

UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 6



**Mecanismo de comunicación en tiempo real para la
representación de indicadores claves de desempeño en
un tablero digital.**

**Tesis presentada en opción al título de
Máster en Informática Aplicada**

Autor:

Ing. Adonis Ricardo Rosales García

Tutora:

MSc. Yeleny Zulueta Veliz

La Habana, julio de 2011

A mi tutora Yeleny, por el apoyo, la confianza y guía certera.

A mi esposa Yasirys, por su ayuda y comprensión desde la distancia.

Al equipo del proyecto Dashboard.

A María y Francisco, mis padres.

A Yasirys, mi esposa.

Declaro por este medio que yo Adonis Ricardo Rosales García, con carné de identidad 85022320027, soy el autor principal del trabajo final de maestría: **Mecanismo de comunicación en tiempo real para la representación de indicadores claves de desempeño en un tablero digital**, desarrollada como parte de la Maestría en Informática Aplicada y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Y para que así conste, firmo la presente declaración jurada de autoría en La Habana a los ____ días del mes de _____ del año 2011.

Adonis Ricardo Rosales García

RESUMEN

Actualmente en el Centro de Tecnologías de Datos de la Universidad de las Ciencias Informáticas se desarrolla el proyecto Sistema de Información de Gobierno de conjunto con la Oficina Nacional de Estadísticas de la República de Cuba. En este se necesita un tablero digital para monitorear el comportamiento de diferentes indicadores claves de desempeño (KPI) en tiempo real. Esta necesidad motivó la presente investigación con el objetivo de diseñar un mecanismo que permite la comunicación entre los KPI y su representación en un tablero digital de manera que se reduzcan los tiempos de respuesta, sin la intervención del usuario y sobre la Web.

Para la definición del mecanismo, primeramente se realizó un análisis crítico acerca de los sistemas de comunicación en tiempo real, así como su aplicación a diferentes áreas. La definición del nuevo mecanismo de comunicación quedó integrada por los protocolos XMPP y BOSH como soporte al modelo de comunicación productor - distribuidor - consumidor; el servidor Ejabberd como servidor de mensajería, un servidor web y el componente implementado. La implementación se basó en una arquitectura en capas y estándares abiertos, que garantiza la soberanía tecnológica. La propuesta se validó a partir del desarrollo experimental de pruebas funcionales y la valoración de los resultados alcanzados. El mecanismo reduce los tiempos de respuesta en que los KPI son mostrados en un tablero digital y la actualización de los datos no depende de la intervención del usuario.

Palabras clave: Comunicación, tiempo real, tablero digital, XMPP.

ABSTRACT

The project Government Information System is developed by the Database Technology Center from the University of Information Sciences, in conjunction with the National Office of Statistics of the Republic of Cuba. This project requires a dashboard component in order to monitoring the behavior of different key performance indicators (KPI) in real time. This necessity motivated this investigation, which was developed to design a mechanism that enables communication between the KPI and dashboard representation, to reduce response times, without user intervention and also on the Web.

To define the mechanism, firstly was performed an analysis of real-time communication systems, and their application to different areas. The new communication mechanism was composed by the protocols XMPP and BOSH, to support the producer-distributor-consumer communication model; Ejabberd server as messaging server; a web server and a software component. The software component implementation was based on a layered architecture and open standards, which guarantees technological sovereignty. The proposal was validated using functional testing that demonstrated the achieved improvements on the performance and efficiency.

Key words: *Communication, real-time, dashboard, XMPP.*

TABLA DE CONTENIDOS

INTRODUCCIÓN	1
CAPÍTULO 1 FUNDAMENTOS TEÓRICOS DE LOS SISTEMAS DE COMUNICACIÓN EN TIEMPO REAL.....	7
1.1 Introducción	7
1.2 Tablero digital e indicadores claves de desempeño	7
1.3 Sistemas de tiempo real	9
1.4 Sistemas distribuidos de tiempo real	11
1.5 Comunicación en tiempo real	13
1.5.1 Modelo de comunicación	14
1.6 Paradigma publicación - suscripción	16
1.7 Jabber y Extensible Messaging and Presence Protocol	18
1.7.1 Arquitectura general del protocolo	19
1.7.2 XMPP sobre la Web.....	20
1.8 Herramientas de comunicación en tiempo real.....	22
1.9 Pentaho BI Suite 3.6 Community Edition.....	25
1.10 Algunas consideraciones sobre la importancia del diseño arquitectónico en los STR.....	27
1.11 Conclusiones	28
CAPÍTULO 2 UN NUEVO MECANISMO DE COMUNICACIÓN EN TIEMPO REAL.....	29
2.1 Introducción	29
2.2 Descripción de la solución propuesta	29
2.2.1 Modelo conceptual	31
2.2.2 Requerimientos	33
2.2.3 Modelo de Sistema	35
2.3 Arquitectura general del sistema	35
2.4 Implementación de la propuesta de solución.....	40
2.4.1 Modelo de despliegue.	43

2.5 Seguridad en XMPP	44
2.6 Conclusiones	45
CAPÍTULO 3 ANÁLISIS DE LOS RESULTADOS	47
3.1 Introducción	47
3.2 Validación de los resultados de implementación	47
3.2.1 Pruebas de integración y sistema	49
3.3 Valoración del aporte práctico de los resultados	53
3.4 Generalización de la propuesta	57
3.5 Conclusiones	58
CONCLUSIONES GENERALES.....	60
RECOMENDACIONES	61
REFERENCIAS BIBLIOGRÁFICAS.....	62
GLOSARIO DE TÉRMINOS.....	66
ANEXOS	67
Anexo 1. Requisitos Funcionales	67
Anexo 2. Diagrama del sistema.....	68

ÍNDICE DE FIGURAS

Figura 1. Representación de los sistemas de tiempo real.....	10
Figura 2. Clasificaciones abordadas de sistemas de tiempo real.....	11
Figura 3. Modelo cliente - servidor.	14
Figura 4. Modelo productor - consumidor.....	15
Figura 5. Paradigma PubSub.	16
Figura 6. Componentes del paradigma PubSub.....	17
Figura 7. Arquitectura de comunicaciones del protocolo XMPP [35].....	20
Figura 8. Descripción de la organización del sistema.	30
Figura 9. Modelo de dominio.	32
Figura 10. Agrupación en capas del sistema.....	35
Figura 11. Protocolo PHP-RPC.	37
Figura 12. Protocolo Comet.....	38
Figura 13. Funcionamiento del <i>long polling</i> usando BOSH escenario 1.....	39
Figura 14. Funcionamiento del <i>long polling</i> usando BOSH escenario 2.....	39
Figura 15. Diagrama de componentes.	40
Figura 16. Diagrama de actividades generales del proceso Publicar.....	42
Figura 17. Modelo de despliegue.	44
Figura 18. Especificación de pruebas aplicadas.	48
Figura 19. Rendimiento por conexiones concurrentes.	53
Figura 20. Comportamiento de los rendimientos.....	55
Figura 21. Diagrama de CUS.	68

ÍNDICE DE TABLAS

Tabla 1. Análisis de necesidades y herramientas estudiadas.	24
Tabla 2. Actores del sistema.....	34
Tabla 3. Rendimiento para crear nodos PubSub.	50
Tabla 4. Rendimiento para suscribir a nodo.	51
Tabla 5. Rendimiento para publicar datos.	52
Tabla 6. Comparación de rendimientos.	54
Tabla 7. Comparación cualitativa.....	56

INTRODUCCIÓN

En el entorno empresarial actual, tan altamente competitivo, donde se producen cambios frecuentes, las empresas deben dedicar importantes recursos humanos, financieros y tiempo a medir el trabajo que realizan para alcanzar sus objetivos estratégicos [1, 2]. El interés de los directivos ya no queda solamente en hacer más con menos, también se ha movido tanto a la atención física, cultural y profesional de sus trabajadores como al análisis de las necesidades e intereses de los clientes.

Medir el desempeño de una empresa en todos sus renglones siempre ha sido una tarea compleja. Para hacer valoraciones o arribar a conclusiones desde el punto de vista gerencial, hay que considerar numerosos aspectos. Los cuadros de mando integral (CMI) o *Balanced Scorecard* como se les conoce en inglés, devienen como elemento clave para la realización de la toma de decisiones de manera adecuada, organizada y precisa. El objetivo principal de un CMI es convertir la visión y la estrategia de una organización en metas, indicadores e iniciativas y que el desarrollo de dicha estrategia no se centre solamente en el marco financiero, sino además en un esquema integrado de seguimiento y mejora [3, 4]. El CMI permite organizar las decisiones más importantes de una empresa u organismo con relación a los Indicadores Claves de Desempeño (KPI del inglés *Key Performance Indicators*) [5, 6].

Utilizar sistemas informáticos que apoyen la realización efectiva de los CMI es una necesidad por las dimensiones de la información a tratar, el desarrollo tecnológico actual y el creciente intercambio de datos a través de la red.

A partir de la definición de los KPI como parte de los CMI, se insertan los tableros digitales (TD) o *Dashboard*, para garantizar su representación gráfica [7]. Esta representación, en tiempo real o no, constituye una de las vistas más integrales de la información dado que el escenario de factores que influyen en

las decisiones se mantiene visible y accesible para los decisores. Una de las características deseadas en un TD es el dinamismo para mostrar las informaciones sin que el usuario tenga que pedir una actualización del estado de los KPI, por la comodidad que representa.

El mercado de software cuenta con una gama amplia de sistemas con soluciones de CMI, estas soluciones forman parte de lo que se conoce como Inteligencia de Negocios [8]. Debido a la novedad e importancia de estas herramientas, en muchos casos son soluciones privativas de código cerrado, dentro de las cuales se pueden citar *Dashboard – Sixtina* [9] y *Bingo análisis* [10], que presentan buenas soluciones, pero están desarrolladas en la plataforma .NET de Microsoft y solo están disponibles para el sistema operativo Windows. Esto representa limitaciones en cuanto a soporte y precios, así como dependencia tecnológica.

Las alternativas libres más utilizadas no están acordes a las exigencias del mercado [11], debido a que en ocasiones muestran limitaciones en cuanto a la actualización de los datos tratados. Además, siempre existe el riesgo de que sean absorbidos por empresas como Oracle o IBM y se conviertan en soluciones privativas.

En el área de Inteligencia de Negocios, una de las empresas que se destaca por sus resultados es *Pentaho Business Intelligence* [12], pues su solución brinda herramientas que permiten hacer el análisis de los datos. Esta brinda dos soluciones: una *Enterprise* y otra *Community*. La solución *Enterprise* brinda un conjunto de funcionalidades avanzadas que facilitan el diseño de las tareas que puede realizar, pero es necesario comprarla. Por su parte, en la *Community*, las funcionalidades no facilitan el diseño y elaboración de las tareas requeridas. A pesar de ser una solución de código abierto, no cuenta con ningún tipo de soporte, por lo que se debe instalar, configurar y mantener el software por esfuerzo propio, lo que supone un esfuerzo extra para las organizaciones [13].

La Oficina Nacional de Estadísticas (ONE), es el órgano rector de la estadística en Cuba, su objetivo es captar, analizar y difundir los datos recogidos a lo largo

y ancho de todo el país para realizar una mejor toma de decisiones y medir los KPI de interés. Este organismo cuenta con una serie de modelos en los cuales recoge toda la información de los sectores de la economía y la sociedad, en aras de mejorar su gestión, actualmente en el Centro de Tecnologías de Datos (DATEC) de la Universidad de las Ciencias Informáticas (UCI) se desarrolla el proyecto Sistema de Información de Gobierno (SiGOB). Este proyecto se propone como una solución tecnológica que permita a través de una red centralizada y un sistema bien estructurado recopilar la información que necesita el Gobierno y los Órganos de Trabajo del Consejo de Defensa para su gestión. Además debe brindar una vista global de los aspectos relevantes a todos los niveles para que apoye a la dirección de los mismos en la toma de decisiones y para ello se necesita la presencia de un TD. Para responder a esta necesidad, se ha adoptado la solución que brinda *Pentaho BI Suite 3.6 Community Edition*. Esta versión no cuenta con un diseñador de TD, ni da soporte a la visualización de KPI en tiempo real, no existe una versión actualmente que de soporte a esta necesidad. Esto constituye un problema para la representación gráfica de la información, y requiere mayor esfuerzo para el diseño y comprensión de la misma. Cuando se habla de actualización en tiempo real, se refiere a aquellos KPI de los cuales se necesita conocer sus valores y monitorizar su comportamiento constantemente, sin la intervención del usuario. Esto daría a los decisores información actualizada de los procesos que se desarrollan en la empresa, y una acertada base sobre la cual tomar las decisiones.

A partir del análisis realizado en la situación anterior se plantea como **problema científico** de la investigación: ¿Cómo lograr una reducción de los tiempos de respuesta en la comunicación en tiempo real entre los KPI y su representación en un tablero digital?

Para la solución del problema planteado, la investigación se enfoca en el **objeto de estudio** sistemas informáticos de comunicación en tiempo real; y se precisa la comunicación en tiempo real para la representación de KPI en un tablero digital como **campo de acción**.

El **objetivo general** de la investigación es diseñar un mecanismo de comunicación en tiempo real para la representación de KPI en un tablero digital, de manera que se reduzcan los tiempos de respuesta. Este objetivo estará sustentado en los **objetivos específicos** siguientes:

1. Caracterizar los sistemas de tiempo real y los modelos de comunicación existentes.
2. Definir la arquitectura de un componente de comunicación en tiempo real.
3. Implementar el componente de comunicación en tiempo real.
4. Validar los resultados de implementación a partir de indicadores de calidad.

Se ha identificado como **hipótesis** de esta investigación:

Si se diseña un mecanismo que permita la comunicación entre los KPI y su representación en un tablero digital entonces se reducirán los tiempos de respuesta en que la información será mostrada.

La **novedad y aporte teórico** del presente trabajo consiste en la elaboración de un mecanismo que permita la comunicación en tiempo real para actualizar los KPI de un TD; además, la **novedad y aporte práctico** consiste en: (1) La obtención de un componente de comunicación en tiempo real entre aplicaciones, así como la implementación sobre estándares abiertos, garantizando la soberanía tecnológica y (2) la integración de un nuevo activo a la línea de producción de software del centro DATEC.

Para la presente investigación se utilizarán los siguientes métodos:

Métodos teóricos:

Histórico - Lógico: para estudiar la evolución y desarrollo de los sistemas informáticos de comunicación en tiempo real y comprender lógicamente cuales son las tendencias actuales de estos.

Analítico - Sintético: para realizar el análisis de la comunicación y actualización de datos a partir de la relación que existe entre los elementos que lo conforman; y a su vez, la síntesis se produce sobre la base de los resultados de este análisis. Se utiliza además para el procesamiento de la información y arribar a las conclusiones de la investigación.

Métodos empíricos:

Medición: como procedimiento para obtener información numérica acerca de algunas características de la solución obtenida, como por ejemplo su rendimiento.

Experimental: para realizar las pruebas en los niveles de Integración y Sistema de la solución, mediante el desarrollo de casos de prueba experimentales previamente diseñados.

Técnicas de recopilación de información:

Entrevista: se utiliza para obtener información en forma verbal, a través de preguntas que se proponen para conocer en detalle sobre la comunicación en tiempo real, las tecnologías libres existentes así como las herramientas que le dan soporte.

El trabajo se presenta está estructurado en tres capítulos:

El Capítulo 1 Fundamentos teóricos de los sistemas de comunicación en tiempo real tiene como objetivo abordar los diferentes aspectos teóricos y conceptuales que constituyen la base para el desarrollo de la propuesta de solución. Se valoran de forma crítica las tendencias y tecnologías actuales, así como los antecedentes asociados con los principales conceptos que se relacionan con el objeto de estudio y que son necesarios para entender posteriormente la propuesta de solución.

En el Capítulo 2 Un nuevo mecanismo de comunicación en tiempo real se realiza una descripción en la que se tiene en cuenta el análisis de un modelo conceptual que contiene las principales definiciones que se utilizan en la propuesta, así como la relación que se establecen entre ellas. Se especifican

los requerimientos funcionales que forman parte del mecanismo. Finalmente se describe su arquitectura, se realiza el diagrama de componentes, la representación de los principales procesos que se pueden realizar con el componente y el diagrama de despliegue.

El Capítulo 3 Análisis de los resultados aborda una valoración crítica tanto desde un enfoque cuantitativo como cualitativo de los resultados obtenidos, a partir del desarrollo de un conjunto de pruebas experimentales realizadas a los resultados de implementación obtenidos. A partir del registro de los tiempos de respuesta del componente, se hace un análisis sobre su efectividad.

