información histórica de los procesos, por ejemplo para saber cómo fueron evolucionando o para tener conocimiento estadístico de lo que ocurrió en un lapso considerable de tiempo.
De modo que los indicadores elaborados se dividieron en los mencionados **indicadores estáticos**, es decir indicadores con información histórica, e **indicadores dinámicos** para los que se procesan los datos en tiempo real y permiten tomar decisiones de forma rápida, cuando las situaciones no apropiadas suceden.

Los indicadores que se idearon en primera instancia fueron:

- Promedio del tiempo de ejecución (indicador estático)
- Tiempo de ejecución de los procesos activos (indicador dinámico)
- Control de tareas fallidas (indicador dinámico)

- Porcentaje de finalización (indicador histórico): para cada proceso calcular el porcentaje de los casos que llegaron a finalizar.
- Porcentaje de tareas fallidas por proceso (indicador histórico): para cada proceso calcular el porcentaje de tareas fallidas.
- Tiempo de ejecución de las tareas de un proceso activo (indicador dinámico): mostrar el tiempo de transcurrido de las tareas que se van ejecutando, como así también su tiempo restante derivado de un indicador límite que al sobrepasarse se informa un alerta.

Los tres primeros indicadores se llegaron a desarrollar de forma completa. Los tres restantes se pensaron como para aumentar la funcionalidad del tablero en el futuro; las bases en cuanto a la implementación son similares a lo desarrollado.

Una vez establecido qué información capturar de los procesos y cómo recolectarla, se dispuso a buscar la forma de procesar dicha información respetando los principios de una solución BAM. Era indispensable tener en cuenta que los datos se capturan en forma de eventos, que la cantidad de eventos podría escalar exponencialmente, y que un grupo de estos eventos se deben procesar en tiempo real.

Ante este escenario, lo primero que se hizo fue buscar una herramienta que provea alguna funcionalidad requerida. Allí fue cuando se descubrió **WSO2 BAM** aplicación capaz de recibir eventos disparados por los procesos de negocio, procesarla e incluso visualizarla a través de componentes de control predefinidos.
El problema que surgió al usar esta aplicación fue que tardaba demasiado tiempo en procesar la información; el mapeo de los datos desde la base de datos de la aplicación a otra base de datos relacional mediante scripts Hive demandaba mucho tiempo, de modo que el concepto de "tiempo real" todavía estaba lejano.
Para solucionar este inconveniente se dispuso a investigar aplicaciones cuya característica primordial sea el procesamiento de eventos en tiempo real, entre las que destacan Apache Storm y **WSO2 CEP**. Se inclinó por esta última debido a que trabaja con los mismos agentes que WSO2 BAM para capturar eventos, y ya se contaba con tal implementación; además por cuestiones de compatibilidad, ya que ambas aplicaciones pueden trabajar paralelamente sobre una misma arquitectura.
A raíz de este problema, se profundizó el conocimiento sobre WSO2 BAM descubriendo que su utilidad radica en el procesamiento de múltiples eventos, pero para un tipo de procesamiento por lotes; es decir un gran volumen de información procesada cada cierto tiempo, no en tiempo real.

Teniendo entonces dos funcionalidades trascendentales cubiertas por ambas aplicaciones se dispuso a investigar si ambas podían coexistir y sobre qué tipo de arquitectura, llegando a la conclusión que la **Arquitectura Lambda** era la solución. WSO2 BAM montada sobre la capa de procesamiento por lotes, y WSO2 CEP correspondiendo a la capa de procesamiento veloz. Bonita OS para ejecutar los procesos y disparando eventos a través de sus conectores hacia estas aplicaciones. Fue importante cambiar los "thrift ports" de alguna de las aplicaciones así una apunta a un puerto diferente y no se solapan.

