Universidad de las Ciencias Informáticas

FACULTAD 6

Título: **Componente de Captura de Métricas v3.0 para Naire**

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autor: Carlos López Durañona

Tutores: Ing. Yoan M. Pérez Piñero

Ing. Rosnel Venero Acosta

La Habana, Junio 2013

"Año 55 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro ser autor del presente trabajo de diploma y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Carlos López Durañona

_____

Firma del Autor

Ing. Yoan M. Pérez Piñero

_____

Firma del Tutor

Ing. Rosnel Venero Acosta

_____

Firma del Tutor

DATOS DE CONTACTO

Tutor:

Ing. Yoan M. Pérez Piñero

Universidad de las Ciencias Informáticas, La Habana, Cuba

Correo electrónico: ymperez@uci.cu

Tutor:

Ing. Rosnel Venero Acosta

Universidad de las Ciencias Informáticas, La Habana, Cuba

Correo electrónico: rvacosta@uci.cu

AGRADECIMIENTOS

Las palabras nunca alcanzan ni aparecen cuando más se les necesita, como en este instante de agradecer a las personas que me han ayudado a llegar hasta aquí.

Quiero agradecer por encima de todo a mi familia que no ha dejado nunca de darme su apoyo, a mi madre que es mi fuerza, y a mi padre que es mi ejemplo. A mis hermanas por su amor y comprensión, y a mi sobrina por su alegría.

Siempre has estado ahí para ayudarme, alegrarme, entristecerme y llenarme de sueños y esperanzas aunque no te lo hayas propuesto. Gracias por todo Rosa, tu siempre serás la única flor de mi jardín.

Gracias a mis tutores Yoan y Rosnel por guiarme y aconsejarme, y a todos mis compañeros que de una u otra forma me han ayudado.

DEDICATORIA

A mis padres que son mi mayor tesoro. Espero que se sientan orgullosos de mí, así como yo lo estoy de ustedes.

RESUMEN

La Universidad de las Ciencias Informáticas constituye un pilar importante en el desarrollo de aplicaciones y soluciones en Cuba. Como parte de su crecimiento surge el Centro de Tecnologías de Gestión de Datos, en el cual se utilizan servidores PostgreSQL para el almacenamiento y gestión de toda la información generada, y la necesidad de contar con un sistema capaz de monitorizar de forma continua dichos servidores. El presente trabajo de diploma describe el proceso de desarrollo de la versión 3.0 del Componente de Captura de Métricas de la herramienta de monitorización de servidores PostgreSQL llamada Naire, permitiendo que la misma sea capaz de monitorizar múltiples clústeres de PostgreSQL por servidor, independientemente de su versión, así como agregar nuevas métricas que permitan obtener una información más detallada del funcionamiento de los servidores.

PALABRAS CLAVE: clúster, métrica, monitorización, PostgreSQL, servidores.

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Con el surgimiento de la informática, se facilitaron un gran número de trabajos repetitivos y monótonos. A medida que se fueron automatizando un mayor número de procesos, la productividad comenzó a incrementarse paulatinamente y se redujo, como consecuencia, el costo de todas las actividades productivas. Hoy en día esta disciplina se aplica a múltiples áreas del conocimiento y la actividad humana, como por ejemplo en la gestión de todo tipo de negocios, industria, robótica, comunicaciones, control de transporte, investigación, diseño computarizado, física, química, almacenamiento y consulta de información, monitorización y control de procesos, entre otras.

El gobierno de Cuba identificó, desde hace muchos años, el desarrollo de la informática en el país como un factor de importancia innegable para el futuro de la economía. En los años 80 del pasado siglo se crea el programa de los Joven Club de Computación, a partir de una idea del líder de la Revolución Fidel Castro Ruz con el principal objetivo de socializar la enseñanza de la computación y la electrónica, dando prioridad a los niños y jóvenes. Comenzó así una serie de transformaciones en el país, como la informatización de múltiples sectores, empresas y la digitalización de los datos manejados por las mismas, por lo que el crecimiento de los volúmenes de información hizo necesario el uso de Sistemas Gestores de Bases de Datos (SGBD) para almacenar, modificar y tener acceso de forma rápida y eficiente a toda la información generada. (1)

Los SGBD constituyen un tipo de software muy específico, dedicado a servir como interfaz entre la base de datos, el usuario, y las aplicaciones que los utilizan. Existen muchos programas en el mercado actual dedicados a la gestión de bases de datos, tales como MySQL, Oracle, Microsoft Acces, Paradox, PostgreSQL, Firebird, SQLite, entre otros. La gran mayoría de los SGBD existentes en la actualidad son de licencia privativa, por lo que se ha optado en el país por el uso de PostgreSQL, de licencia pública, como parte de la política de migración al software libre. PostgreSQL como muchos otros proyectos de código abierto no es manejado por una sola persona y/o empresa, sino por una comunidad de desarrolladores que trabajan de forma desinteresada y libre, denominada PGDG (PostgreSQL Global Development Group).

La generalización del uso de este gestor como SGBD en Cuba ha dado como resultado la creación de la Comunidad Técnica Cubana de PostgreSQL, en la que se ofrece soporte y soluciones a problemas surgidos debido al uso del gestor. Una de las tareas más importantes que realiza un administrador de

bases de datos es la monitorización de los sistemas a su cargo para conocer si se están comportando dentro del margen esperado y poder prevenir futuros problemas. Surge así la necesidad de contar con software que facilite la monitorización y generación de reportes del funcionamiento y uso de servidores PostgreSQL.

En el seno del programa de informatización de la sociedad cubana surge la Universidad de las Ciencias Informáticas (UCI), el 23 de septiembre del 2002, con la misión de formar profesionales comprometidos con su Patria y altamente calificados en la rama de la informática, producir aplicaciones y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación, y servir de soporte a la industria cubana de la informática. La UCI se ha sumado a la política de migración al software libre impulsada en el país, desarrollando diferentes tipos de software y sistemas que sustituyan a sus equivalentes con licencias privadas.

El Departamento PostgreSQL, del Centro de Tecnologías y Gestión de Datos (DATEC) perteneciente a la UCI, ha sido uno de los grupos especializados en sistemas de licencia pública ya existentes, creado con el objetivo de facilitar el uso del gestor del mismo nombre y brindar asesoría a otros proyectos que lo necesiten, además desde su surgimiento ha trabajado para masificar la utilización de PostgreSQL como SGBD por excelencia en el país.

Por la necesidad en Cuba de contar con un sistema de monitorización que permita controlar de manera eficiente múltiples servidores PostgreSQL, se crea Naire, cuyo objetivo es facilitar el trabajo de los administradores de bases de datos en cuanto a la supervisión de los servidores de este tipo, mediante gráficas y otras representaciones de datos.

El flujo de trabajo de Naire utiliza un demonio[1], instalado en cada estación donde se encuentre el servidor PostgreSQL que se desea supervisar, que recoge y envía diferentes tipos de parámetros de las bases de datos mediante consultas directas al gestor. Los datos son recibidos por una aplicación Web encargada de visualizar la información obtenida y construir los reportes.

El desarrollo del sistema comenzó haciendo uso de la versión 8.4 de PostgreSQL y actualmente está funcional para las versiones entre 8.4 y 9.2. La comunidad internacional de este gestor continúa

[1] Disk And Execution MONitor (Daemon). Tipo especial de proceso informático que se ejecuta en segundo plano en lugar de ser controlado directamente por el usuario.

desarrollando funcionalidades y contribuciones que culminan en nuevas versiones del producto, y al realizarse cambios en el catálogo del gestor influyen en el funcionamiento del Componente de Captura de Métricas que forma parte del sistema. Así mismo es posible encontrar empresas que usen versiones inferiores a 8.4 y que no puedan hacer uso de Naire por no tener soporte para ellas.

Además en una misma estación de trabajo (dígase servidor) pueden estar instalados dos o más versiones diferentes de PostgreSQL, en ese caso el sistema solo muestra el último que fue instalado, por lo que no se puede monitorizar el comportamiento de ambos. En la implementación del Componente de Captura de Métricas fue utilizado el paquete de Unix python-pygresql. PyGreSQL es uno de los más antiguos controladores, y entre los cambios de las últimas versiones (4.1) se encuentra que ya no soporta versiones de PostgreSQL anteriores a la 8.3 por lo que dificultaría su uso en empresas que utilicen antiguas versiones de PostgreSQL.

En función de lo antes expuesto se ha identificado el siguiente **problema de investigación**: ¿Cómo mejorar la compatibilidad y la monitorización en el sistema de captura de métricas de Naire?

Se define como **objeto de estudio**: Sistemas de monitorización de bases de datos, enmarcado en el campo de acción: Componentes de captura de métricas en sistemas de monitorización de servidores de base de datos.

Para resolver el problema de investigación se trazó como **objetivo general**: Desarrollar el Componente de Captura de Métricas v3.0 para Naire, que permitirá monitorizar múltiples clústeres de PostgreSQL por servidor independientemente de su versión.

A partir del análisis del objetivo general se derivan los siguientes **objetivos específicos**:

- Analizar las herramientas existentes con funciones similares a Naire así como las versiones anteriores del Componente de Captura de Métricas.
- Diseñar la arquitectura del Componente de Captura de Métricas.
- Implementar el Componente de Captura de Métricas.
- Validar el Componente de Captura de Métricas.

Para dar cumplimiento a los objetivos específicos se trazaron las siguientes **tareas de la investigación**:

- Caracterización de las herramientas y tecnologías a utilizar en el desarrollo del nuevo Componente de Captura de Métricas de Naire.
- Caracterización de la metodología a utilizar en el desarrollo de la investigación.
- Migración del componente utilizando un nuevo controlador de conexión.
- Rediseño de la arquitectura del Componente de Captura de Métricas.
- Definición de las clases de la nueva arquitectura del componente.
- Implementación de las clases.
- Definición de las pruebas que se realizarán al componente.
- Aplicación de las pruebas definidas.

Luego de la realización de las tareas antes mencionadas se espera tener como resultado: Componente de Captura de Métricas v3.0 para Naire.

El presente trabajo se ha estructurado de la siguiente manera:

Capítulo 1: Fundamento Teórico.

El capítulo comprende el estudio del Sistema Gestor de Bases de Datos PostgreSQL, las herramientas de monitorización en tiempo real existentes en el mundo así como sus respectivas tecnologías, enfocado a mejorar las funcionalidades y compatibilidad del sistema de monitorización de Naire con respecto a sus versiones anteriores. También se mencionan las herramientas, tecnologías y la metodología a utilizar para el desarrollo de la nueva versión del componente de monitorización de la presente investigación.

Capítulo 2: Descripción de la solución.

Este capítulo describe las funcionalidades y procesos de las versiones anteriores del Componente de Captura de Métricas así como las que se agregarán en la nueva versión. Además se describen las principales características de este, los requerimientos e Historias de Usuario y se realiza un diagrama de clases del diseño el cual se utiliza como complemento de la metodología definida para lograr un mejor entendimiento del componente.

Capítulo 3: Validación de la solución.

En este capítulo se realizará la validación del componente propuesto anteriormente. Se investigan las diferentes estrategias de prueba definidas por la metodología XP, que permiten verificar el correcto

funcionamiento del componente. Las funcionalidades agregadas al componente serán sometidas a diferentes escenarios de prueba y se comprobará que sus resultados son los esperados, además serán corregidos los errores encontrados.

CAPÍTULO 1: FUNDAMENTO TEÓRICO

Introducción

En la actualidad los SGBD ocupan un lugar de innegable importancia en el manejo de la información, por lo que su constante monitorización es importante para su correcto funcionamiento y futuras mejoras. El presente capítulo comprende el estudio de diferentes definiciones de software y tecnologías referentes a la monitorización de los SGBD. Además de otras herramientas necesarias para el desarrollo de la nueva versión del Componente de Captura de Métricas de Naire.

1.1 Conceptos asociados a la investigación

A continuación se definen una serie de conceptos asociados a la investigación, referentes a las diferentes tecnologías de monitorización de SGBD y los sistemas de comunicación en tiempo real.

Sistemas Gestores de Bases de Datos

Se denomina Sistema Gestor de Base de Datos al software que permite la utilización y/o la actualización de los datos almacenados en una o varias bases de datos por uno o varios usuarios desde diferentes puntos de vista. El objetivo fundamental de un SGBD consiste en suministrar al usuario las herramientas que le permitan manipular, en términos abstractos, los datos, o sea, de forma que no le sea necesario conocer el modo de almacenamiento de los mismos en la computadora, ni el método de acceso empleado. (2)

1.1.1 Monitorización de servidores de bases de datos

La monitorización puede detectar las posibles interferencias que pudieran presentarse en el curso de alguna acción y puede dar lugar a corregir el procedimiento antes de llegar a un resultado final.

Una de las tareas más importantes de un administrador de bases de datos es monitorizar los sistemas a su cargo para saber cómo están funcionando y planear futuras modificaciones y actualizaciones de los mismos.

Existen dos tipos de monitorización (3):

- **Ad Hoc**: Monitorización específica en caso de problemas o pruebas. Se utiliza generalmente para investigar una situación puntual en la que se intenta encontrar una explicación a un suceso, cambio o problema.
- **Preventiva**: Detecta interrupciones de servicios, alerta sobre posibles problemas y crea gráficos con tendencias y datos históricos sobre los sistemas. Este tipo de monitorización está automatizada y ayuda a descubrir cambios en los sistemas que provocan o pueden provocar problemas en un futuro cercano.

Existen numerosos programas que pueden ayudar a realizar la monitorización de sistemas PostgreSQL. A continuación se listan algunos de los software de monitorización de código abierto usados en el mundo:

- **Ad Hoc**: vmstat, iostat, sar, ps, top, iotop, htop.
- **Preventiva**: "Nagios" para detectar y alertar sobre posibles problemas y "Munin" para crear gráficas históricas con las tendencias y uso de los sistemas.