CAPÍTULO

1

FUNDAMENTOS TEÓRICOS DE LOS SISTEMAS DE COMUNICACIÓN EN TIEMPO REAL

1.1 Introducción

Este capítulo tiene como objetivo abordar los diferentes elementos que brindan la base teórico conceptual para el desarrollo de la solución propuesta. Se hace una valoración crítica sobre los modelos de comunicación y tecnologías actuales en cuanto a la comunicación en tiempo real. Se realiza el estudio de algunas herramientas que utilizan comunicación en tiempo real sobre la base de sus características y se analiza el impacto del diseño. Estos elementos son necesarios para entender posteriormente la propuesta de solución.

1.2 Tablero digital e indicadores claves de desempeño

El CMI, es una metodología que permite transmitir las estrategias definidas por una organización de una manera clara y eficiente a todos los integrantes de la misma y a la vez, poder traducir dichas estrategias en objetivos, acciones y medidas concretas, que permitan saber si las mismas se están alcanzando [4, 5, 14].

Una de las herramientas que se encuentra incluida dentro de un CMI lo constituye el TD. Esta se puede definir como una herramienta de visualización gráfica de la información más importante para alcanzar uno o mas objetivos;

organizada en una sola pantalla en la que la información puede ser monitorizada de un solo vistazo [7]. La información se refiere a los diferentes KPI definidos por la organización.

De manera general, un KPI es una medida que ofrece una estimación o evaluación de determinados atributos derivados de un modelo definido con respecto a las necesidades de información. Los KPI son la base para el análisis y la toma de decisiones. Se trata de lo que debería ser presentado a los usuarios de la medición. La medición se basa siempre en una información imperfecta, por tanto cuantificar la incertidumbre, la precisión, o la importancia de los indicadores es un componente esencial de la presentación del valor real del KPI [15].

La base para la representación de un TD son los KPI y las métricas, pues son las herramientas más eficaces para ubicar al usuario en cuanto a:

- Cómo se encuentran con respecto a los objetivos.
- Describir los resultados de rendimiento.
- Proporcionar una visión instantánea de la estrategia organizativa.
- Dar lugar a cambios en el comportamiento organizativo.

Las métricas son una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado [16]. Representan un conjunto de datos del negocio, es decir aquellos conceptos cuantificables que permiten medir el proceso de negocio [17]. A partir de una evaluación de las métricas se pueden obtener los KPI, que serían los resultados a presentarse del proceso.

Los KPI son indicadores que están vinculados a un objetivo de negocio. Definen mediciones que determinan qué tan bien se está desempeñando el proceso de negocio para alcanzar la meta: si está por encima o por debajo de una meta predeterminada [18]. Generalmente se muestran como una tasa o porcentaje y están diseñados para permitir que un usuario de negocios pueda saber instantáneamente si están dentro o fuera de su plan sin que tenga que buscar información adicional.

A continuación se mencionan las características de un correcto KPI [19, 20]:

- Es una métrica no financiera.
- Debe mostrar el objetivo de la organización.
- Debe ser monitoreado de forma constante.
- Debe estar basado en datos reales.
- Debe ser atendido por la dirección de la empresa.
- Debe ser lo suficientemente claro como para identificar perfectamente al responsable del mismo.
- Dice de forma precisa qué acciones deben ejecutarse.

Las métricas y KPI dependen de datos consistentes, pues el resultado que aportan es esencial para determinar una estrategia [19]. Cada empresa posee sus propias métricas y KPI que apoyan el cumplimiento de un objetivo, por lo que se hace necesario que estén en correspondencia con las metas estratégicas de la organización.

Los KPI más utilizados en cada organización le ayudan a estas a determinar si se están manejando acertadamente los recursos y costos, contribuyendo a que la gerencia tenga una noción clara de lo que acontece en un momento específico para tomar medidas correctivas oportunamente. En este sentido, se hace necesaria la comunicación de los KPI hacia el TD en tiempo real.

1.3 Sistemas de tiempo real

Los sistemas de tiempo real (STR) son sistemas computacionales que están fuertemente ligados a un entorno que cambia con el tiempo. Por esta causa, en los sistemas de tiempo real el funcionamiento correcto no sólo depende de los resultados del cálculo, sino también del instante de tiempo en el que estos cálculos son finalizados.

Básicamente los sistemas de tiempo real se definen como sistemas informáticos que tienen la capacidad de interactuar rápidamente con su entorno

físico, el cual puede realizar funciones de supervisión o control para su propio beneficio [21].

Todos los STR tienen la facultad de ejecutar actividades o tareas en intervalos de tiempo bien definidos. Todas las tareas son ejecutadas inmediatamente en una forma concurrente, esto es para sincronizar el funcionamiento del sistema con la simultaneidad de acciones que se presentan en el mundo físico.

Los intervalos de tiempo en que se ejecutan las tareas se definen por un esquema de activación y por plazos de ejecución. En lo que respecta al esquema de activación puede ser periódico, es decir, en intervalos regulares, o también puede ser aperiódico, es decir, en respuesta a sucesos externos que ocurren de forma irregular [21].

La mayoría de los STR son utilizados cuando existen requerimientos de tiempo muy rígidos en las operaciones o en el flujo de datos, generalmente son requeridos como sistemas de control en una aplicación dedicada. La predictibilidad es la característica principal de este tipo de sistemas.

Este tipo de sistemas se caracterizan por tener que producir una salida, como respuesta a una entrada, en un tiempo determinado, esto se representa en la figura 1. El intervalo de tiempo que se presenta entre la entrada y la salida debe ser muy pequeño para que la respuesta temporal del sistema sea aceptable.

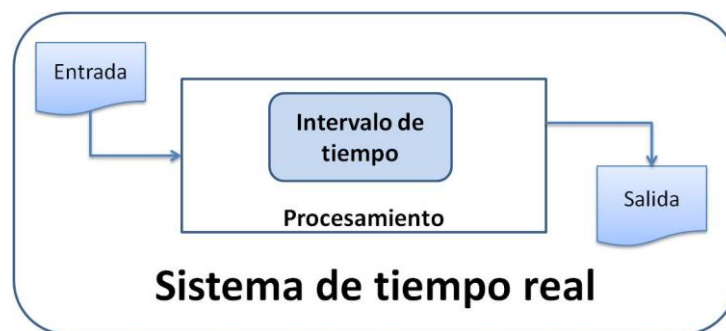


Figura 1. Representación de los sistemas de tiempo real.

A continuación se presenta un mapa conceptual sobre las clasificaciones de los sistemas de tiempo real, a partir del cual se guía la presente investigación.

Como se puede observar en la figura 2, solo se abordarán los conceptos involucrados en la definición del mecanismo de comunicación que se elabora.

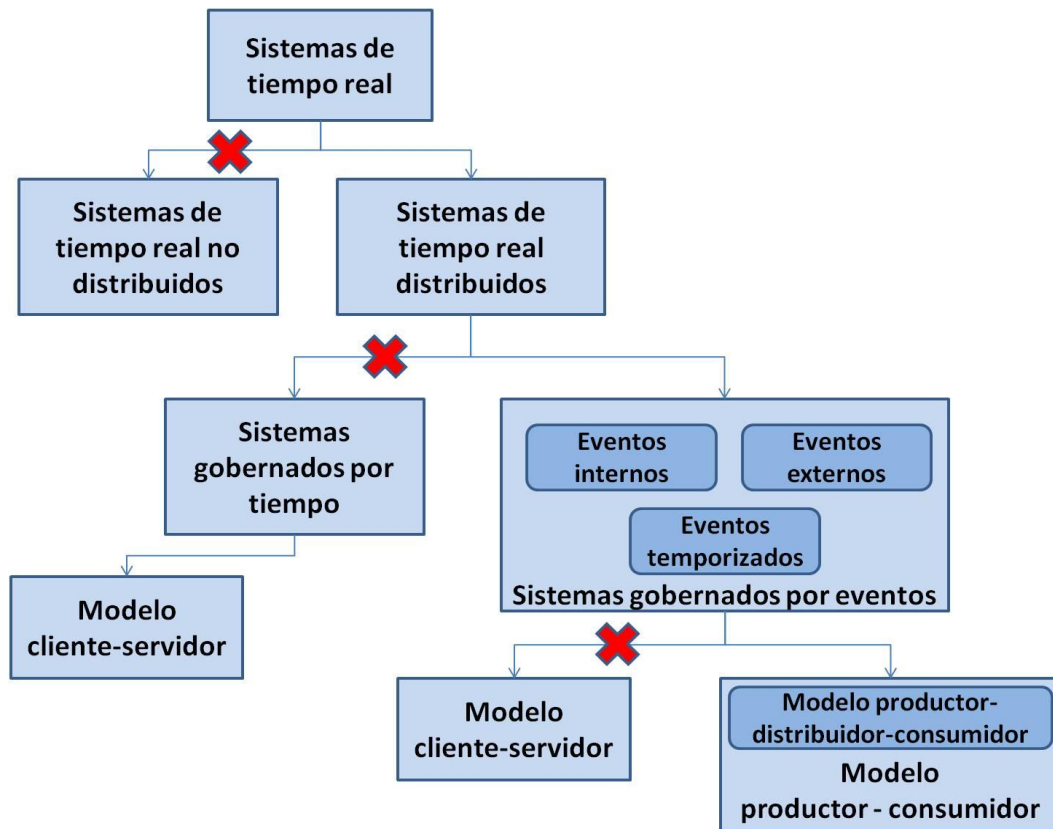


Figura 2. Clasificaciones abordadas de sistemas de tiempo real.

1.4 Sistemas distribuidos de tiempo real

Se considera un sistema distribuido de tiempo real a una colección de ordenadores autónomos conectados por una red de computación y equipados con software distribuido. El software distribuido permite a los computadores coordinar sus actividades y compartir los recursos del sistema - hardware, software y datos. Los usuarios de un sistema distribuido bien diseñado deben percibir una facilidad de computación simple e integrada aunque pueda estar implementada por muchos computadores en diferentes localizaciones [22].

Un sistema de tiempo real distribuido será, entonces, un sistema distribuido con restricciones temporales, es decir, un sistema distribuido donde existen actividades con requisitos de tiempo real. Para cumplir con dichos requisitos temporales, estas actividades además de necesitar ejecutar sus tareas en una

unidad de procesamiento, pueden, ocasionalmente, necesitar intercambiar datos con otras tareas ubicadas en otras unidades. Para que el sistema completo cumpla sus restricciones temporales, tendrá que ser capaz tanto de realizar el procesamiento de dichas tareas en cada procesador, como de intercambiar datos entre las distintas entidades, todo de manera determinista, ajustándose a las cotas temporales impuestas para cada una de las actividades de tiempo real existentes en el sistema [23]. Los sistemas multiprocesadores y distribuidos de tiempo real tienen una importancia creciente en los sistemas de control de hoy día, sobre todo, debido a que el bajo costo de las redes de comunicación permite la interconexión de múltiples dispositivos y de sus controladores en un gran sistema.

La arquitectura típica de un sistema de control distribuido de tiempo real consta de varios nodos procesadores interconectados a través de una o más redes de comunicación [24]. El software del sistema se compone de varias tareas concurrentes que normalmente están localizadas de forma estática en los nodos procesadores, ya que los efectos de una localización dinámica son muy difíciles de predecir para sistemas de tiempo real estricto. Normalmente, la comunicación entre las tareas del sistema (dentro del propio nodo o entre diferentes nodos) se realiza a través del intercambio de mensajes. También es normal que los mensajes intercambiados entre tareas de diferentes nodos se asignen de forma estática a las redes de comunicación. Además, existe la posibilidad de que las tareas de un mismo nodo compartan datos o recursos mediante los mecanismos de sincronización habitualmente utilizados en sistemas de memoria compartida [25].

Los sistemas distribuidos de tiempo real se pueden clasificar en dos grupos [26]:

- Sistemas gobernados por eventos: las acciones son activadas por la ocurrencia de eventos, que caracterizan el flujo de control del programa.
- Sistemas gobernados por tiempo: las acciones son activadas por el sistema en instantes de tiempo predeterminados, habitualmente de forma

cíclica a intervalos de tiempo regulares. La acción del reloj es, en estos casos, el único evento que activa las operaciones en el sistema.

Por otra parte, en un sistema distribuido se deben caracterizar los eventos que disparan la ejecución de las acciones. Una clasificación comúnmente aceptada es la basada en la fuente que origina dicho evento:

- Eventos temporizados: producidos por temporizadores o relojes propios del sistema.
- Eventos externos: originados en el sistema físico que se está controlando y que desencadenan una serie de acciones en el sistema como respuesta a ese evento.
- Eventos internos: producidos dentro del sistema, necesarios para la correcta sincronización del mismo.

Una vía común para la activación de las tareas del sistema distribuido puede ser bien la llegada de un evento de disparo, que puede ser generado por un dispositivo externo del entorno de la implementación (como un temporizador), o bien la llegada de un mensaje enviado por otra tarea del sistema. Muchos de estos eventos tienen asociados ciertos requisitos temporales como el plazo de principio-a-fin, que es el máximo tiempo de respuesta permitido desde la llegada del evento hasta el final de la ejecución de su respuesta. En la ejecución de la respuesta a un evento se pueden ver implicadas varias tareas localizadas en diferentes nodos que se intercambian mensajes a través de las redes del sistema distribuido [11].

1.5 Comunicación en tiempo real

Para el correcto funcionamiento de un STR, no solo es necesario que las tareas que se ejecutan en cada procesador lo hagan dentro de su plazo, sino que la información intercambiada entre los nodos cumpla con los requisitos de tiempo previamente especificados.

1.5.1 Modelo de comunicación

En un sistema distribuido, como ya se dijo anteriormente, los nodos cooperan entre sí para alcanzar un fin común. Para esto deben intercambiar datos entre ellos. La misma información se puede necesitar en distintos nodos o los nodos pueden necesitar datos de más de un nodo; así, la comunicación puede ser uno a uno, uno a muchos, muchos a uno, o muchos a muchos. La manera en la que la información se distribuye por la red se denomina modelo de comunicación, existiendo dos modelos básicos: cliente-servidor y productor-consumidor, así como una extensión de este último.

Modelo cliente - servidor: en este modelo, todo nodo que contiene información relevante es un servidor, como se muestra en la figura 3 y los nodos que necesitan dicha información son los clientes. La comunicación se inicia en el cliente, realizando una petición, a la que el servidor responde. Así, la comunicación es uno a uno. Un ejemplo típico son las transacciones en una base de datos [27].

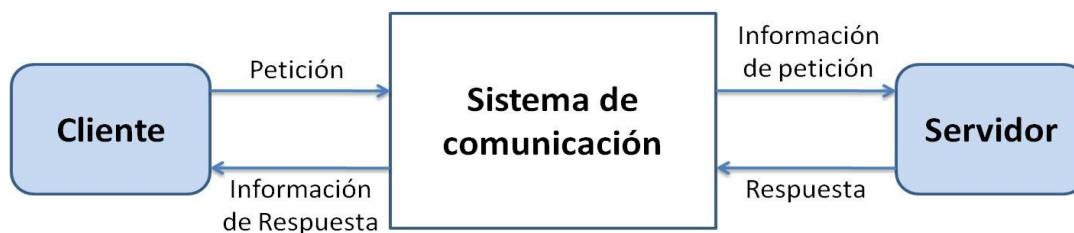


Figura 3. Modelo cliente - servidor.

Las extensiones a este modelo, en sistemas distribuidos, pueden llevar a inconsistencias en los datos. Por ejemplo, en sistemas cliente-servidor uno a muchos, con un servidor y varios clientes, el servidor ha de secuenciar las respuestas a los distintos clientes y si, antes de terminar, los datos que envía cambian, se producen inconsistencias espaciales entre los distintos datos relativos a la entidad que guardan los clientes. Por otra parte, si un cliente necesita datos de múltiples servidores, deberá realizar secuencialmente las peticiones, pudiéndose producir inconsistencias temporales.

Modelo productor - consumidor: en este modelo, todo nodo que contiene información relevante es un productor, y aquellos que necesitan dicha información son consumidores. Los datos no se discriminan por la fuente o destino de los mismos, sino por un identificador lógico que llevan asociado. El productor de información pone el dato a disposición de los consumidores, como se puede observar en la figura 4, quienes atendiendo al identificador lo procesan. Los productores no necesitan saber quiénes ni cuántos son los consumidores, ni viceversa. Así, en este modelo se soporta, inherentemente, un aislamiento entre productores y consumidores. Las transacciones son iniciadas en el productor [27].



Figura 4. Modelo productor - consumidor.

Este modelo soporta comunicaciones uno a uno y uno a muchos, siendo necesario para éstas últimas, sólo que la red de comunicaciones ofrezca el envío de mensajes *broadcast* o *multicast* de forma atómica. Además, en este caso, no se producen inconsistencias espaciales en los datos, ya que la misma información llega a todos los nodos consumidores de la misma entidad. En caso de que la red no ofrezca dicha facilidad, sí pueden producirse inconsistencias espaciales. Además, en este modelo pueden producirse inconsistencias temporales si un consumidor necesita datos de múltiples productores. La existencia de múltiples productores puede provocar que en el proceso de contienda por el medio, algunos mensajes se retrasen, siendo enviados más tarde cuando ya no son válidos.