En cuanto a la parte visual, no alcanzaba con los componentes que ofrecían las aplicaciones utilizadas debido a sus limitaciones y a la falta de personalización. Por lo tanto, se decidió investigar sobre el uso de portales web y la elección fue un poco más sencilla, encontrando en **Liferay** una plataforma muy amplia y de gran impacto visual para el desarrollo del tablero de control. El concepto de portlets como componentes del tablero y la facilidad con la que se pueden desplegar, jugó un rol fundamental para tener una visualización agradable y efectiva. Por otro lado el uso del

lenguaje JAVA, y la capacidad alta de personalización de las aplicaciones resultaron oportunos al momento de desarrollarlas.

En conclusión, luego de un minucioso trabajo de investigación, se logró encontrar las herramientas adecuadas para el propósito inicial. De hecho se han investigado opciones alternativas para procesar eventos en tiempo real y mostrar la información de forma precisa; sin embargo se creyó más oportuna la elección de las aplicaciones y arquitectura descriptas en el trabajo, tanto por cuestiones de eficiencia como de compatibilidad entre herramientas para lograr una arquitectura unificada. Terminó resultando satisfactorio el empleo de estas aplicaciones diferentes que sobre una misma arquitectura conviven y cooperan para un propósito en común.

Si bien no se llegó a probar la aplicación en un escenario estrictamente BigData, por no poder contar con centros de cómputo o una arquitectura que soporte el envío de millones de eventos en tiempo real, sí se llegó a probar que las aplicaciones trabajan en forma correcta y eficiente, tanto por el procesamiento que se manejó en el caso práctico, como por las referencias analizadas en el marco teórico.

Como premisa para el futuro, se buscará incrementar la funcionalidad de la aplicación incorporando nuevos conectores que obtengan diferentes tipos de información de los procesos, y mediante el desarrollo de nuevos KPIs, se pueda agregar nuevos portlets al portal/tablero de control. Se mencionó previamente tres indicadores cuya implementación no difiere de lo que ya se desarrolló.
Además, para los indicadores de información histórica es posible agregar filtros, tales como pueden ser el filtrado temporal; es decir mostrar los datos de un mes, año, o lapso determinado.
También se pensó en tener cierto control desde el tablero, sobre la instancia de proceso que se está ejecutando; por ejemplo en el portlet de control de errores, donde se muestran las tareas fallidas, permitir al usuario del tablero que finalice el proceso o por el contrario que le permita reiniciar la tarea que falló.
Asimismo, se buscarán casos de uso con una cantidad de eventos más idónea como para probar la potencia real de la aplicación respecto al procesamiento de múltiples eventos sea en tiempo real o no.

Referencias Bibliográficas

[1] Weske, M. – *Business Process Management: Concepts, Languages*, Architectures Springer-Verlag Berlin Heidelberg 2007

[2] Luckham, D. (2008, October). The power of events: An introduction to complex event processing in distributed enterprise systems. In *International Workshop on Rules and Rule Markup Languages for the Semantic Web* (pp. 3-3). Springer Berlin Heidelberg.

[3] Sosinsky, B. (2010). *Cloud computing bible* (Vol. 762). John Wiley & Sons.

[4] Leavitt, N. (2009). Complex-event processing poised for growth. *Computer*,*42*(4), 17-20.

[5] Havey, M. (2008). CEP and SOA: Six Letters Are Better than Three. *Feb-2009.[Online]. Available: http://www. packtpub. com/article/cep-complex-event-processing-soa-service-oriented-architecture.[Accessed: 31-Jan-2013]*.

[6] Kavelar, A., Obweger, H., Schiefer, J., & Suntinger, M. (2010, July). Web-based decision making for Complex Event Processing systems. In *2010 6th World Congress on Services* (pp. 453-458). IEEE..

[7] Etzion, O., & Niblett, P. (2010). *Event processing in action*. Manning Publications Co.

[8] Fülöp, L. J., Tóth, G., Rácz, R., Pánczél, J., Gergely, T., Beszédes, A., & Farkas, L. (2010, July). Survey on complex event processing and predictive analytics. In *Proceedings of the Fifth Balkan Conference in Informatics* (pp. 26-31).

