Universidad de las Ciencias Informáticas

Facultad 6

Tema: Desarrollo de un componente de monitoreo para el Servidor de Gestión PostgreSQL

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Yixander De La Paz Milán

Tutores: Ing. Yoan Manuel Pérez Piñero

Ing. Marcos Luis Ortíz Valmaseda

La Habana, Cuba

"Año 53 de la Revolución"

Junio 2011

La revolución no se lleva en los labios para vivir de ella, se lleva en el corazón para morir por ella.

Ernesto Che Guevara

DECLARACIÓN DE AUTORÍA

Declaro ser autor del presente trabajo de diploma y autorizo a la Facultad 6 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Yixander De La Paz Milán

_____

Firma del Autor

Ing. Yoan Manuel Pérez Piñero

_____

Firma del Tutor

Ing. Marcos Luis Ortíz Valmaseda

_____

Firma del Tutor

DATOS DE CONTACTO

Ing: Yoan Manuel Pérez Piñero. **Correo electrónico:** ymperez@uci.cu

Ing: Marcos Luis Ortiz Valmaseda. **Correo electrónico:** mlortiz@uci.cu

AGRADECIMIENTOS

Agradezco a mis padres Nicia Milán y Aventurado De La Paz por ser los protagonistas de este sueño hecho realidad, a ellos no tengo como pagarle todo lo que han hecho por mí.

A mi familia Nano, Yane, Ricardito, Marlon, Malcon, Mariano, a tía María, Rolando, Marlenis, Rolandito, a mi hermana Yixian, a mi hermano Yixel y a toda la familia completa.

A mis tutores Marcos y Yoan que me ayudó muchísimo en todo. Agradecerle también al tribunal de tesis y a todos los profesores que aportaron su granito de arena en mi formación profesional.

A ex compañera de tesis Mercedes, sin ella no hubiese podido desarrollar este trabajo, agradecimientos especiales para ella.

A mis amigos Raiko y Mikel por brindarme su apoyo incondicional, ellos siempre me han ayudado en las buenas y en las malas, a Mario por ayudarme en todo, principalmente en los temas de Linux.

A mi novia Ariannis por entenderme día a día que ha pasado, a ella le debo noches sin dormir, a ti minina gracias.

A Osdel, a el le quiero decir algo, chama entrena duro, que como decía mi abuela, todo esfuerzo tiene en esta vida una recompensa, ¡Serás el campeón del mundo algún día!

A Pedro Paneque, Pablo, al Yanko, a Lassie, al salvaje, Yadrian, Orelvi, a mis socios del apto, al Boza, Daniel, a Yuliel.

A mis colegas de Bayamo, Alberto, Yasmani, Miguel, Lorenzo, Blas, a Pedrito, gracias a ustedes.

Un agradecimiento especial a Ivette y a Yudelki, que siempre me han brindado su mano cuando la he necesitado, muchas gracias a ellas.

A todas la niñas del 6509, en fin le agradezco a todas aquellas personas que han hecho de mi un ingeniero y un profesional. Disculpa pido a todas aquellas personas que me faltan por mencionar, a ustedes también les debo, gracias.

A nuestro comandante en jefe por darme la oportunidad de estudiar en esta gran escuela.

De lo más profundo de mi corazón, de mi alma, de mi mente y de mi ser, gracias, muchas gracias a todos.

DEDICATORIA

Les dedico este trabajo a todas las personas que de una forma permitieron que me graduara como siempre soñé. Pero no podría faltar las personas que más quiero en este mundo, mis padres por ser inspiradores de este regalo, por ser aquellas personas que me han brindado su apoyo, confianza, respeto, amor, cariño, valores revolucionarios, principios, entre otros muchos valores. A ellos les debo la vida y ni volviendo a nacer podría pagarle todo el esfuerzo que han hecho por mí, toda la preocupación que han tenido, todo el apoyo que me han dado. Para ti mamá y para ti papá los quiero del tamaño del Universo y los seguiré queriendo aún después de muerto.

RESUMEN

La Universidad de las Ciencias Informáticas promueve el desarrollo de la ciencia y la tecnología mediante la realización de proyectos productivos. Como parte del desarrollo y soporte a las tecnologías de bases de datos surge el Centro de Tecnologías de Gestión de Datos, en el cual se quiere monitorear y controlar el desempeño de los servidores PostgreSQL por la importancia que tiene saber los detalles sobre el comportamiento de los mismos por parte de los administradores.

El presente trabajo de diploma tiene como objetivo el desarrollo de un componente de monitoreo para el servidor de gestión PostgreSQL llamado DNaire-Client v1.0, software que permite a los administradores de bases de datos llevar un control específico del funcionamiento y rendimiento de los servidores PostgreSQL. Para ello se realiza un estudio de las herramientas de monitoreo existentes y las métricas definidas para monitorear servidores de bases de datos. Se utiliza el lenguaje de programación Python por presentar características afines para el desarrollo de la aplicación informática. Se aplica la metodología ágil Programación Extrema y se realizan pruebas funcionales al componente para verificar su correcto funcionamiento.

Palabras claves: servidores, PostgreSQL, monitoreo, métricas, servidor de gestión

Índice de contenido

Índice de figuras.

Índice de tablas.

INTRODUCCIÓN

El inexorable paso del tiempo, ha provocado que el ser humano se encuentre en un mundo exigente y cambiante. A medida que el hombre evoluciona, junto a él se perfeccionan los diferentes campos de la ciencia y la tecnología. La demanda de los productos de software y los servicios de información tecnológica tienen una de las tasas de crecimiento mundial más elevados en la actualidad. Debido a esto, las empresas y grandes compañías, requieren cada vez más de soluciones rápidas y efectivas que cumplan con los requisitos y exigencias que demanda el campo laboral.

Las Tecnologías de la Información y las Comunicaciones (TICs) juegan un papel relevante en la economía mundial. Su relación con el inminente avance de las ciencias informáticas y la necesidad del desarrollo económico, ha provocado el crecimiento de la industria cubana del software, por eso el Gobierno Cubano se dio a la tarea de desarrollar sistemas para informatizar la sociedad.

La aplicación de los medios informáticos ha revolucionado la gestión de las empresas cubanas. El crecimiento y difícil manejo de grandes volúmenes de información explica las razones que progresivamente obligan a las organizaciones a desarrollar las tecnologías de bases de datos, pues el valor de una información actualizada ha crecido tanto que para incrementar o mantener la productividad se deben gestionar eficientemente todos los datos. Son los Sistemas Gestores de Bases de Datos (SGBD) los que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada.

Dado que existen diversos SGBD como Oracle, SQLServer, MySQL y PostgreSQL, resulta imprescindible seleccionar el más conveniente de acuerdo a las necesidades de las empresas y organismos cubanos, como lograr una disminución de los gastos por pago de licencias de software y soporte relacionado con los SGBD. En contribución al fortalecimiento de la soberanía tecnológica cubana se adoptó PostgreSQL, que como otros proyectos de código abierto no es desarrollado por una sola empresa, sino es dirigido por una Comunidad de Desarrolladores (PGDG[1]) y organizaciones comerciales.

La Universidad de las Ciencias Informáticas (UCI) se desempeña como principal impulsor de la aplicación de estas tecnologías en el país y con el objetivo de aumentar la productividad ha creado centros de desarrollo, conceptualmente pequeñas empresas dirigidas a la producción de software en una rama determinada, para desplegar productos, servicios y soluciones informáticas. Ejemplo de esto es el Centro de Tecnologías de Gestión de Datos (DATEC) creado en septiembre del 2008 con los

[1]PostgreSQL Global Development Group.

objetivos de proveer soluciones integrales, soporte y consultorías relacionadas con tecnologías de bases de datos y análisis de información, desarrollar nuevas tecnologías de bases de datos, de procesamiento y representación de la información y contribuir con la formación y la producción de software (1). De acuerdo a su estructura interna está organizado por departamentos productivos y uno de ellos es el departamento de PostgreSQL.

Actualmente el PGDG solo desarrolla el motor de datos del gestor PostgreSQL junto a un pequeño número de utilidades, por lo que desplegar nuevas funcionalidades que complementen dicho motor, como los sistemas de monitoreo a servidores y bases de datos constituye una necesidad para las instituciones que utilizan los mismos. Se desconoce el desempeño de los servidores y no se controlan los servicios de bases de datos en éstos, como uso de espacio en disco, procesos activos, estadísticas del servidor y las bases de datos en general. El desconocimiento de estos parámetros dificulta a los administradores de bases de datos llevar el control específico del funcionamiento y rendimiento de los servidores PostgreSQL.

En función de lo antes expuesto se ha identificado el siguiente **problema científico:** ¿Cómo supervisar en tiempo real el desempeño de los servidores PostgreSQL desde un servidor de gestión?

Se define como **objeto de estudio**: Sistemas de monitoreo en tiempo real enmarcado en el **campo de acción**: Componentes de monitoreo de servidores de bases de datos PostgreSQL.

Para resolver el problema se trazó como **objetivo general**: Desarrollar un componente de monitoreo para el servidor de gestión PostgreSQL, que supervise en tiempo real el desempeño de los servidores inspeccionados por los administradores.

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

1- Elaborar el marco teórico de la investigación.

2- Definir las métricas a evaluar para el componente de monitoreo.

3- Diseñar el componente de monitoreo a partir de los requisitos identificados.

4- Implementar el componente de monitoreo.

5- Validar la aplicación para verificar su funcionamiento

Para dar cumplimiento a los objetivos específicos se trazaron una serie de **tareas de la investigación:**

1- Revisión y definición de aspectos teóricos conceptuales sobre las diferentes herramientas de monitoreo existentes.

2- Estudio y selección de las métricas a evaluar para el componente de monitoreo.

3- Definición de los requerimientos del componente de monitoreo.

4- Realización del diagrama del modelo de historias de usuario.

5- Realización del diseño del componente de monitoreo.

6- Diseño de la base de datos MongoDB.

7- Implementación del componente de monitoreo.

8- Definición de las pruebas a realizar.

9- Describir los tipos de pruebas.

10- Realización de pruebas funcionales para verificar el funcionamiento del componente de monitoreo.

Luego de la realización de las tareas antes mencionadas se espera tener los siguientes **resultados**:

Componente de monitoreo que verifique el rendimiento y desempeño en tiempo real de los servicios en las bases de datos de los servidores PostgreSQL controlados por el administrador, permitiendo inspeccionar de forma centralizada el comportamiento de los servidores.

La realización de esta investigación ha estado guiada por los siguientes Métodos Científicos:

Métodos Teóricos:

Análisis Histórico-Lógico: se utilizó con el objetivo de conocer las tendencias actuales de los sistemas de monitoreo existentes y realizar un estudio de las métricas definidas por diferentes empresas para el monitoreo de servidores de bases de datos. También permitió recopilar información para describir las tendencias tecnológicas a utilizar, aprovechando elementos teóricos esenciales para el desarrollo de la aplicación.

Analítico-Sintético: se utilizó con el fin de extraer lo esencial de la bibliografía consultada, realizando una división mental del objeto de estudio en sus múltiples relaciones y componentes. Esto permite comprender sus características generales tras sintetizar éstas como un todo, concretando los elementos más importantes relacionados con los sistemas de monitoreo en tiempo real.

Inductivo-deductivo: facilita el análisis de elementos generales a elementos más particulares, o sea, se dispone de un grupo de conocimientos generales sobre las tendencias actuales y características

generales de los sistemas de monitoreo y se pretende llegar a particularizar lo conocido en función de la situación problemática presentada.

El presente trabajo se ha estructurado de la siguiente manera:

Capítulo 1: Fundamentación teórica.

El presente capítulo comprende un estudio de los servidores de gestión, los SGBD específicamente PostgreSQL, las herramientas de monitoreo en tiempo real existentes en el mundo, y las métricas que definen las diferentes empresas para realizar el monitoreo a servidores de bases de datos. También se mencionan las herramientas, metodología y tecnologías a utilizar para el desarrollo del componente de monitoreo de la presente investigación.

Capítulo 2: Descripción de la solución.

En el capítulo se describe el flujo actual de los procesos involucrados, que permitirá conocer particularidades del entorno donde se desarrollará el componente de monitoreo. Además se describen las principales características de este, se definen los requerimientos e historias de usuario y se realiza un diagrama de clases del diseño que se utiliza como complemento de la metodología definida, para lograr un mejor entendimiento del componente.

Capítulo 3: Validación de la solución.

En el presente capítulo se realiza la validación del componente propuesto en el capítulo anterior. Se utiliza como método de validación las Pruebas Unitarias definidas por la metodología XP, que permite verificar el correcto funcionamiento del componente.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

El presente capítulo comprende un estudio de los servidores de gestión, los SGBD, específicamente PostgreSQL, las herramientas de monitoreo en tiempo real existentes en el mundo, y las métricas que definen las diferentes empresas para realizar el monitoreo a servidores de bases de datos. También se mencionan las herramientas, metodología y tecnologías a utilizar para el desarrollo del componente de monitoreo de la presente investigación.

1.1. Servidores de Gestión.

Un 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 éste. Un mismo ordenador puede tener diferentes programas de servidor funcionando al mismo tiempo y también puede cumplir con las funciones de servidor y cliente de manera simultánea. Un servidor de gestión permite administrar de forma centralizada las aplicaciones que se ejecuten en él.

Existen diferentes tipos de servidores, entre ellos se destacan los servidores de archivos que almacenan documentos y los distribuyen a los clientes de la red, servidores de correo que guardan, reciben y envían correos electrónicos y los servidores de bases de datos que se encargan del almacenamiento, acceso y análisis de los datos estructurados. Estos últimos surgen para solucionar los problemas de las empresas que manejan grandes volúmenes de información.

Los servidores de gestión de bases de datos deben proporcionar un conjunto de aplicaciones que permitan administrar, gestionar y monitorear el propio sistema SGBD y a su vez facilitar servicios de mejora para el mismo de forma fiable, rentable y con alto rendimiento.

1.2. 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) base(s) 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 datos en la computadora, ni el método de acceso empleado (2).

Otros especialistas, como Henry Korth, definen los sistemas de bases de datos como una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. Plantean que el objetivo principal de un SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto práctica como eficiente. Además permite gestionar grandes cantidades de información, lo que implica tanto la definición de estructuras para almacenar la información como la provisión de mecanismos para la manipulación de la información (3).

De manera general, los SGBD son un tipo de software dedicado a servir de interfaz entre las bases de datos, los usuarios y las aplicaciones que las utilizan, permitiendo así la creación y manipulación de los objetos y las propias bases de datos.

1.2.1. El Sistema Gestor de Bases de Datos PostgreSQL.

PostgreSQL es un potente Sistema Gestor de Base de Datos, de código abierto del sistema de base de datos objeto-relacional. Cuenta con más de 15 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 idiomas). Cuenta con interfaces de programación nativo de 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 (4).

Existen disímiles empresas que utilizan este gestor, como la EnterpriseDB, líder proveedor de productos de clase empresarial y servicios basados en PostgreSQL, que además de contribuir con el proyecto ofreciendo diversas opciones, brinda soporte técnico para ayudar al desarrollo y despliegue de aplicaciones basadas en PostgreSQL, comercializa productos como Postgres Plus Standard Server y Postgres Plus Advanced Server (5).

Command Prompt es la empresa más antigua proveedora de PostgreSQL dedicada a brindar soporte en América del Norte. Desde 1997 ha venido desarrollando, apoyando, desplegando y promoviendo el uso del gestor de bases de datos más avanzado del mundo (6).

Otra de las empresas más significativas es Cybertec una compañía que ofrece servicios de PostgreSQL y soporte al mismo desde el año 2000 atendiendo clientes de todo el mundo. Sus principales actividades están basadas en consultorías, replicación y migración de base de datos, uno

de sus productos principales es Cybercluster una solución de replicación síncrona para PostgreSQL (7).

