Universidad de las Ciencias Informáticas
Facultad 6

Componente de reportes para la monitorización a servidores de bases de datos PostgreSQL

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Raiko Pulido Avila

Tutor: Ing. Yoan M. Pérez Piñero.

Co-Tutor: Ing. Marcos Luis Ortiz Valmaseda.

Consultantes: Ing. Yudisney Vázquez Ortiz, Ing. Jorge L. Valdés González.

La Habana, Cuba

"Año 53 de la Revolución"

Junio 2011

"(...) Quien pretenda en las aulas universitarias vivir solo con el oxígeno que en ellas se respira, morirá a causa de una asfixia cultural (...)"

J. F. Bulté

DECLARACIÓN DE AUTORÍA

Declaro que somos los únicos autores de este trabajo y autorizamos 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 _____.

_____ _____

Tutor: Yoan M. Pérez Piñero **Autor:** Raiko Pulido Avila

DATOS DEL CONTACTOS

Contacto: Ing. Yoan M. Pérez Piñero.
Datos: graduado en la Universidad de las Ciencias Informáticas.
Correo electrónico: ymperez@uci.cu.

AGRADECIMIENTOS

A mis padres por ser mi apoyo incondicional en todo momento, y por estar siempre a mi lado cuando más los necesité.

A mi hermana, la persona que más quiero en este mundo, por ser mi luz y mi guía.

A mi queridísima Taidy que aunque su nombre no se encuentre en este trabajo hoy, está plasmado en cada línea, en cada capitulo, en cada palabra que pueda decir, a ella que es mi compañera de tesis de siempre, a ti muchas gracias por enseñarme que todo se puede, por darme fuerzas, por mostrarme el camino y guiarme cuando estaba perdido, por estar siempre firme a mi lado.

A Jorge por siempre brindarme su mano, por apoyarme en la búsqueda de soluciones, por ser un guía, por habernos adiestrado en el tema de la investigación, para ti mis más sinceros agradecimientos.

A mi tutor Yoan por la tutoría de este trabajo, por sus sugerencias, por su confianza, por dedicarme cada minuto, cada hora, cada día, y lo mas importante por haberse convertido en un amigo.

A Marco por ser uno de los principales consejeros y guías de esta investigación.

A Yudelki, Teresa, Ivette, por estar siempre pendiente de todos mis problemas, por guiarme cuando voy por mal camino, por ser mas que amigas, por ser mis hermanas.

A Rosnel, por haber aportado todo sus conocimientos y su tiempo libre para que el trabajo saliera a flote.

A Yixander que es un amigo incondicional y estuvo presente en cada línea de código que programé y en cada interfaz que diseñé, a ti mi amigo te agradezco que siempre estuviste a mi lado brindándome tu más sincero apoyo.

A Mikel por apoyarme todo estos años.

A la música por ser mi refugio en los momentos más duros, a mi grupo musical por comprenderme siempre y brindarme su apoyo.

A Yosel, Orisbel, Osdel, Orelmi, Paneque, Mario, Paulo, Yasel, Yadrian, Orelvi por apoyarme y brindarme su mano cuando la necesite.

A todo el Departamento de PostgreSQL Empresarial, ustedes son los mejores, de verdad le agradezco su apoyo en especial a la profesora Yudisney.

A todos ustedes muchas gracias.

DEDICATORIA

Dedicado este trabajo especialmente a mis padres que siempre esperaron ver este sueño hecho realidad y hoy le hago este regalo a ustedes que se sacrificaron para que yo hoy pudiera plasmar estas líneas y decirle en cada fragmento de este trabajo, en cada código, todo el amor que siento por ustedes. A mi hermana por estar siempre pendiente de mis problemas, por darme lo más grande de este mundo, que es su cariño.

RESUMEN

En Cuba se han ido dando pasos de avance en el desarrollo de soluciones informáticas que utilizan a PostgreSQL como Sistema Gestor de Base de Datos. Un elemento que se tiene en cuenta es garantizar un máximo nivel de soberanía tecnológica. Debido a esto, se decide abrir varios temas de investigación, entre ellos la implementación de un Servidor de Gestión.

El objetivo de este trabajo es desarrollar un componente de reporte que contribuya a mejorar la facilidad de uso, la eficiencia y la satisfacción de los usuarios en el proceso de monitorización a servidores de bases de datos PostgreSQL. La interfaz de usuario que provee el gestor para la ejecución de comandos e instrucciones que tributen información para el proceso de monitorización, no garantiza al usuario final transparencia en dicho proceso.

La investigación arrojará una aplicación web capaz de generar reportes de las métricas asociadas al rendimiento de los servidores de base de datos PostgreSQL, contribuyendo totalmente a la soberanía tecnológica.

Palabras Claves: Sistema Gestor de Bases de Datos, Servidor de Gestión, facilidad de uso, monitorización, servidores, reporte, PostgreSQL

Índice

Índice de figuras:

Índice de tablas:

INTRODUCCIÓN

La difusión masiva de las Tecnologías de la Información y las Comunicaciones (TIC) en Cuba ha constituido la principal estrategia para lograr un desarrollo tecnológico que responda a los principales intereses del Estado y el pueblo cubano. Esto se traduce en convertir los conocimientos y las TIC, en instrumentos a disposición del avance y las profundas transformaciones revolucionarias. Cuantiosos son los logros que se han alcanzado hasta el momento como resultado de los esfuerzos realizados por el Estado cubano, y esto ha traído aparejado la inserción de las TIC en casi todas las ramas de la sociedad.

En la actualidad se sigue perfeccionando el trabajo y ampliando el radio de acción de las nuevas tecnologías en beneficio de todas las instituciones cubanas. En este sentido, la proyección de ambiciosas metas no constituye un mero sueño o utopía, sino un claro reflejo en la continuidad de los resultados obtenidos hasta el momento, en materia de desarrollo tecnológico. Un ejemplo de ello es lo alcanzado a partir de los avances en la industria cubana para el desarrollo de sistemas informáticos que, con la participación de la Universidad de las Ciencias Informáticas (UCI) y otras instituciones del país, se han magnificado permitiendo de esta manera ganar terreno en el mercado internacional y sustituir innecesarias importaciones de productos informáticos, potenciando la construcción de insumos tecnológicos propios.

La UCI organiza su modelo de producción con el fin de responder a sus necesidades específicas, cuyo propósito principal es obtener un avance tecnológico que garantice un aumento continuo de exportaciones y lograr apoyar desde la gestión de los proyectos y el perfeccionamiento de tecnologías, al trabajo político ideológico y los servicios de formación. Con la intención de cumplir con sus compromisos, desde el enfoque de la gestión de los proyectos y las tecnologías que lo soportan, el modelo de desarrollo tecnológico de la organización integra varios niveles:

- Centros de desarrollo.
- Servicios centralizados (soporte, auditoría y control).
- Desarrollo estratégico en la alta gerencia de la producción.

A partir de esta concepción estructural y de las proyecciones estratégicas de la UCI de contar con un centro que garantice el desarrollo de aplicaciones fomentando el uso de técnicas de base de datos libres, se crea en septiembre de 2008, el Centro de Tecnologías de Gestión de Datos (DATEC). Entre los principales objetivos del centro está el de contribuir a la soberanía

tecnológica, potenciando tecnologías de bases de datos libres tomando como base el Sistema Gestor de Bases de Datos (SGBD) PostgreSQL (1).