En la *Figura 1* se muestran los elementos a monitorizar, que son todos aquellos que pueden ofrecer información sobre cómo el sistema está funcionando. Algunos de los elementos más importantes que se suelen monitorizar son los siguientes:

- **Servidor**: Disponibilidad y posibles problemas del hardware.
- **CPU**: Carga del sistema y uso de la CPU.
- **Memoria**: Carga y uso de la memoria, uso de la memoria de intercambio (swap).
- **Red**: Disponibilidad de los componentes de red, tráfico de entrada y salida.
- **Discos / almacenamiento**: Espacio utilizado, I/O (Input/Output) del sistema.
- **PostgreSQL**: Número de conexiones, número de transacciones, transacciones abiertas, bloqueos, espacio usado, tipo de comandos usados.

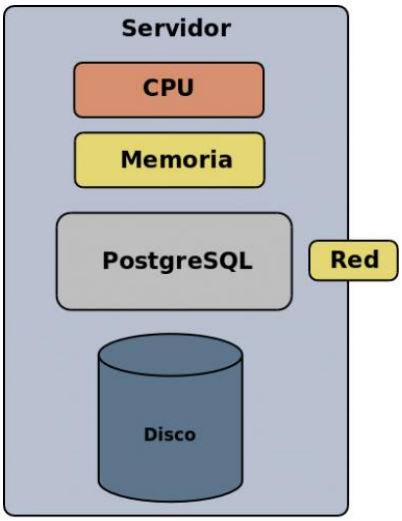

Figura 1: Elementos a monitorizar en un cliente.

Además de estos elementos, suelen existir características específicas de los sistemas que cada administrador deberá identificar y monitorizar de la mejor manera posible. (3)

1.1.2 Sistemas en Tiempo Real

Las aplicaciones de tiempo real son las aplicaciones informáticas en las que la obtención de los resultados está sujeta a unas restricciones temporales impuestas por el entorno en que se ejecutan. Los sistemas que soportan la ejecución de aplicaciones de tiempo real y aseguran el cumplimiento de los requerimientos temporales se denominan sistemas de tiempo real. (4)

El aumento de los volúmenes de información manejados por las bases de datos, hizo necesaria la aparición de la monitorización de servidores en tiempo real, con el principal objetivo de permitir conocer el rendimiento de los mismos. Algunos de los parámetros a monitorizar son: el tiempo de respuesta, el consumo de los recursos del servidor, así como su comportamiento ante determinadas operaciones. La monitorización en tiempo real ayuda al mantenimiento, la verificación y control constante de los parámetros del servidor, y brinda la posibilidad de actuar de forma rápida ante cualquier mal funcionamiento del mismo.

Protocolos de comunicación en tiempo real

Un protocolo es un conjunto de reglas de comunicaciones entre dispositivos (computadoras, teléfonos, enrutadores, conmutadores). Los protocolos gobiernan el formato, sincronización, secuencia y control de errores. Sin estas reglas, los dispositivos no podrían detectar la llegada de bits. (5)

El componente actual de monitorización de Naire hace uso del protocolo STOMP para la implementación del sistema de comunicación en tiempo real, ya que este protocolo trabaja con el formato de datos JSON el cual presenta una estructura muy similar a los documentos de las BD NoSQL de MongoDB, y además permite enviar los datos en tiempo real en el mismo formato que se mostrarán.

Patrones de mensajería

En arquitectura de software un patrón de mensajería es un patrón arquitectónico orientado a redes el cual describe cómo dos partes diferentes de un sistema de intercambio de mensajes se conectan y comunican uno con el otro. Un patrón de intercambio de mensajes (MEP por sus siglas en inglés), en telecomunicaciones, describe el patrón de mensajes requerido por un protocolo de comunicaciones para establecer o utilizar un canal de comunicación. Existen dos tipos generales de patrones de mensajería: el patrón Publicación – Subscripción, y de Una Vía. Por ejemplo HTTP es un protocolo que utiliza el patrón Publicación – Subscripción, y UDP utiliza Una Vía. (6)

El patrón de mensajería que usa el componente de monitorización de Naire actualmente es Publicación – Subscripción ya que mediante este patrón el usuario o aplicación es capaz de subscribirse a un canal específico de información, logrando así una diferenciación de la misma. Una vez subscrito, se les puede enviar información constantemente sin que las aplicaciones clientes la soliciten.

Tecnologías de comunicación con la web

Existen muchas tecnologías y técnicas para lograr comunicación en tiempo real con la web, muchas de ellas se han desarrollado a partir de AJAX[2]. La comunicación en tiempo real ha evolucionado a lo largo del tiempo. En un primer momento la herramienta de comunicación instantánea por excelencia era el chat, después apareció la comunicación sobre voz y las videoconferencias.

Algunas de las tecnologías utilizadas en el mundo en la comunicación en tiempo real son:

[2] Acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo Web para crear aplicaciones interactivas o RIA (Rich Internet Applications).

- Comet: No es una tecnología en sí, sino un conjunto de técnicas mediante las cuales se puede describir e implementar un modelo de diseño web mediante el cual una petición HTTP que se encuentre abierta puede permitir al servidor enviar datos al navegador, sin que este último los solicite.

- BOSH: Stream-Bidireccional sobre HTTP sincrónico, BOSH por sus siglas en inglés, es una tecnología para la comunicación bidireccional a través del Protocolo de Transferencia de Hipertexto (HTTP).

- Orbited: Ofrece un socket de JavaScript/HTML en el navegador. Se trata de una pasarela y firewall web que permite integrar aplicaciones web con sistemas arbitrarios.

El componente de monitorización de Naire utiliza Orbited como tecnología web para la comunicación en tiempo real con el Componente de Reportes. Orbited está desarrollado usando como base a Twisted, el cual es un framework para el desarrollo de aplicaciones orientadas al trabajo con las redes y basado en eventos que permite utilizar tanto XMPP, IRC[3] como STOMP. Mediante el uso de MorbidQ el sistema envía los datos obtenidos al servidor Orbited. MorbidQ es una cola de mensajes simples STOMP para desarrolladores y su principal ventaja es que dicha cola se construye en el servidor Orbited, por lo que es simple habilitarla y configurarla.

1.1.3 Sistema Gestor de Base de Datos PostgreSQL

Es un sistema de gestión de base de datos relacional y libre. Cuenta con más de 20 años de desarrollo activo y una arquitectura probada que se ha ganado una sólida reputación de fiabilidad, integridad y corrección de los datos. Se ejecuta en los principales sistemas operativos, incluyendo Linux, UNIX, Mac OS X, Solaris y Windows. Tiene soporte completo para claves foráneas, uniones, vistas, disparadores y procedimientos almacenados (en varios lenguajes). Cuenta con interfaces de programación nativa para C/C++, Java, Net, Perl, Python, Ruby y posee una documentación excepcional. Es altamente escalable, tanto en la enorme cantidad de datos que puede manejar, como en el número de usuarios concurrentes que puede acomodar. (7)

[3] Protocolo de comunicación en tiempo real basado en texto, que permite debates entre dos o más personas.

Adaptadores de bases de datos PostgreSQL para Python

Un adaptador (o controlador) de bases de datos es una biblioteca que permite a las aplicaciones comunicarse con un SGBD, mediante el uso de funciones específicas y consultas.

El actual Componente de Captura de Métricas de Naire hace uso de PyGreSQL, el cual es un módulo de Python que permite que un programa escrito en este lenguaje pueda establecer una conexión a PostgreSQL. Para sustituirlo se hace una investigación sobre los demás módulos con similares prestaciones, que se describen a continuación:

- Py-bpgsql y Pg8000 son controladores escritos en Python puro, ninguno de los dos se pueden ejecutar en ambientes de producción y no son compatibles con los frameworks más populares.

- Py-Postgresql es actualmente un controlador experimental escrito en Python puro. Usando la interface PG-API, las declaraciones preparadas a nivel de protocolo pueden ser creadas y usadas varias ocasiones. Utiliza la conveniente interfaz COPY para directamente copiar datos de una conexión a otra sin necesidad de archivos intermedios.

- Psycopg2 es un adaptador de bases de datos PostgreSQL para el lenguaje de programación Python. Sus características principales son el soporte completo de la DB API 2.0 de Python y que los hilos pueden compartir las conexiones. Además soporta las versiones de PostgreSQL desde la 7.4 hasta la 9.2. Fue diseñado para pesadas aplicaciones multi-hilos que pueden crear y destruir muchos cursores, además de hacer un gran número de inserciones y actualizaciones concurrentes. La distribución de Psycopg2 incluye el ZPsycopgDA, un adaptador de bases de datos de Zope. (8)

Psycopg2 es mayormente implementado en C como una envoltura de libpq, resultando ser más seguro y eficiente. Algunas de sus características son: cursores del lado del cliente y del servidor, comunicaciones y notificaciones asincrónicas, soporte para COPY TO/COPY FROM, y un flexible sistema de adaptación de objetos. Muchos tipos básicos de Python son soportados y mapeados para concordar con los tipos de datos de PostgreSQL, tales como cadenas (ambas bytes y Unicode), números (ints, longs, floats, decimals), booleans y objetos de hora-fecha, y muchos tipos de objetos binarios. También está disponible el mapeo entre listas y arreglos de PostgreSQL de cualquier tipo soportado, entre diccionarios e historias

de PostgreSQL, y entre tuplas/tuplas-nombradas y tipos compuestos de PostgreSQL. Psycopg2 es compatible con Unicode y Python 3. (8)

Después de la investigación realizada se decide que la opción más óptima es elegir Psycopg2 como nuevo controlador para realizar la conexión con PostgreSQL desde Python, pues el mismo soporta una amplia gama de versiones de PostgreSQL lo que mejoraría la compatibilidad del Componente de Captura de Métricas, además de contar con un amplio soporte y un número de funcionalidades para las comunicaciones asincrónicas mencionadas anteriormente.

Catálogos de PostgreSQL

El catálogo del sistema es el lugar donde un sistema de administración de bases de datos relacionales almacena metadatos de los esquemas, tales como información acerca de tablas y columnas, e información interna. El catálogo de PostgreSQL es un sistema de tablas regulares. Las tablas pueden ser eliminadas y recreadas, por ejemplo cuando se crea una nueva base de datos mediante el comando CREATE DATABASE se inserta una fila en el catálogo pg_database. (9)
La versión actual del Componente de Captura de Métricas de Naire hace uso de los siguientes catálogos del sistema de PostgreSQL:

- pg_database(datname)
- pg_stat_database(datname,numbackends,xact_commit,xact_rollback,tup_deleted,tup_inserted,tup_updated,tup_returned)
- pg_stat_all_tables(n_dead_tup,n_live_tup,idx_scan,seq_scan,seq_tup_read)
- pg_locks()
- pg_stat_all_indexes()

El catálogo de PostgreSQL ha cambiado mucho en cada una de sus versiones, por lo que es probable que siga cambiando. Para solucionar el problema de compatibilidad se plantea diseñar una arquitectura que permita, independientemente de la versión de PostgreSQL instalada en el servidor que se desea monitorizar, hacer uso de los catálogos que estén disponibles y no detener el funcionamiento del componente al ocurrir un error de compatibilidad.

1.2 Herramientas para la monitorización de aplicaciones y bases de datos

La monitorización en tiempo real de los servidores es una opción excelente para empresas que necesiten tener un control específico y regulado, con el objetivo de analizar diferentes tipos de información y estadísticas que permitan optimizar el desempeño de los mismos.
Existen en el mundo una gran cantidad de herramientas que permiten hacer de la monitorización una tarea sencilla.

Con el objetivo de conocer sus características y determinar si dichas herramientas resuelven los problemas actuales del componente de monitorización, se realiza un estudio de algunas de estas aplicaciones, enfocado principalmente en cómo realizan la monitorización de múltiples clústeres de PostgreSQL y sus compatibilidades con diferentes versiones del mismo.

Nagios: Es un sistema de monitorización de redes de código abierto ampliamente utilizado, que vigila los equipos (hardware) y servicios (software) que se especifiquen, alertando cuando el comportamiento de los mismos no sea el deseado. Entre sus características principales figuran la monitorización de servicios de red, de los recursos de sistemas hardware, independencia de sistemas operativos, y la posibilidad de programar plugins específicos para nuevos sistemas. (10)

Esta herramienta presenta una interfaz con la cual es complejo trabajar y además como herramienta de monitorización no está destinada para los servidores de bases de datos sino más bien para servicios de red, por lo que no representa una solución a los problemas del actual Componente de Captura de Métricas de Naire.

PostgreSQL Enterprise Manager: Ha sido diseñado para administradores de bases de datos e instalaciones con grandes volúmenes de datos. Su arquitectura distribuida gestiona todas las instancias de PostgreSQL, en las instalaciones o máquinas virtuales. Los agentes de vigilancia de recogida y envío de datos permiten tener un sistema de administración centralizada al cual se puede acceder desde múltiples consolas o Interfaces de Usuario. Los paneles de control de rendimiento proporcionan métricas personalizadas para la memoria, Entrada/Salida, almacenamiento, la actividad de objetos, sistema operativo y sistema en espera, cada uno con sus propios gráficos, y además permite generar alertas a las condiciones de excepción. Esta es una de las herramientas más recientes de EnterpriseDB enfocada directamente al trabajo de los administradores de bases de datos a nivel empresarial. (11)

PostgreSQL Enterprise Manager es sin duda una excelente herramienta de administración pero presenta el inconveniente de tener licencia privativa, por lo que su uso está restringido por una serie de reglas y derechos de autor, lo cual contrasta con la actual política de migración al software libre impulsada en el país.