PostgreSQL es el gestor de bases de datos de código abierto más avanzado en la actualidad, esencialmente por su robustez, lo que implica que hoy en día sea el gestor más usado con respecto a gestores libres existentes.

1.3. Monitoreo a servidores de bases de datos.

El término monitoreo se asocia a la acción de controlar o supervisar cuidadosamente una actividad durante un tiempo acordado. Algunos autores definen monitoreo como:

"…proceso continuo y sistemático mediante el cual verificamos la eficiencia y la eficacia de un proyecto mediante la identificación de sus logros y debilidades y, en consecuencia, recomendamos medidas correctivas para optimizar los resultados esperados…" (8).

La teoría de la planificación del desarrollo también define el seguimiento o monitoreo como: *"…un ejercicio destinado a identificar de manera sistemática la calidad del desempeño de un sistema, subsistema o proceso a efecto de introducir los ajustes o cambios pertinentes y oportunos para el logro de sus resultados y efectos en el entorno..."* (9).

1.4. Sistemas en Tiempo Real

Los Sistemas en Tiempo Real (STR) son un área de estudio dentro de las ciencias de la computación, existen diferentes universidades y centros de investigación que destinan laboratorios exclusivos para su estudio alrededor del mundo. Algunos autores definen STR como:

"…Un sistema en Tiempo Real es cualquier sistema donde el tiempo en que se produce su salida es significante. Esto es debido a que generalmente la entrada corresponde a algún instante del mundo físico y la salida tiene relación con ese mismo instante. El retraso transcurrido entre la entrada y la salida debe ser lo suficientemente pequeño para considerarse una respuesta puntual…" (10)

Un sistema de tiempo real es aquel donde para que las operaciones computacionales estén correctas no depende solo de que la lógica e implementación de los programas computacionales sea correcto, sino también en el tiempo en el que dicha operación entregó su resultado. (11)

Teniendo en cuenta los conceptos antes descritos, los STR no son más que sistemas informáticos que tienen la capacidad de interactuar rápidamente con su entorno físico, cumpliendo con las restricciones de tiempo establecidas.

El monitoreo en tiempo real a servidores de bases de datos surge de la necesidad de verificar indicadores que permitan evaluar el desempeño de los mismos y conocer el rendimiento de su infraestructura. Entre esos parámetros se puede mencionar el consumo de recursos del servidor, tiempo de respuesta y comportamiento de las bases de datos en una determinada tarea o consulta. Es necesario saber los valores de esos indicadores con el objetivo de conseguir la mayor cantidad de información posible y obtener datos suficientes para realizar análisis comparativos que permitan establecer patrones de comportamiento que ayuden de manera práctica a la toma de decisiones y así asegurar la continuidad operacional de las aplicaciones fundamentales, teniendo conocimiento de su calidad de operación, eficiencia y productividad.

1.5. Herramientas que monitorean servidores de aplicaciones y bases de datos.

Para que las diferentes empresas puedan llevar un control específico del funcionamiento interno de sus sistemas informáticos se sugiere como una excelente opción el monitoreo en tiempo real de servidores de aplicaciones, con el objetivo de coleccionar mediciones de rendimiento que permitan analizar la eficiencia de éstos y los servicios brindados.

Para facilitar y hacer más sencillo el servicio de monitoreo existe una amplia gama de herramientas capaces de adaptarse al monitoreo de complejas aplicaciones. Algunas de éstas se basan específicamente en la monitorización de sistemas operativos, servidores de aplicaciones, virtualización, plataformas distribuidas, aplicadas e integradas y otras que a su vez proporcionan estadísticas de rendimiento de los servidores de bases de datos, web y servicios de red más comunes.

De las diferentes herramientas que realizan monitoreo a servidores de aplicaciones y bases de datos se encuentran:

Nagios. Un sistema de monitoreo 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 el monitoreo de servicios de red, de los recursos de sistemas hardware, independencia de sistemas operativos, y la posibilidad de programar plugins específicos para nuevos sistemas (12)**.** Esta herramienta presenta una interfaz engorrosa que no es de gran usabilidad y no está enfocada al monitoreo de servidores de bases de datos, sino más bien a los servicios de red.

Applications Manager. Provee capacidades de monitoreo para los servidores Linux fuera del sistema. Ayuda a asegurar que los servidores se encuentran funcionando y que se ejecuten a su máximo desempeño monitoreando la utilización del CPU, utilización de la memoria, procesos, utilización, entre

otros (13)**.** Este producto de la empresa ManageEngine es una solución de monitoreo sin agentes, lo que no es muy conveniente para los administradores pues hay que establecer una conexión remoto a cada servidor y el monitoreo se realiza servidor a servidor.

Postgre Plus HQ Monitor. Supervisa la disponibilidad, el rendimiento, la utilización y la infraestructura de las bases de datos. Presenta funcionalidades esenciales como: auto-descubrimiento de recursos supervisando las bases de datos, servidores de aplicaciones y sistemas operativos, realiza monitoreo en tiempo real y envía alertas automáticas (14). Es una herramienta desarrollada por la EnterpriseDB una compañía privada que proporciona soporte de clase empresarial para PostgreSQL.

Hyperic HQ: Se destaca en el monitoreo de aplicaciones y proporciona estadísticas de rendimiento sobre las aplicaciones y servidores de bases de datos, web y servicios de red más comunes (15). Los elementos clave de la arquitectura son el servidor central, que proporciona una gestión centralizada y la persistencia de los datos, y el agente o demonio, que permite el seguimiento y control de cada plataforma.

La definición de su arquitectura permite monitorear aplicaciones mediante el uso de un modelo basado en agentes, ejecuta un agente (*HQ Agent*) en cada servidor que se desea administrar. Al iniciarlo por primera vez, descubre el software que se ejecuta en la máquina, y periódicamente hace un re-análisis de los cambios de configuración. Los agentes recopilan disponibilidad, utilización y métricas de rendimiento de procesamiento, realiza el seguimiento de eventos, y acciones de control de software, por ejemplo, iniciar y detener los servidores web y de aplicaciones. Los agentes envían los datos de inventario y métricas al servidor central (*HQ Server*). El servidor central recibe los datos y los almacena en la base de datos *(HQ Database)* (16). La Figura 1 representa la arquitectura de dicha herramienta.

Figura 1: Arquitectura de Hyperic HQ, herramienta de monitoreo basada en agentes.

Hyperic es una herramienta desarrollada por Spring Source, compañía especialista en el desarrollo de aplicaciones empresariales, la cual es una división de VMware, empresa americana líder en virtualización, que plantea en su contrato de licencia con Hyperic, que el software es de origen estadounidense y se prohíbe las operaciones de exportación del mismo a cualquier persona que sea ciudadano, nacional o residente del gobierno de Cuba (17). Está implementada en Java, por lo que necesita instalar el JDK (Java Development Kit), el cual después de un estudio de varios expertos se están cuestionando si es verdaderamente libre o no.

Partiendo de lo anteriormente expuesto se observa que la adquisición de las herramientas antes mencionadas no garantiza, ni contribuye a alcanzar la soberanía tecnológica para Cuba, pues aunque son de código abierto, reciben soporte de compañías propietarias que impiden la colaboración desde Cuba con nuevas funcionalidades y no permiten que sean reconocidas las contribuciones que se realicen. Otro de los inconvenientes de algunas herramientas son los servicios que brindan que no son los más adecuados para el sistema que se desea desarrollar, aunque las experiencias globales que la herramienta Hyperic HQ ofrece hacen que sea una referencia relevante para la concepción del mismo, pues propone elementos primordiales de gran fortaleza para la realización del mismo, como su arquitectura basada en agentes, que proporciona mayor organización al proceso de monitoreo, brindando también la posibilidad de monitorear en tiempo real más de un servidor de forma centralizada.

El monitoreo a servidores de bases de datos también puede ser realizado por herramientas específicas que trae el propio SGBD, por ejemplo PostgreSQL lo hace a través de logs, vistas estadísticas y

consultas intensivas al catálogo. Esto trae como inconveniente que se realiza a nivel de consola, algo que dificulta el trabajo de los administradores por su complejidad, pues requiere de un conocimiento avanzado de comandos para su ejecución. Existen también otras herramientas externas específicas para el monitoreo a PostgreSQL como son: el PgBench, iotop, lopp, pgstat, PgFouine y el PgAdmin que por sus características no son las más idóneas para monitorear un conjunto de servidores de forma centralizada, tan solo permiten monitorear un solo servidor.

Existen otras técnicas de monitoreo 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 stress sobre los servidores para conocer los límites de prestación del mismo (18).

Los sistemas de monitoreo que adquieren mayor relevancia son los que permiten supervisar el rendimiento en tiempo real de varios servidores a la vez de forma centralizada. Parte fundamental de estos sistemas lo constituye el componente de monitoreo, también llamado agente o demonio. Este es el encargado de recolectar y capturar en tiempo real los valores de los indicadores que mostrarán mediante resultados medibles el comportamiento de los diferentes servidores de bases de datos y que a su vez otorga mayor importancia a los servicios y aplicaciones que realmente son relevantes para el administrador.

1.6. Métricas a aplicar en el proceso de monitoreo a servidores de bases de datos.

Para lograr realizar un adecuado monitoreo a servidores de bases de datos, es necesario definir métricas, con el objetivo de verificar el rendimiento y desempeño de los mismos, realizar estudios comparativos entre los resultados obtenidos, mostrando un estado de mejora o decadencia y optimizar así los procedimientos definidos en el servidor.

En la palabra métrica se encuentran encerrados diferentes términos con los que es muy común asociarla como medición y medida, que por sus similitudes no debieran confundirse. Algunos autores asocian la palabra métrica como:

Una medida "*proporciona una indicación cuantitativa de extensión, cantidad, dimensiones, capacidad y tamaño de algunos atributos de un proceso o producto*" (19).

Sin embargo, la medición "*es el proceso por el cual los números o símbolos son asignados a atributos o entidades en el mundo real tal como son descritos de acuerdo a reglas claramente definidas*" (20).

Mientras que las métricas se pueden definir como: "*Una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado"* (21).

El uso de métricas proporciona un dato cuantitativo sobre el proceso de monitoreo, dando estimaciones sobre el estado de los diferentes servicios de las bases de datos en los servidores y el comportamiento de estos de manera general.

Entre las disímiles empresas en el mundo que desarrollan sistemas de monitoreo se encuentra Manage Engine que ofrece capacidades integrales de monitoreo al sistema gestor PostgreSQL, para ello proporciona diferentes métricas (22), clasificadas de la siguiente manera:

- Métricas de supervisión.
    - Utilización del disco / memoria.
    - Estadísticas de conexión.
- Métricas de rendimiento.
    - Escaneo de índices.
    - Transacciones.
- Métricas de realización de una consulta.
    - Estadísticas de consultas.
    - Bloqueo de las estadísticas

Otra significativa empresa es la EnterpriseDB que define un conjunto de métricas de seguimiento para realizar el monitoreo a servidores de bases de datos PostgreSQL (23), agrupadas en diferentes categorías:

- Métricas de disponibilidad
- Métricas de desempeño.
    - Transacciones.
    - VACUUM / ANALYZE.
- Métricas de rendimiento.
    - Commits / Rollbacks.
    - Bloqueos / Buffer Hits.
- Métricas de utilización.

- Filas insertadas, actualizadas y eliminadas.
- Escaneo de índices.

Hyperic es otra de las empresas que está dirigida al monitoreo de aplicaciones y para esto define métricas para supervisar el comportamiento de servidores de bases de datos PostgreSQL (15), especificadas en cuatro grupos fundamentales:

- Métricas generales
  - Backends / Bloqueos.
  - Commits / Rollbacks.
  - Uso de espacio en disco / Uso de espacio de índices.
- Métricas de tablas
  - Número de filas insertadas.
  - Número de filas actualizadas.
  - Número de filas eliminadas.
- Métricas de índices.
  - Escaneo de índices.
  - Lectura de índices.
- Métricas específicas para el control PostgreSQL.
  - Analyze.
  - Reset Statistics.
  - Vacuum.

La definición de métricas permitirá establecer patrones de comportamiento para los servidores de bases de datos que serán monitoreados, partiendo de que éstas no pueden interpretar por sí solas un concepto medible, pues necesitan de indicadores. Estos son de gran importancia y tienen gran utilidad porque sirven de base para cuantificar conceptos medibles para una necesidad de información, a métodos cuantitativos de evaluación o predicción y a su vez ofrecen información para la toma de decisiones (24).

Las diversas clasificaciones que se le otorgan a las métricas van aparejado de las necesidades particulares y prioridades que tenga cada administrador y cada empresa, otorgándole mayor importancia a las operaciones y aplicaciones de misión crítica para asegurar el correcto funcionamiento de los servidores de bases de datos. Teniendo en cuenta lo anteriormente planteado se seleccionan y

definen las siguientes métricas e indicadores para aplicar al componente a desarrollar, clasificadas en dos categorías fundamentales.

Métricas generales.

1. **Uso de espacio en disco.**
   - **Descripción:** Esta métrica se refiere al porciento de utilización del disco.
   - ✓ **Indicadores:**
     1. Espacio en uso por PostgreSQL.
     2. Espacio libre en disco.
     3. Espacio en uso por el sistema.
     4. Capacidad total del disco.
2. **Uso de espacio en memoria.**
   - **Descripción:** Esta métrica se refiere al porciento de utilización de la memoria RAM.
   - ✓ **Indicadores:**
     1. Espacio libre en memoria.
     2. Espacio en uso por el sistema.
     3. Capacidad total de la memoria.
3. **Conexiones.**
   - **Descripción:** Esta métrica se refiere a la cantidad de conexiones de red que se están haciendo al servidor PostgreSQL.
   - ✓ **Indicadores:**
     1. Cantidad de conexiones al servidor PostgreSQL (Backends).
4. **Procesos activos por el servidor PostgreSQL.**
   - **Descripción:** Esta métrica se refiere a la cantidad de procesos activos que hay en el servidor.
   - ✓ **Indicadores:**
     1. Cantidad de procesos activos por PostgreSQL.

Métricas de bases de datos.

5. **Transacciones.**

- **Descripción:** Esta métrica se refiere a la cantidad de transacciones confirmadas y fallidas en cada base de datos

✓ **Indicadores:**

1. Cantidad de Commits[2] por bases de datos.
2. Cantidad de Rollbacks[3] por bases de datos.

6. Estadísticas de tablas.

- **Descripción:** Esta métrica se refiere al uso de las tablas por bases de datos, respecto a la cantidad de filas insertadas, actualizadas, eliminadas, retornadas, vivas, muertas, leídas por escaneos secuenciales y escaneos secuenciales en las mismas

✓ **Indicadores:**

1. Cantidad de filas insertadas por bases de datos.
2. Cantidad de filas actualizadas por bases de datos.
3. Cantidad de filas eliminadas por bases de datos.
4. Cantidad de filas en retornadas por bases de datos.
5. Cantidad de filas en vivas por bases de datos
6. Cantidad de filas muertas por bases de datos.
7. Cantidad de filas leídas por escaneos secuenciales.
8. Cantidad de escaneos secuenciales.

7. Uso de los índices

- **Descripción:** Esta métrica se refiere al uso de los índices.

✓ **Indicadores:**

1. Total de Índices.
2. Cantidad de índices escaneados.

[2]Transacciones completadas con éxito en una base de datos.

[3]Transacciones fallidas en una base de datos.

1.7. Entorno Tecnológico.

El uso de tecnologías y herramientas es fundamental para el desarrollo de aplicaciones. Para el desarrollo del componente de monitoreo se utilizarán las siguientes tecnologías definidas en el documento de Arquitectura de Software del Servidor de Gestión PostgreSQL.