El centro DATEC internamente está estructurado por departamentos productivos especializados, como son: el Departamento de Soluciones Integrales, Departamento de Almacenes de datos, Departamento de Bioinformática y Departamento de PostgreSQL Empresarial. Este último se constituye con el objetivo de brindar solución a un grupo de dificultades que presentaba DATEC y la universidad (1), entre las cuales se encontraban:

- Inexistencia en el país de herramientas con entorno gráfico, que ofrezcan facilidades para la migración entre versiones del gestor de bases de datos PostgreSQL.
- Inexistencia de instaladores en modo gráfico y texto, que permitan la personalización del gestor PostgreSQL con funcionalidades de análisis de datos, monitorización, reportes, administración, desarrollo y seguridad, que permita la selección de qué lenguajes procedurales instalar y las opciones de los principales parámetros de configuración para montar un sistema óptimo.
- Inexistencia en el país de un Servidor de Gestión que posibilita la monitorización de los servidores en cuestión y a sus bases de datos, que permita la generación de reportes con determinados indicadores y la planificación de tareas.

El desarrollo de soluciones informáticas orientada a la gestión de las bases de datos no es un fenómeno aislado sino que forma parte indisoluble de la propia dinámica de desarrollo de la sociedad. Esta afirmación tiene su fundamento en lo que significa para la sociedad hoy poder persistir y gestionar de manera eficiente todo el cúmulo de información que se gestiona en cualquiera de los escenarios sobre los que se proyecta. Sin embargo, desde la óptica de los sistemas informáticos, no son las bases de datos por sí solas las que le imprimen, en cierta medida, un valor agregado a la gestión de la información en el ámbito digital, sino que éstas, en su gran mayoría, deberán estar asociadas a un SGBD. De este modo se puede permitir el manejo de forma clara, sencilla y ordenada de un conjunto de datos que posteriormente se convertirán en información relevante para una determinada organización o individuo.

Son pocas las empresas que gozan de éxito en el desarrollo de sistemas de esta naturaleza. Según el histórico del estudio anual que realiza Gartner[1], algunas de estas exitosas empresas en materia de desarrollo de SGBD son las corporaciones Oracle, con el SGBD del mismo nombre y Microsoft con SQL Server, que en este caso desarrollan sus soluciones bajo licencias privativas (2). En cambio existen otros que se distribuyen bajo licencias de código abierto que disfrutan de igual prestigio, popularidad y éxito mundial, como es el caso de PostgreSQL[2,] MySQL, Firebird SQLite, DB2 Express-C, Apache Derby, entre otros; aunque es perfectamente cuestionable hoy día la conveniencia de emplear SGBD como MySQL, producto a que la empresa patrocinadora es la Corporación Oracle, la cual posee el derecho de copia de la mayor parte del código.

Cuba ha potenciado el desarrollo de soluciones informáticas que emplean como SGBD a PostgreSQL respondiendo, principalmente, a la política de migración a plataformas libres. En sentido general, a partir de la voluntad política del Estado y las instituciones cubanas, se han puesto en marcha proyectos nacionales de migración que, en definitiva, constituyen acciones concretas como la propuesta de un documento en el marco del V Taller Internacional de Software Libre, a propósito de la XII Feria Internacional Informática 2009. En ese documento se exponía una guía para que las instituciones organizaran su propia migración según sus características (3). Por otro lado, no ha sido esta la única razón por la cual se ha decidido concentrar esfuerzos en el desarrollo sobre este SGBD.

Otros elementos han sido la necesidad de garantizar un máximo nivel de soberanía tecnológica, lo que se traduce en este caso, en el uso y comercialización de tecnologías informáticas desarrolladas por entidades cubanas. De esta manera se evitan los riesgos que traen las dependencias con entidades foráneas en materia tecnológica, a partir de aprovechar las bondades de las políticas de código abierto, libre distribución y libre modificación, para desarrollar servidores de gestión basados en PostgreSQL ajustados a las necesidades y realidades cubanas.

La razón fundamental para el desarrollo de un Servidor de Gestión de PostgreSQL propio de la UCI, consiste en lo siguiente: PostgreSQL es un SGBD de código abierto con una arquitectura cliente/servidor basada en un núcleo (o motor) desarrollado por la comunidad PGDG (4)

[1]Gartner, Inc. (NYSE: IT) es líder mundial en investigación sobre tecnologías de información y en el asesoramiento de empresas. http://www.gartner.com/technology/home.jsp.

[2]http://www.postgre.org

encargada exclusivamente de esa tarea, salvo en algunas excepciones cuando se han orientado al desarrollo de funcionalidades que vienen a complementar el funcionamiento del motor de datos del propio sistema gestor. En este sentido la programación de funcionalidades especializadas, como optimizadores de consultas, sistemas de monitorización, gestores de tareas programadas, entre otras, deviene en una imperiosa necesidad para aquellas entidades que incorporan el uso de este sistema en la comercialización o insumo de productos informáticos.

Empresas, como la EnterpriseDB, se valen del derecho de modificación que otorga el licenciamiento de PostgreSQL, para crear su propio Servidor de Gestión de PostgreSQL y distribuirlo libremente como un producto propio. La empresa EnterpriseDB desarrolla dos distribuciones de servidores de gestión basados en PostgreSQL denominadas Postgre Plus Standard Server y Postgre Plus Advanced Server, los que pueden adquirirse bajo los mismos términos de código abierto, pero que finalmente presentan dependencias directas con el futuro de las políticas de la propia empresa en cuanto al soporte que se les brinda a estas aplicaciones (5). De manera que desde el punto de vista de la soberanía tecnológica, no son una solución factible para el caso de Cuba.

El presente trabajo centra su atención específicamente en la adecuada presentación de la información resultante del proceso de monitorización a servidores PostgreSQL y en cómo abordar la problemática fundamental que anima el desarrollo del mismo. En este sentido se ha identificado que:

- Para realizar procesos de monitorización a servidores PostgreSQL a través de las interfaces que brinda el gestor, el usuario debe tener dominio de detalles técnicos asociados a comandos y programación de instrucciones, dilatando de este modo la curva de aprendizaje al tener que memorizar comandos y procedimientos, en tanto se eleva la taza de errores cometidos debido a que es el usuario el responsable de programar las instrucciones que conforman las métricas para la monitorización.

- La interfaz de usuario que provee el gestor para la ejecución de comandos e instrucciones que tributen información para el proceso de monitorización, no garantiza al usuario final transparencia en dicho proceso, lo que va en detrimento de la eficiencia y la efectividad para lograr los objetivos sustantivos del proceso en cuestión, además de no incluir la capacidad de brindar información estructurada en gráficos que contribuyan a una adecuada y eficiente interpretación de estos datos.

Todo lo anteriormente expuesto contribuyó a la identificación del siguiente **problema**: Las interfaces que provee el Sistema Gestor de Bases de Datos PostgreSQL para realizar la monitorización a este tipo de servidores, incide negativamente en la facilidad de uso, efectividad y eficiencia para desarrollar este proceso.