Modelo productor - distribuidor - consumidor: variante del modelo anterior en la que se introduce sincronización entre productores y consumidores,

eliminando las inconsistencias temporales. En este modelo se introduce un nuevo nodo que ejerce de coordinador, llamado maestro. Los nodos consumidores y productores se llaman esclavos. En este modelo, todas las restricciones temporales y propiedades de los mensajes son conocidas por el maestro, quien planifica cuándo han de enviarse cada uno de ellos, informando dicha planificación a los productores, quienes sólo enviarán los mensajes cuando estén autorizados [28].

1.6 Paradigma publicación - suscripción

Los sistemas de publicación - suscripción (PubSub) están en todas partes: los periódicos, blogs, la televisión e incluso las listas de correo electrónico [29].

El paradigma de PubSub es una implementación del modelo productor - distribuidor - consumidor. Se trata del envío de mensajes asíncrono mediante el cual los usuarios que publican información (productores) no la envían directamente a los usuarios que solicitan esta información (consumidores), sino que lo hacen mediante un mediador [30].

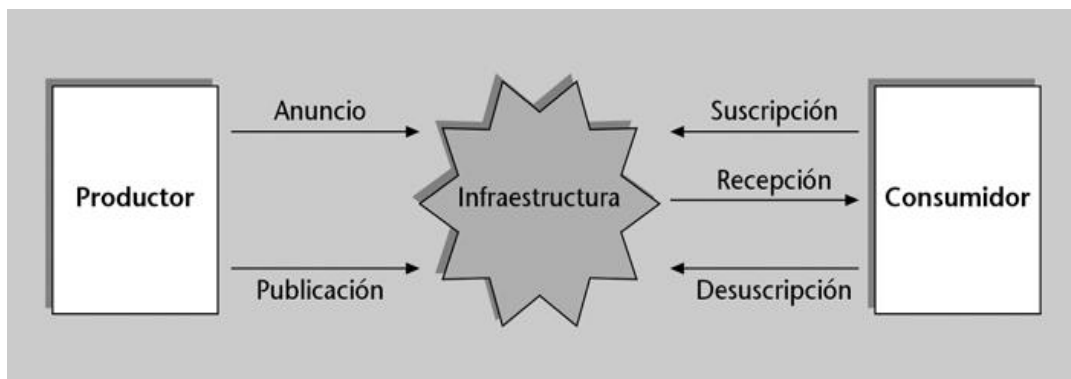


Figura 5. Paradigma PubSub.

La manera en que el paradigma PubSub funciona es haciendo que un productor de la información anuncie la disponibilidad de esta en un canal. El consumidor interesado se debe suscribir a este canal y será avisado cada vez que el productor publique nueva información, como se muestra en la figura 5.

En la figura 6 se muestran los componentes necesarios para que este sistema funcione, los cuales son:

- **Productor de información:** Usuario que tiene la información a difundir. El productor publica la información en el mediador sin tener conocimiento alguno de los usuarios interesados en esa información.
- **Consumidor de la información:** Usuario interesado en recibir información, se suscribe a los temas en los que está interesado y cuando el productor publica información, recibe una notificación indicando que hay nueva información disponible.
- **Mediador:** Está situado entre el productor y el consumidor, y se encarga de recibir la información de los productores y las peticiones de suscripción de los consumidores. Además, se encarga de enviar a los consumidores las notificaciones precisas cuando un productor publica nuevos contenidos.
- **Canal:** Se trata de los conectores lógicos entre el productor y el consumidor. El canal determina algunas de las propiedades de la información (tipo de información, formato). Además, gestiona la forma como se distribuyen los contenidos, si la información expira o es persistente, o si los datos se entregan en el momento de la publicación, o por el contrario el consumidor puede solicitarlos cuando quiera, independientemente del momento en que se generaron.

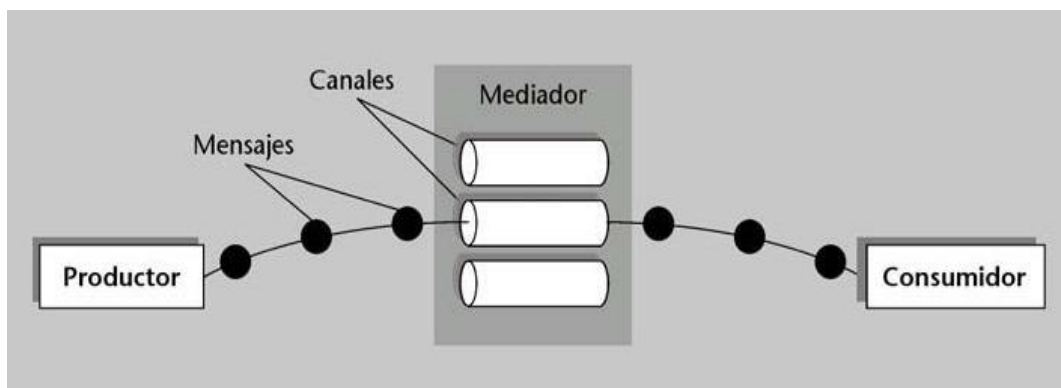


Figura 6. Componentes del paradigma PubSub.

1.7 Jabber y *Extensible Messaging and Presence Protocol*

Si Internet se ha convertido en la revolución de los canales de comunicaciones mundiales, ha sido precisamente por su carácter abierto, estándar y accesible a las tecnologías. El no haber sido desarrollado con carácter propietario ha hecho que Internet evolucione como un ecosistema en el que todos tienen la posibilidad de participar [31]. En el concepto de la mensajería instantánea (MI) ocurre lo mismo. Así nació Jabber, el primer protocolo de MI con carácter abierto.

Extensible Messaging and Presence Protocol (XMPP) es un protocolo abierto basado en el intercambio de datos XML para comunicación en tiempo real, el cual sirve de base para un amplio rango de aplicaciones, incluyendo MI, información de presencia (listas de contactos), charlas con múltiples participantes, voz, video llamadas y colaboración. El protocolo XMPP es el núcleo utilizado por la tecnología Jabber de mensajería instantánea y presencia [32].

Generalmente, XMPP se implementa y se usa como una arquitectura cliente-servidor, pero no necesariamente debe ser así, puede emplearse para establecer una comunicación directa, de extremo a extremo (P2P), entre los clientes. Como su propio nombre indica, también fue diseñado para ser extensible, por lo que se le han añadido nuevas características a lo largo del tiempo. Una de las principales ventajas de este protocolo es que, a diferencia de otros protocolos de mensajería, se trata de un estándar libre. Gracias a esto, y a que es un sistema abierto como el e-mail, cualquier persona que disponga de un dominio puede montar su propio servidor XMPP y contactar con otros usuarios de otros servidores. Dentro de las principales características del protocolo se puede mencionar que [33, 34]:

- Es libre: Jabber es Libre porque no solo se puede ver cómo funciona, sino además el usuario tiene la libertad de implementarlo él mismo, la libertad de adaptarlo a sus necesidades, sin necesitar la aprobación de nadie.

- Es extensible: usando el potencial del lenguaje XML, cualquiera puede extender el protocolo de Jabber para una funcionalidad personalizada.
- Es descentralizado: cualquiera puede montar su propio servidor de Jabber, además está libre de patentes y no depende de ninguna empresa de modo que se puede usar ahora y siempre con total libertad.

La especificación base de Jabber fue inventada en 1998 por Jeremie Miller y tomada como protocolo por la comunidad *open-source* en 1999, donde ha ido creciendo y evolucionando hasta hoy.

Hasta la fecha, Jabber y XMPP han sido los proyectos más aceptados como alternativas libres serias al sistema *MSN Messenger* de Microsoft, al *Yahoo! Messenger* y al ICQ. Y aunque XMPP todavía es un protocolo no muy conocido fuera del mundo de la mensajería instantánea, está creciendo más cada día.

1.7.1 Arquitectura general del protocolo

Generalmente, XMPP se implementa y se usa como una arquitectura cliente/servidor, pero el protocolo no fuerza a hacerlo así. Se puede emplear XMPP para establecer una comunicación directa P2P, entre los clientes.

En cuanto a los protocolos de transporte que sustentan la comunicación en XMPP, normalmente se tienen las conexiones puras de TCP, aunque podrían emplearse otros protocolos como *HTTP Polling* o *HTTP Binding*, que se basan en usos muy particulares del protocolo de aplicación HTTP.

El principal rol en XMPP consiste en gestionar las conexiones y los envíos de paquetes XML hacia los clientes y hacia otros servidores. Los servidores también tienen capacidad de enrutado de mensajes entre ellos. De hecho, el *Internet Assigned Numbers Authority* (IANA) tiene reservado el puerto TCP número 5269 a efectos de este tipo de tráfico XMPP entre servidores.

En el ejemplo que aparece en la figura 7 se puede apreciar como el servidor No. 1 es capaz de enrutar mensajes dirigidos a los clientes pertenecientes al dominio del servidor No. 2 o al dominio del servidor 3.

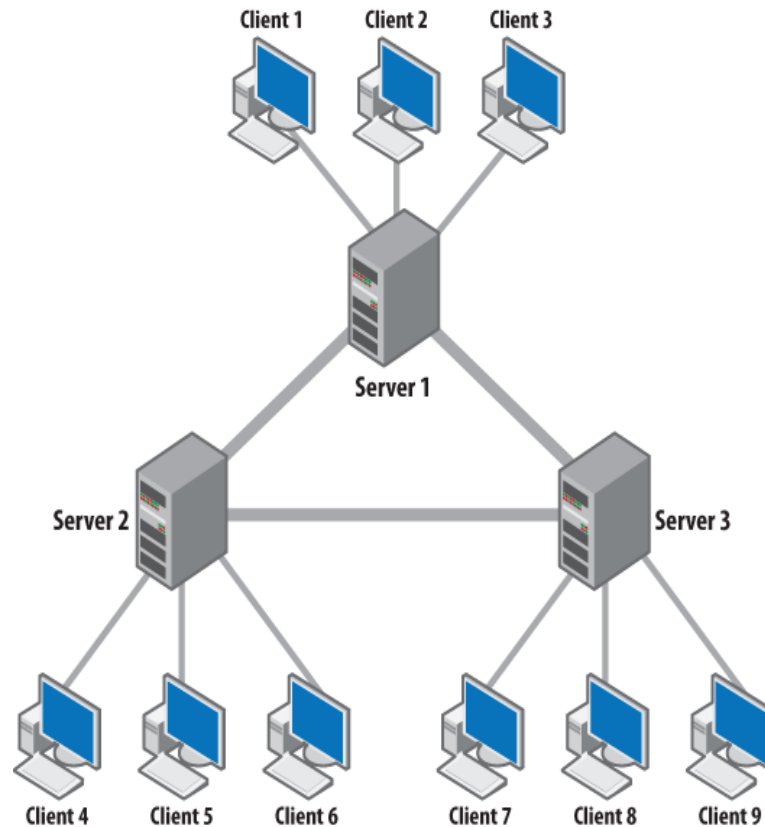


Figura 7. Arquitectura de comunicaciones del protocolo XMPP [35].

Los clientes XMPP inician la conexión contra un servidor u otro cliente y para ello usan el puerto reservado TCP número 5222.

1.7.2 XMPP sobre la Web

A pesar de que varios navegadores están experimentando con características que utilizan XMPP, ninguno de los principales navegadores actualmente proporciona soporte integrado para el protocolo XMPP. Sin embargo, se puede lograr establecer sesiones XMPP a través de conexiones HTTP de manera eficiente y eficaz.

La tecnología que permite este funcionamiento es llamada *HTTP long polling*. Con un protocolo de manejo simple basado en HTTP y un administrador de conexión XMPP, es posible llevar XMPP, y todas sus ventajas, a las aplicaciones HTTP [29].

Long polling

El principio del *long polling* es que cuando se haga una petición al servidor en busca de nuevos datos, en vez de terminar la conexión inmediatamente con una respuesta vacía, la conexión se mantenga abierta por un período de tiempo.

Para ilustrar mejor su funcionamiento, se muestran los casos posibles:

- Si hay nuevos datos para ser enviados: el servidor los envía inmediatamente al cliente.
- Si no hay nuevos datos para ser enviados: el servidor mantiene la conexión abierta. Cuando llegan nuevos datos el servidor responde la petición.
- Si no llegan datos en un período de tiempo definido: el servidor puede enviar una respuesta vacía, así no mantiene muchas conexiones abiertas al mismo tiempo. Una vez que el cliente recibe esta respuesta vacía, envía inmediatamente otra petición, y comienza nuevamente todo el proceso.

Como cada petición estará, potencialmente, abierta por un largo período de tiempo, esta técnica se llama *long polling*. La técnica del *long polling* tiene al menos dos ventajas sobre las peticiones normales:

- El cliente recibe los datos en el instante en que están disponibles (tiempo real).
- Se disminuyen las peticiones al servidor (menor consumo de ancho de banda).

El único cambio real que ocurre es que en lugar de ser el cliente el que espera cierto tiempo para enviar una nueva petición, es el servidor quien debe esperar a que haya nuevos datos que enviar al cliente. Múltiples librerías y protocolos han sido diseñados para implementar *long polling*, pero XMPP es una de las implementaciones más robustas y experimentadas. En XMPP el puente de comunicación con el protocolo HTTP es BOSH [36] (*Bidirectional streams Over*

Synchronous HTTP). Existen otros protocolos como *Comet* o *Reverse HTTP* que han implementado esta tecnología. Un inconveniente de este es que el servidor tiene que ser más inteligente para tratar con el *long polling* [29].

1.8 Herramientas de comunicación en tiempo real

Clientes de mensajería instantánea:

La mensajería instantánea requiere el uso de un cliente de mensajería instantánea que realiza el servicio y se diferencia del correo electrónico en que las conversaciones se realizan en tiempo real [37].

Los sistemas de mensajería tienen unas funciones básicas aparte de mostrar los usuarios que hay conectados y chatear. Unas son comunes a todos o casi todos los clientes o protocolos y otras son menos comunes entre estas se pueden citar:

- Registrar y borrar usuarios de la lista de contactos propia.
- Mostrar varios estados como: son disponibles, ocupado, lejos del computador, invisible.
- Se puede usar un avatar: una imagen o fotografía tipo icono.
- Compartir recurso: envío de archivos.

Los sistemas de mensajería instantáneas son usados muy frecuentemente en el entorno empresarial relacionado con las TIC. Constituye una herramienta para la comunicación y el trabajo colaborativo. Entre los diversos clientes que pueden ser usados, se encuentran: Pidgin, PSI, Neos y Pandion.

Google Wave:

Google Wave es una herramienta en línea que permite a sus usuarios comunicarse y colaborar en tiempo real. La comunicación es a partir de conversaciones en línea, llamadas waves (ondas). Una wave consiste en un documento XML y soporta modificaciones concurrentes y proporciona una baja latencia de actualización entre los participantes de una wave [38]. El proyecto fue anunciado por Google en la conferencia Google I/O, el 28 de mayo de 2009.

Esta aplicación permite unir los servicios de *e-mail*, MI, wiki, y redes sociales. Se enfoca fuertemente en el aspecto colaborativo.

Este protocolo utiliza como base al protocolo XMPP para la comunicación. Está escrito en Java usando OpenJDK y para su interfaz utiliza Google Web Toolkit [39].

El 4 de agosto de 2010, Google anunció que abandona el desarrollo de *Wave*, aduciendo su falta de acogida. Esta herramienta al dejar de tener soporte por sus creadores, se convierte en inestable [40].

Sistemas de supervisión, control y adquisición de datos

Un sistema de supervisión, control y adquisición de datos (SCADA, del inglés *Supervisory Control and Data Acquisition*) se define como una aplicación especialmente diseñada para controlar a través de un ordenador y con dispositivos de campo, las operaciones de control, supervisión y registro de datos de cualquier proceso industrial gobernado por autómatas programables o redes de autómatas [41].

Los sistemas SCADA utilizan la computadora y tecnologías de comunicación para automatizar el monitoreo y control de procesos industriales. Estos sistemas son partes integrales de la mayoría de los ambientes industriales complejos o muy geográficamente dispersos ya que pueden recoger la información de una gran cantidad de fuentes rápidamente, y la presentan a un operador en una forma amigable [41, 42].

Dentro de las principales funciones de un sistema SCADA se encuentran [42]:

- Monitorear: representando los datos con severas restricciones de tiempo de modo que permita a los operadores de la planta el conocimiento del estado de los procesos.
- Supervisar: el mando y la adquisición de datos de un proceso y herramientas de gestión para la toma de decisiones. Tiene además la capacidad de ejecutar programas que puedan supervisar y modificar el

control establecido y, bajo ciertas condiciones, anular o modificar tareas asociadas a los autómatas. Evita una continua supervisión humana.