- **Sistema Operativo: Ubuntu 10.04**

Ubuntu 10.04, 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 como entre sus características se encuentra que el escritorio predeterminado es GNOME líder como escritorio y como plataforma de desarrollo (25).

- **Sistemas de Control de Versiones: Subversion v1.2.1.5297**

Subversion es un sistema de control de versiones libre y de código fuente abierto. Es decir, Subversion maneja ficheros y directorios a través del tiempo. Hay un árbol de ficheros en un repositorio central. El repositorio es como un servidor de ficheros ordinario, excepto porque recuerda todos los cambios hechos a sus ficheros y directorios. Esto le permite recuperar versiones antiguas de sus datos, o examinar el historial de cambios de los mismos. Subversion es un sistema general que puede ser usado para administrar cualquier conjunto de ficheros (26).

- **Gestión de Proyecto: Redmine (GESPRO v1.0)**

Redmine es una herramienta para la Gestión de Proyecto de código abierto, multiplataforma y liberado bajo los términos de Licencia Pública General (GPL), con una administración e interface web más amigable que cualquier otra, y fácil de usar (27).

- **Gestión Documental: Alfresco (GESPRO v1.0)**

Alfresco ofrece el principal sistema de gestión de documentos de código abierto que brinda búsquedas y colaboración de documentos con servicios completos de biblioteca y gestión de ciclo de vida. La gestión documental de Alfresco captura, comparte y retiene contenido, permitiendo a los usuarios hacer versiones, buscar y crear de forma sencilla sus propias aplicaciones de contenido, todo ello con las herramientas utilizadas hoy en día (28).

- **Herramienta de Modelado: Visual Paradigm for UML3.4**

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 (29), con UML se puede:

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

- **Lenguaje de Programación: Python 2.6**

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 (30), entre las que se puede contar:

1. Es un lenguaje expresivo y un programa Python suele ser más corto que su equivalente en lenguajes como C.
2. 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.
3. Ofrece un entorno interactivo que facilita la realización de pruebas.
4. Puede usarse como lenguaje imperativo procedimental o como lenguaje orientado a objetos.
5. Posee un rico juego de estructuras de datos que se pueden manipular de modo sencillo.

- **Metodología de desarrollo de software: Extreme Programming**

Extreme Programming (XP), es una metodología ágil que se centra en potenciar las relaciones interpersonales para lograr el éxito en el desarrollo del software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. Su base se encuentra en la realimentación entre el cliente y el equipo de desarrollo, simplicidad en las soluciones implementadas (31).

- **Base de datos: MongoDB 1.4.4-3**

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 una base de datos del tipo NoSQL orientada a documentos (JSON)[4], entre sus características fundamentales se encuentran (32):

1. 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.

[4]Es un formato de intercambio de datos ligero, una colección de pares nombre / valor.

2. 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.

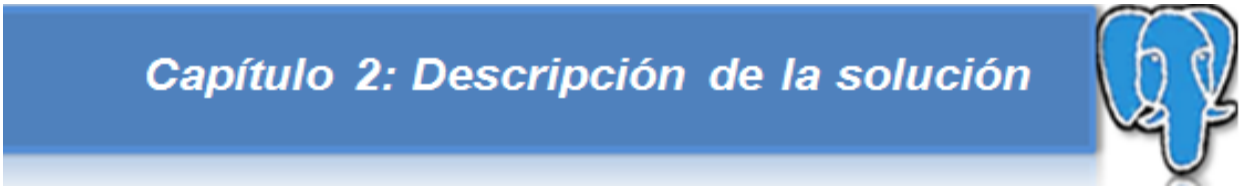
3. Almacenamiento de funciones JavaScript, colecciones de tamaño fijo, almacenamiento de ficheros y un buen rendimiento.

- **Entorno de desarrollo: Aptana Studio 3.0**

Aptana Studio, es un robusto y avanzado Entorno de Desarrollo Integrado (IDE), basado en la flexibilidad de Eclipse. Su mayor fortaleza se encuentra en el poderoso motor de desarrollo web que posee. Este IDE es multiplataforma, puede ejecutarse en Mac OS, Windows y Linux. Tiene soporte a HTML, CSS y JavaScript, así como a PHP, Ruby on Rails y, por supuesto, Python. Para usar el IDE con el intérprete para el lenguaje Python se debe tener el plugin PyDevn (33).

1.8. Conclusiones parciales

En el capítulo se han resumido diferentes aspectos teóricos relacionados a los servidores de gestión, los sistemas gestores de bases de datos, el monitoreo a servidores de aplicaciones y bases de datos. Se explica que son los sistemas en tiempo real. Se presentan algunas de las herramientas que realizan monitoreo, cómo lo hacen y las métricas que emplean, lo que sirvió de apoyo para la definición de las métricas a utilizar para el desarrollo del componente de monitoreo, teniendo en cuenta la importancia y necesidad que tiene el empleo de estas últimas para verificar el desempeño de los servidores a monitorear. También se abordaron brevemente las tecnologías y herramientas a utilizar para desarrollar el componente.



CAPÍTULO 2: DESCRIPCIÓN DE LA SOLUCIÓN

Introducción

En el capítulo se describe el flujo actual de los procesos involucrados, que permitirá conocer particularidades del entorno donde se desarrollará el componente de monitoreo. Además se describen las principales características de dicho entorno. Se describen los requisitos, se definen las historias de usuario y se realiza un diagrama de clases del diseño que se utiliza como complemento de la metodología definida, para lograr un mejor entendimiento del componente.

2.1. Flujo actual de procesos.

Si algún administrador de base de datos, directivo de la institución u otra persona, desea conocer el desempeño en tiempo real de los servidores de bases de datos PostgreSQL que soportan la información manejada, debe hacerlo a través de consultas directas al catálogo, vistas estadísticas definidas por el SGBD o comandos de Linux y PostgreSQL que al ejecutarlos extraen los datos y estadísticas de los procesos más importantes que definen el comportamiento de los servidores y las bases de datos. A raíz de esto surgen limitaciones en el proceso de monitoreo, pues si se desea supervisar y conocer el rendimiento de más de un servidor PostgreSQL, se tiene que realizar el proceso servidor a servidor y por consola, lo que se convierte en una tarea tediosa y compleja para los administradores de bases de datos u otro usuario que necesite la información. Actualmente el monitoreo a servidores es poco eficiente, pues no se ofrece la posibilidad de monitorear el rendimiento de los servidores de bases de datos de forma centralizada, evitando obtener una mayor organización al realizar dicho proceso.

2.2. Objeto de automatización.

Será objeto de automatización toda la información almacenada en los servidores de base de datos necesaria para conocer y mantener el buen funcionamiento de estos, a través de la captura constante de métricas de rendimiento e indicadores que serán almacenados en la base de datos central.

2.3. Mapa conceptual.

Teniendo en cuenta el proceso de negocio para la realización del componente se utiliza un mapa conceptual que explica los conceptos más significativos en el dominio del problema. Se identifican

asociaciones para capturar y expresar el entendimiento ganado en el área en el cual se desarrolla. En la Figura 2 se presentan los conceptos identificados en el proceso de monitoreo a servidores de bases de datos.

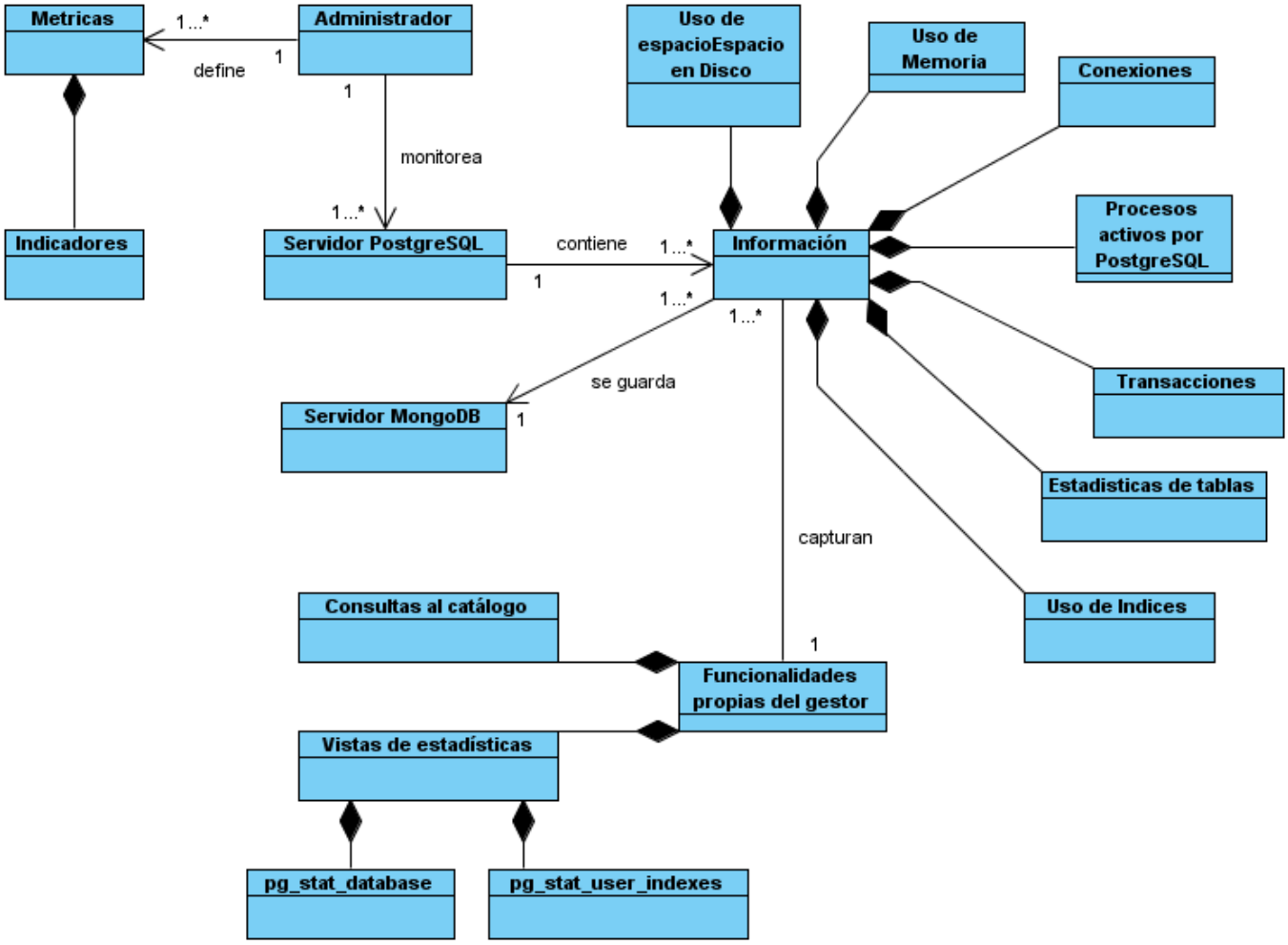


Figura 2: Mapa conceptual del componente de monitoreo.

El administrador define las métricas que están compuestas por indicadores. También realiza el monitoreo a los servidores de bases de datos PostgreSQL. Éstos almacenan información que contiene datos generales del servidor, uso de espacio en disco, uso de la memoria, cantidad de conexiones que se están haciendo al servidor, procesos activos, transacciones, uso de índices y estadísticas de las tablas manera específica. A esta información no se puede acceder de forma rápida, por lo que es necesario utilizar las funcionalidades propias del gestor para capturarlas, como las consultas directas

al catálogo a través de comandos ya definidos o vistas estadísticas como pg_stat_database y pg_stat_user_indexes, que muestra de manera general información acerca de los índices.

2.4. Propuesta del componente a desarrollar.

El componente de monitoreo propuesto a desarrollar es el encargado de recolectar un grupo de métricas, clasificadas en generales y de bases de datos, que permitirá registrar el rendimiento en tiempo real de los servidores PostgreSQL. El administrador debe instalar el componente en cada uno de los servidores a monitorear, una vez instalado se comportará como un proceso del sistema operativo, que de acuerdo a sus características estará enviando constantemente y de forma sincrónica en el intervalo de un segundo el resultado de las métricas capturadas a la base de datos NoSQL MongoDB.

El tiempo establecido para el componente está dado en el momento en que se captura la métrica hasta la inserción de la información en la base de datos MongoDB, éste es de 1s, el cual es necesario que sea cumplido para que sea factible la operación realizada.

La estructura interna del agente[5] está compuesta por diferentes paquetes, uno de ellos es el paquete capturador de métricas donde se va a definir cada una de las métricas e indicadores que permiten analizar el comportamiento tanto del servidor PostgreSQL como del sistema operativo. El paquete con nombre Demonio contiene una serie de módulos encargados de definir todas las funciones que pueden realizarse con el componente, tales como: start, stop, restart, reload, force-reload y status.

Naire.log es un archivo o registro al que se le va añadiendo líneas a medida que se realizan acciones sobre el componente. Las incidencias, errores de conexión, entradas de datos inválidos, cualquier problema con el sistema se guardan en el mismo. Naire.cfg es un archivo de configuración donde se definen parámetros necesarios para la conexión con el servidor PostgreSQL y el servidor basado en documentos MongoDB. En la clase Principal es donde se ejecutan todas las métricas e indicadores capturados de PostgreSQL y del sistema operativo enviando las mismas de forma concurrente hacia el servidor MongoDB. En la base de datos MongoDB se almacena toda la información capturada de cada uno de los servidores PostgreSQL a monitorear. La Figura 3 representa gráficamente la interacción de los paquetes, módulos, archivos, clases y MongoDB.

[5] Componente de monitoreo.

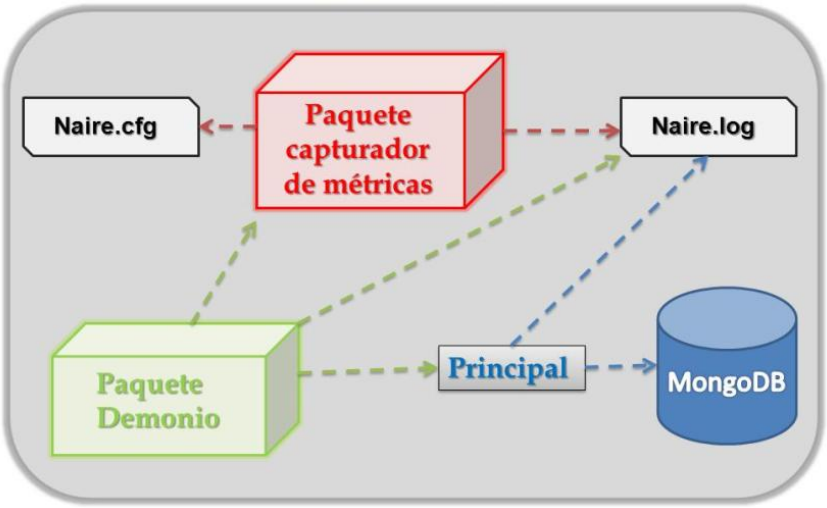


Figura 3: Estructura interna del componente de monitoreo, compuesta por paquetes, módulos, archivos, clases y la base de datos MongoDB, permitiendo el funcionamiento del componente.

2.5. Requerimientos del sistema.

Los requerimientos son declaraciones que identifican atributos, capacidades, características y/o cualidades que necesita cumplir un sistema (o un sistema de software) para que tenga valor y utilidad para el usuario. Los requerimientos pueden ser de dos tipos, funcionales y no funcionales.

2.5.1. Requerimientos funcionales

Los requerimientos funcionales son aquellos que describen qué debe hacer el sistema o software, desde el punto de vista de las necesidades del usuario, son capacidades o condiciones que debe cumplir el sistema y que están fuertemente ligados a las opciones del programa. A continuación se muestran los requerimientos funcionales del sistema.