Hyperic HQ: Se centra en la monitorización de aplicaciones y la gestión del rendimiento para infraestructuras virtuales y físicas, además proporciona estadísticas de rendimiento sobre las aplicaciones y servidores de bases de datos, web y servicios de red más comunes. Los elementos clave de la arquitectura son: el servidor central, que proporciona una gestión centralizada, la persistencia de los datos, y el agente o demonio, que permite el seguimiento y control de cada plataforma. (12)

La definición de su arquitectura permite monitorizar aplicaciones mediante el uso de un modelo basado en agentes. Un agente Hyperic se ejecuta en cada equipo que desea administrar. Al ejecutarlo por primera vez, el agente auto-descubre los recursos de software que residen en la máquina, y periódicamente hace un re-análisis de los cambios de configuración. Los agentes Hyperic reúnen información sobre disponibilidad, utilización, rendimiento, además guardan logs y realizan seguimiento de eventos, también permiten iniciar acciones para el control software, ejemplo: iniciar y detener los servidores web y de aplicaciones. Los agentes Hyperic envían sus datos de inventario y métricas para el Servidor Hyperic Central**.** El servidor central recibe los datos y los almacena en la base de datos (HQ Database). (13)

Hyperic HQ tampoco ofrece una solución a los problemas actuales del Componente de Captura de Métricas de Naire, pues es un software de monitorización muy general y no está enfocado en la obtención de parámetros de servidores PostgreSQL.

Existen otras técnicas de monitorización a servidores de PostgreSQL como los benchmarks que permiten medir el rendimiento del mismo, ya sea con datos ficticios o aleatorios, como con datos reales de una base de datos. Este tipo de evaluaciones suele consumir muchos recursos de memoria, en especial cuando se hacen pruebas de estrés sobre los servidores para conocer los límites de prestación del mismo. (14)

Después de la investigación realizada sobre las principales herramientas de monitorización existentes se concluye que ninguna resuelve o brinda alguna vía para dar solución a los problemas actuales del

Componente de Captura de Métricas de Naire, lo que hace necesario desarrollar la versión 3.0 de dicho componente.

1.3 Metodología de desarrollo

En el desarrollo del nuevo Componente de Captura de Métricas de Naire se continúa, al igual que en las versiones anteriores, utilizando la metodología de desarrollo Xtreme Programming (XP). Dicha metodología es además la utilizada en el Departamento PostgreSQL al cual pertenece el proyecto Naire, y posee una serie de ventajas que se mencionan a continuación.

Programación Extrema o XP por sus siglas en inglés hace hincapié en el trabajo en equipo. Los gerentes, clientes y desarrolladores son socios iguales en un equipo de colaboración. Esta metodología además implementa un ambiente simple pero eficaz, y permite a los equipos ser altamente productivos. El equipo se auto-organiza en torno al problema, para así resolverlo lo más eficientemente posible. El primer proyecto XP se inició el 6 de marzo de 1996.

XP mejora un proyecto de software en cinco aspectos esenciales, la comunicación, la sencillez, la retroalimentación, el respeto y el coraje. Los programadores que utilizan XP están en constante comunicación con sus clientes y colegas programadores. Mantienen su diseño simple y limpio. Reciben retroalimentación probando su software desde el primer día. Entregan el sistema a los clientes tan pronto como sea posible y ponen en práctica los cambios cuando se sugiere. Cada pequeño éxito profundiza su respeto por las contribuciones únicas de cada miembro del equipo, todos y cada uno. Con esta base, los programadores de la metodología XP son capaces de responder con valentía a las necesidades cambiantes de la tecnología. (15)

1.4 Tecnologías y herramientas a utilizar

Las tecnologías y herramientas son imprescindibles para el desarrollo de cualquier aplicación. A continuación se describen las que serán usadas para el diseño e implementación de la nueva versión del Componente de Captura de Métricas de Naire.

Sistema Operativo: Ubuntu 11.04

Ubuntu 11.04, distribución GNU/Linux basada en Debian GNU/Linux, concentra su objetivo en la facilidad de uso, la libertad de uso, lanzamientos regulares y la facilidad en la instalación. También emplea excelentes herramientas de traducción y accesibilidad, proporciona un entorno robusto y funcional, así

entre sus características se encuentra que el escritorio predeterminado es GNOME, líder como escritorio y como plataforma de desarrollo. (16)

Sistemas de Control de Versiones: Git 1.7.9

Git es un sistema distribuido de control de versiones el cual es de código abierto. Está diseñado para manejar todo, desde pequeños a los proyectos más grandes con gran velocidad y eficiencia. (17)

Herramienta de Modelado: Visual Paradigm for UML 6.4

Es una herramienta profesional y multiplataforma, que permite construir diagramas, generar código desde los diagramas y documentación, también brinda al equipo de desarrollo de software la posibilidad de realizar el análisis y diseño de los sistemas con mayor eficacia. Además utiliza UML como lenguaje de modelado (18), con UML se puede:

- Visualizar: permite expresar de una forma gráfica un sistema para que otras personas puedan entenderlo.
- Especificar: esto quiere decir que se puede especificar las características del sistema.
- Construir: el sistema se puede construir a partir de los diseños.

Lenguaje de Programación: Python 2.7.3

Para la implementación se ha optado por utilizar Python, ya que es un lenguaje que presenta una serie de ventajas que lo hacen muy robusto (19), entre las que se puede contar:

- Es un lenguaje expresivo y un programa Python suele ser más corto que su equivalente en lenguajes como C.
- Es muy legible, su sintaxis es elegante y permite la escritura de programas cuya lectura resulta más fácil que si se usara otro lenguaje de programación.
- Ofrece un entorno interactivo que facilita la realización de pruebas.
- Puede usarse como lenguaje imperativo procedimental o como lenguaje orientado a objetos.
- Posee un rico juego de estructuras de datos que se pueden manipular de modo sencillo.

Base de datos: MongoDB 2.0.3

MongoDB es una nueva generación de sistemas de gestión de base de datos que combina un modelo de almacenamiento de documentos con un potente motor de consultas de una manera simple. Es un SGBD del tipo No SQL orientada a documentos (JSON), entre sus características fundamentales se encuentran (20):

- Libre de esquemas: las claves de un documento no están predefinidas o fijas en ninguna forma, esto permite las migraciones masivas de datos y ofrece a los desarrolladores flexibilidad para el trabajo con los modelos de datos.
- Fácil escalado: su modelo de datos orientado a documentos permite que los datos sean divididos a través de la red, ofrece balanceo de datos y de carga a través del clúster.

- Almacenamiento de funciones JavaScript, colecciones de tamaño fijo, almacenamiento de ficheros y un buen rendimiento.

Entorno de desarrollo: Aptana Studio 3

Aptana Studio 3 es un editor profesional para el desarrollo de páginas web. Soporta HTML5, CSS3, JavaScript, PHP, Ruby y Python, facilitando así enormemente el desarrollo de aplicaciones web y de escritorio a través de funciones como auto completar código, historial de archivos, soporte de FTP y sincronización entre servidor y carpeta local y tiene además un navegador que permite acceder a cada carpeta y archivos locales de forma rápida, advierte si estás utilizando código obsoleto o código en desacuerdo con los estándares establecidos por la W3C y si se está utilizando código erróneo, en cada carpeta y cada archivo de la copia local.

Además de las funciones imprescindibles para todo programador, Aptana Studio 3 dispone de funciones que hacen la programación más fácil, tales como formateo de código, atajos de teclado y presenta el código de forma comprensible y agradable a la vista del programador, cambiando los colores del editor a través de sus diversos temas. (21)

Conclusiones del capítulo

En el presente capítulo se han explicado de forma breve los principales elementos teóricos referentes a los sistemas de monitorización, sistemas gestores de bases de datos, protocolos de comunicación y sus

tecnologías más actuales. La investigación realizada permitió elegir Psycopg2 como nuevo adaptador para las conexiones a las bases de datos de PostgreSQL, así como determinar los catálogos del sistema a los que realiza consultas el componente de monitorización, con el objetivo de mejorar la compatibilidad entre las diferentes versiones de PostgreSQL. Se seleccionaron como herramientas para el desarrollo del nuevo componente Ubuntu 11.04, Git 1.7.9, Visual Paradigm 6.4, Python 2.7.3, MongoDB 2.0.3 y Aptana Studio 3.

CAPÍTULO 2: DESCRIPCIÓN DE LA SOLUCIÓN

Introducción

En el presente capítulo será descrito el flujo de procesos de la versión anterior del Componente de Captura de Métricas de Naire (v2.0) así como las nuevas funcionalidades que se añaden al mismo. Se propone además una nueva arquitectura de clases más especializada, que permita mejorar la forma en que el componente realiza las consultas al catálogo de PostgreSQL y la gestión de errores que se puedan generar durante este proceso. También se definirán los requisitos funcionales y no funcionales del nuevo componente, las Historias de Usuario, y el diagrama de clases del diseño como complemento de la metodología definida, propiciando un mejor entendimiento del componente.

2.1 Flujo actual del proceso

El sistema de monitorización Naire hace uso de un instalador que le solicita al usuario una serie de parámetros de configuración, tales como los datos del servidor PostgreSQL y del servidor MongoDB, e instala un grupo de paquetes necesarios para el funcionamiento del demonio que captura las métricas, copia los archivos del mismo en los directorios del sistema y prepara el ambiente de ejecución.

Al iniciarse el demonio comienza la captura de información tanto del sistema operativo como de PostgreSQL. Estos datos son enviados en tiempo real a través de Orbited mediante el protocolo STOMP, y almacenados también en una base de datos en MongoDB. En el proceso de captura de métricas se utiliza el controlador de conexión PygreSQL para realizar las consultas al gestor.

La arquitectura actual del Componente de Captura de Métricas está diseñada para monitorizar solamente un clúster PostgreSQL por cada servidor y la gestión de errores hace imposible el funcionamiento del sistema con versiones antiguas de PostgreSQL, sumándose la compatibilidad limitada de PygreSQL con dichas versiones.

2.2 Modelo de dominio

Suele utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema. Es utilizado además por el analista como un medio para comprender el sector de negocios al cual el sistema va a servir.

El modelo de dominio puede ser tomado como el punto de partida para el diseño del sistema. Cuando se realiza la programación orientada a objetos, el funcionamiento interno del software va a imitar en alguna medida a la realidad, por lo que el mapa de conceptos del modelo de dominio constituye una primera versión del sistema. (22)

En la *Figura 2* se muestra el modelo de dominio del actual Componente de Captura de Métricas de Naire. En el mismo se observa como un administrador monitoriza múltiples servidores PostgreSQL, los cuales contienen información del gestor (capturada a través de consultas al catálogo y vistas estadísticas) y del sistema en el que se ejecuta el componente, que se guarda en un servidor MongoDB y se envía en tiempo real a un servidor Orbited.

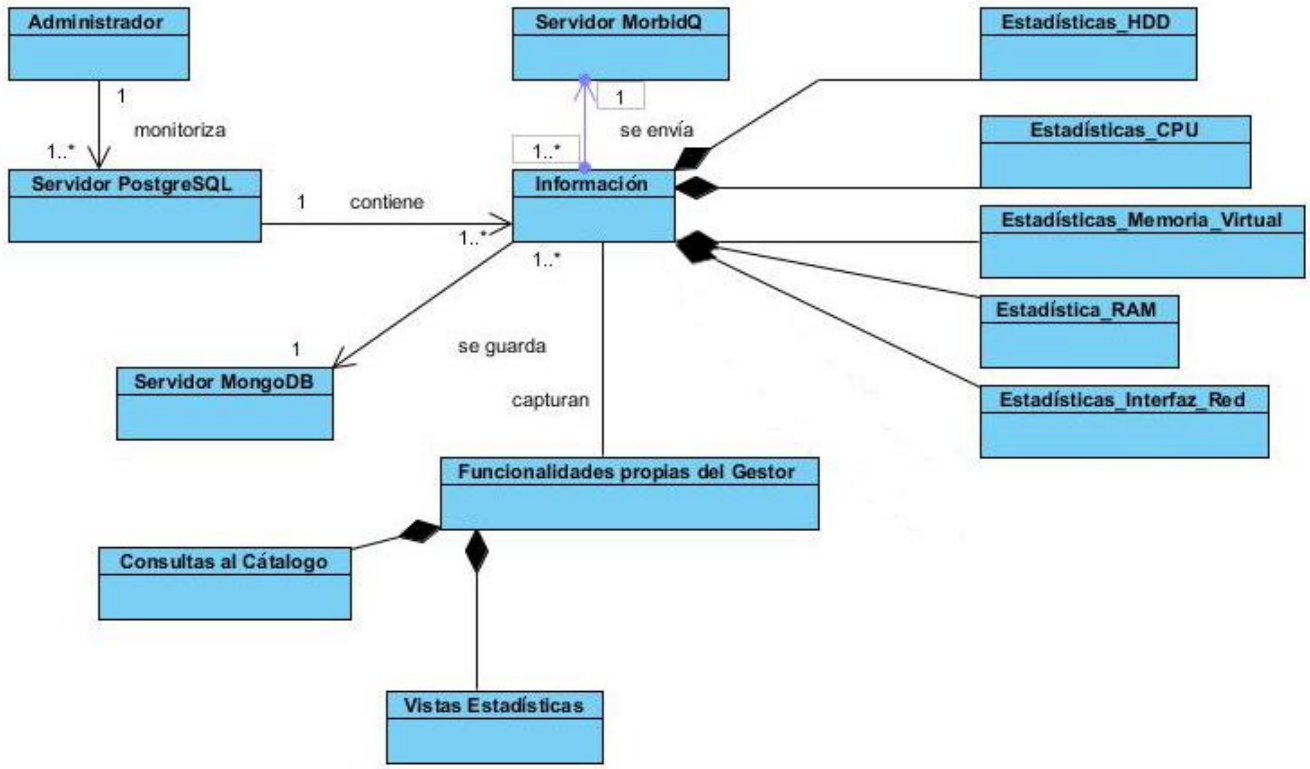

Figura 2: Modelo de dominio del Componente de Captura de Métricas de Naire.

Administrador: Representa al administrador del sistema, es el encargado de comprobar que todos los parámetros del servidor funcionen correctamente.

Servidor PostgreSQL: Son los clústeres de PostgreSQL a monitorizar por parte del sistema y que contienen toda la información que se quiere chequear.

Información: Representa el conjunto de información de las métricas obtenidas de los clústeres de Postgres y del sistema en general.

Servidor Orbited: Es el encargado de enviar en tiempo real toda la información referente a las métricas.

Servidor MongoDB: Representa al servidor MongoDB en el cual se almacena toda la información referente a las métricas.