- Visualizar: mostrando los estados de las señales del sistema (alarmas y eventos) permite el reconocimiento de eventos excepcionales acaecidos en la planta y su inmediata puesta en conocimiento de los operarios para efectuar las acciones correctoras pertinentes.

Ni los clientes de mensajería instantánea, ni *Google Wave*, constituyen opciones para resolver las necesidades plasmados en la investigación, como se muestra en la tabla 1. Sin embargo, pueden utilizarse parcialmente sus principios para obtener una solución que responda al problema planteado. En cuanto a los sistemas SCADA, la solución se acerca más a las necesidades, pero existen incompatibilidades en cuanto al diseño, pues estos sistemas controlan procesos industriales, y monitorean variables físicas del entorno, a través de sensores y dispositivos automáticos. Los sensores convierten las variaciones del fenómeno físico en variaciones proporcionales de una variable eléctrica. Las variables eléctricas más utilizadas son: voltaje, corriente, carga, resistencia o capacitancia.

Tabla 1. Análisis de necesidades y herramientas estudiadas.

Necesidad	Soluciones analizadas
Solución en un entorno Web para intercambio de datos referentes a KPI en tiempo real.	Clientes de mensajería instantánea: Se hace intercambio de mensajes entre usuarios, y a pesar de contar con clientes desarrollados en entornos web, no permiten la integración con un TD.
	<i>Google Wave</i> : Pensado para el trabajo en grupo que pretende unificar en una herramienta varias tecnologías como son el correo electrónico, la mensajería instantánea y los editores colaborativos. Se necesita obligatoriamente una cuenta en el servidor

	de google.
	SCADA: Se centra en la recolección de datos a partir de sensores, haciendo transformaciones de los fenómenos físicos para su interpretación eléctrica. Monitorea procesos. La mayoría de las soluciones son implementadas en entornos de escritorio.
Representación gráfica.	Clientes de mensajería instantánea: No posee.
	<i>Google Wave</i> : Asociada a los servicios que integra.
	SCADA: Visualiza el comportamiento de los procesos, con sistemas de gráficos, semáforos y alarmas. Permite la toma de decisiones.
Simplicidad y configuración.	Clientes de mensajería instantánea: Muy fácil de utilizar. Solo son necesarias algunas configuraciones básicas.
	<i>Google Wave</i> : Como resultado del abandono de esta tecnología por parte de sus creadores, deja de ser sostenible en el tiempo, lo que complica las expectativas para su uso.
	SCADA: Necesita configuraciones muy específicas. La implementación no es trivial.

1.9 Pentaho BI Suite 3.6 Community Edition

Pentaho BI Suite 3.6 Community Edition extiende las capacidades de inteligencia de negocios de código abierto y es la opción sin licencia comercial a la *Pentaho BI Suite 3.6 Enterprise Edition*. Incluye nuevas características que mejoran la simplicidad para diseñar reportes diagramados a nivel de píxeles,

gráficos de autoservicio y grillas de datos para CMI. Esta suite de herramientas de informes de código abierto permite crear informes relacionales y de análisis de una amplia gama de fuentes de datos y tipos de salida, incluyendo: PDF, Excel y HTML. La arquitectura abierta y puntos de gran alcance y extensión de la API aseguran que este sistema puede crecer con sus necesidades. Incluye los siguientes componentes [43]:

- Editor de metadatos: Es la herramienta que permite la administración y gestión de la capa de metadatos centralizados de la solución.
- Diseñador de reportes: Es una herramienta para generar todo tipo de documentos empresariales e informes de formato complejo, posee una interfaz unificada que permite un acceso rápido a los objetos de reportes de uso común, maximizando el área de diseño y una buena flexibilidad para la visualización de la información. Permite acceder a datos de múltiples fuentes y mostrarlos en cualquier parte del reporte.
- Marco de trabajo de esquemas: Esta herramienta permite modelar esquemas multidimensionales.
- Integración de datos: La herramienta de extracción, transformación y carga (ETL) que permite acceder a fuentes de datos y prepararlos para su análisis, minería de datos o informes. Es en general, dónde se debe comenzar si se quiere preparar los datos para su análisis.

En esta solución, a diferencia de la edición *Enterprise*, no se cuenta con un diseñador de TD, y para conseguir estas implementaciones es necesario hacer modificaciones al código. Las soluciones obtenidas a partir de esto, tienen la característica de no soportar la actualización en tiempo real. Podría implementarse a partir de su modelo arquitectónico, un método de *polling*, pero esto sería una implementación ineficiente en cuanto a utilización de recursos se refiere.

1.10 Algunas consideraciones sobre la importancia del diseño arquitectónico en los STR

Todo software lleva implícita una arquitectura, generalmente similar a otro(s) software ya sea en estilo general de sus subsistemas, módulos y capas o en la especificación de los patrones de diseño e implementación de los componentes finales. Al igual que la especificación de la arquitectura de una edificación, la arquitectura de software da confianza a los diseñadores sobre el buen camino que recorrerá el desarrollo, debido a que pueden corroborar con el cliente si los requerimientos se tienen en cuenta, así como comparar con otras herramientas de conocido prestigio que utilizan arquitectura similar [44].

El diseño arquitectónico se realiza a inicios del proceso de diseño del software y facilita que los requerimientos funcionales y no funcionales estén representados en los artefactos de diseño de manera intuitiva y clara. Este proceso está encaminado a la obtención del documento de arquitectura de software, donde se establece un marco para la estructura básica que refleja los principales componentes del sistema y la comunicación entre ellos.

A partir de los conceptos hasta aquí tratados, se decide la utilización de los STR gobernados por eventos, en los cuales la activación se realiza en el instante que ocurren los cambios, con el ánimo de buscar eficiencia en la utilización de recursos. En este tipo de sistemas, los mensajes son enviados por la aplicación cuando ocurre un determinado evento (como un cambio en el valor de alguna de las variables monitoreadas); mientras que en un sistema gobernado por tiempo, los mensajes son enviados sólo en instantes temporales predefinidos. Para ello se hará uso del modelo de comunicación productor-distribuidor-consumidor sustentado en el paradigma de Publicación-Suscripción.

El diseño de la arquitectura en los STR es de gran importancia, debido a que para el desarrollo de estos, es necesario conocer qué características se desean priorizar. Las decisiones más importantes están ligadas a la obtención de un balance entre las respuestas temporales, y el consumo de recursos tanto de procesamiento como de red. Todo esto está relacionado principalmente con el

rendimiento. Para lograr una solución satisfactoria, será necesario un análisis que permita la obtención de resultados superiores a los existentes actualmente.

1.11 Conclusiones

En este capítulo se han expuesto los fundamentos teóricos conceptuales que constituyen la base para la correcta concepción, diseño e implementación de la solución propuesta. Existen diversas clasificaciones para STR; se considera que para la obtención de un mecanismo que permita la comunicación en tiempo real sobre la Web se deben utilizar los STR distribuidos gobernados por eventos.

Después del análisis de cada uno de los modelos de comunicación y teniendo en cuenta las características deseadas en los STR, se seleccionó el modelo de comunicación productor-distribuidor-consumidor sustentado en el paradigma de Publicación - Suscripción. En correspondencia, se decide utilizar el protocolo XMPP como base para el desarrollo del mecanismo de comunicación.

El análisis de soluciones que utilizan comunicación en tiempo real como es el caso de los clientes de mensajería instantánea, *Google Wave* y los sistemas SCADA, arrojó que no son factibles para la solución deseada. Finalmente se considera una necesidad tener en cuenta el diseño arquitectónico como elemento que aporte calidad a la solución propuesta.

CAPÍTULO

2

UN NUEVO MECANISMO DE COMUNICACIÓN EN TIEMPO REAL

2.1 Introducción

En este capítulo se realiza una descripción del nuevo mecanismo de comunicación, para ello se considera un modelo conceptual que contiene las principales definiciones que se utilizan en la propuesta, así como la relación que se establecen entre ellas. Se especifican los requerimientos funcionales que forman parte del componente. Finalmente se describe su arquitectura e implementación y se tienen en cuenta consideraciones sobre la seguridad.

2.2 Descripción de la solución propuesta

Una de las soluciones libres más utilizadas para el análisis de datos con representaciones gráficas, es la suite de *Pentaho BI Suite 3.6 Community Edition*. Con la utilización de esta suite, puede conseguirse la implementación de TD con diferentes tipos de gráficos y representaciones. Sin embargo, los datos son cargados a partir de un almacén de datos y una vez que se muestran los resultados, cualquier variación en ellos, no será mostrada.

Cuando se necesita controlar determinado indicador en tiempo real pierde sentido utilizar esta solución, puesto que para mostrar la información, hay que guardar los datos en el almacén de datos y luego actualizar el TD. Esto

representa una considerable pérdida de tiempo, introduciendo una desviación temporal entre el instante en que los datos fueron modificados y su representación. Es por ello que se necesita un mecanismo que permita la comunicación directa entre el origen de los datos y la aplicación que los gestiona. También es necesario que este mecanismo implemente una forma de actualizar los datos en el instante en que ocurre una modificación, sin la necesidad de que el usuario intervenga y esté al tanto del estado actual de los KPI.

La propuesta de solución constituirá un nuevo mecanismo de comunicación capaz de mantener información actualizada sobre procesos cambiantes en el tiempo.

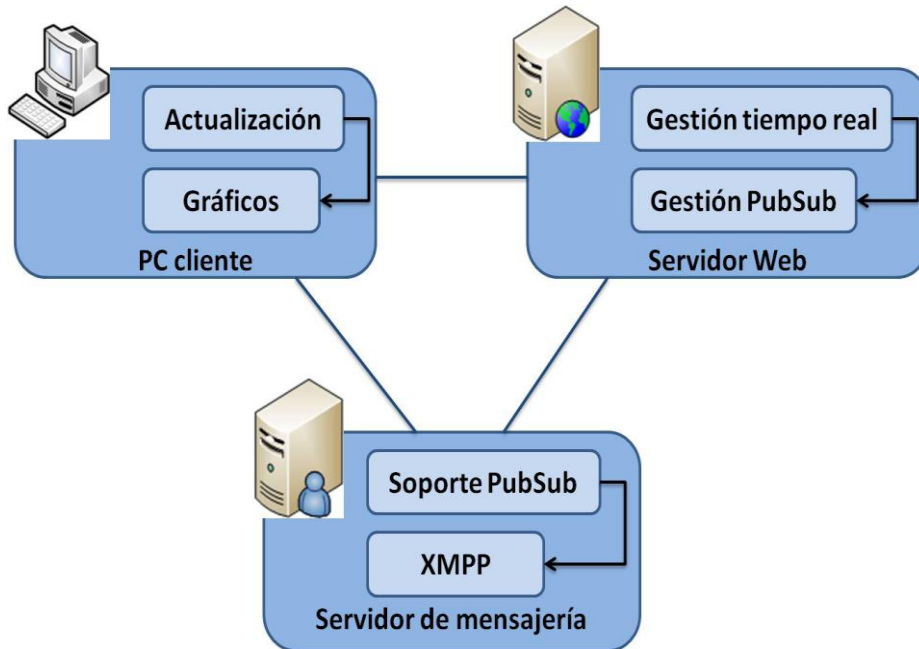


Figura 8. Descripción de la organización del sistema.

En la figura 8 se muestra de forma simple los elementos que debe tener el mecanismo de comunicación. Este estará compuesto por el modelo de comunicación publicador - distribuidor - consumidor, implementado por el servidor de mensajería Ejabberd en su especificación XEP-0060: Publicación - Suscripción. Se deberá contar con tres nodos de procesamiento: un cliente, un servidor web y un servidor de mensajería; aunque pueden ser dos, en cuyo

caso el servidor de mensajería puede compartir el mismo nodo físico con el servidor web. Además se contará con un componente de comunicación implementado en el lenguaje PHP, encargado de la gestión en tiempo real a través del servidor Ejabberd. En la PC cliente, se visualizarán los gráficos, que serán actualizados en la medida que varíen los datos. En el servidor web, se encontrará la lógica de programación que permitirá la gestión de la comunicación en tiempo real y lo referente a la interpretación del paradigma PubSub. El servidor de mensajería constituye una pieza fundamental, pues en este se implementa la infraestructura de la especificación PubSub y se interpreta la estructura del protocolo XMPP.

El componente de comunicación debe permitir de manera integrada la realización de diferentes operaciones, entre las que se encuentran: conectarse a un servidor XMPP, obtener nodos, suscribirse a un nodo y publicar datos en un KPI, de tal manera que todos los abonados a este puedan recibirlos automáticamente. Todas estas acciones, deben estar estrechamente vinculadas en un mismo entorno para, de esta forma, permitir un mejor desempeño del usuario con la aplicación.

Una vez concluido el componente, debe insertarse como un activo en el Paquete de Ayuda para la Toma de Decisiones (PATSI). PATSI incluye un conjunto de herramientas que permiten capturar datos de distintas fuentes, consultar estos datos en forma de reportes, visualizarlos, monitorear datos o KPI y programar acciones de respuesta ante estos eventos.

2.2.1 Modelo conceptual

Un modelo conceptual o modelo de dominio, constituye una representación visual para el usuario de los conceptos u objetos significativos del mundo real para un problema o área de interés. Representa conceptos del mundo real, no de los componentes de software, mediante clases conceptuales del dominio del problema, encargándose de capturar los tipos más importantes de objetos y eventos que suceden en el entorno [45]. La figura 9 muestra el modelo de dominio asociado al sistema.

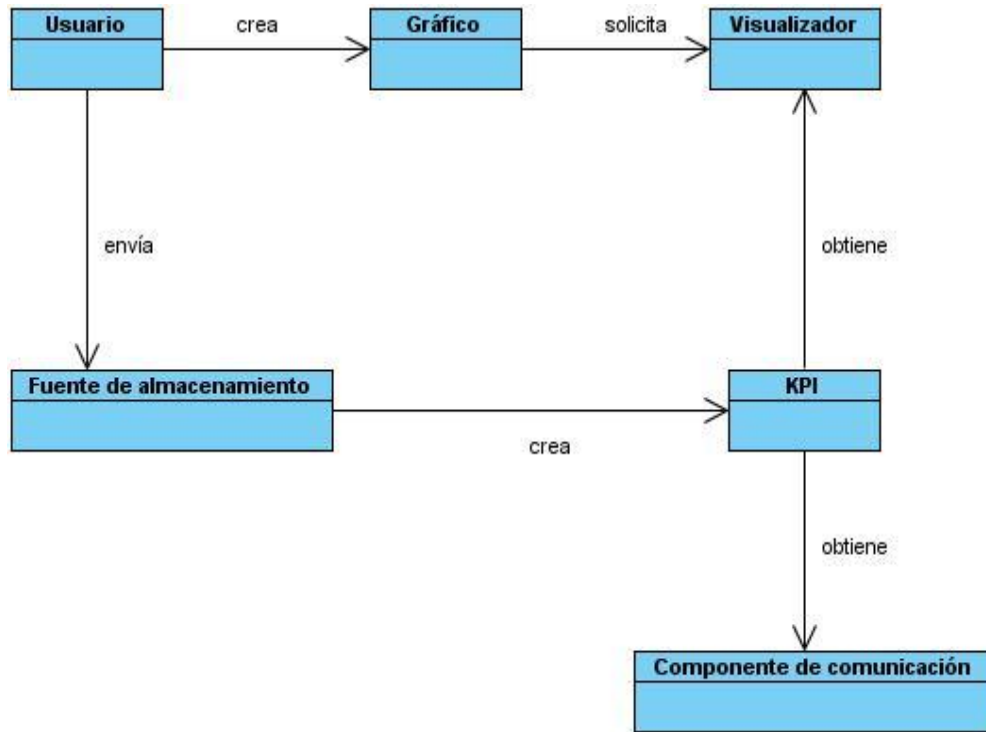


Figura 9. Modelo de dominio.

Conceptos asociados al dominio:

- **Usuario:** su función será realizada por una persona o sistema, que deberá tener en cuenta la entrada de datos indispensable para la creación de los gráficos.
- **Gráfico:** es el encargado de mostrar un conjunto de gráficos, además de realizar la actualización en tiempo real de los datos.
- **Visualizador:** gestiona la información obtenida del proceso de suscripción y la hace llegar a los gráficos, para de esta manera lograr su actualización.
- **Fuente de almacenamiento:** realiza la función de un servidor de datos y contiene además los datos necesarios para la creación y actualización de los gráficos.
- **Componente de comunicación:** es el objeto responsable de gestionar la información para la actualización del servidor Ejabberd.
- **KPI:** realiza un constante monitoreo del comportamiento de los datos y de la actualización en tiempo real, encargándose del envío de los datos.