- ✓ RF-1 Autenticar Usuario.
- ✓ RF-2 Crear conexión a PostgreSQL.
- ✓ RF-3 Ejecutar consultas a PostgreSQL.
- ✓ RF-5 Crear conexión a MongoDB.
- ✓ RF-6 Crear colecciones en MongoDB.
- ✓ RF-7 Insertar colecciones en MongoDB
- ✓ RF-8 Crear archivo de configuración.
- ✓ RF-9 Crear archivo log.
- ✓ RF-10 Capturar espacio del disco en uso por PostgreSQL.
- ✓ RF-11 Capturar espacio del disco en uso por el sistema operativo.
- ✓ RF-12 Capturar espacio libre en disco.

- ✓ RF-13 Capturar capacidad total del disco.
- ✓ RF-14 Capturar espacio de memoria en uso por PostgreSQL.
- ✓ RF-15 Capturar espacio libre en memoria.
- ✓ RF-16 Capturar espacio en uso de la memoria por el sistema.
- ✓ RF-17 Capturar capacidad total de la memoria.
- ✓ RF-18 Capturar cantidad de conexiones que se están haciendo al servidor PostgreSQL.
- ✓ RF-19 Capturar cantidad de procesos activos por PostgreSQL.
- ✓ RF-20 Capturar cantidad de filas insertadas.
- ✓ RF-21 Capturar cantidad de filas actualizadas.
- ✓ RF-22 Capturar cantidad de filas eliminadas.
- ✓ RF-23 Capturar cantidad de filas retornadas.
- ✓ RF-24 Capturar cantidad de filas vivas.
- ✓ RF-25 Capturar cantidad de filas muertas.
- ✓ RF-26 Capturar filas leídas por escaneos secuenciales.
- ✓ RF-27 Capturar escaneos secuenciales.
- ✓ RF-28 Capturar cantidad de Rollbacks.
- ✓ RF-29 Capturar cantidad de Commits.
- ✓ RF-30 Capturar cantidad de escaneo de índices.
- ✓ RF-31 Capturar total de índices.

2.5.2. Requerimientos no funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Describen cómo debe funcionar el sistema o software.

Hardware

- ✓ Para el cliente (componente de monitoreo):
  - Microprocesador Pentium 4 o Superior.
  - 512 MBytes de RAM o superior.
  - 100 MBytes de espacio libre en disco.
- ✓ Para el servidor (MongoDB):
  - Microprocesador Pentium 4 o Superior.
  - 512 MegaBytes de RAM o superior.
  - 500 GibaBytes de espacio libre en disco.

Software

- ✓ Para el cliente:
  - Sistema gestor de bases de datos PostgreSQL.
  - Se aconseja tener instalado como sistema operativo Ubuntu 10.04 o Debían 6.
  - Se requiere tener instaladas las librerías python-pgsql, python-pygresql-dbg, python-psycopg2, python-pymongo, python.
- ✓ Para el servidor:
  - Se aconseja tener instalado como sistema Debían 6.
  - Se requiere tener instalada la librería mongodb.

Restricciones de diseño e implementación: El componente de monitoreo debe estar listo para el despliegue en un tiempo aproximado de tres meses y para el desarrollo del mismo se usará el lenguaje de programación Python.

2.6. Historias de usuarios.

La identificación de las historias de usuario (HU) es la técnica utilizada para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las HU es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. Se definieron para el desarrollo del componente de monitoreo 12 historias de usuarios, de las cuales se muestran en las siguientes tablas las más significativas.

Tabla 1: HU "Administrar conexión a PostgreSQL."

Historia de Usuario	
Número:2	**Nombre de la Historia de Usuario:** Administrar conexión a PostgreSQL.
Cantidad de modificaciones a la Historia de Usuario: Ninguna.	
Usuario: Yixander De La Paz Milán.	**Iteración asignada:**1
Prioridad en negocio: Muy alta.	**Puntos estimados:** 4 días.

Riesgo en desarrollo: Medio.	**Puntos reales:** 4 días.
Descripción: Permite establecer la conexión al sistema gestor PostgreSQL. Para realizar la conexión es necesario establecer una serie de parámetros, entre los que se encuentran, nombre de la base de datos a monitorear, usuario postgres con permisos de super-usuario, contraseña y puerto por el cual escucha el servidor PostgreSQL.	

Tabla 2: HU "Administrar conexión a MongoDB."

Historia de Usuario	
Número:3	**Nombre de la Historia de Usuario:** Administrar conexión a MongoDB.
Cantidad de modificaciones a la Historia de Usuario: Ninguna.	
Usuario: Yixander De La Paz Milán.	**Iteración asignada:**1
Prioridad en negocio: Muy alta.	**Puntos estimados:** 4 días.
Riesgo en desarrollo: Medio.	**Puntos reales:** 4 días.
Descripción: Permite establecer la conexión a la base de datos MongoDB. Para realizar la conexión es necesario establecer una serie de parámetros, entre los que se encuentran, ip del servidor MongoDB y puerto por el cual escucha.	

Tabla 3: HU "Administrar archivo log."

Historia de Usuario	
Número: 4	**Nombre de la Historia de Usuario:** Administrar archivo log.
Cantidad de modificaciones a la Historia de Usuario: Ninguna.	
Usuario: Yixander De La Paz Milán.	**Iteración asignada:** 1

Prioridad en negocio: Muy Alta.	**Puntos estimados:** 2 días.
Riesgo en desarrollo: Medio.	**Puntos reales:** 2 días.
Descripción: Permite al usuario consultar el archivo que contiene trazas de cómo ha sido el comportamiento del componente de monitoreo. Se almacena errores de conexión, consultas nulas y parámetros inválidos.	

Tabla 4: HU "Capturar uso de espacio en disco."

Historia de Usuario	
Número:5	**Nombre de la Historia de Usuario:** Capturar uso de espacio en disco.
Cantidad de modificaciones a la Historia de Usuario: Ninguna.	
Usuario: Yixander De La Paz Milán.	**Iteración asignada:** 2
Prioridad en negocio: Muy Alta.	**Puntos estimados:** 1 semana.
Riesgo en desarrollo: Medio.	**Puntos reales:** 1 semana.
Descripción: Permite capturar los valores de utilización del disco, teniendo en cuenta el espacio que está siendo usado por PostgreSQL, por el sistema operativo, el espacio libre en el disco y el espacio total, para observar si este se llena, pues aunque los datos no se corrompen, si esto llega a suceder el servidor se cae y es necesario tomar las precauciones necesarias.	

Tabla 5: HU "Capturar uso de espacio de la memoria."

Historia de Usuario

Número:6	**Nombre de la Historia de Usuario:** Capturar uso de espacio de la memoria.
Cantidad de modificaciones a la Historia de Usuario: Ninguna.	
Usuario: Yixander De La Paz Milán.	**Iteración asignada:** 2
Prioridad en negocio: Muy Alta.	**Puntos estimados:** 1 semana.
Riesgo en desarrollo: Medio.	**Puntos reales:** 1 semana.
Descripción: Permite capturar los valores de utilización de la memoria, partiendo del espacio que está siendo ocupado por PostgreSQL, por el sistema y el espacio que está libre, para observar si el servidor se queda sin memoria o el proceso de PostgreSQL requiere más memoria de la que el sistema operativo le puede brindar.	

Tabla 6: HU "Autenticar Usuario."

Historia de Usuario	
Número:7	**Nombre de la Historia de Usuario:** Autenticar Usuario
Cantidad de modificaciones a la Historia de Usuario: Ninguna.	
Usuario: Yixander De La Paz Milán.	**Iteración asignada:**1
Prioridad en negocio: Muy alta.	**Puntos estimados:** 4 días.
Riesgo en desarrollo: Medio.	**Puntos reales:** 4 días.
Descripción: Permite la autenticación para establecer la comunicación del componente de monitoreo con el servidor PostgreSQL y con el servidor MongoDB. Los parámetros están compuestos por usuario, contraseña, base de datos que desea monitorear y puerto, siendo estos parámetros necesarios de PostgreSQL. También se define el ip y el	

puerto que identifican al servidor MongoDB.

2.7. Diseño del componente de monitoreo.

Para el diseño de las aplicaciones, la metodología XP no requiere la presentación del sistema mediante diagramas de clases utilizando notación UML, en su lugar se usan otras técnicas como las tarjetas CRC (Contenido, Responsabilidad y Colaboración). No obstante, el uso de estos diagramas puede aplicarse siempre, no como un artefacto de la metodología, pero se pueden utilizar siempre y cuando influyan en el mejoramiento de la comunicación, no sea un peso su mantenimiento, no sean extensos y se enfoquen en la información importante. A continuación se muestran las siguientes tarjetas CRC definidas.

Tabla 7: Tarjeta CRC correspondiente a la clase Métricas.

Tarjeta CRC	
Clase: Métricas.	
Responsabilidades	**Colaboraciones**
→ Establecer la conexión con el servidor PostgreSQL. → Establecer la conexión al servidor MongoDB. → Capturar bases de datos. → Capturar Nombre de la PC. → Capturar espacio en disco. → Capturar uso de la memoria. → 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.	DB Connection MongoDB.

Tabla 8: Tarjeta CRC correspondiente a la clase Demonio.

Tarjeta CRC

Clase: Demonio.	
Responsabilidades	**Colaboraciones**
→ Crear proceso de sistema. → Establecer el inicio del componente. → Establecer la detención del componente. →Establecer el reinicio del componente. → Establecer el estado del componente. → Establecer la ejecución del componente.	

Tabla 9: Tarjeta CRC correspondiente a la clase Monitoreo.

Tarjeta CRC	
Clase: Monitoreo.	
Responsabilidades	**Colaboraciones**
→ Ejecutar métricas. → Comprobar conexión al servidor PostgreSQL. → Comprobar conexión al servidor MongoDB.	Demonio.

Tabla 10: Tarjeta CRC correspondiente a la clase Principal.

Tarjeta CRC	
Clase: Principal.	
Responsabilidades	**Colaboraciones**

→ Iniciar el componente (start). → Parar el componente (stop). → Reiniciar el componente (restart). → Ver Estado del componente (status). → Comprobar funciones del componente.	Monitoreo.

A continuación, en la Figura 4, se representa el diagrama de clases definido para el desarrollo del componente de monitoreo, apoyándose en UML como lenguaje de modelado para facilitar el entendimiento del componente a desarrollar.

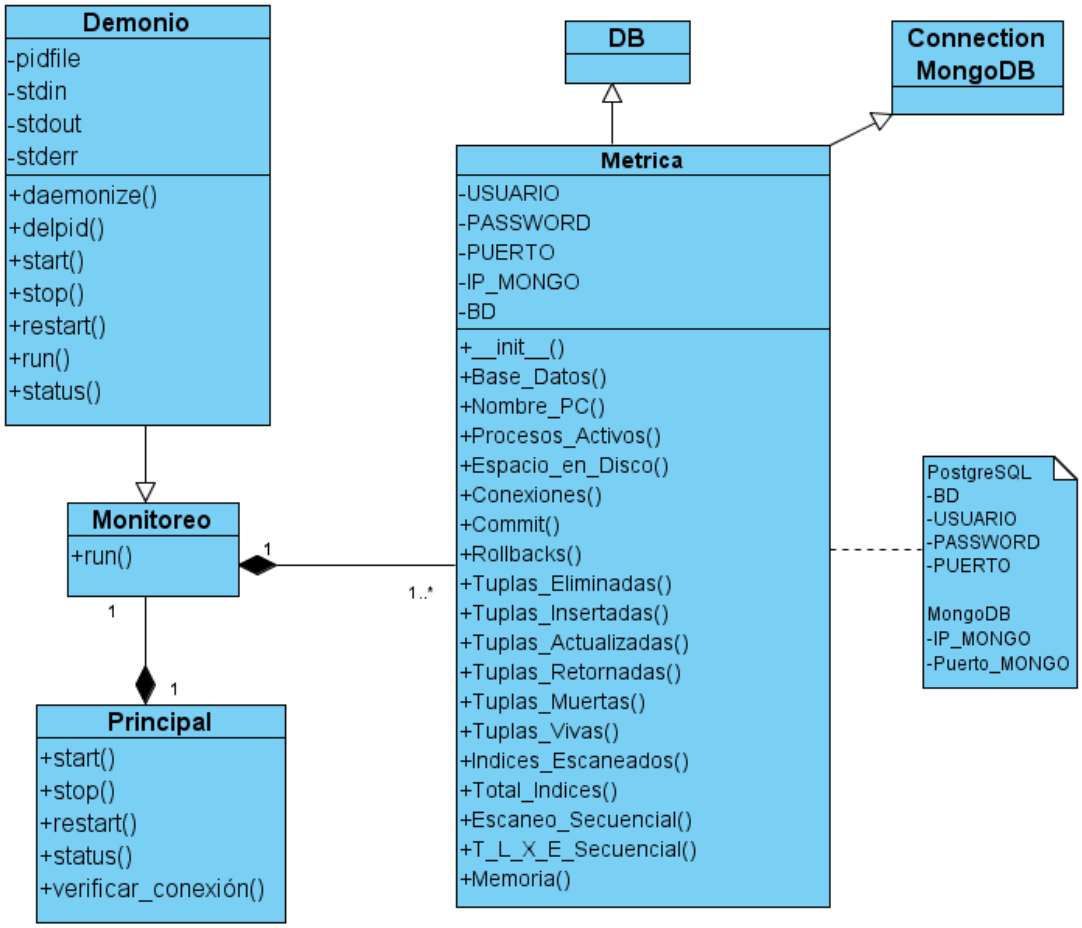


Figura 4: Diagrama de clases del diseño del componente de monitoreo.

Cada clase en el diagrama anterior representa una tarjeta CRC perteneciente al modelo de diseño en XP, mostrando las relaciones entre cada una de ellas.

2.8. Patrones de diseño.

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y, para lograr una mayor calidad en el diseño, se tuvieron en cuenta para utilizar un conjunto de patrones definidos en la investigación realizada en el trabajo de diploma Arquitectura de software del servidor de gestión PostgreSQL, los cuales se mencionan a continuación:

- **Creador.**

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 conecte con el objeto producido en cualquier evento. Le asigna a una clase determinada la responsabilidad de crear una instancia de otro tipo de clase (34). Monitoreo tiene datos de inicialización que se pasan a un objeto de Métrica cuando sea creado (por tanto, Monitoreo es un Experto con respecto a la creación de Métrica). Ver Figura 5 para ver un ejemplo de la aplicación.

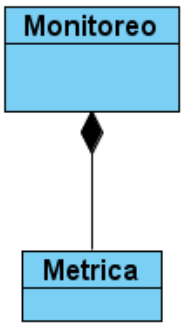


Figura 5: Ejemplo de la aplicación del patrón creador en el sistema.

- **Bajo acoplamiento.**

El acoplamiento coincidido 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. Acoplamiento alto significa que una clase recurre a muchas otras clases (34). A continuación en la Figura 6 se muestra cómo la clase Métrica depende solamente de DB y ConectionMongoDB, por lo que no existe una dependencia entre las clases. También se muestra inexistencia de una herencia muy profunda.

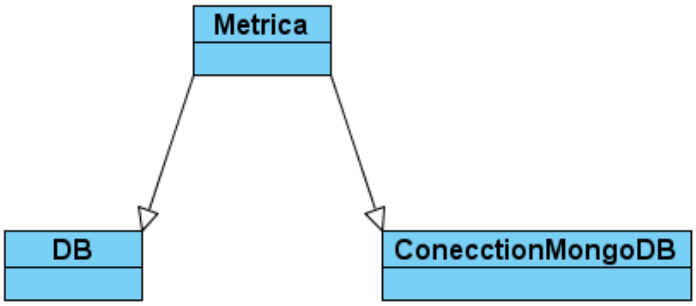


Figura 6: Ejemplo de la aplicación del patrón bajo acoplamiento en el sistema.