Funcionalidades propias del gestor: Representa un conjunto de tablas y vistas estadísticas del gestor PostgreSQL mediante las que se obtienen diferentes métricas del mismo.

Estadísticas_HDD: Representa el conjunto de métricas que obtienen información del estado del disco duro, tales como el espacio utilizado y restante, entre otras.

Estadísticas_CPU: Representa el conjunto de métricas que obtienen información del estado del microprocesador, como el porciento utilizado.

Estadísticas_Memoria_Virtual: Representa el conjunto de métricas que obtienen información del estado de la memoria virtual, tales como la cantidad utilizada y restante.

Estadísticas_RAM: Representa el conjunto de métricas que obtienen información del estado de la memoria RAM, tales como la cantidad utilizada y restante.

Estadísticas_Interfaz_Red: Representa el conjunto de métricas que obtienen información del estado de la red, tales como la cantidad de paquetes enviados y recibidos.

2.3 Arquitectura propuesta

Con el objetivo de lograr que el Componente de Captura de Métricas de Naire realice la monitorización de múltiples clústeres de PostgreSQL y mejore la compatibilidad con los mismos es necesario redefinir la arquitectura actual del sistema, incluyendo el nuevo controlador de conexiones a PostgreSQL Psycopg2 y rediseñar el instalador del demonio de forma tal que le solicite al usuario el número de clústeres a monitorizar y los datos de cada uno por separado. Además con la aplicación de patrones GRASP (Patrones Generales de Software para Asignación de Responsabilidades) se separan responsabilidades de algunas clases lo que conlleva a la creación de otras con funciones más especializadas.

En la *Figura 3* se ilustra el proceso de instalación y la nueva arquitectura del Componente de Captura de Métricas de Naire utilizando Psycopg2 como controlador de conexión a los clústeres de PostgreSQL.

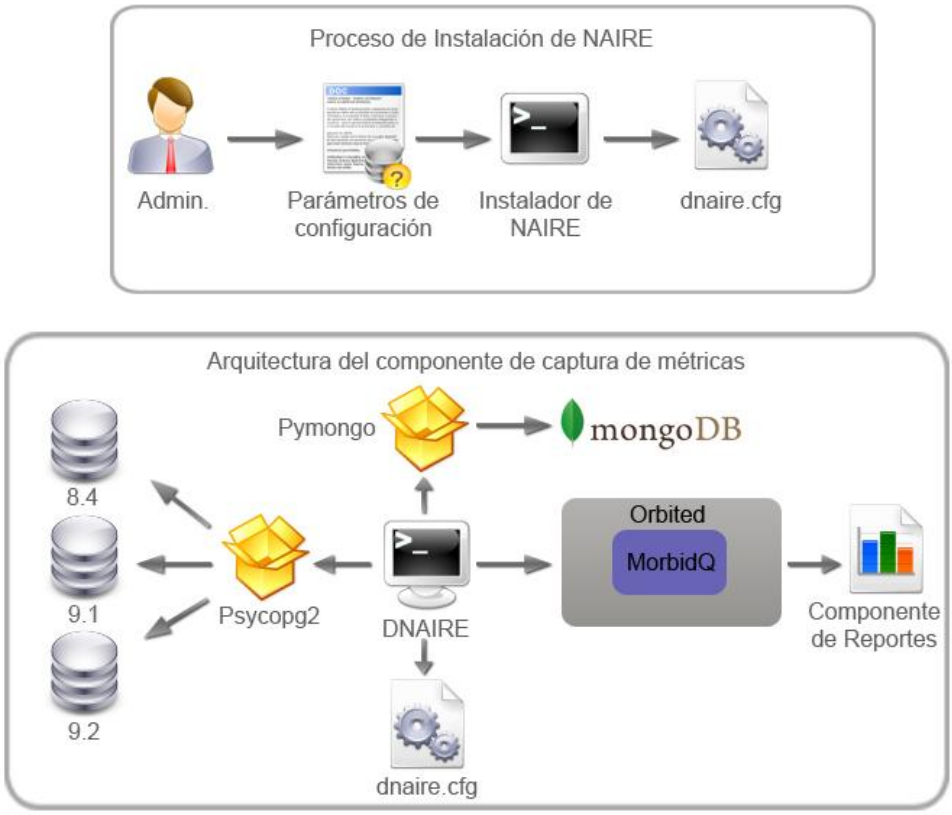

Figura 3: Nueva arquitectura del Componente de Captura de Métricas de Naire.

2.4 Descripción del sistema propuesto

El Componente de Captura de Métricas a desarrollar en su versión 3.0 es el encargado de realizar la monitorización de múltiples clústeres de PostgreSQL independientemente de su versión. El administrador deberá instalar el demonio en el servidor donde se encuentran los clústeres a monitorizar.

El instalador ha sido reprogramado en BASH, que es el intérprete de comandos por defecto en un gran número de distribuciones de Linux, y después de comprobar que el usuario se ha autenticado como administrador, copia e instala todos los paquetes necesarios para el funcionamiento del demonio, además le solicita al administrador el número de clústeres a monitorizar y los datos de cada uno por separado (versión, puerto, usuario y contraseña), le solicita también otros parámetros para la conexión con el servidor MongoDB y Orbited, y el intervalo de tiempo en el que se enviarán los datos.

Al concluir la instalación se almacenan los parámetros de configuración en el archivo dnaire.cfg en la carpeta del sistema /etc/. En dicho archivo a cada clúster se le asigna un número según el orden en que el usuario haya introducido los datos.

Al iniciarse el demonio, carga los datos de configuración y comienza el proceso de obtención y envío de los datos. En la versión anterior del Componente de Captura de Métricas, la conexión tanto a PostgreSQL como a MongoDB se hacía en la misma clase que obtiene las diferentes métricas. La nueva versión del componente separa las clases según sus responsabilidades haciendo uso del patrón experto, así se crean dos clases para realizar las conexiones a los dos SGBD.

En la clase que realiza la conexión a PostgreSQL se utiliza el controlador Psycopg2 en lugar del anterior PygreSQL, sin embargo la forma en que los paquetes obtienen los datos al realizar las consultas no es exactamente igual pues PygreSQL obtiene la información en forma de lista de diccionarios mientras que Psycopg2 la obtiene en forma de tuplas, por lo que fue necesario implementar una función que convirtiera las tuplas obtenidas en los diccionarios requeridos por el componente. Además la clase de conexión a PostgreSQL recibe como parámetro el número del clúster al que se va a conectar y obtiene sus datos del archivo de configuración dnaire.cfg.

La otra clase de conexión es la encargada de realizar las operaciones con MongoDB mediante la biblioteca PyMongo. La clase MetricsDSS es la encargada de obtener las métricas de bases de datos y del sistema. Para lograr que el componente monitorice todos los clústeres de PostgreSQL se crea una instancia de la clase conexión a dicho gestor por cada una de las versiones especificadas por el

administrador y almacenadas en el archivo de configuración. Posteriormente se ejecuta cada métrica de bases de datos para cada una de las conexiones y se almacena la información en una lista para su envío en tiempo real y su almacenamiento además en MongoDB.

En versiones anteriores la gestión de errores no se realizaba de manera eficiente, lo que afectaba directamente la compatibilidad del sistema con las diferentes versiones de PostgreSQL, pues al realizar una consulta al catálogo y no encontrar la tabla o el campo especificado se generaba un error que culminaba en la detención del demonio.

En la versión 3.0 del Componente de Captura de Métricas de Naire se mejora notablemente la gestión de errores, pues cada conexión manipula sus excepciones de manera independiente sin que un error en una consulta afecte el funcionamiento general del sistema, así aunque en una versión antigua de PostgreSQL no exista una tabla o campo requerido para la ejecución de una métrica, el componente registra el error e intenta ejecutar el resto de las métricas sin detener su funcionamiento. Además con las nuevas modificaciones si en el momento de generar las conexiones con cada clúster ocurre un problema con la conexión, ya sea que el clúster no está funcionando correctamente o que los parámetros introducidos por el usuario en la configuración no son válidos, el sistema genera un error e intenta realizar el resto de las conexiones. Esto garantiza la independencia de las mismas y le brinda una mayor estabilidad al componente.

En la nueva versión del componente se adicionan algunas métricas, que amplían la gama de información que se genera y se muestra al usuario, tales como el número de conexiones abiertas a cada clúster en el servidor PostgreSQL, la velocidad de conexión de red, la descripción del sistema operativo en el que se encuentra ejecutándose el demonio, una lista con los procesos que más CPU y memoria RAM están consumiendo, y los procesos activos de PostgreSQL.

2.5 Historias de usuario

Para especificar los requisitos de software se utilizan las Historias de Usuario (HU), que son tarjetas en las cuales el cliente describe brevemente las características que el sistema debe poseer. Estas potencian la participación del equipo en la toma de decisiones, se crean y evolucionan a medida que el proyecto avanza, son peticiones concretas y pequeñas, contienen la información imprescindible y apoyan la

cooperación y conversación entre los miembros del equipo. A continuación se listan las HU definidas en el diseño del nuevo Componente de Captura de Métricas de Naire y se muestran las más significativas:

- HU-1 Crear conexiones a los clústeres de PostgreSQL
- HU-2 Capturar las métricas de cada clúster de PostgreSQL
- HU-3 Salvar las colecciones en el servidor MongoDB
- HU-4 Capturar posibles excepciones de compatibilidad
- HU-5 Capturar número de conexiones a cada clúster de PostgreSQL
- HU-6 Capturar la velocidad de la conexión de red actual
- HU-7 Capturar la descripción del Sistema Operativo
- HU-8 Capturar los procesos que más RAM están consumiendo
- HU-9 Capturar los procesos que más CPU están consumiendo
- HU-10 Capturar los procesos activos de PostgreSQL

En las *Tablas 1, 2 y 3* se muestran algunas de las Historias de Usuario generadas, en las que se especifican requisitos de software necesarios en la implementación del nuevo Componente de Captura de Métricas de Naire. Las demás HU pueden ser vistas en los ANEXOS.

Tabla 1: HU Crear conexiones a los clústeres de PostgreSQL

Historia de Usuario	
Número: 1	Nombre de la Historia de Usuario: Crear conexiones a los clústeres de PostgreSQL
Cantidad de modificaciones a la Historia de Usuario: *Ninguna*	
Usuario: Carlos López Durañona	Iteración asignada: 1

Prioridad en negocio: Muy Alta	Puntos estimados: 1 semana
Riesgo en desarrollo: Muy Alto	Puntos reales: 1 semana
Descripción: Permite la creación de una conexión por cada clúster de PostgreSQL especificado en la instalación del demonio.	
Observaciones: Ninguna.	
Prototipo de interfaces: No presenta prototipo de interfaz de usuario ya que la aplicación es un demonio del sistema operativo.	

Tabla 2: HU Capturar las métricas para cada clúster de PostgreSQL

Historia de Usuario	
Número: 2	Nombre de la Historia de Usuario: Capturar las métricas para cada clúster de PostgreSQL
Cantidad de modificaciones a la Historia de Usuario: *Ninguna*	
Usuario: Carlos López Durañona	Iteración asignada: 1
Prioridad en negocio: Muy Alta	Puntos estimados: 1 semana
Riesgo en desarrollo: Alto	Puntos reales: 1 semana
Descripción: Permite capturar cada métrica del sistema por cada una de las conexiones a los clústeres de PostgreSQL creadas anteriormente.	
Observaciones: Ninguna.	

Prototipo de interfaces: No presenta prototipo de interfaz de usuario ya que la aplicación es un demonio del sistema operativo.

Tabla 3: HU Salvar las colecciones en el servidor MongoDB

Historia de Usuario	
Número: 3	Nombre de la Historia de Usuario: Salvar las colecciones en el servidor MongoDB
Cantidad de modificaciones a la Historia de Usuario: *Ninguna*	
Usuario: Carlos López Durañona	Iteración asignada: 1
Prioridad en negocio: Muy Alta	Puntos estimados: 0.2 semanas
Riesgo en desarrollo: Alto	Puntos reales: 0.2 semanas
Descripción: Permite guardar la información de las métricas recogidas por el demonio en forma de colecciones en una base de datos del servidor MongoDB.	
Observaciones: Ninguna.	
Prototipo de interfaces: No presenta prototipo de interfaz de usuario ya que la aplicación es un demonio del sistema operativo.	

2.6 Lista de reserva del producto

La lista de reserva del producto es un artefacto que contiene los requisitos funcionales que debe cumplir la aplicación que se desea realizar, ordenados según la prioridad de implementación, ubicados en Muy Alta, Alta, Media y Baja, en esta última aparecen los requisitos de menor complejidad además de los requisitos no funcionales del sistema a desarrollar. Indica además la estimación de cada uno de ellos y su implementación por semanas además del rol que hizo la estimación. La *Tabla 4* muestra la lista de reserva

del producto en la que se muestran los requisitos funcionales y no funcionales del nuevo Componente de Captura de Métricas de Naire.

Tabla 4: Lista de reserva del producto del Componente de Captura de Métricas de Naire

Ítem *	Descripción	Estimación	Estimado por
Prioridad: Muy Alta			
1	Crear conexiones a los clústeres de PostgreSQL	1	Programador
2	Capturar las métricas para cada clúster de PostgreSQL	1	Programador
3	Salvar las colecciones en el servidor MongoDB	0.2	Programador
4	Capturar posibles excepciones de compatibilidad	0.2	Programador
Prioridad: Alta			
5	Capturar número de conexiones a cada clúster de PostgreSQL	0.2	Programador
6	Capturar la velocidad de la conexión de red actual	0.2	Programador
7	Capturar la descripción del Sistema Operativo (SO)	0.2	Programador
8	Capturar los procesos que más RAM están consumiendo	0.2	Programador
9	Capturar los procesos que más CPU están consumiendo	0.2	Programador
10	Capturar los procesos activos de PostgreSQL	0.2	Programador
Requisitos no funcionales			
1	**Hardware** Para el cliente (Componente de Captura de Métricas):		

	• Microprocesador Pentium 4 o Superior. • 1 GB de RAM o superior. • 5 GB de espacio libre en disco. **Software** Para el cliente(Componente de Captura de Métricas): • Sistema gestor de bases de datos PostgreSQL. • Se aconseja tener instalado como sistema operativo Debian en su versión 6.0.5 o cualquiera de sus distribuciones. • Se requiere tener instaladas las librerías python-psycopg2, python-psutil, python-pymongo, stompservice, python-twisted, python-demjson, python-stompy, python-netifaces, sysstat, hddtemp, python-setuptools, python-dev.		