El objeto usuario, es el responsable de la creación de los gráficos, por lo que deberá tener en cuenta la entrada de datos a graficar, además de la dirección del KPI y la dirección de la fuente de almacenamiento. También es el encargado de enviar una serie de datos obtenidos de las distintas suscripciones efectuadas a la fuente de almacenamiento para su actualización. Igualmente, el objeto fuente de almacenamiento, constituye un servidor de datos donde se gestiona la información resultante de la creación de los nodos KPI necesarios para la realización y actualización de los gráficos.

El objeto gráfico muestra un paquete de gráficos, que son el resultado de la representación, de un conjunto de datos obtenidos de las suscripciones de los nodos realizadas en la fuente de almacenamiento. También incluye un conjunto de funcionalidades con el fin de lograr un mejor entendimiento por parte del usuario a la hora del proceso de toma de decisiones. Realiza además la actualización en tiempo real de los datos que le son enviados automáticamente de la fuente de almacenamiento haciendo uso del objeto visualizador.

El objeto visualizador obtiene los datos provenientes del KPI para posibilitar la actualización del paquete de gráficos. Por su parte el objeto KPI constituye el nodo de publicación - suscripción creado en la fuente de almacenamiento, el mismo recibe un conjunto de datos de los cuales se requiere su graficación.

El modelo de dominio ocupa un rol protagónico en el desarrollo moderno de software. Además de que puede ser tomado como el punto de partida para el diseño del mismo.

2.2.2 Requerimientos

Los requerimientos se dividen en funcionales y no funcionales para separar los que surgen a partir de reglas del negocio de los que surgen como parte de las características que el software debe presentar para ser aceptado por el cliente. La selección de los requisitos funcionales es una tarea que debe realizarse con precisión, por cumplir un papel primordial en el proceso de producción de software, pues se enfoca en un área fundamental: la definición de lo que se

desea producir, permitiendo describir con mayor claridad el comportamiento del sistema, minimizando los problemas derivados de su desarrollo.

Dentro de las funcionalidades que debe cumplir el componente, figuran como las más importantes las de conectarse y desconectarse a un servidor de mensajería, crear y gestionar nodos de publicación - suscripción en este y permitir las suscripciones a los nodos. Además, debe poderse acceder a la lista de los nodos existentes y publicar datos en el que se seleccione, a partir de las restricciones que se imponen en el funcionamiento del paradigma PubSub.

La lista de requisitos funcionales se puede consultar en el anexo 1.

Para el desarrollo del componente de comunicación se identifican tres actores que intervienen en la propuesta de solución, los cuales son mostrados en la tabla 2.

Tabla 2. Actores del sistema.

Actor	Descripción
Usuario	Actor que representa a toda aquella persona que gestiona los datos con los cuales interactúa el sistema, es el responsable de las acciones: crear, eliminar, editar y obtener un nodo; así como realizar la conexión o desconexión del servidor, recuperar los datos y publicarlos en los nodos, suscribir al nodo y eliminar la suscripción a este, además de graficar los datos.
Administrador Jabber	Actor que se vincula con las acciones relacionadas con la administración del servidor Ejabberd instalado, tales como: crear, editar y eliminar usuario Jabber.
Sistema Externo	Actor que se relaciona con las funcionalidades publicar datos y gestionar nodos de los cuales obtiene información.

2.2.3 Modelo de Sistema

Los casos de uso del sistema (CUS) se utilizan para capturar los requisitos, describen la interacción entre el usuario y el sistema para lograr un objetivo específico. En el desarrollo de la investigación se determinaron un total de 10 casos de uso. El diagrama se encuentra en el anexo 2.

2.3 Arquitectura general del sistema

Para el desarrollo de la solución se utilizó una arquitectura en tres capas la cual se representa en la figura 10. Como protocolo de comunicación fue seleccionado el protocolo XMPP debido a sus características, en especial porque da soporte al paradigma PubSub.

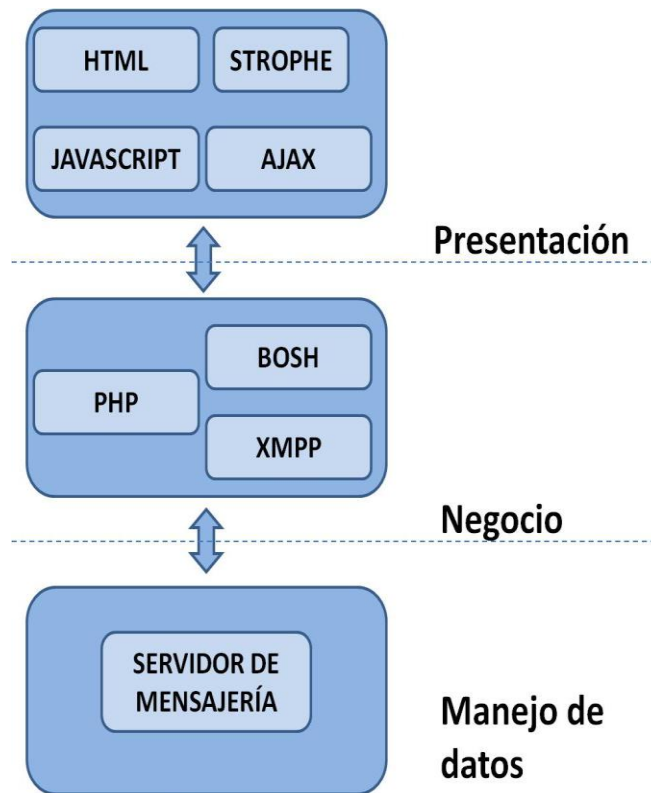


Figura 10. Agrupación en capas del sistema.

En la construcción de la capa de presentación, se utiliza HTML, *JavaScript*, Strophe y Dojo. La librería Strophe utiliza BOSH para emular una conexión persistente, con estado y conexión bidireccional a un servidor XMPP ya que

JavaScript no está preparado para las conexiones TCP persistentes [11]. La interfaz de usuario fue desarrollada utilizando Dojo, que constituye un conjunto de librerías escritas en *JavaScript* para la construcción de aplicaciones. Esta librería cuenta con componentes vistosos y personalizables con una amplia incorporación de AJAX.

La capa de negocio se implementa utilizando el lenguaje de script PHP 5, considerando la librería *Sixties* para la comunicación con el servidor XMPP [11]. A esta librería fue necesario introducirle cambios que permitieran un mejor desempeño en el componente propuesto. La solución será desplegada haciendo uso del servidor web Apache 2.0.

La representación de la capa de acceso a datos fue hecha a partir del servidor Ejabberd en su versión 2.1, no constituyendo este un acceso a Bases de datos, sino un servidor de mensajería a través del cual se tramitarán los datos. Para este propósito se hace uso de la especificación PubSub que este implementa [11].

Para la selección de las herramientas y tecnologías a utilizar se tuvieron en cuenta las tendencias actuales en estos temas y se siguieron estándares abiertos.

Protocolos

Como se trata de una aplicación web en PHP, la utilización de protocolos de comunicación se reduce a aquellos que sean implementados sobre HTTP. Para lograr la comunicación en tiempo real, no solo se consideran importantes los tiempos de respuestas, sino que hay que encontrar una medida entre rendimiento y consumo de recursos.

El protocolo PHP-RPC permite la comunicación entre el cliente y el servidor, es ligero y permite un mecanismo simple y rápido para la llamada a procedimientos remotos en PHP. Sin embargo, la dinámica del protocolo es puramente petición/respuesta, como se puede observar en la figura 11. Para cada petición el navegador realiza una conexión HTTP al servidor web, el servidor devuelve los datos y termina la conexión. La desventaja de este método es que para

poder actualizar los datos en el cliente, el usuario o el cliente debe actualizar o cambiar de página web, lo que hace que esto tome mucho tiempo, sin dejar de lado que hace un mayor consumo de ancho de banda. Esto también trae problemas en cuanto al rendimiento de las páginas del lado del cliente.

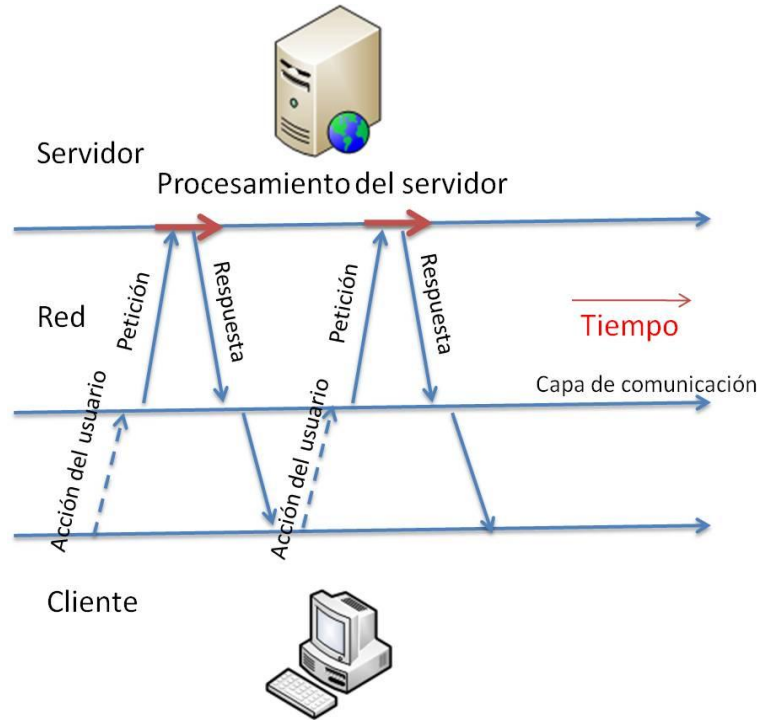


Figura 11. Protocolo PHP-RPC.

Comet es un mecanismo que permite a un servidor web enviar datos al navegador, sin que este los solicite explícitamente, a partir de mantener abierta una petición HTTP. Este método confía en características incluidas por defecto en navegadores, como *JavaScript*, en lugar de sobre *plug-ins* no disponibles por defecto. En Comet se abre una línea de comunicación entre el navegador y el servidor, reduciendo la latencia de enviar paquetes desde el navegador cada vez que se necesite un requerimiento y permitiendo que sea el servidor quien envíe los paquetes al navegador de forma autónoma tal y como se muestra en la figura 12. Dado que Comet no está construido sobre la base de comunicación estándar HTTP petición/respuesta es necesario instalar componentes de parte del servidor; además no presenta soporte para el modelo PubSub.

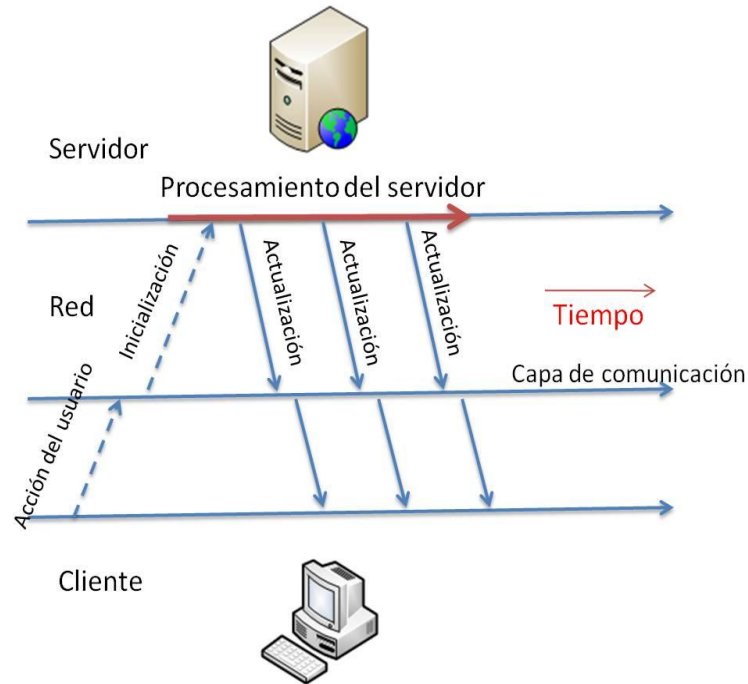


Figura 12. Protocolo Comet.

El protocolo XMPP es una tecnología para la comunicación en tiempo real, que se utiliza principalmente en la mensajería instantánea. En esencia, el XMPP proporciona una vía para enviar piezas pequeñas de XML de una entidad a otra en tiempo real. HTTP no coexiste nativamente con este protocolo, sin embargo, existe una pasarela para poder establecer una conexión a un servidor XMPP desde HTTP, conocida como BOSH. Esto permite que se puedan utilizar todas las funciones de un servidor XMPP, a través de una conexión HTTP. Es posible utilizar características importantes del servidor XMPP y no hay necesidad de contar con bases de datos separadas de usuarios registrados y sus suscripciones, ya que esto estará gestionado por el servidor XMPP. BOSH implementa el *long polling*, así que se puede hacer uso de sus bondades.

La utilización de BOSH, permite reducir el número de peticiones realizadas al servidor, por lo que la carga de este será menor. El hecho de permitir la suscripción a un nodo, hace que sea el servidor quien envíe los datos cuando estos estén disponibles y no que tenga que lidiar con una avalancha de peticiones, cuando puede ser que no existan cambios en estos. Esto está representado en la figura 13. En la figura 14, se puede observar como el

servidor envía la respuesta a los clientes, una vez que los datos están disponibles. Cuando en el servidor ocurre un evento, que en este caso será la publicación de nuevos datos, este será capaz de informarlo a todos los clientes suscritos.

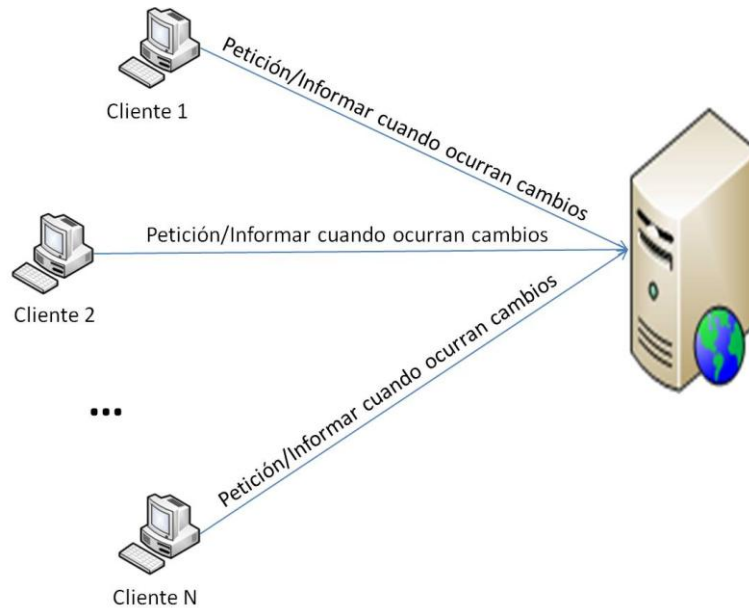


Figura 13. Funcionamiento del *long polling* usando BOSH escenario 1.

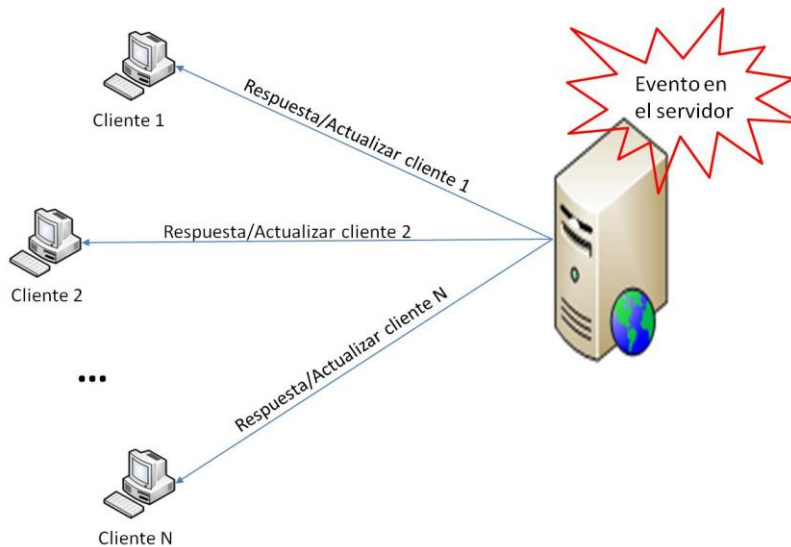


Figura 14. Funcionamiento del *long polling* usando BOSH escenario 2.

Debido al análisis realizado, se decide hacer uso de los protocolos XMPP y BOSH para la implementación de la solución. Con estos se puede implementar

el modelo de comunicación productor - distribuidor - consumidor, en la especificación PubSub. Con el uso de BOSH se hace un consumo más eficiente del ancho de banda.

2.4 Implementación de la propuesta de solución

La implementación es el centro de las iteraciones durante la fase de construcción, aunque también se lleva a cabo la implementación durante la fase de elaboración. El modelo de implementación describe cómo los elementos del modelo de diseño son implementados en términos de componentes, donde se detalla además, su organización de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y el lenguaje o lenguajes de programación utilizados; así como la dependencia que se establece entre estos componentes [45].