- **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 (34). La Figura 7 refleja un ejemplo de uso de este patrón.

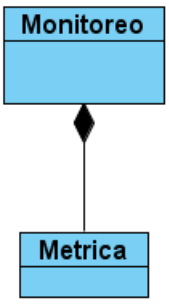


Figura 7: Ejemplo de la aplicación del patrón alta cohesión en el sistema.

- **Experto.**

La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada (34). La clase Daemon es experta en información.

Con la aplicación de este patrón, obtendremos un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento). En la figura 8 se muestra un ejemplo del uso del mismo.

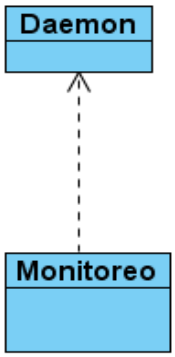


Figura 8: Ejemplo de la aplicación del patrón experto en el sistema.

A continuación se puede observar una tabla donde se ilustran las principales características de los patrones de asignación de responsabilidades aplicados en el sistema.

Tabla 11: Principales características de los patrones de asignación de responsabilidades aplicados en el sistema.

Nombre del patrón	Problema	Solución
Experto	¿Cuál es un principio general para asignar responsabilidades a los objetos?	Asignar una responsabilidad al experto en información – la clase que tiene la información necesaria para la realización de la asignación.
Creador	¿Quién debería ser el responsable de la creación de una nueva instancia de alguna clase?	Asignar a la clase B la responsabilidad de crear una instancia de clase A si se cumple uno o más de los casos siguientes: 1. B agrega objetos de A 2. B contiene objetos de A 3. B registra instancias de

		objetos de A 4. B utiliza más estrechamente objetos de A. 5. B tiene datos de inicialización que se pasarán a un objeto de A cuando sea creado (por tanto, B es un Experto con respecto a la creación de A). 6. B es un creador de los objetos A.
Bajo Acoplamiento	¿Cómo soportar bajas dependencias, bajo impacto del cambio e incremento de la reutilización?	Asignar una responsabilidad de manera que el acoplamiento permanezca bajo.
Alta cohesión	¿Cómo mantener la complejidad manejable?	Asignar una responsabilidad de manera que la cohesión permanezca alta.

2.9. Diseño de la base de datos MongoDB.

Diseñar la Base de Datos es algo que no se puede pasar por alto, producto de que uno de sus objetivos fundamentales es brindar la seguridad en el almacenamiento de la información a gestionar. La base de datos encargada de almacenar la información capturada por el componente a desarrollar es MongoDB. Es una base de datos documental de código abierto desarrollada en C++, escalable y de alto rendimiento. No es una base de datos relacional, por lo que se caracteriza por ser NoSQL, que significa Not Only SQL pues no presenta estructura de tablas y relaciones en las bases de datos. Además, se trata de un software de licencia libre. Funciona en sistemas operativos Windows, Linux, OS X y Solaris.

En MongoDB los datos son almacenados en documentos jerárquicos al estilo JSON, y estos en colecciones de documentos. Un documento equivale al concepto de fila y una colección puede ser considerada como una tabla en bases de datos relacionales (sólo que las colecciones pueden almacenar documentos con muy diferentes formatos, en lugar de estar sometidos a un esquema fijo). Se pueden agrupar las colecciones en bases de datos, al igual que en una base de datos relacional se agrupan tablas. Ver la tabla 12 para un mayor entendimiento. Una sola instancia de MongoDB puede almacenar varias bases de datos, cada una con sus propios permisos y archivos separados.

Tabla 12: Comparativa entre base de datos relacional y MongoDB.

Base de datos relacional	MongoDB
Bases de datos	Bases de datos
Tabla	Colección
Registro	Documento

En el diseño realizado a MongoDB se utilizan los principios antes mencionados, pues por cada servidor de PostgreSQL en el que se instale el componente de monitoreo a desarrollar se creará una base de datos en MongoDB con el nombre del servidor. Las bases de datos contienen una serie de colecciones, que no son más que las métricas a medir y los documentos definidos por MongoDB representan los indicadores de cada una de las métricas, ver Figura 9. Para un mejor entendimiento del diseño de MongoDB remítase al Anexo 1.

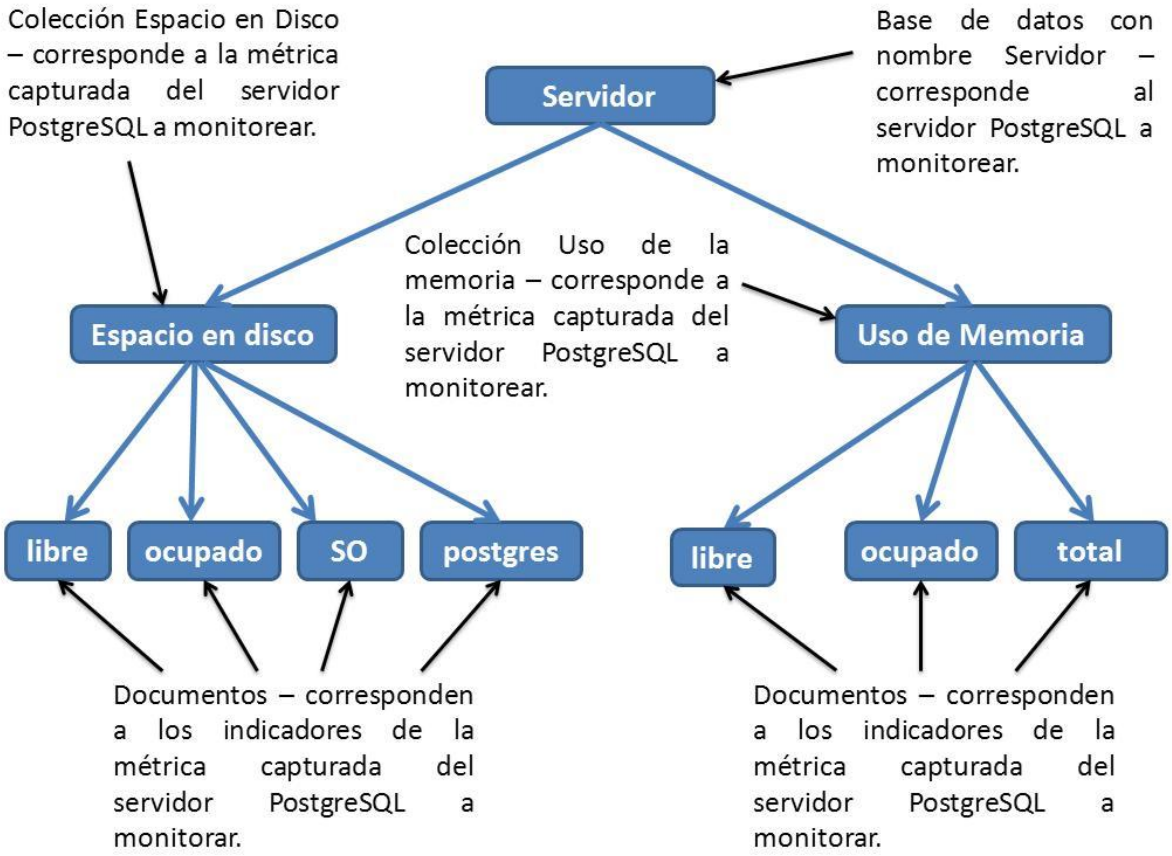


Figura 9: Ejemplo del diseño de la base de datos MongoDB.

2.10. Conclusiones parciales.

En este capítulo se ha logrado definir las características fundamentales del componente a desarrollar. Se identificaron los requerimientos del sistema y se realizó una descripción de las historias de usuario, como paso fundamental en el ciclo de desarrollo. Para lograr un mejor entendimiento del componente a desarrollar se definió un diagrama de clases del diseño en correspondencia con las tarjetas CRC, artefacto generado por la metodología utilizada, necesario para la implementación. Se identificaron los patrones de diseños presentes y se definió la estructura de la base de datos MongoDB para el almacenamiento de los datos.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN

Introducción.

En el presente capítulo se realizará un plan de iteraciones con el fin de recoger una planificación detallada en el proceso de desarrollo del sistema. Se utilizará como método de validación las Pruebas funcionales definidas por la metodología XP utilizada, ejecutándose las mismas al final de cada una de las iteraciones, evaluando cada una de las opciones con las que cuenta el componente informático, permitiendo verificar el correcto funcionamiento del mismo.

3.1. Plan de iteraciones.

Antes de comenzar la implementación la metodología XP propone que una vez que se han definido cuáles son las HU y las mismas se han priorizado de acuerdo al valor que tiene cada una en el negocio, se pasa a definir las iteraciones. Una iteración es un mini-proyecto, donde se implementan una serie de HU, lo más recomendado es colocar las HU de mayor prioridad en las primeras iteraciones. Todo esto se tiene en cuenta para definir el plan de iteraciones que se muestra a continuación.

Iteración 1.

Esta iteración tiene como objetivo realizar las historias de usuarios 1, 2, 3, 4 , 5 y 6 que serán las de vital importancia para el componente a desarrollar, pues son las que permitirán la autenticación de usuario, administrar la conexión a PostgreSQL, MongoDB, Administrar archivo log, capturar uso de espacio en disco y uso de espacio en memoria.

Iteración 2.

El objetivo de esta iteración es implementar las historias de usuarios 7, 8 y 9, que son de una alta prioridad para el componente a desarrollar, como capturar cantidad de conexiones, cantidad de procesos activos por el servidor PostgreSQL y cantidad de transacciones, pues estas son las historias de máximo interés para los administradores de bases de datos, cuando realizan monitoreo.

Iteración 3.

Esta iteración está centrada en desarrollar una parte de las historias de usuario con prioridad media para el componente a desarrollar. Está conformada por las historias de usuario 10 y 11, referente capturar estadísticas de las tablas por bases de datos y capturar uso de los índices, que además no poseen un nivel de riesgo muy elevado para el programador.

3.2. Pruebas.

Uno de los pilares fundamentales de XP es el proceso de pruebas, el cual anima a los desarrolladores a probar constantemente tanto como sea posible. Mediante esta filosofía se reduce el número de errores no detectados así como el tiempo entre la introducción de éste en el sistema y su localización. Todo esto contribuye a elevar la calidad de los productos desarrollados y a la seguridad de los programadores a la hora de introducir cambios o modificaciones.

La metodología XP divide las pruebas en dos grupos: pruebas unitarias, desarrolladas por los programadores, encargadas de verificar el código de forma automática y las pruebas de aceptación, destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente.

3.2.1. Pruebas Funcionales

Las pruebas funcionales permiten confirmar que la HU ha sido implementada correctamente al final de cada iteración. Este período de prueba se conoce también como período de caja negra donde se definirán las entradas al sistema y los resultados esperados de estas entradas. Una HU puede tener todas las pruebas de aceptación que necesite para asegurar su correcto funcionamiento. El objetivo final de éstas es garantizar que los requerimientos han sido cumplidos y que el sistema es aceptable.

A continuación se representan las pruebas funcionales realizadas para las HU del sistema:

3.2.2. Casos de pruebas

Tabla 13: Caso de Prueba 1: HU 1: Autenticar Usuario.

Escenario	Descripción	V1	V2	V3	V4	V5	V6	Respuesta del sistema	Flujo central
EC 1.1 Autenticar usuario correctamente	Permitirá la autenticación de un usuario de postgres dado, el cual deberá poseer privilegios de súper-usuario.	V	V	V	V	V	V	Se muestra un mensaje indicando que el componente de monitoreo se ha iniciado correctamente.	1-El usuario introduce el usuario con permisos de súper – usuario. 2-El usuario introduce la contraseña. 3-El usuario introduce el nombre de la base de datos a monitorear. 4-El usuario introduce el puerto de escucha de PostgreSQL. 5- El usuario introduce el ip de MongoDB. 6- El usuario introduce el puerto de escucha de MongoDB. 7- El usuario presiona la tecla Enter.
EC 1.2 Autenticar usuario con campos incorrectos	El sistema mostrará un error de autenticación.	I	V	V	V	V	V	El sistema muestra un error indicando que la autenticación falló.	
		V	I	V	V	V	V		
		V	V	I	V	V	V		
		V	V	V	I	V	V		
		V	V	V	V	I	V		
		V	V	V	V	V	I		

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
Variable1	Usuario	Campo de texto	No	Usuario de PostgreSQL.
Variable2	Contraseña	Campo de texto	No	Contraseña.
Variable3	Base de Datos	Campo de texto	No	Base de Datos a monitorear
Variable4	Puerto	Campo de texto	No	Puerto de escucha de PostgreSQL.
Variable5	IP de MongoDB	Campo de texto	No	Dirección IP de MongoDB.
Variable 6	Puerto de MongoDB	Campo de texto	No	

Tabla 14: Caso de prueba 2: HU 2: Administrar conexión a PostgreSQL.

Escenario	Descripción	V1	V2	V3	V4	Respuesta del sistema	Flujo central
EC 1.1 Administrar conexión a PostgreSQL.	Permite configurar el archivo que definirá la comunicación del	V	V	V	V	El componente realiza la conexión correctamente.	1-El usuario modifica el campo usuario. 2-El usuario modifica la

	componente de monitoreo con el servidor PostgreSQL.						contraseña. 3-El usuario modifica el nombre de la base de datos a monitorear. 4-El usuario modifica el puerto de escucha de PostgreSQL. 5-El usuario selecciona guardar los cambios en el archivo de configuración.
EC 1.2 Administrar conexión a PostgreSQL con datos incorrectos.	Permite configurar el archivo que definirá la comunicación del componente de monitoreo con el servidor PostgreSQL.	I	V	V	V	Al iniciar el sistema muestra un error con el parámetro que incidió en el fallo de la conexión.	
		V	I	V	V		
		V	V	I	V		
		V	V	V	I		

No	Nombre de campo	Clasificación	Valor Nulo	Descripción

Variable1	Usuario	Campo de texto	No	Usuario de PostgreSQL.
Variable2	Contraseña	Campo de texto	No	Contraseña.
Variable3	Base de Datos	Campo de texto	No	Base de Datos a monitorear.
Variable4	Puerto	Campo de texto	No	Puerto de escucha de PostgreSQL.

Tabla 15: Caso de prueba 3: HU 3: Administrar conexión a MongoDB.

Escenario	Descripción	V1	V2	Respuesta del sistema	Flujo central
EC 1.1 Administrar conexión a MongoDB.	Permite configurar el archivo que definirá la comunicación del componente de monitoreo con el servidor MongoDB.	V	V	El componente realiza la conexión correctamente.	1- El usuario modifica el puerto de escucha de MongoDB. 2- El usuario selecciona guardar los cambios en es archivo de configuración.
EC 1.2 Administrar conexión a MongoDB con datos incorrectos.	Permite configurar el archivo que definirá la comunicación del componente de monitoreo con el servidor MongoDB.	I	V	Al iniciar el sistema muestra un error de conexión.	
		V	I		

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
Variable1	IP de MongoDB	Campo de texto	No	Dirección IP de MongoDB.
Variable 2	Puerto de MongoDB	Campo de texto	No	Puerto de escucha de MongoDB.

Tabla 16: Caso de prueba 4: HU 4: Administrar archivo log.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Administrar archivo log.	Permite al usuario consultar las trazas que todas las operaciones realizadas por el componente de monitoreo, tales como: iniciar, parar y reiniciar el servicio. También podrá consultar errores ocurridos por el componente, así como la hora y fecha en que han ocurrido todas las operaciones.	El sistema muestra el log con valores actualizados.	Una vez instalado el componente se puede consultar el archivo log.