2.7 Tareas de la ingeniería

Son las entradas de trabajo para el equipo de programadores, cada una de estas presenta una característica del sistema.

El equipo de desarrollo evalúa cada escenario, entiéndase por escenario Historias de Usuario o requisitos, y lo divide en tareas donde cada una de estas representa una característica del sistema. A continuación (*Tablas 4 y 5*) aparecen dos tareas de la ingeniería (TI) donde se muestra el encargado de programarlas y una descripción breve de lo que se necesita para hacerlo.

Tabla 5: Tarea de Ingeniería 1. Solicitar datos de los clústeres de PostgreSQL

Tarea de Ingeniería	
Número Tarea: 1	Número Historia de Usuario: 1
Nombre Tarea: Solicitar datos de los clústeres de PostgreSQL	
Tipo de Tarea : Desarrollo	Puntos Estimados: 0.4
Fecha Inicio: 4/02/2013	Fecha Fin: 5/02/2013
Programador Responsable: Carlos Miguel López Durañona	
Descripción: Es la encargada de solicitarle al administrador que introduzca los datos para las conexiones con los clústeres de PostgreSQL en el proceso de instalación del Componente de Captura de Métricas de Naire.	

Tabla 6: Tarea de Ingeniería 2. Cargar configuración de los clústeres de PostgreSQL

Tarea de Ingeniería	
Número Tarea: 2	Número Historia de Usuario: 1
Nombre Tarea: Cargar configuración de los clústeres de PostgreSQL	
Tipo de Tarea : Desarrollo	Puntos Estimados: 0.4
Fecha Inicio: 6/02/2013	Fecha Fin: 7/02/2013
Programador Responsable: Carlos Miguel López Durañona	

Descripción: Es la encargada cargar los parámetros de conexión de los clústeres de PostgreSQL desde el archivo de configuración de Naire.

2.8 Plan de iteraciones

Después de tener ya definidas las Historias de Usuario es necesario crear un plan de iteraciones donde se indiquen las Historias de Usuario que se crearán para cada versión del programa. Un plan de iteraciones se utiliza para la planificación, donde los desarrolladores y clientes establecen los tiempos de implementación ideales de las Historias de Usuario, además de determinar la prioridad con la que serán desarrolladas las funcionalidades y las historias en cada versión del programa.

En la *Tabla 7* se muestra el Plan de Iteraciones del nuevo Componente de Captura de Métricas de Naire, en el que se indican las Historias de Usuario a desarrollar en cada iteración y su duración. En la primera iteración serán implementadas las funcionalidades con una prioridad muy alta, descritas en las HU 1, 2, 3 y 4, con una duración total de dos semanas y dos días. Durante la segunda iteración se desarrollarán el resto de las funciones con una prioridad alta, descritas en las HU 5, 6, 7, 8, 9 y 10, con una duración total de seis días. La duración total de la etapa de implementación será de tres semanas y tres días.

Tabla 7: Plan de Iteraciones del Componente de Captura de Métricas de Naire

Release	Descripción de la iteración	Orden de la HU a implementar	Duración total
1	En esta iteración se realizarán las Historias de Usuario de prioridad Muy Alta, estas Historias de Usuario se encargarán del proceso realizar las conexiones a los clústeres de PostgreSQL así como de capturar, enviar y guardar las métricas por cada uno, también se mejorará el sistema de manejo de excepciones.	1-2-3-4	2.4 semanas
2	En esta iteración se realizarán las Historias de Usuario de prioridad Alta, las que se encargan del proceso de implementación de nuevas métricas que permitan conocer datos de las conexiones de red y del sistema operativo, entre otros.	5-6-7-8-9-10	1.2 semanas

2.9 Modelo del diseño

Un diseño simple influye positivamente en el avance del desarrollo del producto de software ya que existe la posibilidad de que los requisitos cambien, por lo que se debe diseñar lo requerido según las necesidades del problema. La metodología XP permite que el diseño final sea de la manera más simple al agrupar todos los requisitos.

2.9.1 Diagrama de clases

En la *Figura 4* se muestra el diagrama de clases que representa el nuevo Componente de Captura de Métricas. La clase Monitor, que extiende de Daemon, es la que se ejecuta como un proceso del sistema y llama a la función send_data() de la clase DataStompSender. La misma extiende de StompClientFactory

de la que hereda las funcionalidades para el envío en tiempo real de los datos obtenidos de MetricsDSS a través de los canales de información creados en la clase Channels. Las conexiones a PostgreSQL y a MongoDB se realizan mediante las clases Conexión_Postgres y Conexión_Mongo respectivamente. Todos los parámetros de conexión y envío de datos se obtienen del fichero de configuración mediante la clase NaireConfigLoader.

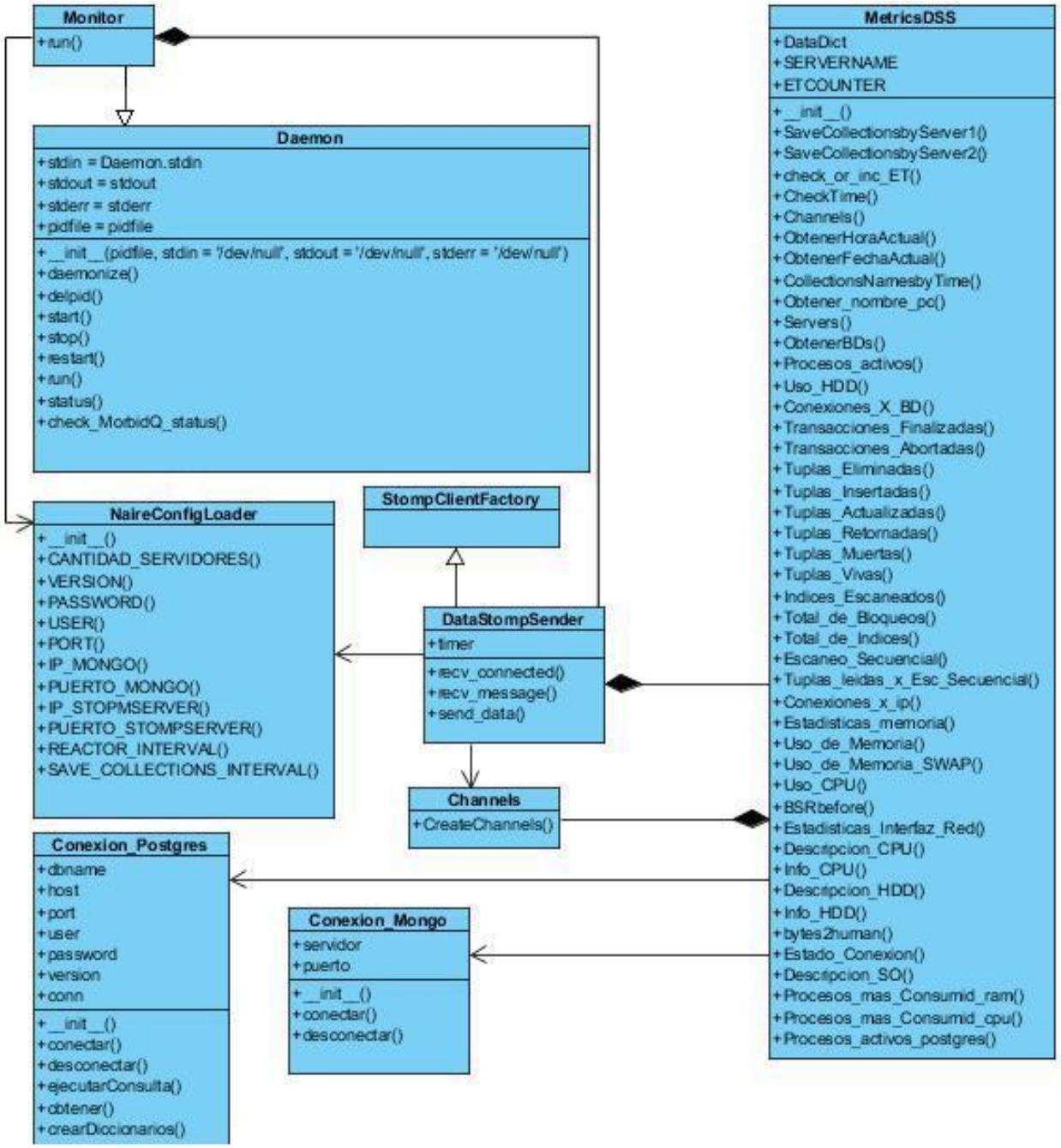

Figura 4: Diagrama de clases del nuevo Componente de Captura de Métricas de Naire.

2.9.2 Tarjetas CRC

Las tarjetas CRC (Clase, Responsabilidad y Colaboración) son artefactos, utilizados en la metodología XP para el diseño de software orientado por objetos. Estas tarjetas deben de estar bien enfocadas para de esta forma contribuir a la comunicación y el buen entendimiento del funcionamiento entre las clases de este trabajo. No deben ser muy largas, pero deben contener la información necesaria para lograr los resultados esperados. A continuación se muestran las clases más relevantes a través de las tarjetas CRC. En las *Tablas 8 y 9* se muestran dos de las tarjetas CRC generadas durante el diseño del nuevo Componente de Captura de Métricas de Naire, donde se observan las responsabilidades de las clases así como sus colaboraciones con otras. Las demás tarjetas CRC pueden ser vistas en los ANEXOS.

Tabla 8: Tarjeta CRC 1. Clase Monitor

Tarjeta CRC	
Clase: Monitor	
Responsabilidades	**Colaboraciones**
• Iniciar, Detener y Reiniciar el demonio. • Iniciar el Reactor de Twisted	Daemon DataStompSender NaireConfigLoader

Tabla 9: Tarjeta CRC 2. Clase MetricsDSS

Tarjeta CRC	
Clase: MetricsDSS	
Responsabilidades	**Colaboraciones**
• Establecer conexión al servidor MongoDB. • Establecer conexiones con los clústeres de PostgreSQL. • Capturar bases de datos. • Capturar nombre de la PC. • Capturar estadísticas de la	Conexion_PostgreSQL Conexion_MongoDB NaireConfigLoader Channels

memoria. • Capturar estadísticas de la memoria virtual. • Capturar estadísticas del disco duro • Capturar cantidad de conexiones. • Capturar cantidad de procesos activos por el servidor PostgreSQL. • Capturar estadísticas de tablas. • Capturar cantidad de transacciones. • Capturar uso de los índices. • Capturar estadísticas del CPU. • Capturar estadísticas de la interfaz de red. • Capturar estadísticas del uso del CPU. • Capturar estadísticas del uso de la memoria RAM. • Capturar descripción del Sistema Operativo. • Capturar estado de la conexión de red. • Crear colecciones JSON por instantes de tiempo.	

2.10 Patrón de arquitectura

Los patrones de arquitectura ayudan a entender el funcionamiento del sistema, para de esta forma lograr su implementación lo más organizada posible, son los encargados de capturar los elementos esenciales de una arquitectura de software**.** Los patrones arquitectónicos brindan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. (23)

En la versión 3.0 se continuó utilizando el patrón de Arquitectura Basada en Eventos usada en la versión anterior.

Las arquitecturas basadas en eventos se vinculan históricamente con sistemas basados en actores, daemons y redes de conmutación de paquetes (publicación-suscripción). Los conectores de estos sistemas incluyen procedimientos de llamada tradicionales y vínculos entre anuncios de eventos e invocación de procedimientos. La idea dominante en la invocación implícita es que, en lugar de invocar un procedimiento en forma directa (como se haría en un estilo orientado a objetos) un componente puede anunciar mediante difusión uno o más eventos. Un componente de un sistema puede anunciar su interés en un evento determinado asociando un procedimiento con la manifestación de dicho evento. (23)

En la *Figura 5* se muestra el uso de la Arquitectura Basada en Eventos aplicada al nuevo Componente de Captura de Métricas de Naire, en la misma se observa cómo el sistema obtiene información de los clústeres de PostgreSQL al ocurrir un evento cada cierto tiempo, definido por el usuario en la instalación del daemon, y publica los datos obtenidos mediante el uso de canales a los que se subscribe el Componente de Reportes para la visualización de la información.

Figura 5: Representación del patrón de arquitectura dirigida por eventos aplicado en Naire.

2.11 Patrones de diseño

Los patrones de diseño permiten lograr soluciones de software de mayor calidad, al ser la base para la búsqueda de soluciones a problemas comunes en el desarrollo de aplicaciones. A continuación se definen los patrones empleados en la nueva versión del Componente de Captura de Métricas de Naire.

Creador

La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos. En consecuencia, conviene contar con un principio general para asignar las responsabilidades concernientes a ella. El diseño, bien asignado, puede soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reusabilidad. El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento. (24)

La clase MetricsDSS tiene datos de inicialización que se pasan a un objeto de la clase Channels (ver *Figura 6*) cuando es creado, además crea instancias de otras clases con las que se relaciona tales como NaireConfigLoader, Conexion_PostgreSQL y Conexion_Mongo.

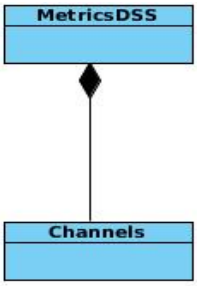

Figura 6: Ejemplo del uso del patrón creador entre las clases MetricsDSS y Channels.

Experto

Experto es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele ser útil en el diseño orientado a objetos. El cumplimiento de una responsabilidad requiere a menudo información distribuida en varias clases de objetos. Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con

ello definiciones de clase sencillas y más cohesivas que son más fáciles de comprender y de mantener. Así se brinda soporte a una alta cohesión. (24)

Las clases Conexion_PostgreSQL y Conexion_Mongo (ver *Figura 7*) son dos ejemplos en los que se utiliza el patrón experto, pues esas clases se especializan en la conexión con sus respectivos gestores.