[9] Paschke, A. (2008). Design patterns for complex event processing. *arXiv preprint arXiv:0806.1100*.

[10] Ammon, R. V., Silberbauer, C., & Wolff, C. (2007, October). Domain specific reference models for event patterns–for faster developing of business activity monitoring applications. In *VIP Symposia on Internet related research with elements of M+ I+ T+* (Vol. 16).

[11] *MsCoy, D. (2002). Business Activity Monitoring: Calm Before the Storm. Recuperado el 6 de agosto de 2012, de https://www.gartner.com/doc/354283/business-activity-monitoring-calm-storm*

[12] Adams, P. (2002). What's going on?[business activity monitoring].*Manufacturing Engineer*, *81*(6), 282-283.

[13] Reybaud, A. L., Moreno, J. J., Novales, R., & Aguilar, L. J. (2005). Business Activity Monitoring y Business Rules para el Manejo de Excepciones en las Políticas en un sistema de Gestión de Procesos de Negocios. Estado del-Arte. In *IV SIMPOSIO INTERNACIONAL DE SISTEMAS DE INFORMACIÓN* (p. 195).

[14] Gassman, B. (2004). How the Pieces in a BAM Architecture Work; Gartner.

[15] McCoy, D. (2004). The convergence of BPM and BAM. *Gartner Research Note (SPA-20-6074)*.

[16] Kolár, J. (2009). Business activity monitoring. *Unpublished Master Thesis. Masaryk University*.

[17] McCoy, D. (2004). The convergence of BPM and BAM. *Gartner Research Note (SPA-20-6074)*.

[18] Bustos Barrera, S. A., & Mosquera Artieda, V. N. (2013). Artículo Científico.-Análisis, diseño e implementación de una solución BUSINESS INTELLIGENCE para la generación de indicadores y control de desempeño, en la empresa OTECEL SA, utilizando la metodología HEFESTO V2. 0.

[19] DAVIS, Jeff. *Open source SOA*. Manning Publications Co., 2009.

[20] von Ammon, R., Emmersberger, C., Ertlmaier, T., Etzion, O., Paulus, T., & Springer, F. (2009, July). Existing and future standards for event-driven business process management. In *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems* (p. 24). ACM.

[21] Malik, S. (2005). *Enterprise dashboards: design and best practices for IT*. John Wiley & Sons.

[22] Few, S. (2006). Information dashboard design.

[23] Mauro Cardarelli, *Maximizing Your Portal Investment with Composite Applications*. URL: http://www.jornata.com/presentations/MaximizingYourPortalInvestment.pdf, 2007.

[24] Antoine, *Portal Server* http://antoine.eduangi.com, 2007.

[25] Juan C. García Gómez, *Los portales Web ante el reto de la generación de negocio en Internet* Servicio de Información Universitario, Universidad de Murcia. URL: http://www.um.es/gtiweb/juancar/curri/portales.htm, 2007

[26] Avila, G., Sánchez, A., de Zayas, L. M. R., Jiménez, P., & Basnuevo, M. (2010). El Portal Holguín una ventana al mundo por la red de redes. *Ciencias Holguín*, *10*(4).

[27] Luis Villa, *Servidores de portales: ¿usabilidad empaquetada?* URL: http://alzado.org, 2006.

[28] Carvajal Ramírez, C. A. (2009). *Tableros de control para un proceso del negocio en una compañía de seguros* (Doctoral dissertation, Universidad Nacional de Colombia).

[29] Bonitasoft BPM. URL: (*http://www.bonitasoft.com/*)

[30] WSO2. URL: (*http://wso2.com/)*

[31] WSO2 Business Activity Monitor (WSO2BAM). URL: (*http://wso2.com/products/business-activity-monitor/*)

[32] WSO2 Complex Event Processor (WSO2CEP). URL: (*http://wso2.com/products/complex-event-processor/)*

[33] Liferay Portal. URL: (*http://www.liferay.com/)*