Tabla 17: Caso de Prueba 5: HU 5: Capturar uso de espacio en disco.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Capturar uso de espacio en disco.	El componente enviará la información del espacio que está siendo usado por PostgreSQL, por el sistema, el espacio libre en el disco y el espacio total hacia la base de datos MongoDB.	El sistema envía la información a MongoDB correctamente.	Una vez iniciado el componente, éste le hará peticiones al sistema para enviar la información capturada hacia la base de datos MongoDB.

Tabla 18: Caso de Prueba 6: HU 6: Capturar uso de espacio de la memoria.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Capturar uso de espacio de la memoria.	El componente enviará la información del espacio en memoria que está siendo usado por el sistema, libres y total hacia la base de datos MongoDB.	El sistema envía la información a MongoDB correctamente.	Una vez iniciado el componente, éste le hará peticiones al sistema para enviar la información capturada hacia la base de datos MongoDB.

Tabla 19: Caso de Prueba 6: HU 6: Capturar cantidad de conexiones.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Capturar cantidad de conexiones.	El componente enviará la información acerca de la cantidad de conexiones que se están haciendo al servidor PostgreSQL hacia la base de datos MongoDB.	El sistema envía la información a MongoDB correctamente.	Una vez iniciado el componente, éste le hará peticiones al servidor PostgreSQL para enviar la información capturada hacia la base de datos MongoDB.

Tabla 20: Caso de Prueba 7: HU 7: Capturar cantidad de procesos activos por el servidor PostgreSQL.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Capturar cantidad de procesos activos por el servidor PostgreSQL.	El componente enviará la información acerca de la cantidad de procesos activos por el servidor PostgreSQL hacia la base de datos MongoDB.	El sistema envía la información a MongoDB correctamente.	Una vez iniciado el componente, éste le hará peticiones al Sistema para enviar la información capturada hacia la base de datos MongoDB.

Tabla 21: Caso de Prueba 7: HU 7: Capturar estadísticas de tablas.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Capturar estadísticas de tablas.	El componente enviará la información acerca de los valores que definen el comportamiento de las tablas hacia la base de datos MongoDB.	El sistema envía la información a MongoDB correctamente.	Una vez iniciado el componente, éste le hará peticiones al servidor PostgreSQL para enviar la información capturada hacia la base de datos MongoDB.

Tabla 22: Caso de Prueba 8: HU 8: Capturar cantidad de transacciones.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Capturar cantidad de transacciones.	El componente enviará la información acerca de la cantidad de transacciones por base de datos hacia la base de datos MongoDB.	El sistema envía la información a MongoDB correctamente.	Una vez iniciado el componente, éste le hará peticiones al servidor PostgreSQL para enviar la información capturada hacia la base de datos MongoDB.

Tabla 23: Caso de Prueba 9: HU 9: Capturar uso de los índices.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Capturar uso de los índices.	El componente enviará la información acerca del uso de los índices hacia la base de datos MongoDB.	El sistema envía la información a MongoDB correctamente.	Una vez iniciado el componente, éste le hará peticiones al servidor PostgreSQL para enviar la información capturada hacia la base de datos MongoDB.

Para validar que la aplicación cumple las actividades o tareas (métricas obtenidas del servidor que se está monitoreando) en intervalos de tiempo definidos, se muestra en la Figura 10 y en la Figura 11 el comportamiento en tiempo real de las HU Capturar cantidad de procesos activos y uso de espacio de la memoria respectivamente, obteniendo para el sistema un alto grado de utilización, cumpliendo con los requerimientos de tiempo establecidos.

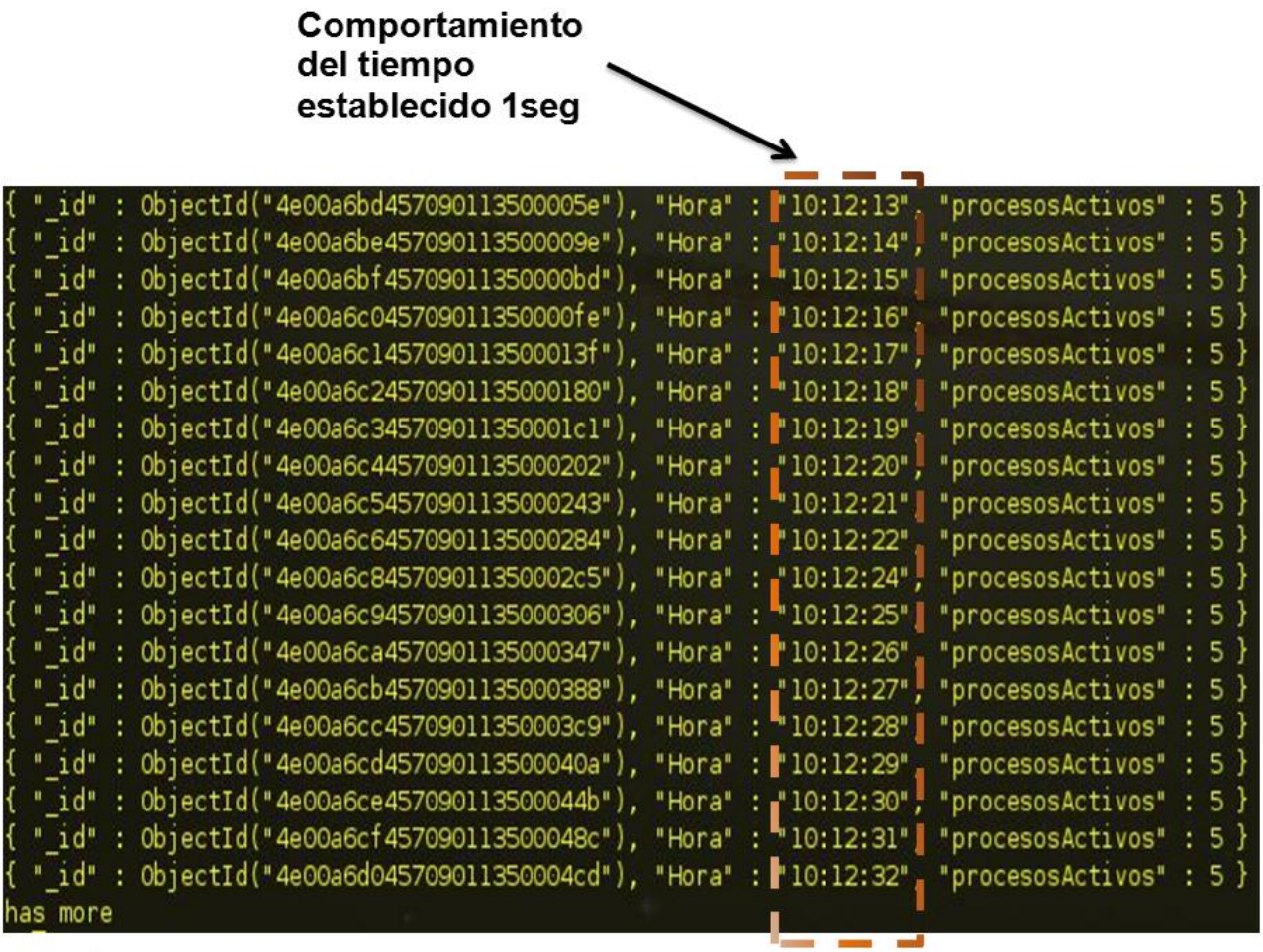


Figura 10: Comportamiento del tiempo establecido para la HU “capturar cantidad de procesos activos”

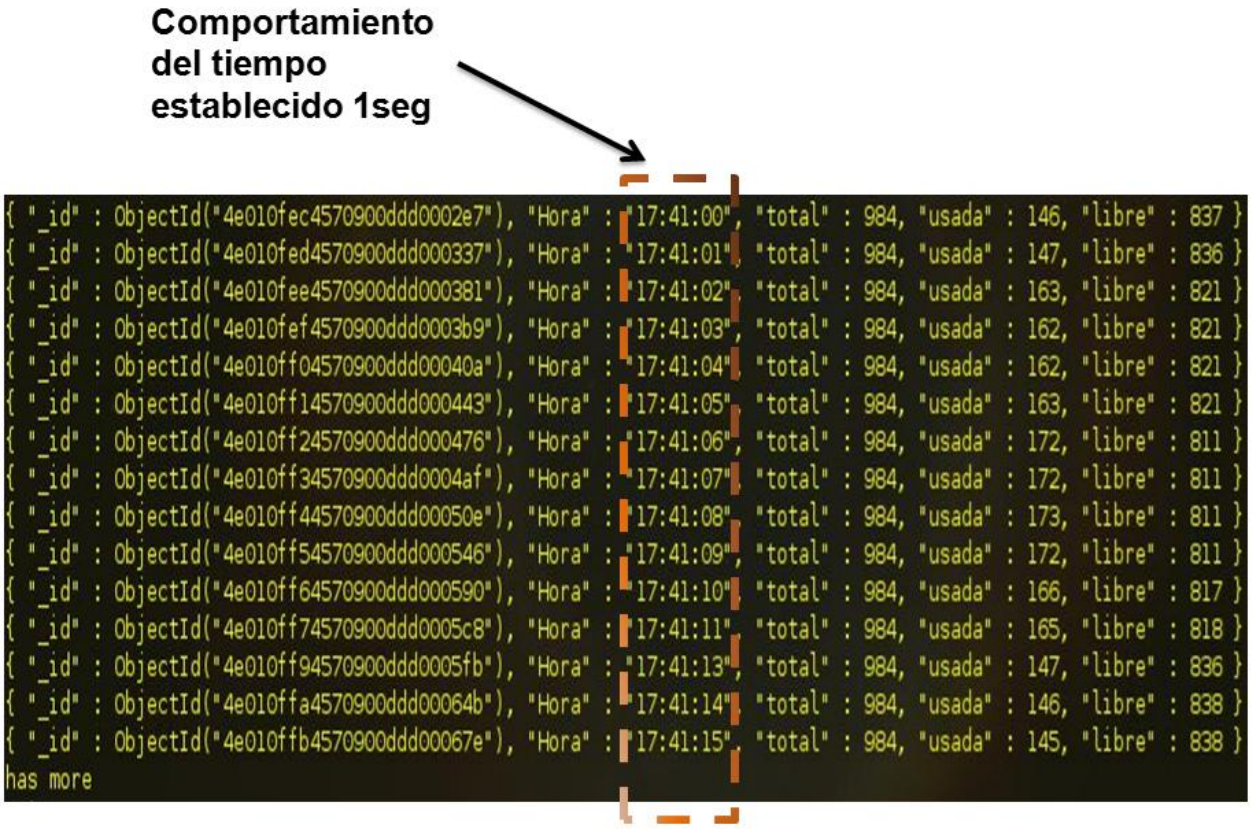


Figura 11: Comportamiento del tiempo establecido para la HU “capturar uso de espacio de la memoria”

3.2.3. No conformidades detectadas

No	Elemento	No conformidad	Aspecto correspondiente	Etapa de detención	Signif	No signif
1	Consola de autenticación.	Se iniciaba el componente y no mostraba error.	Cuando se dejaban los campos vacíos.	Etapa de aplicación de prueba a la historia de usuario Autenticar Usuario.	X	
2	Consola de autenticación.	Se iniciaba el componente y no mostraba el error correspondiente.	Cuando se introducían parámetros inválidos.	Etapa de aplicación de prueba a la historia de usuario Autenticar Usuario.	X	
3	Archivo log.	No se listaba el error de	Cuando se quitaba el cable	Etapa de aplicación de	X	

		conexión con la base de datos MongoGB.	de red o fallaba por cualquier razón la conexión.	prueba a la historia de usuario Administrar archivo log.		
4	Archivo log.	No se mostraba el mensaje de inicio del componente.	Cuando se realizaba la operación iniciar el componente (start)	Etapa de aplicación de prueba a la historia de usuario Administrar archivo log.	X	
5	Servidor MongoDB.	Las métricas de índices no se insertaban en MongoDB.	Cuando se iniciaba el componente.	Etapa de aplicación de prueba a la historia de usuario capturar estadísticas de índices.	X	
6	Consola del sistema operativo y archivo.log	No se registraba el error de conexión.	Cuando se iniciaba el componente.	Etapa de aplicación de prueba a la historia de usuario administrar conexión a PostgreSQL.	X	
7	Consola del sistema operativo.	No mostraba el mensaje de error.	Cuando ejecutaban los comandos sin permisos de root.	Etapa de aplicación de prueba a la historia de usuario Autenticar Usuario.	X	
8	Consola del sistema operativo.	No dejaba espacio entre el parámetro usuario postgres que se pedía y la respuesta del usuario.	Cuando se solicitaban los parámetros para iniciar el componente	Etapa de aplicación de prueba a la historia de usuario Autenticar Usuario.		X
9	Consola del sistema operativo.	No dejaba espacio entre el parámetro contraseña que se pedía y la respuesta de la contraseña.	Cuando se solicitaban los parámetros para iniciar el componente.	Etapa de aplicación de prueba a la historia de usuario Autenticar Usuario.		X
10	Consola del	No dejaba	Cuando se	Etapa de		X

	sistema operativo.	espacio entre el parámetro base de datos que se pedía y la respuesta de la base de datos.	solicitaban los parámetros para iniciar el componente.	aplicación de prueba a la historia de usuario Autenticar Usuario.		
11	Consola del sistema operativo.	No dejaba espacio entre el parámetro puerto de escucha de postgres que se pedía y la respuesta del puerto.	Cuando se solicitaban los parámetros para iniciar el componente	Etapa de aplicación de prueba a la historia de usuario Autenticar Usuario.		X
12	Consola del sistema operativo.	No dejaba espacio entre el parámetro ip del servidor MongoDB que se pedía y la respuesta del ip.	Cuando se solicitaban los parámetros para iniciar el componente	Etapa de aplicación de prueba a la historia de usuario Autenticar Usuario.		X
13	Consola del sistema operativo.	No dejaba espacio entre el parámetro puerto de escucha de MongoDB que se pedía y la respuesta del puerto.	Cuando se solicitaban los parámetros para iniciar el componente	Etapa de aplicación de prueba a la historia de usuario Autenticar Usuario.		X
14	Archivo log.	El formato de los mensajes no era el más adecuado con respecto a los entandares usados por los log.	Cuando se almacenada la información.	Etapa de aplicación de prueba a la historia de usuario Administrar archivo log.	X	
15	Servidor MongoDB.	La métrica capturar cantidad de conexiones no se insertaba en MongoDB.	Cuando se iniciaba el componente.	Etapa de aplicación de prueba a la historia de usuario capturar cantidad de conexiones.	X	
16	Servidor MongoDB.	Las métricas cantidad de transacciones	Cuando se iniciaba el componente.	Etapa de aplicación de prueba a la	X	

		se insertaban con valores irreales.		historia de usuario capturar cantidad de transacciones.		
17	Consola del sistema operativo.	El consumo de memoria del sistema operativo era de 1,1 mbytes/seg.	Cuando se iniciaba el componente y se ejecutaba el comando htop.	Etapa de aplicación de prueba a la historia de usuario capturar cantidad de procesos activos por PostgreSQL.	X	
18	Servidor MongoDB.	Las métricas de tablas mostraban valores irreales.	Cuando se iniciaba el componente.	Etapa de aplicación de prueba a la historia de usuario capturar estadísticas de tablas.	X	

3.2.4. Análisis de los resultados

Con el propósito de comprobar que todos los requisitos del sistema fueron desarrollados y cumplidos completamente, se aplicaron al componente pruebas funcionales, donde se creó un caso de prueba por cada Historia de Usuario obteniendo los resultados que se presentan en la Figura 10, 11, 12 y 13:

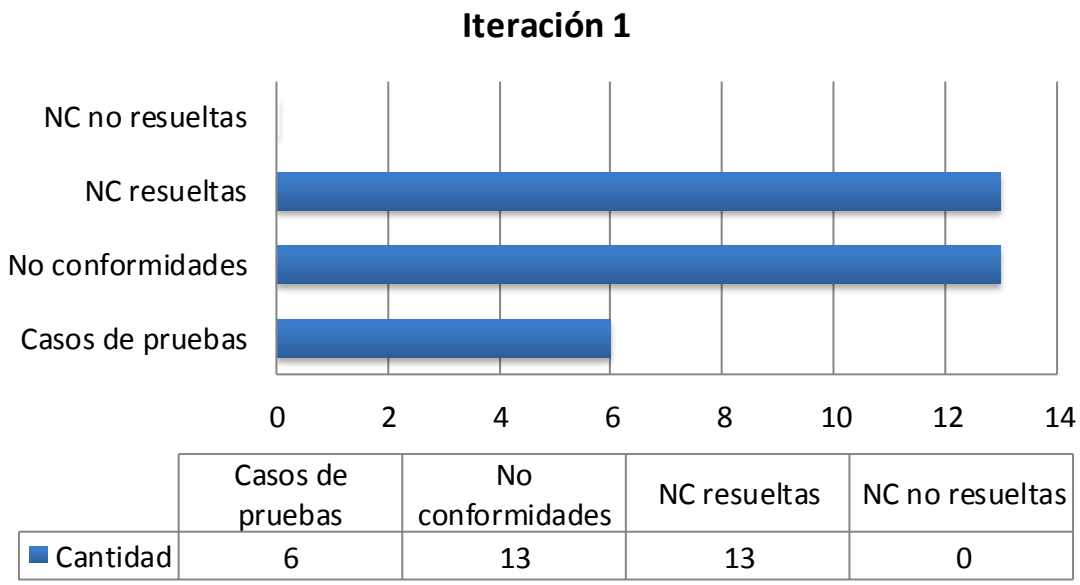


Figura 12: Primera iteración.