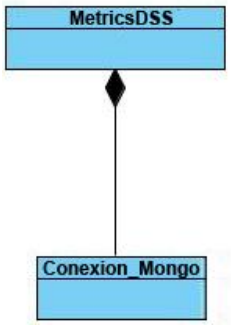

Figura 7: Ejemplo de la aplicación del patrón experto en la clase Conexión_Mongo.

Bajo acoplamiento

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Acoplamiento bajo significa que una clase no depende de muchas clases, mientras que acoplamiento alto significa que una clase recurre a muchas otras clases. Esto presenta los siguientes problemas (24):

- Los cambios de las clases afines ocasionan cambios locales.
- Difíciles de entender cuando están aisladas.
- Difíciles de reutilizar puesto que dependen de otras clases.

Conexion_PostgreSQL (ver *Figura 8*) es un ejemplo de clase en la que se utiliza el patrón bajo acoplamiento pues solo depende de NaireConfigLoader para su funcionamiento.

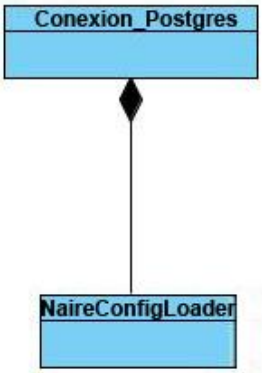

Figura 8: Ejemplo de aplicación del patrón bajo acoplamiento en el sistema.

Alta cohesión

La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Una baja cohesión hace muchas cosas no afines o realiza trabajo excesivo.
Esto presenta los siguientes problemas (24):

- Son difíciles de comprender.
- Difíciles de reutilizar.
- Difíciles de conservar.
- Las afectan constantemente los cambios.

En la *Figura 9* se muestra cómo la clase Monitor está altamente cohesionada con DataStompSender al ser responsables del envío en tiempo real de los datos obtenidos de los clústeres de PostgreSQL y del sistema operativo.

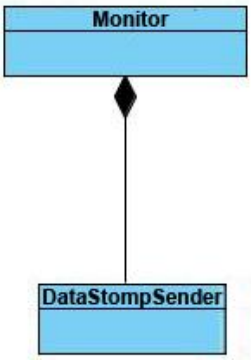

Figura 9: Ejemplo de aplicación del patrón alta cohesión entre las clases Monitor y DataStompSender

2.12 Estándares de codificación

El estilo de programación (también llamado estándares de código o convención de código) es un término que describe convenciones para escribir código fuente en ciertos lenguajes de programación.

El estándar de codificación usado en la implementación del Componente de Captura de Métricas en su versión 3.0 es la Propuesta de Mejora para Python número 8 o PEP-8 por sus siglas en inglés, el cual es el nombre clave de la Guía de estilo para código Python publicada en julio de 2001.

Conclusiones del capítulo

En el presente capítulo fueron definidas las principales características del desarrollo del componente. Se identificaron diez requisitos funcionales y seis no funcionales del sistema, y se describieron las Historias de Usuario de cada una de las diez funcionalidades implementadas. Con el objetivo de tener un mejor entendimiento del componente a desarrollar se definió un modelo de dominio, así como un diagrama de diseño en el que se relacionan las nueve clases del sistema en correspondencia con el mismo número de tarjetas CRC. Fueron definidos, para utilizar en la creación de la versión 3.0 del Componente de Captura de Métricas de Naire, como patrón de arquitectura el basado en eventos, y como patrones de diseño el creador, experto, bajo acoplamiento y alta cohesión.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN

Introducción

En el presente capítulo se valida el componente propuesto anteriormente, seleccionando la estrategia de prueba a realizar y el método de prueba más apropiado de acuerdo a la metodología utilizada, los cuales permiten comprobar el correcto funcionamiento del sistema y las funcionalidades implementadas. Además son mostrados los resultados en cada una de las iteraciones del proceso de desarrollo del software.

3.1 Estrategias de prueba

Pruebas unitarias

Una prueba unitaria es la verificación de un módulo (unidad de código) determinado dentro de un sistema. Son llevadas a cabo por los programadores encargados de cada módulo y aseguran que dicho módulo cumpla con un comportamiento esperado en forma aislada antes de ser integrado al sistema. Se recomienda su uso cuando la interfaz de un método no es clara, la implementación es complicada, para probar entradas y condiciones inusuales, y luego de realizar modificaciones en un módulo.

Este tipo de prueba se considera una actividad fundamental en la metodología XP, ya que los cambios son integrados en lapsos de tiempo muy breves, además brinda una visión de lo que se quiere realizar, demuestra que lo implementado es lo pensado al inicio, y es más sencillo programar teniendo los casos de prueba escritos.

Las pruebas unitarias poseen como beneficios que les brindan al programador una retroalimentación inmediata de como están realizando su trabajo, se pueden realizar cambios de forma segura respaldados por efectivos casos de pruebas, y que permiten conocer si una determinada funcionalidad se puede integrar al sistema sin alterar el funcionamiento del mismo. (25)

Pruebas de aceptación

Las pruebas de aceptación son más importantes que las pruebas unitarias dado que significan la satisfacción del cliente con el producto desarrollado y el final de una iteración y el comienzo de la siguiente. Son creadas a partir de las Historias de Usuario. Durante una iteración las Historias de Usuario seleccionadas durante la reunión de planificación del Plan de Iteraciones serán traducidas a pruebas de

aceptación. El cliente especifica los escenarios para poner a prueba cuando una HU se ha aplicado correctamente.
Estas pruebas se caracterizan por la participación activa del usuario, que debe ejecutar los casos de prueba ayudado por miembros del equipo de pruebas, y están enfocadas a probar las funcionalidades de la aplicación. (26)

Existen dos tipos de pruebas de aceptación:

La prueba alfa

La prueba alfa es aplicada por los usuarios finales en el área de trabajo del desarrollador en un entorno controlado, donde registra los errores y problemas detectados.

La prueba beta

A diferencia de la alfa se aplica en el lugar de trabajo de los usuarios finales, en un entorno no controlado por el desarrollador. El cliente registra todos los problemas (reales o imaginarios) que encuentra durante la prueba beta e informa al desarrollador.

Se decide utilizar para el desarrollo del nuevo Componente de Captura de Métricas de Naire la estrategia de prueba aceptación de tipo alfa, al ser más efectivo que el cliente dé una aceptación del producto que desea en el ambiente de desarrollo, lo que permite determinar con mayor claridad los errores. Además al utilizar la prueba alfa se garantiza una mayor rapidez a la hora de resolver los errores detectados y se garantiza una mejor comunicación entre el desarrollador y el cliente. Estas pruebas conllevan al cliente a precisar lo que la aplicación debe hacer en determinadas circunstancias, por esto el cliente es la persona adecuada para diseñar las pruebas.

3.1.1 Enfoques de diseños de prueba

El único instrumento adecuado para determinar el estado de la calidad de un producto software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos.

Existen diferentes autores que han definido los enfoques de prueba de distintas maneras, a continuación se muestra una selección de estas definiciones:

De acuerdo con Eric J. Braude, existen tres enfoques para el diseño de casos de prueba (27):

- **Enfoque de caja blanca**: su objetivo es probar las líneas de falla más probables de la aplicación. Para realizar pruebas de caja blanca, primero se desglosa el diseño de la aplicación en busca de trayectorias y otras particiones de control y datos. Después se diseñan pruebas que recorran todas o algunas de estas trayectorias y se ejecutan todas las partes o elementos.
- **Enfoque de caja negra**: el único interés es si una aplicación o parte de ella proporciona la salida adecuada, se prueba que se cumpla cada requerimiento al usar la entrada apropiada.
- **Enfoque de caja gris**: consideran el trabajo interior de la aplicación o unidad bajo prueba, pero solo en un grado limitado, también pueden incluir aspectos de caja negra.

De acuerdo con Mario G. Piattini y otros, existen tres enfoques para el diseño de casos de prueba (28):

- **Enfoque estructural o de caja blanca:** consiste en centrarse en la estructura interna (implementación) del programa para elegir los casos de prueba.
- **Enfoque funcional o de caja negra:** consiste en estudiar la especificación de las funciones, la entrada y la salida para derivar los casos.
- **Enfoque aleatorio:** consiste en utilizar modelos (en muchas ocasiones estadísticos) que representen las posibles entradas al programa para crear a partir de ellos los casos de prueba.

De acuerdo con Ron Patton, existen dos enfoques principales para el diseño de casos de prueba (29):

- **Enfoque estructural o de caja blanca:** verifica la estructura interna del componente con independencia de la funcionalidad establecida para el mismo. Por tanto, no se comprueba la corrección de los resultados si éstos se producen.
- **Enfoque funcional o de caja negra:** comprueba el correcto funcionamiento de los componentes del sistema de información, analizando las entradas y salidas y verificando que el resultado es el esperado.

Después de haber consultado definiciones de enfoques de prueba de diferentes autores se decide utilizar el enfoque funcional o de caja negra según Ron Patton, debido a que se considera que es la definición más completa y ajustada al desarrollo de las pruebas a realizar al Componente de Captura de Métricas de Naire, que serán descritas de forma más detallada en el siguiente sub epígrafe.

3.1.2 Enfoque de prueba seleccionado

En la realización de las pruebas realizadas al nuevo Componente de Captura de Métricas de Naire se decide utilizar el enfoque de caja negra (ver *Figura 10*) debido a que permite obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales del sistema. En las pruebas a realizar se pretende comprobar el funcionamiento del componente modificando los factores externos con los que trabaja, tales como paquetes de Linux, las conexiones a los clústeres de PostgreSQL y los catálogos del SGBD, y validar que los resultados obtenidos son los esperados.

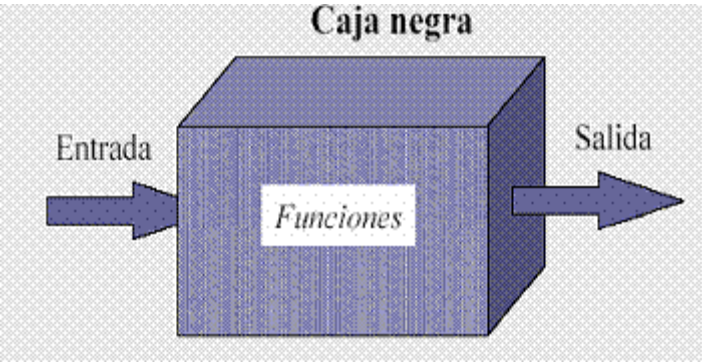

Figura 10: Representación del flujo de las pruebas de caja negra

Las pruebas de caja negra no son una alternativa a las técnicas de prueba de caja blanca. Son un enfoque complementario. Se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación. (30)

Los casos de prueba de la caja negra pretenden demostrar que (30):

- Las funciones del software son operativas.
- La entrada se acepta de forma adecuada.
- Se produce una salida correcta.

- La integridad de la información externa se mantiene.

A diferencia de la prueba de caja blanca, que se lleva a cabo previamente en el proceso de prueba, la prueba de caja negra tiende a ser aplicada en posteriores fases de prueba, ya que la prueba de la caja negra ignora la estructura de control y concentra su atención en el dominio de la información.

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están (32):

- Técnica de la Partición de Equivalencia: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- Técnica del Análisis de Valores Límites: esta Técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- Técnica de Grafos de Causa-Efecto: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Se selecciona la Técnica de Partición de Equivalencia para realizar las pruebas al nuevo Componente de Captura de Métricas de Naire porque es la que más se ajusta a la forma en que se pretende ejercitar las funcionalidades añadidas mediante diferentes entradas, y comprobar el comportamiento del sistema con las mismas.

La partición equivalente es un método de prueba de la caja negra que divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de error (por ejemplo procesamiento incorrecto de todos los datos de carácter) que de otro modo requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. (31)

El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Típicamente una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición booleana (si o no). Las clases de equivalencia se pueden definir de acuerdo a las siguientes directrices:

- Si una condición de entrada especifica un rango, se define una clase de equivalencia válida y dos inválidas.

- Si una condición de entrada requiere un valor específico, se define una clase de equivalencia válida y dos inválidas.
- Si una condición de entrada especifica un miembro de un conjunto, se define una clase de equivalencia válida y una inválida.
- Si una condición de entrada es booleana, se define una clase válida y una inválida.

3.2 Casos de prueba basados en Historias de Usuario

La confirmación de un sistema de software es un proceso continuo en cada etapa del ciclo de vida del software. Los casos de prueba no son más que un conjunto de condiciones o variables bajo las cuáles el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio. En las *Tablas 10, 11 y 12* se muestran tres ejemplos de casos de prueba realizados a las nuevas funcionalidades del Componente de Captura de Métricas.

Tabla 10: Caso de prueba 1: Historia de Usuario 1: Crear conexiones a los clústeres de PostgreSQL

Escenario	Descripción	Variable 1	Respuesta del sistema	Flujo central
EC 1.1 Crear conexiones a los clústeres de PostgreSQL con todos los clústeres funcionando correctamente.	*El componente crea exitosamente las conexiones con todos los clústeres.*	V	*El componente crea las conexiones a los clústeres de PostgreSQL y se encuentra listo para capturar las métricas de bases de datos.*	*Al iniciar el componente, obtiene la cantidad de clústeres a monitorizar del fichero de configuración y realiza la conexión con cada uno.*
EC 1.2 Crear conexiones a los clústeres de PostgreSQL con todos los clústeres fuera de servicio.	*El componente no crea las conexiones con los clústeres.*	F	*El componente registra un error por cada conexión fallida y continúa con su funcionamiento.*	
EC 1.3 Crear conexiones a los	*El componente crea exitosamente las*		*El componente crea las conexiones a los*	

clústeres de PostgreSQL con algunos de los clústeres fuera de servicio.	*conexiones con los clústeres activos y no con los inactivos.*		*clústeres de PostgreSQL que se encuentran activos y registra un error por cada conexión fallida con un clúster inactivo, y continúa con la captura de las métricas.*	

No.	Nombre de campo	Clasificación	Valor Nulo	Descripción
Variable 1	clusteres_activos	Valor numérico	No	Cantidad de clústeres a monitorizar.