En el presente trabajo se establece como **objeto de estudio** los Sistemas de Monitorización en tiempo real a Bases de Datos, siendo el **campo de acción** las herramientas de reportes de los sistemas de monitorización en tiempo real a servidores de bases de datos PostgreSQL. El **objetivo general** consiste en desarrollar un componente de reportes que contribuya a mejorar la facilidad de uso, efectividad y eficiencia en el proceso de monitorización a servidores de bases de datos PostgreSQL. En tanto los **objetivos específicos** son:

1. Elaborar el marco teórico de la investigación.
2. Diseñar el componente de reportes de acuerdo a la metodología seleccionada.
3. Implementar el sistema a partir del diseño realizado.
4. Validar la solución desarrollada a partir de la ejecución de pruebas de software.

Para dar cumplimiento al objetivo general se definen las siguientes tareas de investigación, donde cada una tributa en alguna medida a los objetivos específicos:

1. Análisis del estado del arte y tendencias sobre herramientas de reportes utilizadas en los procesos de monitorización a servidores de bases de datos PostgreSQL.
2. Definición de requerimientos del componente de reportes.
3. Definición de la arquitectura de software del sistema.
4. Implantación del entorno tecnológico para el desarrollo de la solución.
5. Diseño e implantación de la base de datos para el funcionamiento interno del sistema.
6. Diseño de la base de datos para la consulta y actualización de las estadísticas obtenidas de los servidores PostgreSQL.
7. Implementación de las funcionalidades del componente de reportes.
8. Definición de las pruebas a realizar.
9. Ejecución del plan de pruebas para determinar el cumplimiento de los requisitos establecidos del sistema.

El presente trabajo responde a una necesidad concreta del centro DATEC, y más generalmente a una necesidad de Cuba, en función de independizar las tecnologías informáticas de las fluctuaciones del mercado mundial. En este sentido se espera desarrollar un sistema informático de reportes en ambiente Web, para la monitorización a servidores de bases de datos PostgreSQL, como parte de una suite de monitorización (DashBoard) para el Servidor de Gestión PostgreSQL. Ha sido estructurado de la siguiente forma:

En el **capítulo 1** se realiza el análisis del estado del arte sobre servidores de gestión, del desarrollo de componentes tecnológicos orientados a la monitorización en tiempo real y un análisis desde el punto de vista teórico de las tecnologías y la metodología necesarias para la concepción de este tipo de componentes.

En el **capítulo 2** se realiza el diseño de la solución a partir de la metodología adoptada y se realiza además la identificación de los requerimientos del sistema y una descripción general de los elementos conceptuales de la arquitectura del mismo.

En el **capítulo 3** se realiza la validación del sistema a partir del diseño y aplicación de pruebas de contrastación de requerimientos.

CAPÍTULO 1

INTRODUCCIÓN

En este capítulo se realiza una investigación de los procesos de monitorización a servidores de bases de datos PostgreSQL, así como algunas experiencias de éxitos en el mundo que sirvan de punto de partida para el desarrollo de la presente investigación, además de efectuarse un análisis de algunos de los principales conceptos que intervienen en la problemática.

Se realiza conjuntamente una evaluación de las actividades de monitorización a servidores PostgreSQL, así como los elementos negativos que presenta este gestor para la realización de diferentes procesos como es el caso la monitorización en tiempo real.

1.1 Servidores de Gestión de Bases de Datos

Los servidores de gestión de bases de datos surgen por la necesidad existente en las empresas de poder monitorizar los grandes y complejos volúmenes de datos. Estos son utilizados para brindar información a los ordenadores que se conecten a él y además deben ser capaces de proveer un conjunto de aplicaciones que permitan administrar, gestionar y monitorizar el propio SGBD al cual le debe su existencia. A estas tres características es necesario añadirle una más, y es que el servidor debe ser capacitado para prestar servicios hacia el desarrollo y manejo de su propio gestor.

No deben confundirse los términos de Servidor de Gestión de Bases de Datos y Sistemas Gestores de Bases de Datos. Los sistemas gestores de bases de datos, entre otras funciones, se encargan de la normalización y estructura de los datos; incorporan lenguajes de acceso que permiten además construir programas especializados (consultas, reportes, entre otros) que se encargan de tratar con dichos datos en un formato estandarizado; los que devienen en una interfaz más amigable[3] entre usuarios y otros sistemas computacionales, con respecto a la interfaz del sistema de ficheros de los sistemas operativos y los dispositivos físicos de almacenamiento (6). En tanto los servidores de gestión se encargan de enriquecer la concepción

[3] Desde el punto de vista de la transparencia en el acceso a procedimientos e información sobre estándares y protocolos más complejos.

de lo que son los Sistemas Gestores de Bases de Datos, incorporando otras funcionalidades y abstracciones, permitiendo de este modo contar con un sistema que posee otras características particulares, como pueden ser la capacidad de monitorización y administración remota, entre otras.

En esencia estos dos conceptos, los Servidores de Gestión de Bases de Datos y los Sistemas Gestores de Bases de Datos, no se contraponen sino que se complementan.

1.2 Sistemas Gestores de Bases de Datos

Los sistemas gestores de base de datos se definen como el software que permite la utilización y/o la actualización de los datos almacenados en una o varias bases de datos por uno o varios usuarios desde diferentes puntos de vista. Su objetivo fundamental es el de suministrar al usuario las herramientas que le permitan manipular, en términos genéricos, los datos, es decir, de forma que no le sea necesario conocer el modo de almacenamiento de los mismos en la computadora, ni el método de acceso empleado (7).

La IEEE define como Sistema Gestor de Base de Datos a *"(…) un sistema informático compuesto por hardware, software o ambos, que proporciona una técnica sistemática para la creación, el almacenamiento, el procesamiento y la consulta de la información almacenada en base de datos. Un SGBD actúa como un intermediario entre las aplicaciones y los datos, o bien entre los datos y la base de datos (…)"* (8). Otra forma de definirlo es como el componente de software más importante de un sistema informático y constituye un complejo conjunto de programas que son controlados por la organización para almacenar, gestionar y recuperar datos (9).

1.2.1 El Sistema Gestor de Bases de Datos PostgreSQL

PostgreSQL es un potente SGBD relacional orientado a objetos de uso libre y de código abierto. Usa la licencia BSD (*Berkeley Software Distribution*), lo que significa que el código fuente licenciado debe mantener su información de derecho de copia y licencia (10). La licencia BSD es certificada por la Iniciativa de Código Abierto (OSI).

Este gestor es soportado por un gran número de plataformas, entre las que se encuentran Windows, GNU/Linux, Solaris y sistemas modernos Unix. Además de permitir su ejecución nativa sobre sistemas basados en Microsoft, Windows NT como Win2000 SP4, WinXP y Win2003. Incluso, las versiones de Windows basadas en MSDOS (Win95, Win98, WinME) pueden correr

PostgreSQL usando Cygwin[4]. Está basado en un arquitectura cliente/servidor capaz de ser embebido siempre y cuando no se requiera del servidor PostgreSQL para realizar procesos de solicitud, ya que PostgreSQL define procesos separados para cada cliente y servidor. Tiene interfaces disponibles para los lenguajes C, PHP, TCL, Perl, Python, entre otros (10). Con una gran capacidad máxima de almacenamiento de las bases de datos (en los niveles de los Tera Byte), una inmensa comunidad para el desarrollo y el goce de un prestigio mundial, PostgreSQL se ha convertido en una de las principales opciones de innumerables empresas, organizaciones y países en toda la extensión del globo.