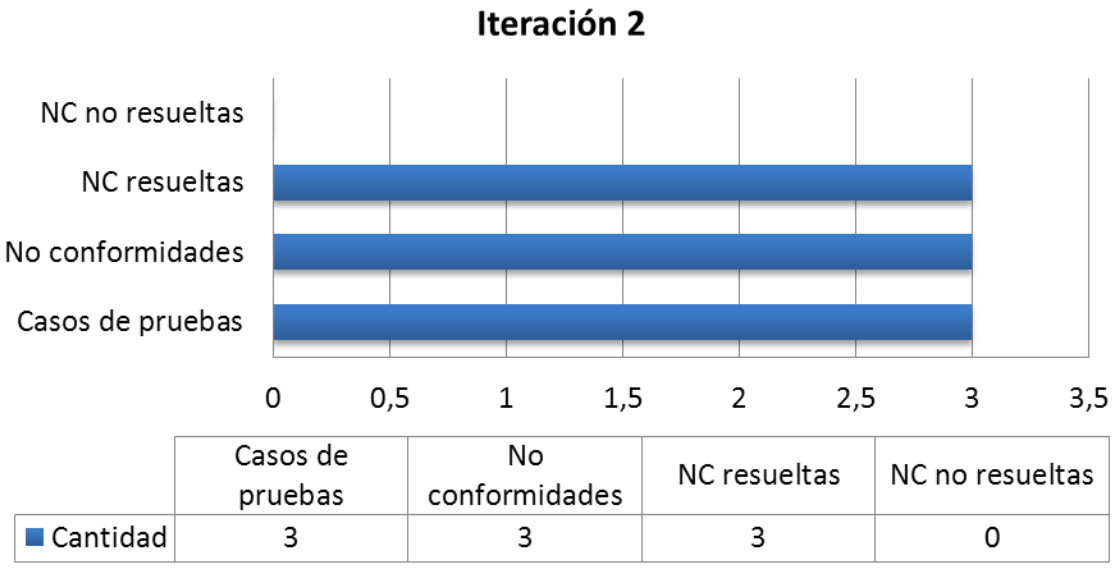


Figura 13: Segunda Iteración.

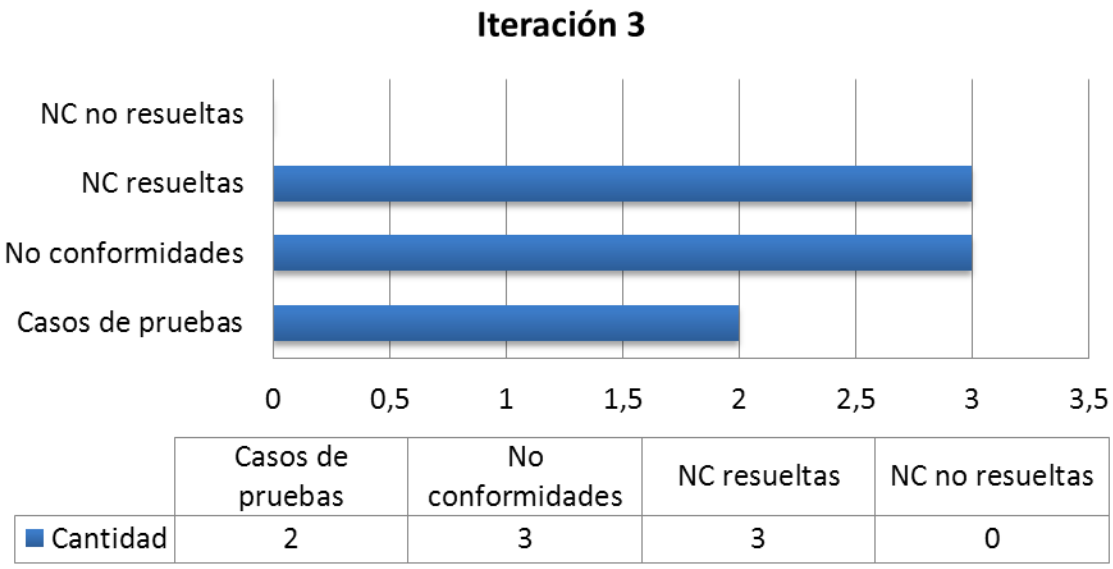


Figura 14: Tercera Iteración.

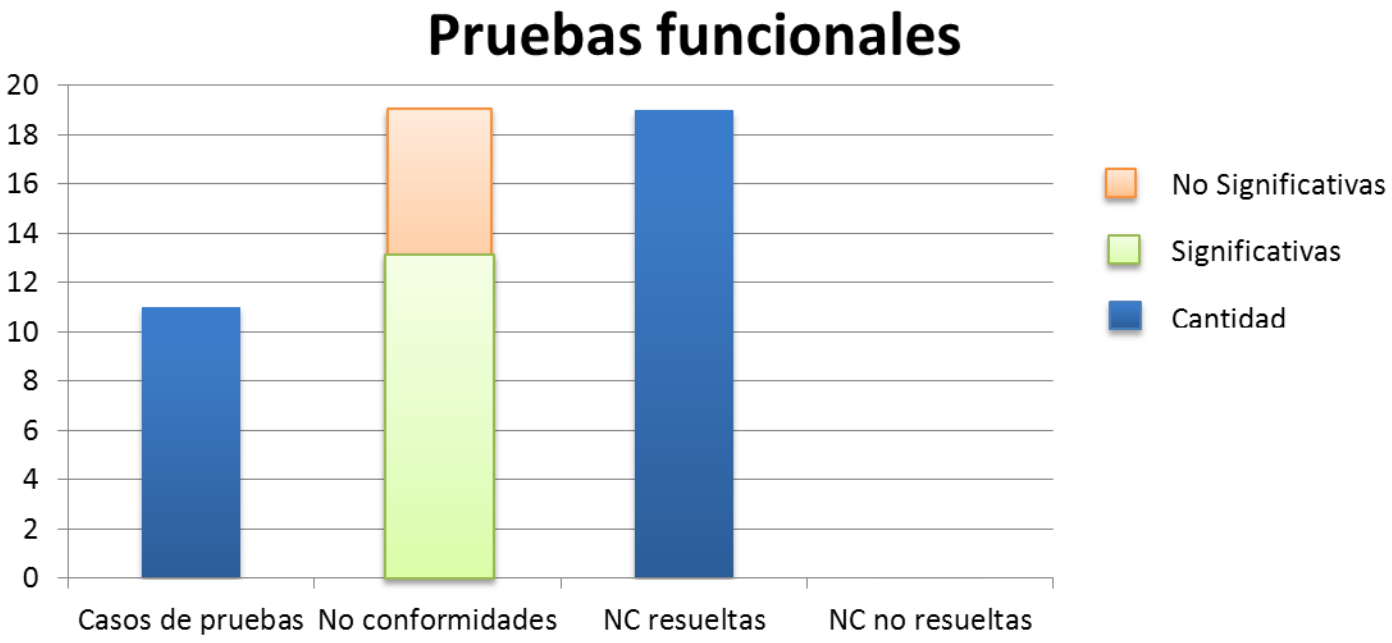


Figura 15: Pruebas funcionales.

3.3. Conclusiones Parciales.

Después que se definieron las HU y que se les dio una prioridad, se definieron las iteraciones en las que se encuentran cada una de estas HU de acuerdo al nivel de prioridad que poseen.

Con el fin de lograr conformidad y seguridad ante las funcionalidades del sistema se aplicaron las pruebas funcionales capaces de probar el sistema completamente, se aplicaron un total de 11 casos de pruebas, uno por cada HU, los cuales arrojaron 19 no conformidades, 13 en la primera iteración, 3 en la segunda y 3 en la tercera, las fueron solucionadas satisfactoriamente para un mejor funcionamiento del sistema.

CONCLUSIONES

Con el desarrollo del componente de monitoreo, se cumplió con los objetivos trazados en la investigación, obteniendo como resultado un componente integrado al Servidor de Gestión, capaz de monitorear y controlar el desempeño de los servidores PostgreSQL.

- El estudio de las herramientas de monitoreo existentes en Cuba y el mundo arrojó gran cantidad de información relevante para el desarrollo de la aplicación.
- El componente de monitoreo constituye una importante solución para administradores de bases de datos que permite llevar un control estricto de los servidores PostgreSQL que están bajo su supervisión.
- La realización de pruebas funcionales verificó el correcto funcionamiento del componente.

RECOMENDACIONES

Con el objetivo de lograr mejoras en el funcionamiento del componente de monitoreo se recomienda:

→ Continuar con el proceso de desarrollo del componente con el fin de agregarle nuevas métricas de acuerdo a las necesidades que se tenga.

→ Desarrollar la herramienta aplicando protocolo XMPP para el envío de mensajes instantáneos, en este caso los mensajes serían las métricas capturadas en cada unos de los servidores que se desea monitorear.

GLOSARIO DE TÉRMINOS

ManageEngine: Empresa productora de la Enterprise IT Managmet Software perteneciente a la corporación Zoho Corp.

PgBench: Herramienta muy útil e informativa para el monitoreo.

iotop: Herramienta desarrollada en Python, sencilla que permite ver los recursos consumidos por procesos y se puede encontrarla en el sitio web

lopp: Herramienta desarrollada en Postgres que sirve para medir las estadísticas de entrada/salida (i/o) por proceso.

pgstat: Herramienta desarrollada en Python, puede ser descargada del PgFoundry.

PgFouine: Herramienta más popular de revisión de logs y está escrita en PHP.

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: Es un 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.

pg_stat_database: Vista que define el gestor de bases de datos PostgreSQL utilizada para mostrar información relevante acerca de las bases de datos.

pg_stat_user_indexes: Vista que define el gestor de bases de datos PostgreSQL utilizada para mostrar información relevante acerca de los índices.

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

NC: no conformidades.

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.

Base de datos: Conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora. O sea, que una base de datos puede considerarse una colección de datos variables en el tiempo.

Sistema Gestor de Bases de Datos: Es el software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista.

Monitoreo: 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.

Componente de monitoreo: Es el encargado de recolectar la información de cada servidor PostgreSQL supervisado, así como capturar las métricas a medir, que permitirán evaluar el desempeño de los servidores.

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

Indicadores: Son valores que sirven de base para cuantificar conceptos medibles para una necesidad de Información, métodos cuantitativos de evaluación o predicción y ofrecen información para la toma de decisiones.

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.

REFERENCIAS BIBLIOGRÁFICAS

1. **Vázquez, Yudisney.** *Infraestructura Productiva.* PostgreSQL, UCI. 2010. pág. 10. Proyecto Técnico .

2. **García, Rosa María Mato.** *Diseño de bases de datos.* 1999. pág. 51.

3. **Korth, Henry y Silberschatz, Abraham.** *Fundamentos de las bases de datos.* Cuarta edición. 2002.

4. **PostgreSQL Global Development Group.** PostgreSQL. [En línea] 1996 – 2011. [Citado el: 19 de Octubre de 2010.] htttp://www.postgresql.org/.

5. **EnterpriseDB.** EnterpriseDB. *EnterpriseDB.* [En línea] [Citado el: 15 de Noviembre de 2010.] http://www.enterprisedb.com/products-services-training/products-overview.

6. **Command Prompt.** CMD. *CMD.* [En línea] 2010-2011. [Citado el: 19 de Octubre de 2010.] http://www.commandprompt.com/about/..

7. **Cybertec.** Cybertec. *Cybertec.* [En línea] 2010. [Citado el: 10 de diciembre de 2010.] http://www.cybertec.at/en/postgresql_products/cybercluster.

8. **Organización Internacional del Trabajo.** CINTENFOR. *CINTENFOR.* [En línea] 1996-2010. [Citado el: 5 de noviembre de 2010.] http://www.cinterfor.org.uy/public/spanish/region/ampro/cinterfor/temas/gender/em_ca_eq/m_eva.htm.

9. **Valle, Otto.** Monitoreo e Indicadores. [En línea] 2010. Texto de apoyo al proceso de construcción del Sistema Regional de Indicadores sobre Atención y Educación Inicial. http://www.oei.es/idie/mONITOREOEINDICADORES.pdf.

10. **A, Burns A. and Wellings.** *Real-time systems and programming languages.* University of York, Addison Wesley : s.n., 1996.

11. **José de Jesús Medel Juárez, Pedro Guevara López, Asdrúbal López Chau.** Aleph Zero. *Aleph Zero.* [En línea] http://aleph.cs.buap.mx.

12. **Nagios Enterprises.** Nagios. *Nagios.* [En línea] 2009-2011. [Citado el: 18 de Octubre de 2010.] http://www.nagios.com/..

13. **Manage Engine.** ZMA. *ZMA.* [En línea] 2010. [Citado el: 30 de octubre de 2010.] http://www.zma.com.ar/productos/infraestructura.php?producto_id=9.

14. **EnterpiseDB.** EnterpiseDB. *EnterpiseDB.* [En línea] 2010. [Citado el: 15 de Noviembre de 2010.] http://www.enterprisedb.com/products-services-training/products/postgres-plus-hq.

15. **Hyperic.** Spring Source Hyperic. *Spring Source Hyperic.* [En línea] 2010. [Citado el: 20 de noviembre de 2010.] http://www.hyperic.com/products/postgresql-monitoring.

16. **Hyperic**. Spring Source Hyperic. *Spring Source Hyperic.* [En línea] 2010. [Citado el: 16 de Noviembre de 2010.] http://www.hyperic.com/products/applications-monitoring.

17. **Vmware.** Spring Source. *Spring Source.* [En línea] 2011. [Citado el: 14 de Enero de 2011.] http://www.springsource.com/license/hyperic-enterprise.

18. **Consorcio SIU.** *Monitoreando y midiendo el rendimiento de un servidor Postgresql en sistemas Windows, Linux y Solaris.* pág. 10. Parte 1.

19. **Pressman, Roger S.** *Ingeniería del Software. Un enfoque práctico.* [ed.] Concepción Femández Madrid. 5ta. 1998. pág. 642. ISBN:84-481-3214-9.

20. **Fenton, Norman E. y Lawrence, Pfleeger Shari.** *Software Metrics. A rigorus and Practical Approach.* 1998. ISBN:0534954251 .