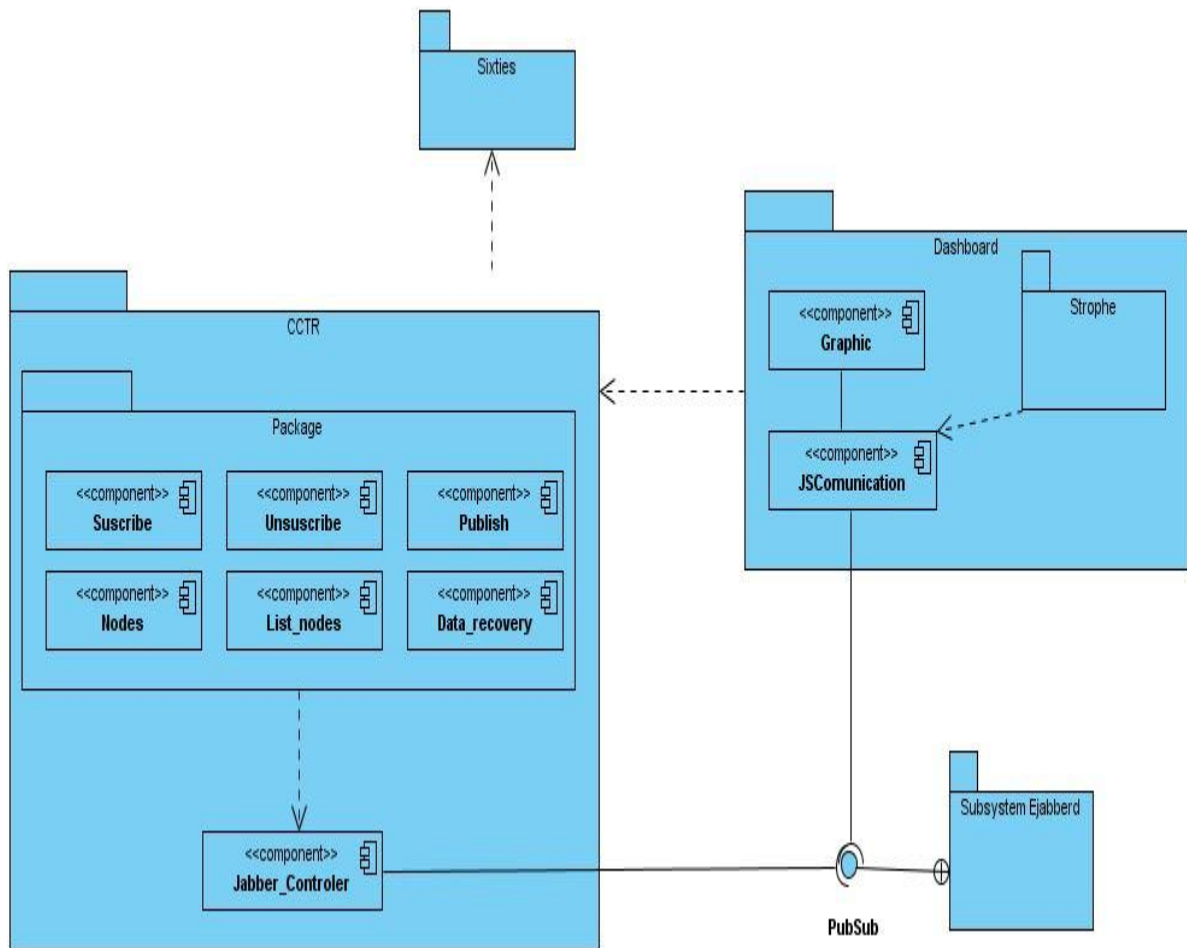


Figura 15. Diagrama de componentes.

La figura 15 muestra el diagrama de componentes de la solución propuesta. En este se observan las dependencias que se establecen entre los paquetes, los componentes y las diferentes librerías que se utilizan para lograr la funcionalidad deseada. El componente *Jabber_Controller* se encarga de la gestión de la comunicación en tiempo real mediante la especificación PubSub, directamente con el subsistema Ejabberd, a través de los nodos de publicación/suscripción representados como la interface PubSub. Este componente brinda funcionalidades a los componentes *Suscribe*, *Unsuscribe*, *Publish*, *Data_Recovery*, *Nodes* y *List_Nodes*. Todos estos se encuentran dentro del paquete CCTR para mejorar su organización. Para lograr el correcto funcionamiento, el paquete CCTR utiliza a la librería *Sixties*.

Dentro del paquete *Dashboard*, se encuentra el componente *Graphic* que manejará la representación gráfica, para lo cual se asocia al componente *JSComunication* que será el encargado de entregar los datos. Para poder implementar la comunicación a través del protocolo BOSH, se deben utilizar funciones que se encuentran en la librería *Strophe*.

A continuación se describe en términos de implementación, los procesos de manejo de mensajes y suscripciones, considerándose que son los más importantes desde el punto de vista del funcionamiento del componente de comunicación.

Gestionar nodo

Este proceso constituye uno de los fundamentales, pues garantiza la infraestructura necesaria para la comunicación en tiempo real usando el paradigma PubSub. Para crear un nodo se debe conocer en qué servidor se creará, así como el nombre con que se accederá al nodo, que en este caso, siempre debe ser el nombre de un KPI sobre el cual se desee tener control sobre su funcionamiento.

Para editar un nodo es necesario introducir los campos que se desean editar, así como las opciones de configuración pertinentes, luego se envía esta

información en un formulario al servidor. Para Eliminar un nodo se necesita ser el propietario (*owner*) de dicho nodo, y conocer su nombre, luego se envía el formulario al servidor realizando así la eliminación.

Publicar datos

En este proceso, el usuario selecciona los datos y el nodo en el que se realizará la publicación de los mismos. Este evento ha de tener formato XML, por lo que en la aplicación desarrollada, el tipo de eventos que se envía está dentro de una etiqueta XML. Si el nodo seleccionado no existe, este debe ser creado y se procede a enviar los datos. Si el nodo existe, se envían los datos a este. Cuando se publican datos, en realidad se está enviando un mensaje hacia un nodo PubSub, por lo cual todos los que estén suscritos a ese nodo recibirán ese mensaje, como se muestra en la figura 16. El tipo de datos y el formato lo decide el ente que los publica, pero debe ser especificado con anterioridad para lograr la comprensión de los mismos por parte de los suscriptores. Es necesario destacar que para realizar este proceso, el usuario debe encontrarse autenticado en el servidor Jabber con sus credenciales de usuario y contraseña.

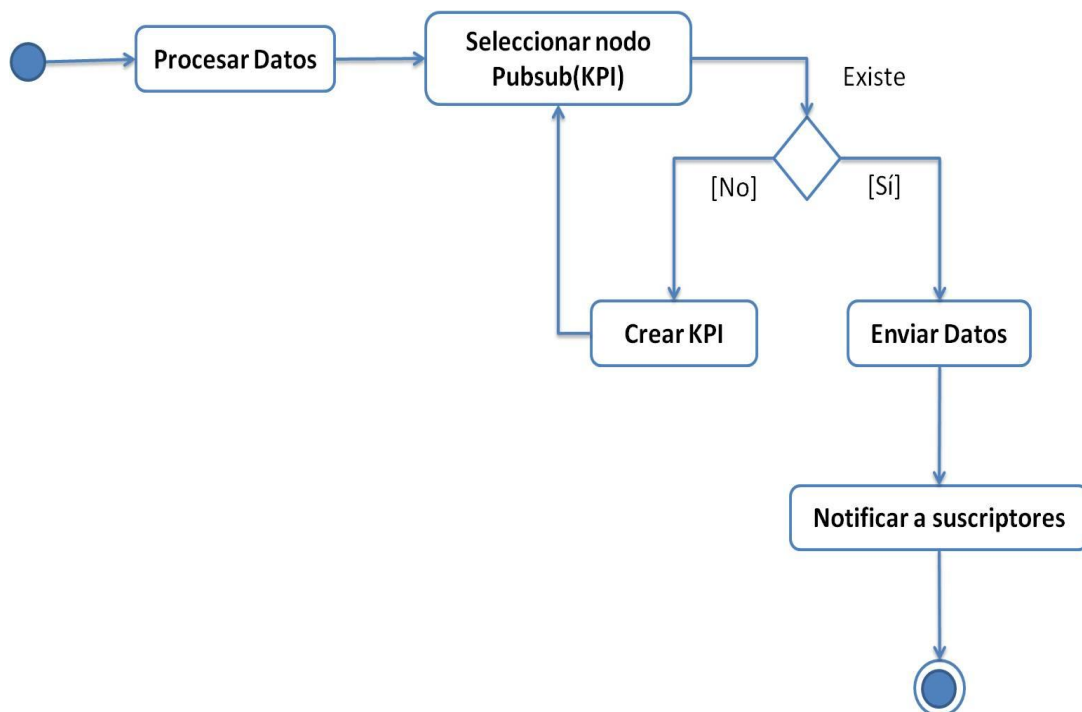


Figura 16. Diagrama de actividades generales del proceso Publicar.

En el mecanismo de comunicación descrito, se observa una limitación sobre el sistema de publicación. Esta viene asociada a que solo puede publicar datos una entidad, esta condición viene impuesta por el paradigma PubSub. Podría pensarse que constituye una limitación crítica, cuando se piense en el mecanismo implantado donde un mismo KPI puede modificarse a partir de diferentes fuentes físicas. Esto viene dado, porque se propone que sobre un nodo PubSub solo pueda publicar el usuario que posee el rol de propietario, que es el que lo crea. El protocolo XMPP permite varias configuraciones sobre un nodo, entre ellas permitir que los entes suscritos puedan publicar contenidos, pero esto violaría la independencia entre el publicador y los suscriptores. En el caso que un KPI se actualice de diferentes fuentes físicas, esta dificultad se resuelve a través de mecanismos de autenticación que establece la tecnología XMPP, pues sin importar el recurso físico donde varíen los datos, siempre se pueden publicar los cambios ocurridos con las credenciales del usuario propietario.

Suscribirse a nodo

Este proceso se realiza cuando se necesita recibir los datos que se publican en tiempo real sobre el comportamiento de determinado indicador. Cuando se requiere realizar la suscripción a un nodo PubSub se necesita el nombre del servicio PubSub y el nombre del nodo, para comenzar a recibir los datos que sean publicados en el mismo. Una vez un cliente se suscribe, mantiene esta suscripción aunque se desconecte. La única manera de dejar de estar suscrito es borrando la suscripción.

2.4.1 Modelo de despliegue.

La vista de despliegue define la arquitectura física del sistema por medio de nodos interconectados. Estos nodos son elementos hardware sobre los cuales pueden ejecutarse los elementos de software. El sistema estará distribuido como se muestra en la figura 17.

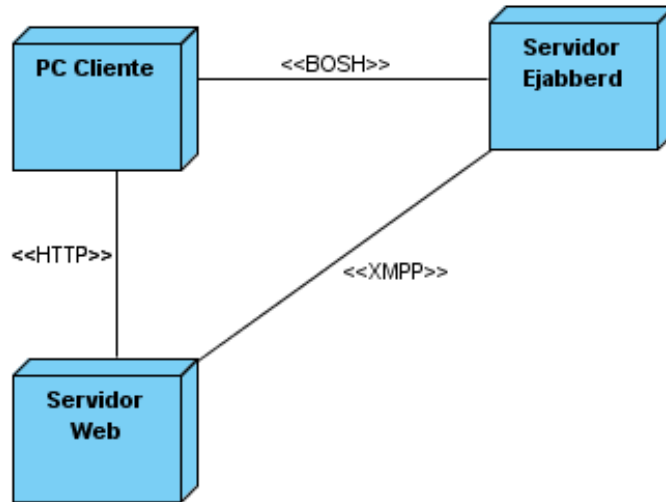


Figura 17. Modelo de despliegue.

La aplicación será desplegada de la siguiente manera: la PC cliente que podrá estar en cualquier lugar utiliza el protocolo BOSH para emular una conexión persistente, con estado y conexión bidireccional al servidor Ejabberd (nodos administrados y suscripciones). El servidor Ejabberd a través del protocolo de comunicación XMPP es el encargado de gestionar las conexiones y los envíos de paquetes XML entre el cliente y los servidores. El servidor web es el encargado de gestionar la distribución de las operaciones que deben realizarse para obtener el resultado.

De esta forma, se garantizará el funcionamiento del componente, y el despliegue del mismo en cualquier entorno que cuente con una red local o acceso a Internet.

2.5 Seguridad en XMPP

La seguridad en las comunicaciones XMPP está garantizada a través de dos mecanismos: el protocolo de seguridad de la capa de transporte (TLS del inglés *Transport Layer Security*) aplicado a la capa de transporte y el protocolo de autenticación simple y seguridad (SASL del inglés *Simple Authentication and Security Layer*).

TLS es capaz de autenticar en ambos lados de la comunicación, y crea una conexión cifrada entre las dos. El protocolo TLS puede ser extendido, esto

significa que nuevos algoritmos pueden ser utilizados para cualquiera de los propósitos, con la condición de que tanto el cliente como el servidor conozcan dichos algoritmos. La principal propiedad del protocolo TLS, es ofrecer privacidad e integridad de los datos, entre dos aplicaciones que se comunican.

En los inicios del protocolo XMPP la autenticación se realizaba mediante el protocolo jabber:iq:auth, actualmente esto se realiza empleando SASL. Este provee a XMPP de un método generalizado para la autenticación. Para ello se han aplicado ciertas normas:

- Si el servidor soporta SASL, deberá informar de sus tipos de autenticaciones con la etiqueta *<mechanisms/>* en la contestación de la etiqueta de inicio de sesión, si es que el cliente soporta la conexión SASL.
- Durante la negociación SASL, ninguno de los dos deberá enviar algún carácter en blanco como separación entre elementos, esta prohibición ayuda a asegurar la precisión a nivel de byte.

Finalmente, para la comunicación haciendo uso del protocolo BOSH, todo el intercambio de información debe ocurrir bajo conexiones HTTP encriptadas. Para esto, debe negociarse la seguridad de la conexión con tecnologías como *Secure Sockets Layer* (SSL) o TLS, explicado anteriormente. Se concluye parcialmente que el transporte de datos haciendo uso del mecanismo garantiza la seguridad.

2.6 Conclusiones

En este capítulo se realizó una descripción completa de la solución propuesta, con énfasis en la implementación de la misma. Se definió el diseño arquitectónico del mecanismo de comunicación que está compuesto por los protocolos XMPP y BOSH como soporte al modelo de comunicación productor - distribuidor - consumidor; el servidor Ejabberd en su versión 2.1 como servidor de mensajería, el servidor web Apache 2.0 y el componente implementado.

Con el diagrama de componentes, la representación de los principales procesos que se pueden realizar y el diagrama de despliegue se describió el componente implementado.

CAPÍTULO

3

ANÁLISIS DE LOS RESULTADOS

3.1 Introducción

En este capítulo se realiza un análisis mediante un conjunto de criterios de medida que permiten evaluar los resultados obtenidos. Se muestran los resultados de la validación de la propuesta, basada en las pruebas funcionales al componente. A partir del registro de los tiempos de respuesta del componente, se hace un análisis sobre su efectividad. Además, se establece una comparación entre el mecanismo propuesto y otras soluciones existentes, sobre la base del rendimiento. Finalmente se especifican las posibilidades de generalización del mecanismo de comunicación.

3.2 Validación de los resultados de implementación

Para lograr la validación del componente implementado, se aplicó un conjunto de pruebas realizadas en dos iteraciones, en los niveles de Integración y de Sistema. Como se observa en la figura 18, las pruebas en el nivel de Sistema son más abarcadoras, ya que incluyen las técnicas definidas para este nivel, como las que se definieron para el nivel de Integración.

Durante la primera iteración, se realizaron pruebas en el nivel de Integración, con el objetivo de identificar errores introducidos por la combinación de componentes. Se integraron componentes para asegurar que la comunicación, enlaces y datos compartidos ocurran apropiadamente. En este nivel, se utilizó el

tipo de prueba funcional, en la cual se verifica el cumplimiento apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados.

En esta iteración, se utiliza el método de caja negra, que establece la realización de las pruebas sobre la interfaz, teniendo en cuenta las entradas y salidas del programa. No es necesario conocer la lógica de programación, únicamente la funcionalidad que debe realizar. Para la aplicación de este método, se seleccionó la técnica de partición de equivalencia, que divide el campo de entrada en clases de datos que tienden a ejercitar las diferentes funciones de la aplicación. Como herramientas para guiar las pruebas, se utilizaron los diseños de caso de prueba basados en las funcionalidades definidas en los casos de uso, comparando cada funcionalidad implementada con la descrita.