Una de las empresas que utiliza este gestor es Cybertec, la que desarrolla un producto de código abierto para la replicación síncrona y asíncrona en bases de datos. Cybercluster, denominación que recibe la herramienta de réplica, es capaz de reproducir y sincronizar datos a través de una red de manera eficiente. En su versión 2.0, está basado completamente en PostgreSQL 9.0, y es capaz de escalar la infraestructura de bases de datos de manera rápida y eficiente (11).

Como se ha mencionado en el inicio del presente trabajo, otra importante empresa que desarrolla soluciones informáticas basadas en PostgreSQL es la EnterpriseDB con los sistemas Postgre Plus Standard Server y Postgre Plus Advanced Server (5). Esta corporación constituye una interesante experiencia de obligatoria referencia, para la concepción de un Servidor de Gestión de PostgreSQL para Cuba.

En otro orden, existen otras grandes empresas en todo el mundo que son asiduos usuarios del SGBD PostgreSQL, tales como (4), Badische Anilin und Soda Fabrik (BASF), cuyo portal comercial (Arkdom Commerce Enterprise Edition) es desarrollado por Web Commerce Group (WCG) y que en la concepción de su arquitectura despliega servidores PostgreSQL para garantizar bajos costos, proveer de una mejor tecnología y brindar mejores servicios de mantenimiento y soporte a sus clientes (12).

Unos de los usuarios que cuenta con gran experiencia en el desarrollo e implementación de aplicaciones usando PostgreSQL como gestor, es la empresa Vanten Inc, la cual desarrolla soluciones informáticas de código abierto para el mercado japonés. Es respaldada por Daiei OMC, Japan.s. Vanten ha seleccionado PostgreSQL como uno de los componentes clave para

[4] Es una colección de herramientas desarrollada por Cygnus Solutions para proporcionar un comportamiento similar a los sistemas Unix en Windows.

su desarrollo, confiando en el poder y la extensibilidad de este gestor, además de asegurar con éste que su negocio sea más eficiente y rentable (4).

Otra significativa empresa que utiliza como gestor a PostgreSQL es Shannon Medical Center, la que se dedica a realizar análisis de datos para los sistemas de asistencia financiera, operativa, y clínicas de los hospitales e instituciones sanitarias. Con el uso de PostgreSQL como gestor, Shannon Medical Center garantizó un sistema con la máxima flexibilidad y control y a un costo mínimo, según Andrew Gould. Gracias a los estándares abiertos que utiliza PostgreSQL y la facilidad de uso en una compleja infraestructura de las TIC, la empresa cuenta con un gran éxito en el despliegue de la tecnología, con poca dificultad y aportando gran beneficio para sus clientes (4).

Según entrevistas realizadas a especialistas de la Empresa de Telecomunicaciones de Cuba (ETECSA), la Empresa Cubana de Aeropuertos y Servicios Aeronáuticos SA (ECASA), la división de la Empresa para el Desarrollo de Software en Cuba (DESOFT) en la provincia Ciego de Ávila y a la Dirección de Tecnología de la UCI; así como a un especialista de redes informáticas en el Ministerio de Finanzas y Precios, con el objetivo de evaluar la utilización de PostgreSQL, se identificó que:

- Excepto la división de la empresa DESOFT en la provincia de Ciego de Ávila, todas las entidades entrevistadas, que representan el 80%, emplean PostgreSQL como gestor de bases de datos.
- El 80% de los especialistas entrevistados consideran que el desarrollo en Cuba de un servidor de gestión de PostgreSQL con tecnología libre, orientado a brindar servicios de monitorización, favorecería la independencia tecnológica del país, en tanto solo un 20% se pronunció a favor de limitarse a emplear experiencias existentes en el mundo que ya hayan tenido un nivel de aceptación y prueba en la comunidad internacional.
- En la entrevista realizada a un especialista de la Dirección de Tecnología de la UCI, se identificó que al menos un 51.87% de los proyectos de la universidad emplean como gestor de bases de datos PostgreSQL, lo que indica un índice de uso significativo, respecto a otros SGBD (13).

1.3 Sistemas de monitorización y servidores de gestión de bases de datos. Algunas experiencias internacionales de implementación y uso

Según el Centro Internacional de Investigaciones para el Desarrollo (IDRC), la técnica de monitorización se puede definir como "*un proceso permanente para verificar sistemáticamente que las actividades o procesos planificados se llevan a cabo según lo esperado o que se está progresando en el logro de los resultados planificados*" (14). Una interpretación de esta definición aplicada al ámbito de la informática, específicamente al monitorizar servidores, conduce al siguiente razonamiento: monitorizar servidores de bases de datos implica una constante verificación de los indicadores o métricas de funcionamiento y rendimiento, evaluación del cumplimiento de configuraciones y tareas programadas, así como una sistematización en el registro de históricos que permita realizar análisis comparativos, de evolución y auditorías de procesos.

Son varias las empresas en el mundo que han orientado su desarrollo a la obtención de herramientas para la monitorización de servidores, sistemas y otros recursos, tales son los casos de la empresa Manage Engine con su producto Applications Manager; la empresa Hyperic con Hyperic HQ; Postgres Plus HQ Monitors de la EnterpriseDB; la empresa Pentaho con sus soluciones Enterprise y Community; Nagios con su nuevo producto Nagios XI y otros que han sido incorporados a empresas como la propia Hyperic que provee mecanismos de integración entre HypericHQ y Nagios, además del caso específico del centro DATEC de la UCI, así mismo en esta universidad se utiliza también Munin, para realizar la monitorización a servidores GESPRO[5] entre otros ejemplos. La mayoría de estas empresas desarrollan sistemas para la monitorización de servidores en sentido general, sin embargo las herramientas como Hyperic HQ, Postgres Plus HQ Monitors y la Applications Manager están dirigidas específicamente a los servidores de PostgreSQL.

1.4 Componentes para la generación de reportes, paneles de Control (DashBoard). Algunas experiencias de implementación y uso

Los procesos generales de toma de decisiones en cualquier empresa se nutren de informaciones estructuradas a partir de la conjunción de datos obtenidos de alguna vía oficial o extraoficial. De esta manera un primer paso para lograr garantía de un adecuado desenvolvimiento de estos procesos en cualquier contexto, es la correcta presentación de los datos en algún soporte que

[5] Paquete para la Gestión de Proyectos desarrollado por la Universidad de las Ciencias Informáticas UCI http://portal.dt.prod.uci.cu/

haga posible la estructuración de información valiosa y con sentido para los involucrados e interesados de la empresa.