Tabla 11: Caso de prueba 2: Historia de Usuario 2: Capturar las métricas por cada clúster de PostgreSQL

Escenario	Descripción	Variable 1	Respuesta del sistema	Flujo central
EC 1.1 Capturar las métricas con el clúster funcionando correctamente.	*El componente captura exitosamente las métricas.*	V	*El componente captura las métricas por cada clúster y las almacena para su envío.*	*Después de creadas las conexiones con cada clúster se comienza con la captura de métricas por cada uno y se almacenan para su envío.*
EC 1.2 Capturar las métricas con el clúster funcionando incorrectamente.	*El componente no captura las métricas.*	F	*El componente registra un error por cada conexión fallida y continúa con su funcionamiento.*	

No.	Nombre de campo	Clasificación	Valor Nulo	Descripción
Variable 1	cluster_activo	Valor booleano	No	Especifica si el clúster está activo o no.

Tabla 12: Caso de prueba 3: Historia de Usuario 3: Salvar las colecciones en el servidor MongoDB

Escenario	Descripción	Variable 1	Respuesta del sistema	Flujo central
EC 1.1 Guardar las colecciones con el servidor MongoDB funcionando correctamente.	*El componente guarda las colecciones correctamente.*	V	*El componente guarda las colecciones obtenidas de las métricas en el servidor MongoDB.*	*Después de capturadas las métricas de cada clúster se intenta salvarlas en el servidor MongoDB.*
EC 1.2 Guardar las colecciones con el servidor MongoDB funcionando incorrectamente.	*El componente no guarda las colecciones.*	F	*El componente registra un error informando del mal funcionamiento del servidor MongoDB.*	

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
Variable 1	conexion_activa	Valor booleano	No	Especifica si el servidor de MongoDB se encuentra activo.

A continuación se muestran algunos conjuntos de datos que evidencian el comportamiento del nuevo Componente de Captura de Métricas en los escenarios de prueba a los que se sometió el sistema.

En la *Tabla 13* se muestran los datos obtenidos por la funcionalidad que captura los cinco procesos que más CPU están consumiendo. En la misma se observa además la fecha, el nombre del servidor, el consumo de memoria de dichos procesos, su nombre e identificador.

Tabla 13: Lista de procesos que más CPU están consumiendo

Fecha: 2013-05-04		**ServerName: f6-datec6-301-13**		
No.	**Memoria**	**Proceso**	**Identificador**	**CPU**
1	15.4	firefox	3035	3.8%
2	1.0	python	3105	1.5%
3	1.1	Xorg	1107	0.5%
4	6.3	mongod	1013	0.3%
5	1.6	compiz	1874	0.2%

La *Tabla 14* muestra la información obtenida por la funcionalidad que captura el número de conexiones a los clústeres de PostgreSQL instalados en el servidor. Se muestra la versión del clúster, la fecha en que se captura la información, el nombre del servidor, la cantidad de conexiones al clúster y la dirección IP desde la que se realizan las conexiones.

Tabla 14: Conexiones a los clústeres de PostgreSQL del servidor

Clúster: 9.1	**Fecha: 2013-05-04**	**ServerName: f6-datec6-301-13**
Cant.	**Client_addr.**	
1	127.0.0.1	
Clúster: 8.4	**Fecha: 2013-05-04**	**ServerName: f6-datec6-301-13**
Cant.	**Client_addr.**	
1	10.56.13.13	

3.3 Resultado de las pruebas

Con el objetivo de verificar que todos los requisitos funcionales fueron desarrollados correctamente se aplicaron pruebas específicas por cada una de las Historias de Usuario. La *Figura 11* muestra el número de Casos de Prueba (CP) realizados, las No Conformidades (NC) detectadas y las No Conformidades Resueltas (NCR) durante las dos iteraciones realizadas. En la primera iteración se detectaron dos No Conformidades que fueron solucionadas completamente. Durante la segunda iteración fue detectada una No Conformidad que fue debidamente solucionada.

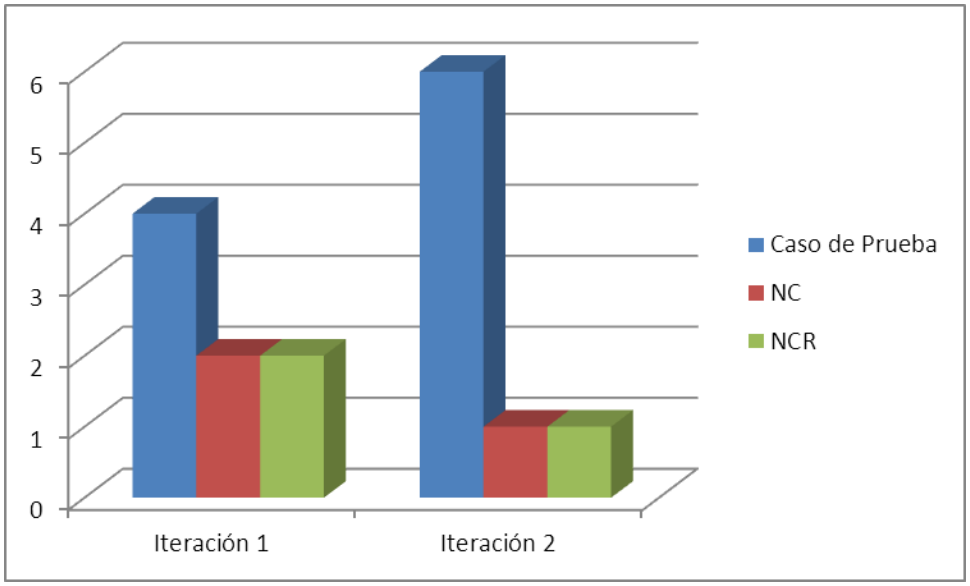

Figura 11: Casos de Prueba, No Conformidades y No Conformidades Resueltas por cada Iteración

La *Tabla 15* muestra las dos no conformidades detectadas en la primera iteración y sus respectivas respuestas por parte del desarrollador.

Tabla 15: No conformidades detectadas durante la primera iteración

Elemento	No.	No conformidad	Aspecto correspondiente	Etapa de detección	Clasificación	Estado NC	Respuesta del Equipo de Desarrollo
Aplicación	*1*	*El componente deja de funcionar cuando se detiene uno de los clústeres que está monitorizando.*	*Conexión a los clústeres de PostgreSQ*	*Prueba*	*S*	*PD: 7/05/13 RA:: 8/05/13*	Se agrega una función que verifique el estado de los clústeres antes de intentar realizar una

			L.				conexión con los mismos.
Aplicación	*2*	*El componente no monitoriza los clústeres que estaban inactivos y comienzan a funcionar mientras el componente se encuentra en ejecución.*	*Conexión a los clústeres de PostgreSQL.*	*Prueba*	*S*	*PD: 7/05/13 RA:: 8/05/13*	Se agrega una función que verifique el estado de los clústeres antes de intentar realizar una conexión con los mismos.

Conclusiones del capítulo

Con el objetivo de verificar el correcto comportamiento y la estabilidad de las nuevas funcionalidades del Componente de Captura de Métricas de Naire fueron aplicadas una serie de pruebas funcionales capaces de determinar si la nueva versión del componente ofrece los resultados que se esperaban. Se aplicaron un total de diez casos de prueba durante las que se detectaron dos No Conformidades en la primera iteración y una en la segunda, quedando resueltas completamente. La realización de las pruebas permitió confirmar el correcto funcionamiento del nuevo componente así como su estabilidad en diferentes escenarios, ofreciendo los resultados esperados.

CONCLUSIONES

Con el desarrollo de la versión 3.0 del Componente de Captura de Métricas, se cumplió con los objetivos trazados en la investigación, al obtenerse como resultado un sistema estable capaz de monitorizar múltiples clústeres de PostgreSQL independientemente de sus versiones. El estudio realizado permitió seleccionar Psycopg2 como nuevo controlador de conexiones entre Python y PostgreSQL e incorporar nuevas funcionalidades al Componente de Captura de Métricas de Naire para ampliar la información que se muestra de los servidores monitorizados. La realización de pruebas utilizando la estrategia de prueba seleccionada permitió comprobar el correcto funcionamiento del componente ante varios escenarios posibles.

RECOMENDACIONES

Con el objetivo de mejorar el funcionamiento y utilidad del Componente de Captura de Métricas de Naire se recomienda:

- Implementar un sistema que permita al usuario decidir qué métricas específicas desea monitorizar en cada servidor para disminuir el consumo innecesario de recursos en los mismos.
- Implementar una interfaz amigable que permita modificar los parámetros de configuración del componente.
- Realizar un estudio para extender el uso del componente a otras distribuciones de Linux y a sistemas Windows.

REFERENCIAS BIBLIOGRÁFICAS

1. EcuRed. [En línea] [Citado el: 13 de 11 de 2012.] http://www.ecured.cu/index.php/Joven_Club_de_Computaci%C3%B3n_y_Electr%C3%B3nica.

2. García, Rosa María Mato. *Diseño de Bases de Datos.* 1999.

3. Martínez, Rafael. PostgreSQL. [En línea] [Citado el: 13 de 11 de 2012.] http://www.postgresql.org.es/node/582.

4. Universidad de Valencia. [En línea] [Citado el: 13 de 11 de 2012.] www.uv.es/gomis/Apuntes_SITR/Conceptos.pdf.

5. Eveliux. [En línea] [Citado el: 13 de 11 de 2012.] http://www.eveliux.com/mx/protocolos-de-comunicaciones.php.

6. Erl, Thomas. *Service Oriented Architecture: Concepts, Technology, and Design.* Indiana: Pearson Education, 2005. ISBN 0-13-185858-0.

7. Group, PostgreSQL Global Development. PostgreSQL. [En línea] [Citado el: 13 de 11 de 2012.] http://www.postgresql.org/.

8. INIT D. [En línea] [Citado el: 13 de 11 de 2012.] www.initd.org/.

9. PostgreSQL. [En línea] [Citado el: 13 de 11 de 2012.] www.postgresql.org.

10. Nagios. [En línea] [Citado el: 13 de 11 de 2012.] http://www.nagios.com/.

11. EnterpriseDB. [En línea] [Citado el: 13 de 11 de 2012.] http://www.enterprisedb.com/products-services-training/products/postgres-enterprise-manager.

12. SourceForge. [En línea] [Citado el: 13 de 11 de 2012.] http://sourceforge.net/projects/hyperic-hq/.

13. Hyperic. [En línea] [Citado el: 13 de 11 de 2012.] http://www.hyperic.com/products/applications-monitoring.

14. SIU, Consorcio. *Monitoreando y midiendo el rendimiento de un servidor PostgreSQL en sistemas Windows, Linux y Solaris.*

15. Extreme Programming. [En línea] [Citado el: 13 de 11 de 2012.] http://www.extremeprogramming.org/.

16. Comunidad de Ubuntu. [En línea] [Citado el: 20 de 11 de 2012.] http://doc.ubuntu-es.org/Sobre_Ubuntu.

17. Torvald, Linus. GIT. [En línea] [Citado el: 20 de 11 de 2012.] http://git-scm.com/.

18. Visual Paradigm. [En línea] [Citado el: 20 de 11 de 2012.] http://www.visual-paradigm.com/.

19. Varó, Marzal y Luengo, Gracia. *Introducción a la Programación con Python.* 2003.

20. Chodorow, Michel. *MongoDB: The Definitive Guide.* 2010.

21. Softigation. [En línea] [Citado el: 20 de 11 de 2012.] http://es.softigation.com/aptana-studio-3-2-2/120.

22. Bass L., Clements P., Kazman R. *Software Architecture in Practice: Second Edition.* s.l.: Addison-Wesley, 2005.

23. Chung, Martin. *Publish-Subscribe Toolkit documentation for Microsoft BizTalk.* s.l.: MSDN Library, 2002.

24. Marcello Visconti, Hernán Astudillo. *Fundamentos de Ingeniería de Software.* s.l. : Universidad Técnica Federico Santa María.

25. Malfara, Dayvis. Facultad de Ingeniería de la Universidad de Uruguay. [En línea] [Citado el: 24 de 4 de 2013.] www.fing.edu.uy.

26. Xtreme Programming. [En línea] [Citado el: 26 de 4 de 2013.] http://www.extremeprogramming.org/rules/functionaltests.html.

27. Braude, Eric J. *Ingeniería de software, una perspectiva orientada a objetos.* s.l. : Alfaomega Grupo Editor, 2003.

28. Mario G. Piattini, José A. Calvo-Manzano, Joaquín Cervera Bravo, Luis Fernández Sanz. *Análisis y diseño de aplicaciones informáticas de gestión, una perspectiva de ingeniería del software.* s.l. : Alfaomega, 2004.

29. Patton, Ron. *Software Testing.* s.l.: Sams Publishing, 2005.

30. Universidad del Valle, Colombia. [En línea] [Citado el: 29 de 4 de 2013.] http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/.

31. Pressman, Roger S. *Ingeniería de Software Un Enfoque Práctico.* s.l. : McGraw-Hill , 2005. ISBN: 9701054733

BIBLIOGRAFÍA

1. Bass L., Clements P., Kazman R. *Software Architecture in Practice: Second Edition.* s.l.: Addison-Wesley, 2005.

2. Beck, Kent. *Extreme Programming Explained. Addison-Wesley Professional, s.n.,* 1999, *ISBN: 0321278658.*

3. Bertino, Elisa y Martino, Lorenzo. *Sistemas de bases de datos orientados a objetos.* s.l.: Ediciones Díaz de Santos, 1995, ISBN: *0201653567*.