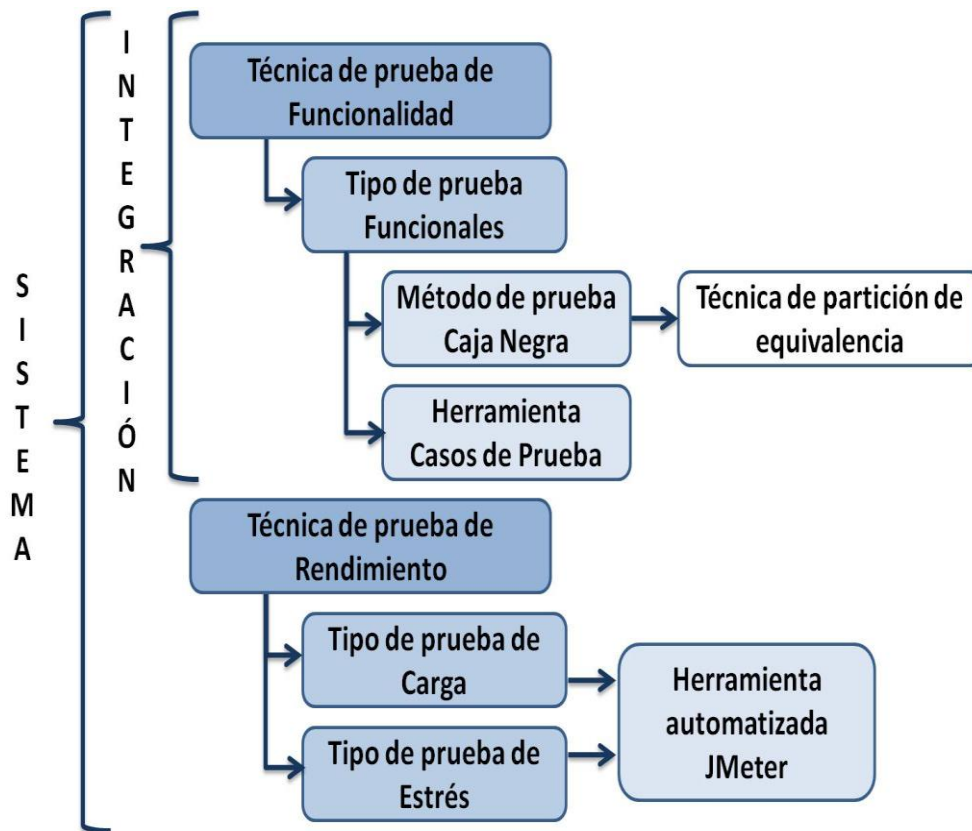


Figura 18. Especificación de pruebas aplicadas.

En la segunda iteración, se realizaron las pruebas en el nivel de Sistema para asegurar que todos los módulos trabajen como un sistema sin errores. Estas pruebas examinan qué tan bien se cumple con los requisitos, su utilidad y desempeño. Se utilizó además, la técnica de prueba de rendimiento, para determinar cuán rápido realiza una tarea el sistema bajo condiciones particulares de trabajo, dentro de la cual se aplicaron las pruebas de carga y estrés. Las pruebas de carga se aplican para observar el comportamiento de la aplicación bajo una cantidad de peticiones esperadas. En el caso de las pruebas de estrés, se utiliza para hacer colapsar la aplicación, y así determinar si rendirá lo suficiente.

Para la realización de las pruebas de carga y estrés se hizo uso de la herramienta de software libre JMeter, que ofrece la posibilidad de realizar pruebas sobre diferentes aspectos de una aplicación Web. Esta herramienta es de fácil configuración y permite conocer los tiempos de respuesta experimentados por una aplicación cuando se tiene un número N de usuarios conectados y el número real de transacciones procesadas por unidad de tiempo. Los resultados pueden ser visualizados en diferentes formatos, lo cual facilita las labores de análisis de los mismos.

El desarrollo de las pruebas se realizó en una computadora Haier, con 1 GB de memoria RAM y un procesador Intel Pentium 4 a 3.0 Ghz de frecuencia, donde fue desplegado el componente de comunicación, incluyendo el servidor Ejabberd.

3.2.1 Pruebas de integración y sistema

Para la ejecución de las pruebas de integración se hizo una selección de los casos de uso, de manera que se tuviera una representación de todas las funcionalidades del sistema distribuidas por todos sus módulos. Sin embargo, a continuación se hace un análisis sobre el rendimiento de las tres funcionalidades críticas para el mecanismo de comunicación propuesto.

Los resultados para la funcionalidad de crear nodos PubSub, se muestran en la tabla 3.

Tabla 3. Rendimiento para crear nodos PubSub.

Usuarios Concurrentes	Rendimiento (ms)
50	50.1
100	95.6
150	140.6
200	182.6
250	218.6
300	258.8
350	302.5
400	330.9

Se puede observar que para una concurrencia de 400 peticiones al servidor, este es capaz de resolverlas en la tercera parte de un segundo aproximadamente, considerándose esto una aproximación aceptada para lo que se demanda del componente. Se establece como límite 400 conexiones concurrentes, pues se espera que no supere esta cifra una vez desplegado el componente.

En el caso de los resultados para la opción de suscribir descrita en el capítulo 2, son mostrados en la tabla 4. Es necesario aclarar, que los niveles de concurrencia están dirigidos a la conexión sobre un mismo nodo.

Tabla 4. Rendimiento para suscribir a nodo.

Usuarios Concurrentes	Rendimiento (ms)
50	352,6
100	366,0
150	389,0
200	466,9
250	547,9
300	582,2
350	597,3
400	623,5

En este caso, los resultados son mayores, lo cual se explica a partir de que las peticiones están dirigidas al mismo nodo de PubSub. Esto significa que para un entorno donde haya 200 o menos conexiones, el cliente tarda menos de medio segundo en informar al servidor su interés en recibir los datos de determinado canal. Para un nivel de concurrencia mayor que 200 y hasta 400, tarda poco más de medio segundo. Debido a la restricción de tiempo real, se establece 400 como límite máximo de usuarios concurrentes para la funcionalidad de suscribir a nodo.

Para el caso de la funcionalidad publicar datos los resultados son los mostrados en la tabla 5.

Tabla 5. Rendimiento para publicar datos.

Usuarios Concurrentes	Rendimiento (ms)
50	371,3
100	390,6
150	427,3
200	473,4
250	519,5
300	549,2
350	570,6
400	588,2

El tiempo medido para la funcionalidad publicar datos es desde que se publican los datos hasta que se reciben y se considera que el resultado es bueno. Se mejora el rendimiento de la red porque es el servidor el que informa la disponibilidad de los datos, y no el cliente haciendo peticiones constantes a este.

Esta es la funcionalidad principal del componente, y en la que se hace la utilización de las ventajas del mecanismo propuesto, pues para el envío de los datos, se utiliza el paradigma PubSub. En este caso, mediante el protocolo BOSH, los datos fluyen entre el servidor y el cliente una vez que están disponibles.

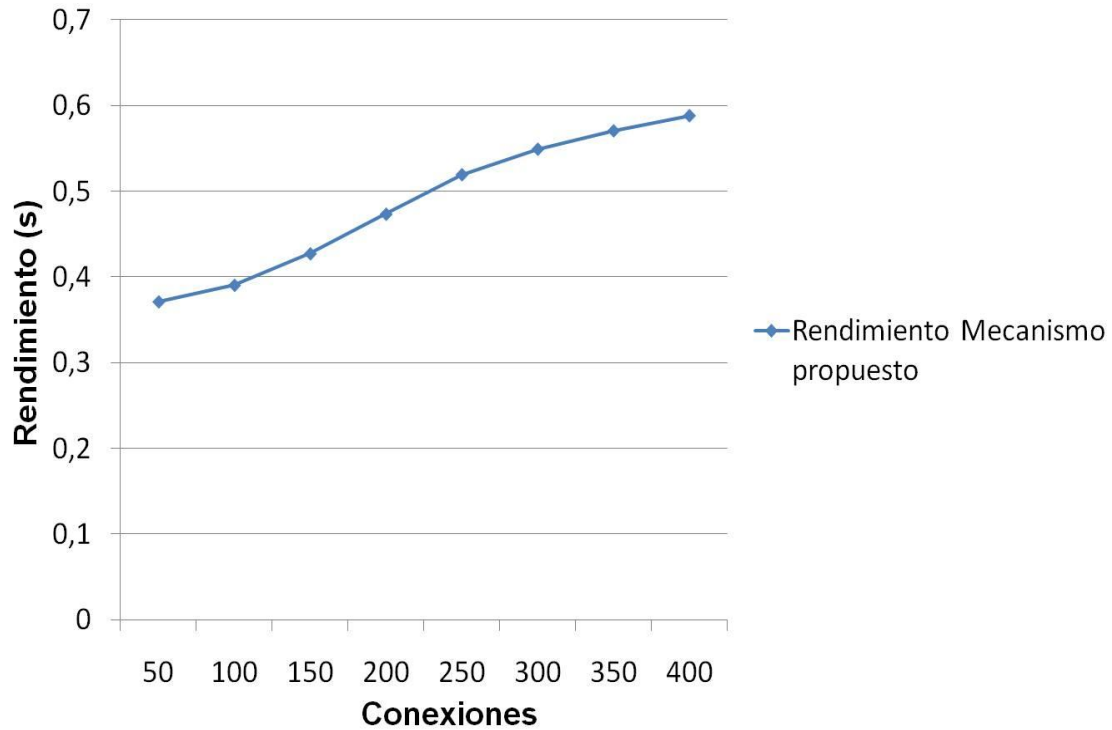


Figura 19. Rendimiento por conexiones concurrentes.

En la figura 19 se muestra la gráfica de rendimiento contra conexiones concurrentes.

3.3 Valoración del aporte práctico de los resultados

Para conocer si el mecanismo cumple con el objetivo que se persigue, es necesario hacer una comparación con respecto a otras soluciones. Con esta intención, se realizaron pruebas de rendimiento al Dashboard Energía, correspondiente al mercado de datos portadores energéticos, que se encuentra implementado dentro del Proyecto Sala Situacional UCI. Una vez obtenidos los resultados, estos se registran y se procede a su análisis.

En la tabla 6 se muestran los resultados de la comparación en cuanto a rendimiento del mecanismo propuesto y el utilizado por la *suite* de Pentaho.

Tabla 6. Comparación de rendimientos.

Usuarios concurrentes	Solución Pentaho Rendimiento (segundos)	Mecanismo propuesto Rendimiento (segundos)
50	0,979	0,3713
100	1,803	0,3906
150	2,385	0,4273
200	3,226	0,4734
250	3,551	0,5195
300	4,142	0,5492
350	4,181	0,5706
400	5,253	0,5882

A partir de los resultados obtenidos por niveles de concurrencia se observa la mejora en cuanto a rendimiento del mecanismo implementado. Para solo 50 usuarios concurrentes, se manifiesta que la solución propuesta es más eficiente 2,63 veces. Para 100 usuarios lo supera en 4.61 veces, y para 400 usuarios en al menos 8,93 veces.

Como se puede observar en la figura 20, en la cual se muestran las gráficas de rendimientos de las dos soluciones contra conexiones concurrentes, el factor de crecimiento de la función correspondiente a la solución de Pentaho es mayor. Finalmente puede decirse que a partir del desarrollo experimental de los casos de prueba aplicados y de los tiempos medios de respuesta obtenidos en forma

aproximada, se puede reducir hasta 8 veces el tiempo medio de respuesta de la actualización de los datos en el TD.

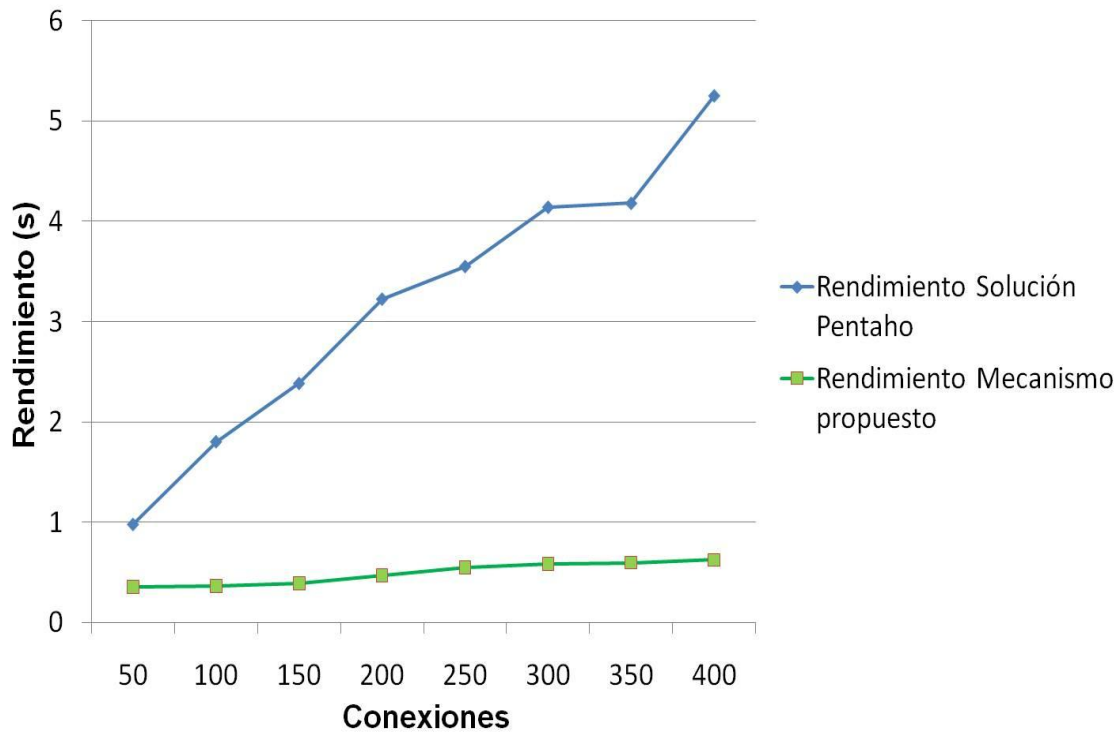


Figura 20. Comportamiento de los rendimientos.

Con el objetivo de evidenciar los beneficios que significa el uso del nuevo mecanismo de comunicación, la tabla 7 muestra una comparación entre este, las soluciones tradicionales (representadas por la suite de Pentaho) y la solución de los sistemas SCADA. Para la comparación se tienen en cuenta los aspectos de actualización, rendimiento y usabilidad.

Tabla 7. Comparación cualitativa.

Criterio	Soluciones analizadas
Actualización	Soluciones tradicionales: Para mostrar una actualización, es necesario ejecutar nuevamente el TD, o recargar la página.
	Soluciones SCADA: Para la actualización utiliza características de los STR distribuidos gobernados por tiempo y los STR gobernados por eventos. Para ello se ejecutan eventos temporizados cíclicamente para conocer el estado de los procesos monitoreados por los sensores.
	Solución propuesta: La actualización se realiza automáticamente, cuando hay una variación en los datos.
Rendimiento	Soluciones tradicionales: Son mayores los tiempos de respuesta, dado por el hecho de utilizar el modelo cliente - servidor, donde se hacen peticiones constantemente al servidor, lo que puede resultar en que este deje de atender solicitudes por sobrecarga.
	Soluciones SCADA: Obtiene resultados de tiempo real a costa de una sobreutilización de recursos, dígase consumo de CPU así como de la infraestructura de comunicaciones.
	Solución propuesta: Mejores tiempos de respuesta, debido a la utilización del paradigma de publicación -

	suscripción, donde los datos son enviados por el servidor una vez que estos están disponibles, esto permite atender un mayor número de peticiones.
Usabilidad	Soluciones tradicionales: Es necesario implementar un mecanismo de actualización por tiempo, donde se hacen peticiones al servidor periódicamente o se necesita la intervención directa del usuario, para la actualización de la página.
	Soluciones SCADA: Posee características de configuración que demandan mayor conocimiento por parte del usuario, en cuanto a dispositivos, medios de comunicación, períodos de actualización y métodos de interconexión.
	Solución propuesta: La actualización ocurre automáticamente, sin necesidad de la intervención del usuario. Esta se realiza a partir de eventos transparentes al usuario.

3.4 Generalización de la propuesta

Para la utilización del mecanismo propuesto en otras soluciones que utilicen o necesiten la comunicación en tiempo real, es necesario tener en cuenta un conjunto de características:

- Se debe tener acceso a una red local y contar con una PC destinada para usarse como servidor Web, en la cual debe estar instalado Apache 2.0.
- Inicialmente el uso del componente de comunicación está desarrollado en el lenguaje de programación PHP, por lo que se debe contar con la implementación de la lógica de negocio en este lenguaje.