El análisis anterior es perfectamente aplicable al caso de la monitorización a servidores de bases de datos. Aquellos sistemas informáticos encargados de realizar esta operación, deberán garantizar precisamente una adecuada presentación de los datos y la información. Es aquí donde intervienen los componentes especializados en esta tarea.

La generación de reportes, específicos o genéricos, es una actividad orientada esencialmente a los usuarios de los sistemas y que tributa directamente al proceso de toma de decisión en el correspondiente ámbito. Las herramientas o componentes informáticos que la realizan están dotados de otros requerimientos, además de la propia generación de reportes, que complementan las funcionalidades del sistema (ver epígrafe 2.2), como son la exportación de los datos en varios formatos, la capacidad de impresión de reportes, ampliación detallada de la información, entre otros.

Estos componentes adquieren un mayor significado cuando los reportes son emitidos en la evaluación de indicadores o métricas en tiempo real; como es el caso del componente del presente trabajo. En la mayoría de los casos esta tipología de componente se integra en otros sistemas informáticos haciendo posible una considerable disminución en el tiempo de desarrollo de estos. Esta capacidad de integración responde a más de un requerimiento particular, a una tendencia en el uso de componentes especializados. En los últimos años se ha visto un aumento en el uso de componentes comerciales en prácticas de reutilización de software. Concretamente, estos componentes comerciales comúnmente conocidos con el nombre de COTS (Commercial Off The Shelf) están siendo considerados con mayor frecuencia para la construcción de sistemas complejos, distribuidos y abiertos.

En este caso en particular, el componente de reporte constituye un elemento de un sistema de monitorización a servidores de gestión de PostgreSQL, que responde a la tipología de sistema DashBoard. En sentido general, el objetivo fundamental de estos sistemas es brindar una interfaz de usuario sencilla, fácil de usar, que provea información en tiempo real de indicadores que pueden estar asociados a diferentes ámbitos de aplicación. Algunos ejemplos se identifican en la esfera empresarial, específicamente en la Inteligencia de Negocio; en la Seguridad y Protección Física con los sistemas de vigilancia y en el ámbito Informático con los sistemas de monitorización en tiempo real a servidores, redes y otros sistemas.

Los DashBoard conjugan exquisitez en la presentación visual de la información, eficiencia y transparencia en su funcionamiento, para derivar en un excelente atractivo para una nueva generación de sistemas de reportes. Incorporan disímiles tipos de gráficos que permiten enriquecer y facilitar el análisis de datos para una correcta y pertinente toma de decisiones. Permiten además trabajar con varios orígenes de datos, y gracias a mecanismos de consolidación implementados, posibilitan la presentación y análisis de la información en una interfaz gráfica orientada a la Web o a un entorno de escritorio.

Algunos ejemplos que utilizan estos fundamentos y que vienen a redundar en este epígrafe sobre lo que ya se ha explicado, son la empresa EnterpriseDB, la empresa Hyperic y la Pentaho Bussines Inteligent; las que desarrollan sistemas de monitorización utilizando DashBoard, además de que en la UCI la Dirección Tecnológica utiliza diferentes Dashboard para mostrar la información referente a la monitorización de los servidores GESPRO desplegados en la institución.

Motivados por la concepción arquitectónica, requerimientos funcionales y no funcionales, licenciamiento, fundamento científico-técnico, entre otros aspectos, el proyecto Personalización de PostgreSQL de DATEC (Documento de Arquitectura del Servidor de Gestión) estableció la definición y desarrollo del DashBoard a partir de las experiencias de la empresa Hyperic; en este sentido la herramienta que presenta el trabajo conjuga algunas de las prácticas de arquitectura y funcionamiento de esta empresa y al mismo tiempo incorpora contribuciones particulares de la solución propuesta.

Postgre Plus HQ Monitor: Es una herramienta privada desarrollada por la EnterpriseDB, la cual realiza un control y supervisión de la disponibilidad, el rendimiento, la utilización y la infraestructura de las bases de datos.

Applications Manager: Es un software de administración que ayuda a las empresas a asegurar una alta disponibilidad en el desempeño de sus aplicaciones de negocio. El mismo proporciona un sistema de monitorización a servidores de aplicaciones, bases de datos y servicios web. Applications Manager establece una conexión remota a cada uno de los servidores que monitoriza lo que imposibilitaría monitorizar todo un lote de servidores de manera simultánea.

Nagios: Es un sistema de monitorización que proporciona una gran versatilidad para consultar prácticamente cualquier parámetro de interés de un sistema, ya sea hardware o servicios. El

mismo genera alertas, que pueden ser recibidas por los responsables correspondientes mediante correo electrónico y mensajes SMS. Visualiza el estado de la red en tiempo real a través de interfaz web, con la posibilidad de generar informes de comportamiento de los sistemas monitorizados, o sea, el sistema esta orientado a la monitorización de servicios de red y no a los servidores de base de datos, presenta una interfaz con bajo nivel de facilidad de uso lo que dificulta el rápido aprendizaje de los usuarios implicados en su uso.

1.4.1 Hyperic HQ: Arquitectura y funcionamiento general

Hyperic HQ es una plataforma de monitorización de aplicaciones y recursos de red basada en Java, que implementa portlets, en su interfaz web que le permiten al usuario adaptar el sistema a sus necesidades. A partir de una revisión general de su arquitectura, se identifican dos componentes fundamentales: agentes (HQ Agent) y el servidor (HQ Server); los que mantienen una estrecha comunicación a partir del intercambio de información asociada a parámetros de localización (IP, puertos, dominios), valores de métricas para la monitorización, entre otros aspectos.

Entre sus funcionalidades se encuentran la detección automática de recursos a partir de los Agentes HQ, los que recolectan información como arquitectura de la plataforma, datos sobre la memoria RAM, velocidad de la CPU, dirección IP y nombre de dominio. Esta funcionalidad puede ser extendida a diferentes plataformas o sistemas operativos, mediante el desarrollo de plugin a partir de los que han sido proveídos por la comunidad. Otras funcionalidades son el inventariado de software y recursos; la monitorización a partir de métricas; control y administración remota de recursos; notificación y alertas; visualización, seguimiento y análisis de rendimientos y disponibilidad, entre otras (15).

De acuerdo a la arquitectura mostrada en la Figura 2, se puede percibir la existencia de un portal web (DashBoard) que constituye la página principal de la interfaz del sistema HypericHQ. Éste está conformado por diversos portlets que permiten proveer información, entre otros aspectos, acerca del estado de alertas, recursos, reportes de métricas, control de acciones y disponibilidad. El DashBoard del sistema puede ser personalizado a partir de cuatro acciones básicas: adición y eliminación de portlets; cambio del diseño de la página a partir del cambio de localización de los portlets y la modificación del comportamiento de los portlets según los requerimientos del usuario (17).