4. Braude, Eric J. *Ingeniería de software, una perspectiva orientada a objetos.* s.l.: Alfaomega Grupo Editor, 2003.

5. Chodorow, Michel. *MongoDB: The Definitive Guide.* 2010.

6. Chung, Martin. *Publish-Subscribe Toolkit documentation for Microsoft BizTalk.* s.l.: MSDN Library, 2002.

7. Erl, Thomas. *Service Oriented Architecture: Concepts, Technology, and Design.* Indiana: Pearson Education, 2005. ISBN 0-13-185858-0.

8. García, Rosa María Mato. *Diseño de Bases de Datos.* 1999.

9. Guerrero, Lugo Manuel Barbosa. *Arquitectura de software como eje temático de investigación*. 2006.

10. Marcello Visconti, Hernán Astudillo. *Fundamentos de Ingeniería de Software.* s.l.: Universidad Técnica Federico Santa María.

11. Mario G. Piattini, José A. Calvo-Manzano, Joaquín Cervera Bravo, Luis Fernández Sanz. *Análisis y diseño de aplicaciones informáticas de gestión, una perspectiva de ingeniería del software.* s.l.: Alfaomega, 2004.

12. Patton, Ron. *Software Testing.* s.l.: Sams Publishing, 2005.

13. Pressman, Roger S. *Ingeniería de Software Un Enfoque Práctico.* s.l.: McGraw-Hill, 2005. ISBN: 9701054733

14. SIU, Consorcio. *Monitoreando y midiendo el rendimiento de un servidor PostgreSQL en sistemas Windows, Linux y Solaris.*

15. Varó, Marzal y Luengo, Gracia. *Introducción a la Programación con Python.* 2003.

16. Visconti, Marcello y Astudilo, Hernán. Fundamentos de Ingeniería de Software.

GLOSARIO

AJAX: Acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo Web para crear aplicaciones interactivas o RIA (Rich Internet Applications).

benchmarks: Técnica utilizada para medir el rendimiento de un sistema o componente del mismo o conjunto de procedimientos programas de computación para evaluar el rendimiento de un ordenador.

Demonio: Tipo de proceso no interactivo, es decir, que se ejecuta en segundo plano permitiendo que los usuarios interactúen y trabajen de forma normal en la PC mientras este hace su trabajo.

Clúster: Se aplica a los conjuntos o conglomerados de computadoras o programas construidos mediante la utilización de hardware y software común y que se comportan como si fuesen un único sistema.

Firebird: Sistema de administración de base de datos relacional (o RDBMS) (Lenguaje consultas: SQL) de código abierto, basado en la versión 6 de Interbase, cuyo código fue liberado por Borland en el año 2000.

Framework: Estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software.

htop: Comando que muestra en tiempo real el estado de los procesos del sistema. Éste no es exclusivo de Linux, sino más bien un legado de BSD para todos los miembros de la familia UNIX.

iostat: Se utiliza para informar las estadísticas de entrada y salida de disco, y para generar medidas de rendimiento, uso, longitudes de cola, tasas de transacciones y tiempo de servicio.

IRC: Protocolo de comunicación en tiempo real basado en texto, que permite debates entre dos o más personas.

JSON: (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos.

libpq: Interfaz para los programadores de aplicaciones en C para PostgreSQL. libpq es un conjunto de rutinas de biblioteca que permiten a los programas cliente trasladar consultas al servidor de Postgres y recibir el resultado de esas consultas.

Iotop: El comando o herramienta iotop permite monitorizar las entradas y salidas (I/O) del kernel Linux (2.6.20 o superior) mostrando en una tabla el uso actual de las mismas por los diferentes procesos o subprocesos en el sistema.

Métricas: Una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado.

Microsof Acces: Sistema de gestión de bases de datos o (SGBD) incluido en el paquete de programas de Microsoft Office. Es igualmente un gestor de datos que recopila información relativa a un asunto o propósito particular, como el seguimiento de pedidos de clientes o el mantenimiento de una colección de música.

MongoDB: Es una nueva generación de sistemas de gestión de base de datos que combina un modelo de almacenamiento de documentos con un potente motor de consultas de una manera simple.

MorbidQ: Cola de mensajes simples STOMP para desarrolladores.

Monitorización: Se asocia a la acción de controlar o supervisar cuidadosamente una actividad durante un tiempo acordado. Es el proceso continuo y sistemático mediante el cual verificamos la eficiencia y la eficacia de un proyecto mediante la identificación de sus logros y debilidades. Asimismo, es el responsable de preparar y aportar la información que hace posible sistematizar resultados y procesos.

Munin: Herramienta multiplataforma basada en web, utilizada en el monitoreo de los recursos en red.

MySQL: Sistema de gestión de bases de datos relacional, multihilo y multiusuario

Oracle: Sistema de gestión de bases de datos objeto-relacional (u ORDBMS por el acrónimo en inglés de Object-Relational Data Base Management System), desarrollado por Oracle Corporation.

Orbited: Tecnología web para la comunicación en tiempo. Orbited está desarrollado usando como base a Twisted.

Paradox: Base de datos relacional para entorno MS Windows, anteriormente disponible para MS-DOS y Linux, desarrollada actualmente por Corel e incluida en la suite ofimática WordPerfect Office.

pg_database: Catálogo que almacena información acerca de las bases de datos disponibles.

pg_stat_database: Vista que contiene una fila por cada base de datos en el clúster, y que muestra información estadística.

pg_stat_all_indexes: Vista que contiene información estadística sobre los índices de las tablas de las bases de datos en el clúster.

pg_stat_all_tables: Vista que contiene información estadísticas sobre las tablas en las bases de datos del clúster.

pg_locks: La vista pg_locks provee acceso a información acerca de los bloqueos mantenidos por transacciones abiertas dentro del servidor de base de datos.

ps: Comando de Linux que se usa para informar del estado del proceso. Ps es la abreviatura de Process Status (Estado de Proceso).

Servidor: Es un ordenador que forma parte de una red y que provee servicios a otros ordenadores o sea, sirve información a los que están conectados a él.

SQLite: Sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña (~275 KB) biblioteca escrita en C.

sar: El nombre sar proviene de las siglas de "system activity report" (informe de la actividad del sistema). En Linux se encuentra normalmente en el paquete sysstat. El paquete sysstat incluye programas y scripts para recopilar y mostrar información sobre el rendimiento del sistema, generando informes detallados.

STOMP: El protocolo STOMP de colas de mensajes se utiliza para comunicación distribuida entre computadoras, principalmente en aplicaciones de informática. STOMP es el Protocolo de Mensajes Orientado a Texto y en Flujos (Streaming Text Oriented Messaging Protocol).

top: Se utiliza para averiguar el estado del servidor, ofreciendo información sobre la carga actual, el número de días que lleva encendido sin reiniciarse, el número de usuarios conectados por SSH y en definitiva, información sobre el servidor en tiempo real.

Twisted: Es un framework para el desarrollo de aplicaciones orientadas al trabajo con las redes y basado en eventos que permite utilizar tanto XMPP, IRC como STOMP.

vmstat: Comando que permite obtener un detalle general de los procesos, E/S, uso de memoria/swap, estado del sistema y actividad del CPU.

XMPP: Protocolo abierto y extensible basado en XML, originalmente ideado para mensajería instantánea.

Zope: Proyecto comunitario activista de un entorno de desarrollo para la creación de sitios web dinámicos y/o aplicaciones web usando un servidor de aplicaciones web orientado al objeto, escrito en el lenguaje de programación Python.

ANEXOS

Anexo 1: HU 4: Capturar posibles excepciones de compatibilidad

Historia de Usuario	
Número: 4	**Nombre de la Historia de Usuario:** Capturar posibles excepciones de compatibilidad
Cantidad de modificaciones a la Historia de Usuario: *Ninguna*	
Usuario: *Carlos López Durañona*	**Iteración asignada:** 1
Prioridad en negocio: Muy Alta	**Puntos estimados:** 0.2 semanas
Riesgo en desarrollo: Alto	**Puntos reales:** 0.2 semanas
Descripción: Permite definir y capturar excepciones que puedan ser generadas en el proceso de obtención de métricas de bases de datos.	
Observaciones: Ninguna	
Prototipo de interfaces: No presenta prototipo de interfaz de usuario ya que la aplicación es un demonio del sistema operativo.	

Anexo 2: HU 5: Capturar número de conexiones a cada clúster de PostgreSQL

Historia de Usuario	
Número: 5	**Nombre de la Historia de Usuario:** Capturar número de conexiones a cada clúster de PostgreSQL
Cantidad de modificaciones a la Historia de Usuario: *Ninguna*	
Usuario: *Carlos López Durañona*	**Iteración asignada:** 2

Prioridad en negocio: Alta	**Puntos estimados:** 0.2 semanas
Riesgo en desarrollo: Alto	**Puntos reales:** 0.2 semanas
Descripción: Permite capturar el número de conexiones a cada clúster de Postgres.	
Observaciones: Niguna	
Prototipo de interfaces: No presenta prototipo de interfaz de usuario ya que la aplicación es un demonio del sistema operativo.	

Anexo 3: HU 6: Capturar la velocidad de la conexión de red actual

Historia de Usuario	
Número: 6	**Nombre de la Historia de Usuario:** Capturar la velocidad de la conexión de red actual
Cantidad de modificaciones a la Historia de Usuario: *Ninguna*	
Usuario: *Carlos López Durañona*	**Iteración asignada:** 2
Prioridad en negocio: Alta	**Puntos estimados:** 0.2 semanas
Riesgo en desarrollo: Alto	**Puntos reales:** 0.2 semanas
Descripción: Permite capturar la velocidad actual de la conexión de red.	
Observaciones: Ninguna	
Prototipo de interfaces: No presenta prototipo de interfaz de usuario ya que la aplicación es un demonio del sistema operativo.	

Anexo 4: HU 7: Capturar la descripción del SO

Historia de Usuario	
Número: 7	**Nombre de la Historia de Usuario:** Capturar la descripción del Sistema Operativo (SO)
Cantidad de modificaciones a la Historia de Usuario: *Ninguna*	
Usuario: *Carlos López Durañona*	**Iteración asignada:** 2
Prioridad en negocio: Alta	**Puntos estimados:** 0.2 semanas
Riesgo en desarrollo: Alto	**Puntos reales:** 0.2 semanas
Descripción: Permite capturar la descripción del SO en el que se está ejecutando el demonio.	
Observaciones: Ninguna	
Prototipo de interfaces: No presenta prototipo de interfaz de usuario ya que la aplicación es un demonio del sistema operativo.	

Anexo 5: HU 8: Capturar los procesos que más RAM están consumiendo

Historia de Usuario	
Número: 8	**Nombre de la Historia de Usuario:** Capturar los procesos que más RAM están consumiendo
Cantidad de modificaciones a la Historia de Usuario: *Ninguna*	
Usuario: *Carlos López Durañona*	**Iteración asignada:** 2
Prioridad en negocio: Alta	**Puntos estimados:** 0.2 semanas
Riesgo en desarrollo: Alto	**Puntos reales:** 0.2 semanas

Descripción: Permite capturar y generar una lista con los procesos que más memoria RAM están consumiendo.
Observaciones: Ninguna
Prototipo de interfaces: No presenta prototipo de interfaz de usuario ya que la aplicación es un demonio del sistema operativo.

Anexo 6: HU 9: Capturar los procesos que más CPU están consumiendo

Historia de Usuario	
Número: 9	**Nombre de la Historia de Usuario:** Capturar los procesos que mas CPU están consumiendo
Cantidad de modificaciones a la Historia de Usuario: *Ninguna*	
Usuario: *Carlos López Durañona*	**Iteración asignada:** 2
Prioridad en negocio: Alta	**Puntos estimados:** 0.2 semanas
Riesgo en desarrollo: Alto	**Puntos reales:** 0.2 semanas
Descripción: Permite capturar y generar una lista con los procesos que más recursos del CPU están consumiendo.	
Observaciones: Ninguna	
Prototipo de interfaces: No presenta prototipo de interfaz de usuario ya que la aplicación es un demonio del sistema operativo.	

Anexo 6: HU 9: Capturar los procesos activos de PostgreSQL

Historia de Usuario	
Número: 10	**Nombre de la Historia de Usuario:** Capturar los procesos activos de PostgreSQL
Cantidad de modificaciones a la Historia de Usuario: *Ninguna*	
Usuario: *Carlos López Durañona*	**Iteración asignada:** 2
Prioridad en negocio: Alta	**Puntos estimados:** 0.2 semanas
Riesgo en desarrollo: Alto	**Puntos reales:** 0.2 semanas
Descripción: Permite capturar y generar una lista con los procesos activos que tiene ejecutándose Postgres.	
Observaciones: Ninguna	
Prototipo de interfaces: No presenta prototipo de interfaz de usuario ya que la aplicación es un demonio del sistema operativo.	

Anexo 7: Tarjeta CRC 3: NaireConfigLoader

Tarjeta CRC	
Clase: NaireConfigLoader	
Responsabilidades	**Colaboraciones**
• *Cargar los parámetros de configuración del archivo dnaire.cfg.*	

Anexo 8: Tarjeta CRC 4: DataStompSender

Tarjeta CRC	
Clase: DataStompSender	
Responsabilidades	**Colaboraciones**
• *Realizar las conexiones continuas a través del reactor de Twisted en el intervalo de tiempo definido.*	NaireConfigLoader

Anexo 8: Tarjeta CRC 5: Channels

Tarjeta CRC	
Clase: Channels	
Responsabilidades	**Colaboraciones**
• *Se encarga de la creación de los canales para el envío de las métricas.*	

Anexo 9: Tarjeta CRC 6: Conexión_Postgres

Tarjeta CRC	
Clase: Conexión_Postgres	
Responsabilidades	**Colaboraciones**
• *Se encarga de gestionar la conexión al clúster de PostgreSQL especificado mediante el uso de la librería Psycopg2.*	NaireConfigLoader

Anexo 10: Tarjeta CRC 7: Conexión_Mongo

Tarjeta CRC
Clase: Conexión_Mongo

Responsabilidades	Colaboraciones
• *Se encarga de gestionar la conexión al servidor MongoDB especificado mediante el uso de la librería pymongo.*	NaireConfigLoader