- Se debe contar en la organización donde se desee implantar con el servidor XMPP Ejabberd en su versión 2.1, o en caso de no tenerlo, instalarlo.
- Debe existir la necesidad de manejar el intercambio de datos en tiempo real entre aplicaciones.

El mecanismo es extensible a situaciones donde la prioridad sea la obtención de los datos en tiempo real, y es capaz de abstraerse del procesamiento y uso posterior de estos. Brinda una solución con un rendimiento adecuado y un uso racional de recursos.

Hoy día muchas organizaciones y empresas cuentan con servicios de mensajería instantánea entre sus integrantes. Esto garantiza uno de los requisitos necesarios para la generalización del mecanismo porque uno de los componentes fundamentales de la propuesta lo constituye el servidor de mensajería, bastaría hacer algunas modificaciones y configuraciones adicionales.

En cuanto a condiciones de licencia, no existen limitaciones, puesto que las tecnologías utilizadas son libres y de código abierto. Esto garantiza la soberanía tecnológica de la solución.

Una variante de uso puede ser la implementación de servicios web, que exporten la funcionalidad del intercambio en tiempo real, usando un servidor XMPP externo. Sin embargo, esto debería ser sometido a pruebas, para evaluar si cumple con los requerimientos de tiempo necesarios.

3.5 Conclusiones

Se realizaron pruebas a las funcionalidades y al rendimiento del sistema que demostraron la reducción en los tiempos de respuestas obtenidos con el uso del mecanismo propuesto. A través de la comparación con soluciones similares, se establecieron las características que son mejoradas con el mecanismo propuesto, haciendo énfasis en las características de rendimiento, actualización y usabilidad. El principal requisito para la generalización del mecanismo lo constituye que su utilización debe hacerse en un entorno web y para la

comunicación en tiempo real de algún tipo de datos, así como la presencia del servidor de MI Ejabberd 2.1 y el servidor web apache 2.0.

CONCLUSIONES GENERALES

Al finalizar el desarrollo de la presente investigación se puede arribar a las siguientes conclusiones:

- El nuevo mecanismo de comunicación permite la reducción de los tiempos de respuesta en comparación con otras soluciones, mejorando la eficiencia, validado a partir del desarrollo experimental de pruebas funcionales y la valoración de los resultados alcanzados. Los tiempos registrados son una mejor aproximación a respuestas en tiempo real que los del resto de las soluciones analizadas.
- Se realizó un análisis crítico acerca de los sistemas de comunicación en tiempo real, así como su aplicación a diferentes áreas, que permitió identificar los principales componentes o elementos estructurales necesarios para el desarrollo de un nuevo mecanismo de comunicación.
- El diseño arquitectónico permitió la utilización de estándares abiertos para su implementación, garantizando la soberanía tecnológica del mecanismo de comunicación. Este quedó integrado por los protocolos XMPP y BOSH como soporte al modelo de comunicación productor - distribuidor - consumidor; el servidor Ejabberd en su versión 2.1 como servidor de mensajería, el servidor web Apache 2.0 y el componente implementado.
- Se realizó la implementación de un componente de comunicación en tiempo real, que permite la visualización en un TD, de los valores de KPI. Además se consiguió la integración de un nuevo activo a la línea de producción de software del centro DATEC.

RECOMENDACIONES

Para el desarrollo de trabajos futuros asociados con los resultados obtenidos a partir del desarrollo de la presente investigación se recomienda:

- Aplicar el mecanismo de comunicación a otros entornos y no solo para la comunicación en tiempo real de valores de KPI sino para la transmisión de datos generados por eventos.
- Valorar la factibilidad de la implementación de servicios web capaces de exportar las funcionalidades de comunicación en tiempo real, para independizar la generalización del mecanismo propuesto de la tecnología utilizada.

REFERENCIAS BIBLIOGRÁFICAS

1. Niven, P., *El cuadro de mando integral paso a paso*. Vol. I. 2003: Ediciones Gestión 2000. 401.
2. Valeria, P. (2004) *Administración de los recursos humanos*. 49.
3. Argudín, J. *El cuadro de mando integral como guía para la acción*. 2005 [cited 2011 febrero]; Available from: http://www.padep.org.bo/c2website/UAGRM-Santa%20Cruz/Material%20de%20Lectura/Modulo_3/Guia_CMI.pdf.
4. Kaplan, R. and D. Norton, *The Balanced Scorecard: Measures that Drive Performance*. Harvard Business Review, 1992: p. 10.
5. Kaplan, R. and D. Norton, *Putting the Balanced Scorecard to Work*. Harvard Business Review 1996.
6. Kaplan, R. and D. Norton., *The Balanced Scorecard - Translating Strategy into Action*. Harvard Business Review, 1996.
7. Few, S., *Information Dashboard Design*. 1st Edition ed. Vol. I. 2006: O'Reilly. 224.
8. Turban, E., R. Sharda, and D. Delen, *Decision Support and Business Intelligence Systems*. 9th ed, ed. P. Hall. 2010: Prentice Hall 780.
9. Group, S.C. *Sixtina Dashboard* 2010 [cited 2011 marzo]; Available from: <http://www.sixtinagroup.com>.
10. Intelligence, B. *Business Intelligence para seres humanos*. 2010 [cited 2011 marzo]; Available from: <http://www.bingointelligence.com/analisis/index.html>.

11. Rosales, A. and Y. Terry (2010) *Componente de comunicación en tiempo real de valores de los KPIS de un dashboard*. 16.
12. Bouman, R. and J. Dongen, *Pentaho Solutions: Business Intelligence and Data Warehousing with Pentaho and MySQL* 2009: Wiley Publishing 648.
13. Corporation, P., *Introducing the Pentaho BI Suite 3.5 Community Edition*. 1st ed. Vol. I. 2009: Pentaho Corporation. 42.
14. Kaplan, R. and D. Norton, *The Balanced Scorecard – Translating Strategy into Action*. Harvard Business School Press, 1996.
15. ISO. *ISO/IEC 15939:2007*. 2007 [cited 2011 junio]; Available from: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?cnumber=44344.
16. IEEE, *Software Engineering Standards*. 1993. p. 47-48.
17. Manuel, G. (2010) *Métricas e indicadores de gestión*.
18. Choboy, I. (2011) *Dashboard: Conocimiento completo sobre la marcha de la empresa*.
19. Guevara, A. (2008) *Diseño de KPIs para proyectos de TI*. 176.
20. Rivas, M. *Los KPI y el Balanced Scorecard vinculado*. 2008 [cited 2011 mayo]; Available from: http://www.miguelrivas.cl/archivos/KPI_y_Balanced_Scorecard_vinculado.pdf.
21. Burns, A. and A. Wellings, *Real-Time Systems and Programming Languages* 2009: Addison Wesley Longmain 602.
22. Coulouris, G., J. Dollimore, and T. Kindberg, *Distributed Systems: Concepts and Design*. 4nd edition edition ed, ed. I. Addison-Wesley Longman. 2005.
23. Palencia, J. and M. Gonzalez. *Schedulability Analysis for Tasks with Static and Dynamic Offsets*. in *Real-Time Systems Symposium*. 1998.

24. Rivas, J., *Algoritmo de asignación de plazos globales en sistemas distribuidos de tiempo real con palmificación EDF: comparativa de estrategias de planificación*. 2009, Universidad de Cantabria. p. 53.
25. Pérez, H., *Adaptación y optimización para una plataforma distribuida de tiempo real de un middleware basado en los estándares de RT-CORBA y ADA*. 2008. p. 51.
26. Kopetz, H., *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, ed. K.A. Publishers. 1997.
27. Defaz, M. (2007) *Estudio del protocolo CAN (Controller Area Network) y su aplicación en redes de control*. 41.
28. Gómez, C. *Protocolo WORLDFIP*. 2011 [cited 2011 abril]; Available from: <http://es.scribd.com/doc/51568141/World-FIP>.
29. Moffitt, J., *Professional XMPP Programming with JavaScript and jQuery*. First ed. 2010: Wiley Publishing, Inc. 484.
30. Torres, A. *Uso de XMPP para el transporte de información cooperativa y de contexto*. 2009 [cited 2011 abril]; Available from: <http://www.upcommons.upc.edu/pfc/bitstream/2099.1/6907/1/memoria.pdf>.
31. Hevia, J. and J. Díaz (2007) *Como las administraciones públicas pueden beneficiarse de los estándares abiertos de la mensajería instantánea y P2P*. 10.
32. Díaz, J. *Estudio del protocolo XMPP de mensajería instantánea, de sus antecedentes, y de sus aplicaciones civiles y militares*. 2009 [cited 2011 marzo]; 235]. Available from: <http://oa.upm.es/1481/>.
33. Dean, J., et al., *Client/server messaging protocols in serverless environments*. Academic Press Ltd. London, 2011. **Volume 34** p. 13.

34. Pohja, M., *Server push with instant messaging*. ACM symposium on Applied Computing 2009: p. 653-658.
35. Saint-Andre, P., K. Smith, and R. Tronçon, *XMPP: The Definitive Guide Building Real-Time Applications with Jabber Technologies*. First Edition ed. Vol. 1. 2009: O'Reilly Media, Inc. 306.
36. Paterson, I., et al. *XEP-0124: Bidirectional-streams Over Synchronous HTTP (BOSH)*. 2010 2010-07-02 [cited 2011 febrero]; Available from: <http://xmpp.org/extensions/xep-0124.html>.
37. Martínez, C., *Bases tecnológicas de la Telemedicina. Internet. Servicios y Contenidos. Posibilidades futuras.*, in *Reduca. Serie Medicina*. 2009: Madrid. p. 20.
38. Baxter, A., et al. (2009) *Google Wave Federation Protocol Over XMPP*.
39. Lassen, S. and S. Thorogood (2009) *Google Wave Federation Architecture*.
40. Martín, M., *Comunicación de nuevos productos 2.0. El lanzamiento de Google Wave a través de The Official Google Blog*, in *Revista Latina de Comunicación Social* 2010: Universidad de La Laguna. Islas Canarias. p. 14.
41. Antunez, Y. and L. Ravelo (2010) *Módulo de comunicación para sistemas SCADA usando tecnologías libres*. 19.
42. Sehara, Y. and Y. Companioni (2010) *Herramienta de configuración para sistemas SCADA*. 21.
43. Bouman, R. and J. Dongen, *Pentaho Solutions* 1st ed. 2009: John Wiley & Sons. 648.
44. Sommerville, I., *Ingeniería de software*. 7ma ed. 2005: PEARSON Educación, Madrid.
45. Pressman, R., *Ingeniería de Software, un enfoque práctico.*, ed. E.F. Varela. Vol. 1. 2005, La Habana, Cuba.

GLOSARIO DE TÉRMINOS

AJAX: *Asynchronous JavaScript And XML*, es una técnica de desarrollo web para crear aplicaciones interactivas.

CMI: cuadro de mando integral.

CPU: *Central Processing Unit*, unidad central de procesamiento.

Ejabberd: es un servidor de mensajería instantánea de código abierto implementado en erlang.

HTTP: *Hypertext Transfer Protocol* en español protocolo de transferencia de hipertexto.

IBM: *International Business Machines*, es una empresa multinacional estadounidense que fabrica y comercializa herramientas, programas y servicios relacionados con la informática.

ICQ: *I Seek You*, es un cliente de mensajería instantánea.

Jabber: *Extensible Messaging and Presence Protocol* (XMPP) anteriormente conocido como Jabber es un protocolo libre para mensajería instantánea, basado en el estándar XML y gestionado por *XMPP Standards Foundation*.

MSN Messenger: Programa de comunicación instantánea de la empresa Microsoft.

Oracle: es una empresa con sede central en California, EUA, con operaciones en 145 países.

PC: *Personal Computer*, computadora personal.

PHP: es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas.

TD: tablero digital o *Dashboard*.

XML: *Extensible Markup Language*, lenguaje extensible de marcas.

Yahoo! Messenger: es un cliente de mensajería instantánea.

ANEXOS

Anexo 1. Requisitos Funcionales

RF 1.1 Autenticar usuario.

RF 1.2 Conectar al servidor XMPP.

RF 2.1 Listar los nodos existentes.

RF 3.1 Insertar nombre del servicio PubSub.

RF 3.2 Realizar la suscripción del nodo.

RF 4.1 Recuperar datos publicados antes de la suscripción.

RF 4.2 Visualizar conjunto de datos.

RF 5.1 Eliminar suscripción el nodo.

RF 5.2 Actualizar listado de nodos suscritos.

RF 6.1 Desconectar de servidor XMPP.

RF 7.1 Crear nodo.

RF 7.2 Editar nodo.

RF 7.2.1 Actualizar nodo.

RF 7.3 Eliminar nodo.

RF 8.1 Seleccionar los datos a publicar.

RF 8.2 Publicar datos.

RF 9.1 Crear usuario jabber.

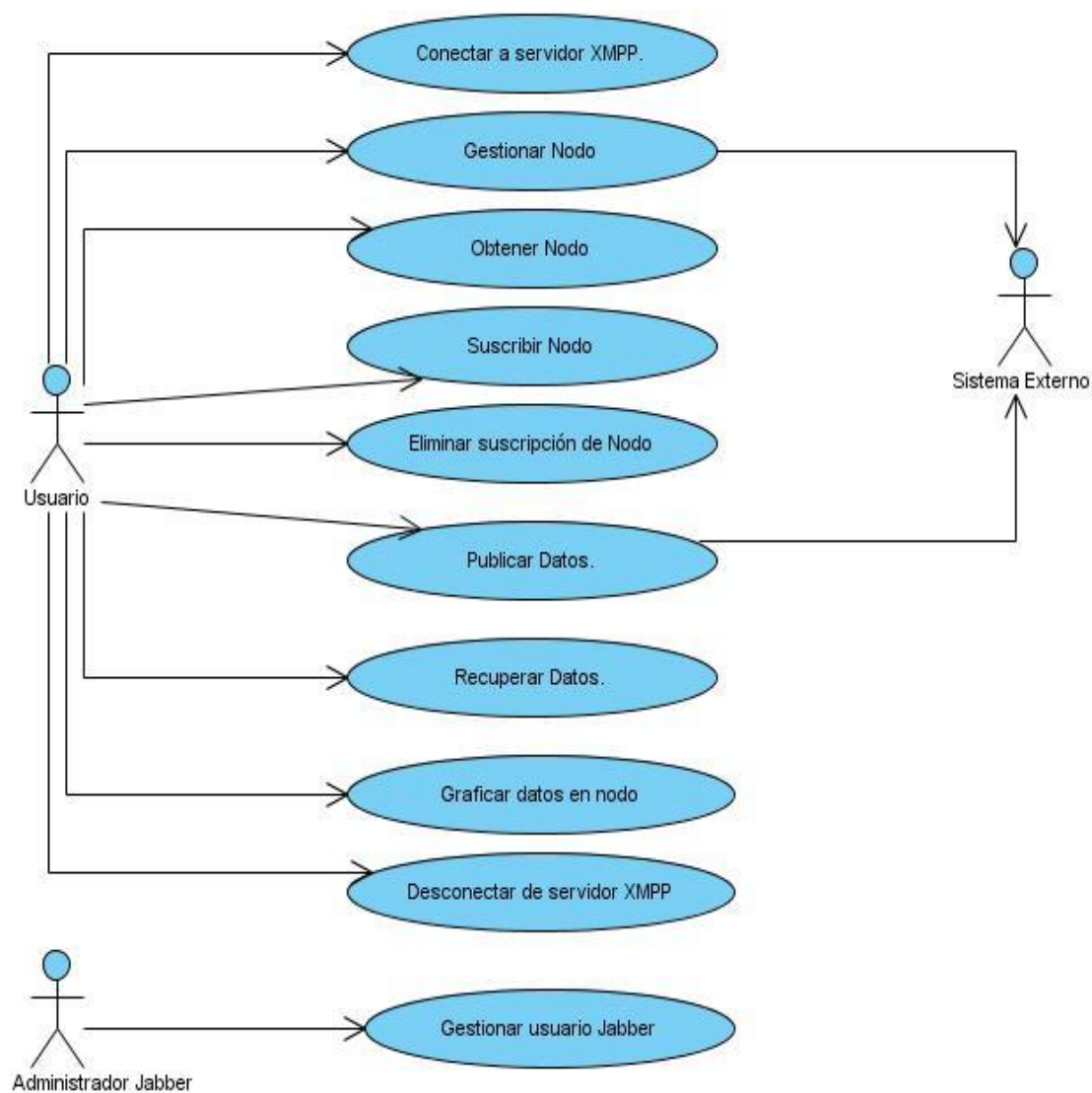
RF 9.2 Editar usuario jabber.

RF 9.2.1 Buscar y visualizar usuario jabber.

RF 9.2.2 Actualizar usuario jabber.

RF 9.3 Eliminar usuario jabber.

RF 10.1 Graficar datos.

Anexo 2. Diagrama del sistema.**Figura 21.** Diagrama de CUS.