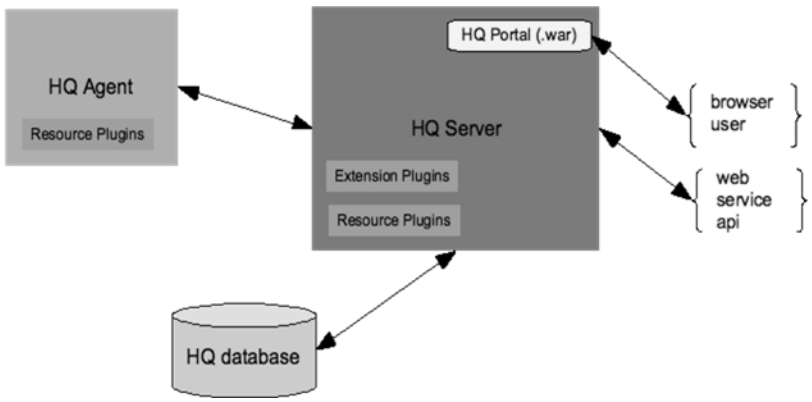

Figura 1: Arquitectura de Hyperic para una mejor concepción del sistema (16).

Hyperic provee dos versiones de sus productos, una versión de código abierto licenciada bajo los términos de GNU GPLv2, y otra distribuida libremente como versión de prueba con una licencia comercial, la que realiza todas las funcionalidades de la primera, además de la automatización avanzada y funciones de control para la gestión de aplicaciones web a gran escala, aunque la versión de prueba se limita solamente a la administración de cincuenta plataformas (16).

Una de las funcionalidades más útiles que permite realizar el DashBoard de Hyperic como contribución al análisis posterior de los datos, es la de exportar los reportes generados por el sistema a los formatos PDF, HTML, Excel y CSV (18).

Como parte de su estrategia de integración con otros sistemas, HypericHQ brinda una API (Application Programming Interface) para servicios web que permite acceder a plataforma, servidores, servicios, grupos, usuarios, entre otros, bien sea mediante una línea de comandos que provee la propia interfaz del sistema y que la mayoría de las veces retorna un objeto serializado en un fichero XML que puede ser editado y aplicado como actualización a la base de datos HQ; o mediante la versión de la API para el lenguaje Java (19). En sentido general, la interfaz de usuario del sistema HypericHQ está dividida en cuatro secciones generales:

1. IU-DashBoard (HQ Dashboard)
2. IU-Recursos
3. IU-Análisis

4. IU-Administración

La primera, como se había mencionado anteriormente, es la página principal del sistema y contiene toda la información asociada a reportes, alertas, métricas. Las interfaces de recursos proveen un conjunto de funcionalidades para la búsqueda y filtrado de recursos (servidores, servicios, aplicaciones, plataformas, clúster); navegación entre gráficos; creación de grupos, aplicaciones, plataformas; selección y adición de plataformas, servidores, servicios a grupos nuevos o existentes; edición y eliminación de recursos, entre otras. Las interfaces de análisis incluyen formularios para centros de alertas, eventos, operaciones y reportes. En tanto las interfaces de administración se ocupan de la gestión y administración de roles y usuarios, proveer mecanismos de autenticación y autorización, configuración del servidor HQ, administración de plugins, entre otras funcionalidades a las que solo podrán acceder usuarios con credenciales de administrador (20).

Esta revisión de Hyperic HQ posibilitó identificar elementos que constituyen buenas prácticas asociadas específicamente con la definición de requerimientos funcionales y no funcionales de la solución informática propuesta; estructuración y formatos de presentación de la información referente a los reportes de monitorización; elementos arquitectónicos particulares del sistema de reporte.

1.4.2 Procedimientos de monitorización en servidores PostgreSQL

La monitorización a servidores PostgreSQL es una tarea de gran envergadura para los administradores de base de datos, lo que se debe sobre todo, a la cantidad de prestaciones que ofrece este SGDB.

En sentido general, la monitorización a servidores PostgreSQL puede dividirse en dos grandes grupos:

1. Comportamiento del servidor: asociado a rendimiento, actividad del micro, consumo de recursos, tiempo de respuesta, entre otros aspectos.
2. Actividad del motor de PostgreSQL: referente a índices, tablas, investigación de consultas, entre otros.

Lo cierto es que muchas de las maneras para monitorizar servidores de PostgreSQL mediante el propio gestor de bases de datos son a través de comandos e instrucciones, lo que demanda del usuario conocimientos técnicos, limitando la actividad de monitorización generalmente solo a

administradores y técnicos especialistas. Procedimientos tan triviales como verificar si el servidor está en ejecución o no, podrían ser de mucha complejidad para usuarios estándares. Incluso, no solo para ejecutar la acción en cuestión, sino hasta para interpretar la salida o resultados de los comandos que permiten conocer sobre esta información.

Quizás algunas instrucciones y comandos sean fáciles de realizar y den como resultado una salida en cierta medida amigable para cualquier usuario. Evidentemente el término amigable introduce un margen de relatividad a este análisis, pero la idea es que pudieran existir mecanismos que permitan al usuario contar con un reporte lo más entendible posible, como puede ser el caso de los comandos ***free*** y ***vmstat*** para el análisis del uso de la RAM y la SWAP[6] en sistemas Linux, cuando el servidor PostgreSQL requiere de más recursos de la RAM que el sistema operativo le puede otorgar y comienza a utilizar el archivo de intercambio. El resultado de estos comandos es un reporte como el de la Figura 3.

```
             total       used       free     shared    buffers     cached
Mem:        605000     296120     308880          0        516     201072
-/+ buffers/cache:      94532     510468
Swap:       979956          0     979956
procs -----------memory---------- ---swap-- -----io---- -system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa
 0  0      0 308880    516 201072    0    0    34     2  263   73  1 45 54  0
```

Figura 2: Ejemplo de salida de los comandos free y vmstat para el análisis del uso de la RAM y la SWAP.

Una revisión general de los mecanismos, comandos, herramientas e instrucciones que provee PostgreSQL y los sistemas operativos para realizar la monitorización a servidores de bases de datos PostgreSQL, bien sea para analizar el comportamiento del servidor o la actividad del motor, indican en todos los casos el requisito de dominar por parte de los usuarios que lo realizan, detalles técnicos sobre la sintaxis de las instrucciones y comandos, así como significados de cada uno de los elementos (siglas, indicadores, identificadores) que se muestran en las salidas de cada instrucción o comando.

[6] Partición de intercambio en una zona del disco (un fichero o una partición), que se usa para guardar las imágenes computacionales de los procesos que no han de mantenerse en memoria física.

1.5 Sistemas en tiempo real

Los sistemas en tiempo real (STR), como el resto de sistemas informáticos, tienen una importancia creciente en nuestra sociedad. Una característica primordial de estos sistemas es la obligación de completar sus actividades en determinados plazos de tiempo, de otro modo el sistema controlado no funcionará correctamente. Los STR son aquellos sistemas digitales que interactúan activamente con un entorno con dinámica conocida en relación con sus entradas, salidas y restricciones temporales, para darle un correcto funcionamiento de acuerdo con los conceptos de predictibilidad y estabilidad.

Algunos autores especializados en el tema han dado varias definiciones sobre lo que es un sistema en tiempo real, Donald Gillies plantea que: "*Un sistema de tiempo real es aquel en el que para que las operaciones computacionales sean correctas no solo es necesario que la lógica e implementación de los programas computacionales sea correcto, sino también el tiempo en el que dicha operación entregó su resultado. Si las restricciones de tiempo no son respetadas el sistema se dice que ha fallado"* (21).