21. **IEEE.** [En línea] 17 de Julio de 1990. [Citado el: 15 de Noviembre de 2010.]

22. **ZOHO Corp.** Manage Engine. *Manage Engine.* [En línea] 2010. [Citado el: 16 de noviembre de 2010.] http://www.manageengine.com/products/applications_manager/postgresql-monitoring.html.

23. **EnterpriseDB.** EnterpriseDB. *EnterpriseDB.* [En línea] 2010. [Citado el: 15 de noviembre de 2010.] http://www.enterprisedb.com/products/postgres_plus_hq.do.

24. **Luis Olsina.** *Métricas e Indicadores.* 2003.

25. **Comunidad de Ubuntu.** doc.ubuntu-es. *doc.ubuntu-es.* [En línea] 7 de diciembre de 2009. [Citado el: 20 de noviembre de 2010.] http://doc.ubuntu-es.org/Sobre_Ubuntu.

26. **Collins Sussman, Ben.** Control de versiones con Subversion. [En línea] 2004. [Citado el: 16 de noviembre de 2010.] Revision 3820. http://svnbook.red-bean.com/nightly/es/svn-ch-1-sect-1.html.

27. **Jean Philippe Lang.** Redmine. *Redmine.* [En línea] 2006-2011. [Citado el: 14 de Enero de 2011.] http://www.redmine.org/.

28. **Alfresco Entrerprise Edition.** Alfresco. *Alfresco.* [En línea] 2011. [Citado el: 14 de Enero de 2011.] http://www.alfresco.com/.

29. **Visual Paradigm.** Visual Paradigm. *Visual Paradigm.* [En línea] 2011. [Citado el: 14 de Enero de 2011.] http://www.visual-paradigm.com/.

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

31. **Extreme Programming.** Extreme Programming. *Extreme Programming.* [En línea] 20 de Septiembre de 2009. [Citado el: 14 de Enero de 2010.] http://www.extremeprogramming.org/.

32. **Chodorow, Kristina y Dirolf, Michael.** *MongoDB: The Definitive Guide.* [ed.] Julie Steel. Primera. 2010. pág. 216. ISBN: 978-1-449-38156-1.

33. **Desarrollo Web.** Desarrollo Web. *Desarrollo Web.* [En línea] 30 de Junio de 2010. [Citado el: 15 de Enero de 2011.] http://www.desarrolloweb.com/actualidad/aptana-studio-3-beta-disponible-3659.html..

34. **Marcello Visconti; Hernán Astudillo.** Fundamentos de Ingeniería de Software. *Fundamentos de Ingeniería de Software.* [En línea] 2010. [Citado el: 30 de Enero de 2011.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.

BIBLIOGRAFÍA

1. **Alfresco Entrerprise Edition.** Alfresco. *Alfresco.* [En línea] 2011. [Citado el: 14 de Enero de 2011.] http://www.alfresco.com/.

2. **Chodorow, Kristina y Dirolf, Michael. 2010.** *MongoDB: The Definitive Guide.* [ed.] Julie Steel. Primera. 2010. pág. 216. ISBN: 978-1-449-38156-1.

3. **Collins Sussman, Ben. 2004.** Control de versiones con Subversion. [En línea] 2004. [Citado el: 16 de noviembre de 2010.] Revision 3820. http://svnbook.red-bean.com/nightly/es/svn-ch-1-sect-1.html.

4. **Command Prompt. 2010-2011.** CMD. *CMD.* [En línea] 2010-2011. [Citado el: 19 de Octubre de 2010.] http://www.commandprompt.com/about/..

5. **Comunidad de Ubuntu. 2009.** doc.ubuntu-es. *doc.ubuntu-es.* [En línea] 7 de diciembre de 2009. [Citado el: 20 de noviembre de 2010.] http://doc.ubuntu-es.org/Sobre_Ubuntu.

6. **Consorcio SIU.** *Monitoreando y midiendo el rendimiento de un servidor Postgresql en sistemas Windows, Linux y Solaris.* pág. 10. Parte 1.

7. **Cybertec. 2010.** Cybertec. *Cybertec.* [En línea] 2010. [Citado el: 10 de diciembre de 2010.] http://www.cybertec.at/en/postgresql_products/cybercluster.

8. **Desarrollo Web. 2010.** Desarrollo Web. *Desarrollo Web.* [En línea] 30 de Junio de 2010. [Citado el: 15 de Enero de 2011.] http://www.desarrolloweb.com/actualidad/aptana-studio-3-beta-disponible-3659.html..

9. **EnterpiseDB. 2010.** EnterpiseDB. *EnterpiseDB.* [En línea] 2010. [Citado el: 15 de Noviembre de 2010.] http://www.enterprisedb.com/products-services-training/products/postgres-plus-hq.

10. **EnterpriseDB. 2010.** EnterpriseDB. *EnterpriseDB.* [En línea] 2010. [Citado el: 15 de noviembre de 2010.] http://www.enterprisedb.com/products/postgres_plus_hq.do.

11. **EnterpriseDB.2010.** EnterpriseDB. *EnterpriseDB.* [En línea] [Citado el: 15 de Noviembre de 2010.] http://www.enterprisedb.com/products-services-training/products-overview.

12. **Extreme Programming. 2009.** Extreme Programming. *Extreme Programming.* [En línea] 20 de Septiembre de 2009. [Citado el: 14 de Enero de 2010.] http://www.extremeprogramming.org/.

13. **Fenton, Norman E. y Lawrence, Pfleeger Shari. 1998.** *Software Metrics. A rigorus and Practical Approach.* 1998. ISBN:0534954251 .

14. **García, Rosa María Mato. 1999.** *Diseño de bases de datos.* 1999. pág. 51.

15. **Guevara, P.** *Control de motores de corriente continua en Tiempo Real con capacidad de telecontrol y telemonitoreo. Tesis de Maestría en ciencias de la computación.* Instituto Politécnico Nacional, México. : s.n., 1999.

16. **Gutierrez, Jorge A. Saavedra. 2007.** El Mundo Informático. *El Mundo Informático.* [En línea] Jorge A. Saavedra Gutierrez, 8 de mayo de 2007. http://jorgesaavedra.wordpress.com/category/patrones-grasp/.

17. **Hyperic. 2010.** Spring Source Hyperic. *Spring Source Hyperic.* [En línea] 2010. [Citado el: 20 de noviembre de 2010.] http://www.hyperic.com/products/postgresql-monitoring.

18. **Hyperic. 2010.** Spring Source Hyperic. *Spring Source Hyperic.* [En línea] 2010. [Citado el: 20 de Noviembre de 2010.] http://www.hyperic.com/products/open-source-systems-monitoring.

19. **Hyperic. 2010.** Spring Source Hyperic. *Spring Source Hyperic.* [En línea] 2010. [Citado el: 16 de Noviembre de 2010.] http://www.hyperic.com/products/applications-monitoring.

20. **IEEE. 1990.** [En línea] 17 de Julio de 1990. [Citado el: 15 de Noviembre de 2010.]

21. **Jean Philippe Lang. 2006-2011.** Redmine. *Redmine.* [En línea] 2006-2011. [Citado el: 14 de Enero de 2011.] http://www.redmine.org/.

22. **José de Jesús Medel Juárez.** Aleph Zero. *Aleph Zero.* [En línea] http://aleph.cs.buap.mx.

23. **Korth, Henry y Silberschatz, Abraham. 2002.** *Fundamentos de las bases de datos.* Cuarta edición. 2002.

24. **Luis Olsina. 2003.** *Métricas e Indicadores.* 2003.

25. **Manage Engine. 2010.** ZMA. *ZMA.* [En línea] 2010. [Citado el: 30 de octubre de 2010.] http://www.zma.com.ar/productos/infraestructura.php?producto_id=9.

26. **Marcello Visconti; Hernán Astudillo. 2010.** Fundamentos de Ingeniería de Software. *Fundamentos de Ingeniería de Software.* [En línea] 2010. [Citado el: 30 de Enero de 2011.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.

27. *METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES.* **Figueroa, Robert G., J. Solis, Camilo y Cabrera, Armando A.** pág. 9. Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación.

28. **MongoDB**. *MongoDB.* [En línea] Rafael Hernampérez Martín, 06 de Agosto de 2010. [Citado el: 10 de 05 de 2011.] http://www.mongodb.org/display/DOCSES/Inicio.

29. **Nagios Enterprises. 2009-2011.** Nagios. *Nagios.* [En línea] 2009-2011. [Citado el: 18 de Octubre de 2010.] http://www.nagios.com/..

30. **Oracle Corporation. 2010.** Oracle. *Oracle.* [En línea] 2010. [Citado el: 20 de Enero de 2011.] http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html.

31. **Organización Internacional del Trabajo. 1996-2010.** CINTENFOR. *CINTENFOR.* [En línea] 1996-2010. [Citado el: 5 de noviembre de 2010.] http://www.cinterfor.org.uy/public/spanish/region/ampro/cinterfor/temas/gender/em_ca_eq/m_eva.htm.

32. **Ortíz, Yudisney Vázquez. 2010.** *Infraestructura Productiva.* PostgreSQL, UCI. 2010. pág. 10. Proyecto Técnico.

33. **PostgreSQL Global Development Group. 1996 – 2011.** PostgreSQL. [En línea] 1996 – 2011. [Citado el: 19 de Octubre de 2010.] htttp://www.postgresql.org/.

34. **Pressman, Roger S. 1998.** *Ingeniería del Software. Un enfoque práctico.* [ed.] Concepción Femández Madrid. 5ta. 1998. pág. 642. ISBN:84-481-3214-9.

35. **Python Software Foundation . 2010.** python. *python.* [En línea] 2010. http://www.python.org/.

36. **Valle, Otto. 2010.** Monitoreo e Indicadores. [En línea] 2010. Texto de apoyo al proceso de construcción del Sistema Regional de Indicadores sobre Atención y Educación Inicial. http://www.oei.es/idie/mONITOREOEINDICADORES.pdf.

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

38. **Vázquez, Yudisney. 2010.** *Infraestructura Productiva.* PostgreSQL, UCI. 2010. pág. 10. Proyecto Técnico .

39. **Visual Paradigm International . 2010.** Visual Paradigm. *Visual Paradigm.* [En línea] 2010. http://www.visual-paradigm.com/.

40. **Visual Paradigm. 2011.** Visual Paradigm. *Visual Paradigm.* [En línea] 2011. [Citado el: 14 de Enero de 2011.] http://www.visual-paradigm.com/.

41. **Vmware. 2011.** Spring Source. *Spring Source.* [En línea] 2011. [Citado el: 14 de Enero de 2011.] http://www.springsource.com/license/hyperic-enterprise.

42. **ZOHO Corp. 2010.** Manage Engine. *Manage Engine.* [En línea] 2010. [Citado el: 16 de noviembre de 2010.] http://www.manageengine.com/products/applications_manager/postgresql-monitoring.html.

ANEXOS

Anexo 1.

Nombre de la base de datos: lp00-003-14

Colección: cantidad_base_datos

[{“datname”: “postgres”, “hora”: “15:00”},
{“datname”: “pruebas”, “hora”: “15:01”},
{“datname”: “biblioteca”, “hora”: “15:02”},
{“datname”: “componente”, “hora”: “15:03”}]

Colección: cantidad_de_commit

[{"datname" : "postgres", "xact_commit" : 1668 },
{"datname" : "pruebas", "xact_commit" : 5 },
{"datname" : "biblioteca", "xact_commit" : 54 },
{"datname" : "componente", "xact_commit" : 3 }]

Colección: cantidad_de_rollbacks

[{"datname" : "postgres", "xact_rollback" : 0 },
{ "datname" : "pruebas", "xact_rollback" : 0 },
{ "datname" : "biblioteca", "xact_rollback" : 0 },
{ "datname" : "componente", "xact_rollback" : 0 }]

Colección: cantidad_de_conexiones

[{ "numbackends" : 0, "datname" : "postgres" },
{ "numbackends" : 0, "datname" : "Pruebaa" },
{ "numbackends" : 0, "datname" : "biblioteca" },
{ "numbackends" : 1, "datname" : "componente" }]

Colección: espacio_en_disco

[{ "postgres" : 83084, "libre" : 19780644, "sistema" : 8664784, "otros" : 117214660, "total" : 153374544 },
{ "postgres" : 83084, "libre" : 19780644, "sistema" : 8664784, "otros" : 117214660, "total" : 153374544 },
{ "postgres" : 83084, "libre" : 19780644, "sistema" : 8664784, "otros" : 117214660, "total" : 153374544 },
{ "postgres" : 83084, "libre" : 19780644, "sistema" : 8664784, "otros" : 117214660, "total" : 153374744 }]

Colección: indices_escaneados

[{"Indices Escaneados" : 12 },
{ "Indices Escaneados" : 12 },
{ "Indices Escaneados" : 12 },
{ "Indices Escaneados" : 12 },
{ "Indices Escaneados" : 12 }]

Colección: scaneo_secuencial

[{ "Escaneo Secuencial" : 24 },
{ "Escaneo Secuencial" : 24 },
{ "Escaneo Secuencial" : 24 },
{ "Escaneo Secuencial" : 24 },
{ "Escaneo Secuencial" : 24 }]

Colección: total_de_indices

[{ "Total de Indices" : 72 },
{ "Total de Indices" : 72 },
{ "Total de Indices" : 72 },
{ "Total de Indices" : 72 }]

Colección: tuplas_actualizadas

[{ "datname" : "template1", "tup_updated" : 0 },

{ "datname" : "template0", "tup_updated" : 0 },
{ "datname" : "postgres", "tup_updated" : 0 },
{ "datname" : "Pruebaa", "tup_updated" : 0 }]

Colección: tuplas_eliminadas

[{ "datname" : "template1", "tup_deleted" : 0 },
{ "datname" : "template0", "tup_deleted" : 0 },
{ "datname" : "postgres", "tup_deleted" : 0 },
{ "datname" : "Pruebaa", "tup_deleted" : 0 }]

Colección: tuplas_insertadas

[{ "datname" : "template1", "tup_inserted" : 0 },
{ "datname" : "template0", "tup_inserted" : 0 },
{ "datname" : "postgres", "tup_inserted" : 0 },
{ "datname" : "Pruebaa", "tup_inserted" : 0 }]

Colección: tuplas_leidas_x_scaneo_secuenciales

[{ "Tuplas leidas por escaneo secuenciales" : 0 },
{ "Tuplas leidas por escaneo secuenciales" : 0 },
{ "Tuplas leidas por escaneo secuenciales" : 0 },
{ "Tuplas leidas por escaneo secuenciales" : 0 }]

Colección: tuplas_muertas

[{ "Tuplas Muertas" : 0 },
{ "Tuplas Muertas" : 0 },
{ "Tuplas Muertas" : 0 },
{ "Tuplas Muertas" : 0 }]

Colección: tuplas_retornadas

[{ "datname" : "template1", "tup_returned" : 0 },
{ "datname" : "template0", "tup_returned" : 0 },
{ "datname" : "postgres", "tup_returned" : 462841 },

{ "datname" : "Pruebaa", "tup_returned" : 0 }]

Colección: tuplas_vivas

[{"Tuplas Vivas" : 0 },
{ "Tuplas Vivas" : 0 },
{ "Tuplas Vivas" : 0 },
{ "Tuplas Vivas" : 0 }]

Colección: uso_de_memoria

[{ "total" : 984, "usada" : 636, "libre" : 348 },
{ "total" : 984, "usada" : 638, "libre" : 345 },
{ "total" : 984, "usada" : 640, "libre" : 343 },
{ "total" : 984, "usada" : 638, "libre" : 346 }]

La estructura de documento y colecciones anteriormente representada será la misma para cada servidor PostgreSQL añadido a MongoDB.