1.6 Entorno tecnológico para el desarrollo del componente

Para asumir el desarrollo del componente fueron identificadas un conjunto de herramientas, donde el fundamento para el uso de cada una de ellas puede encontrarse en el Documento de Arquitectura del Software. Algunas de estas herramientas son:

1.6.1 Herramientas Horizontales

- **Sistema Operativo:** Ubuntu 10.04. Distribución GNU/Linux que ofrece un sistema operativo predominantemente enfocado a ordenadores de escritorio aunque también proporciona soporte para servidores. La versión estable más reciente de Ubuntu es la 10.04, nombre clave "Lucid Lynx" liberada el 29 de abril del 2010 (22).
- **Servidor Control de Versiones:** Subversión 1.2.1.5297. Sistema para el control de versiones usado para mantener ajustes actuales e históricos y los cambios de archivos. Unas de sus principales ventajas es que debido a que el trabajo es versionado, no existe la posibilidad de que la calidad del mismo empeore, ya que si se produce algún cambio incorrecto de los datos, sólo hace falta deshacerlo (23).
- **Gestión de Proyecto:** Redmine (GESPRO v1.0). Herramienta de gestión y planificación de proyectos con interface web orientado a la coordinación de tareas. Se distribuye bajo la licencia GPL y es de código abierto. Algunas de las características fundamentales de

Redmine son el soporte para múltiples proyectos, la flexibilidad basada en funciones de control de acceso, soporte para múltiples bases de datos, entre otras (24)**.**

- **Gestión Documental:** Alfresco (GESPRO v1.0). Plataforma de contenidos para la gestión documental que captura, comparte y retiene contenido, permitiendo a los usuarios versionar, buscar y crear de forma sencilla sus propias aplicaciones (25).
- **Metodología de desarrollo:** eXtreme Programming. Es una de las metodologías de desarrollo de software más exitosas en la actualidad utilizada especialmente para proyectos de corto plazo y equipo de desarrollo pequeño. Esta es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo (26).

1.6.2 Herramientas Verticales

- **Herramienta de Modelado:** Visual Paradigm for UML versión 3.4. Herramienta CASE[7] orientada al diseño UML[8]. Soporta el ciclo de vida completo del desarrollo de software (27).
- **Lenguaje de Programación:** Python v2.6. Lenguaje de alto nivel orientado a objetos e interpretado, creado por Guido van Rossum a principios de los años 90. Posee una sintaxis simple y sencilla; de tipado dinámico, con un gestor de memoria, entre otros aspectos (28).
- **Ambiente de Desarrollo Integrado:** Aptana Studio v3.0**.** Entorno que posee grandes capacidades para constituir aplicaciones en Ruby & Rails, PHP, Python, HTML, CSS, y javaScript.
- **Framework base:** Django v1.2 es un framework para el desarrollo de aplicaciones Web, de código abierto, basado en el lenguaje de programación Python y que sigue el patrón de diseño Modelo Vista Controlador (MVC) (29).
- **Framework GUI:** jQuery v1.4.2**.** Es una biblioteca o framework de Javascript**,** que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM,

[7] Ingeniería de Software Asistida por Computación (CASE).Se define como la aplicación de métodos ytécnicas a través de las cuales se hacen útiles a las personas comprender las capacidades de las computadoras, por medio de programas, de procedimientos y su respectiva documentación.
[8] *Unified Modeling Language*. Es un lenguaje de modelado que le ayuda a especificar, visualizar y documentar modelos de sistemas de software, incluida su estructura y diseño, de manera que cumpla con todos estos requisitos.

manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas web (30).

- **Servidor de aplicación (web):** Nginx**.** Es un servidor HTTP de alto rendimiento, y un servidor proxy para IMAP/POP3/SMTP. Fue desarrollado por Igor Sysoev para *Rambler.ru*, el segundo sitio web más visitado de Rusia, donde ha estado funcionando en producción más de dos años y medio. Igor ha lanzado el código fuente bajo una licencia estilo BSD. Aunque aún se encuentra en una etapa beta, Nginx es conocido por su estabilidad, gran conjunto de características, configuración simple, y bajo consumo de recursos (31).
- **Diseño Gráfico:** Gimp. Es un programa de distribución libre para tareas como retoque fotográfico, composición y creación de imagen. Es expansible y extensible. Está diseñado para ser ampliado mediante plug-ins y extensiones. La interfaz de scripting avanzado permite todo, desde las tareas más simples a los complejos procedimientos de manipulación de imágenes para hacer fácilmente secuencias de comandos (32).
- **Estándar de interoperabilidad:** JSON (Javascript Object Notation) es un formato de intercambio de datos ligero. Es fácil para que las máquinas puedan analizarlo y generarlo y es un formato de texto que es completamente independiente del lenguaje. Estas propiedades hacen JSON un lenguaje ideal para el intercambio de datos.
- **Base de datos:** MongoDB v1.4.4. Es una base de datos documental de gran alcance, sin esquema, escalable y de alto rendimiento, además de que es del tipo NoSQL orientada a documentos (JSON). Combina la capacidad de escala con muchas de las características más útiles de las bases de datos relacionales, tales como índices secundarios, rangos, y la clasificación. MongoDB es una gran herramienta para el seguimiento de las métricas en tiempo real por las siguientes razones (33):

    - Las operaciones de Upsert permite enviar un único mensaje para crear un documento de seguimiento, sobre los nuevos incrementos que han tenido los contadores en un documento existente.

    - El Upsert enviado no espera una respuesta, sino que este “dispara” y “olvida”. Esto permite evitar el bloqueo en cada actualización de análisis que se realice. No

necesita esperar y ver si la operación tiene éxito, ya que en caso de un error en el análisis el código, este no es reportado a los usuarios del sistema.

- Se puede utilizar diferentes actualizaciones para incrementar un contador sin tener que hacer una consulta por separado y operación de actualización. También elimina los problemas de contención en caso de múltiples cambios sucedan simultáneamente.

- El rendimiento de MongoDB al actualizar es muy buena, así que hacer una o más actualizaciones por solicitud de análisis es razonable.

Conclusiones parciales

La revisión de la bibliografía y fuentes consultadas permitió identificar soluciones informáticas orientadas a la monitorización de servidores de bases de datos PostgreSQL, las que han contribuido a la definición de los requerimientos funcionales y no funcionales, así como la arquitectura del componente de reportes para el servidor de gestión. Independientemente de que la mayoría de los sistemas de monitorización analizados son distribuidos bajo licencias de código abierto, se partió del principio de soberanía tecnológica para abordar un desarrollo de un nuevo componente que tome las experiencias de empresas multinacionales pero que responda a las necesidades e intereses de las empresas cubanas. En este sentido se considera poco conveniente adoptar las herramientas de monitorización consultadas dado que el soporte a las mismas implicaría gastos económicos al país, lo cual constituye otro elemento, en tanto se correría el riesgo de incurrir en litigios legales futuros en los que se le negase al país el servicio de soporte al sistema.

CAPÍTULO 2

INTRODUCCIÓN

El objetivo fundamental de este capítulo es analizar los aspectos más relevantes de funcionamiento y arquitectura del componente de reportes desarrollado. Primeramente se describe el contexto en el que se enmarca el componente, las principales funcionalidades con que cuenta el sistema además de sus propiedades y cualidades. Luego se describe la arquitectura, los principales patrones utilizados, las diferentes clases y las relaciones que existen entre ellas. Se modela además la base de datos en la cual el componente almacena la información útil para su adecuado funcionamiento y finalmente se presenta un modelo físico que establece la manera en que se despliegan los componentes en la infraestructura tecnológica.

2.1 Descripción del Negocio

Con el desarrollo de un servidor de gestión de base de datos PostgreSQL en el centro DATEC, surge la necesidad de crear una interfaz gráfica altamente amigable que posibilite un alto grado de efectividad, facilidad de uso y eficiencia en el proceso de monitorización.

Naire es el nombre de código del proyecto, que significa domador de Elefante. Constituye una solución para la captura en tiempo real del rendimiento de los sistemas de gestión de base de datos, presentación de vistas de estadísticas de servidores PostgreSQL, generación de reportes automáticos, además de proveer sugerencias para la configuración y administración del servidor en producción.

Naire está basado en dos componentes principales:

- **Servidor Naire:** Provee el sistema central para la visualización de los reportes y monitorización de las estadísticas de rendimiento y funcionamiento del servidor, además de la planificación de las tareas.
- **Agentes de Naire:** Se encarga principalmente de realizar el cálculo de las diferentes métricas y planificar las tareas de las bases de datos en los diferentes hosts en que estará desplegado.

La generación de reportes a partir de la monitorización de indicadores de rendimiento y almacenamiento a servidores PostgreSQL la desarrollan usuarios avanzados o administradores de servidores, con el objetivo, principalmente, de conducir la toma de decisiones sobre el funcionamiento de sistemas de cómputo y/o equipamientos tecnológicos requeridos, precisamente, para garantizar un adecuado funcionamiento de dichos sistemas.

El hecho de construir este servidor de gestión trae consigo el desarrollo de un conjunto de subsistemas para complementar el funcionamiento íntegro de la aplicación; entre la que se encuentra una herramienta de reportes para la presentación de la información de las métricas de monitorización a servidores PostgreSQL, el que constituye uno de los principales elementos para la creación de un sistema de monitorización DashBoard.

El principal encargado de suministrar los datos de las métricas a monitorizar en los diferentes servidores, es el componente de monitorización, que es el trabajador principal del sistema. Este se encarga de enviar la información a la base de datos con el fin de que el componente de reportes pueda acceder a la recuperación de éstos y generar así los reportes de las métricas, mostrando finalmente al usuario la información en un formato preestablecido. La Figura 4 muestra una representación gráfica del flujo de funcionamiento general de dicho sistema.

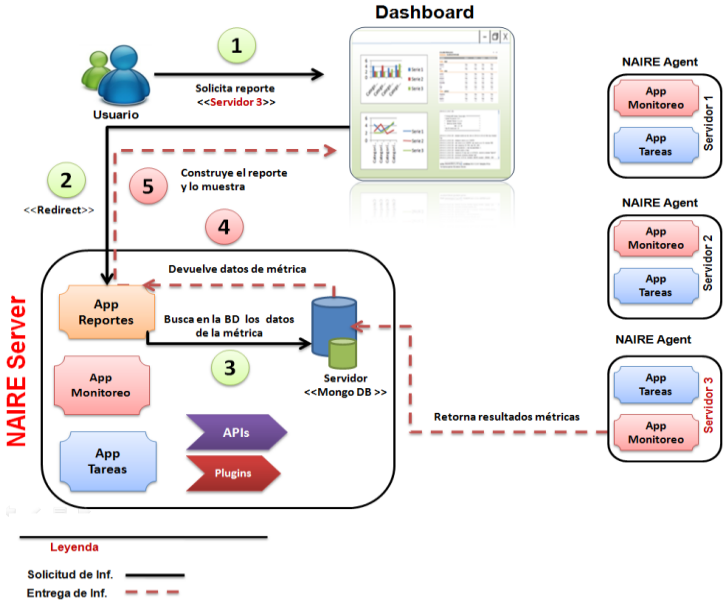

Figura 3: Flujo de datos sobre la solicitud de un reporte presentada por un usuario en el sistema.

2.2 Requerimientos del sistema

Según la IEEE 610.12-1990 (34) un requerimiento es: *"(…) Una condición o capacidad requerida por un usuario para resolver un problema o alcanzar un objetivo (…)"*, o *"(…) Una condición o capacidad que debe ser poseída por un sistema o componente del sistema para satisfacer un contrato, estándar, especificación, u otro documento formalmente impuesto (…)".* Estos pueden clasificarse en (35):

- **Requisitos funcionales:** no son más que las capacidades o condiciones que el sistema debe cumplir. Estos requisitos funcionales se mantienen invariables sin importar con qué propiedades o cualidades se relacionen, además deben especificar las acciones que un sistema debe ser capaz de realizar sin tomar en cuenta las limitaciones físicas.

  A continuación se muestran algunos de los requerimientos funcionales del sistema más significativos:

Ítem *	Descripción
1.	**Buscar servidores a Monitorizar.**
1.1.	Listar servidores a Monitorizar.
2.	**Buscar Bases de Datos a monitorizar.**
2.1.	Listar Bases de Datos a monitorizar.
3.	**Generar alertas del sistema.**
3.1.	Mostrar aletas del sistema.
3.2.	Actualizar estado de alertas.
3.3.	Eliminar alertas del sistema.
4.	**Adicionar los parámetros de configuración del servidor SMTP.**
4.1.	Eliminar parámetros existentes.
4.2.	Verificar conexión con el servidor SMTP.
5.	**Configurar parámetros de alertas del sistema.**
5.1.	Adicionar alertas del sistema.
5.2.	Eliminar alertas del sistema.
5.3.	Modificar alertas del sistema.
6.	**Notificar Alertas del Sistema por Correo Electrónico.**
6.1.	Enviar alerta.
6.2.	Actualizar estado de la alerta (enviado o no enviado).
6.3.	Buscar Administradores de alertas.
7.	**Generar reporte de métricas.**
7.1.	Commits.
7.2.	Rollbacks.
7.3.	Uso estado del disco duro.
7.4.	Estado de espacio libre de memoria.
7.5.	Procesos activos.
7.6.	Filas insertadas.

7.7.	Filas eliminadas.
7.8.	Filas Actualizadas.
7.9.	Filas Retornadas.
7.10.	Filas leídas por escaneo secuencial.
7.11.	Filas muertas.
7.12.	Filas vivas.
7.13.	Escaneo secuencial.
7.14.	Índices escaneados.
8.	**Generar histórico de valores de una métrica en un servidor.**
8.1.	Buscar histórico de servidor.
8.2.	Buscar métrica del servidor.
8.3.	Mostrar histórico de servidor.
8.4.	Eliminar reporte generado anteriormente.
9.	**Generar reporte de métricas por Bases de Datos.**
9.1.	Filas insertadas.
9.2.	Filas eliminadas.
9.3.	Filas Actualizadas.
9.4.	Filas Retornadas.

- **Requisitos no funcionales:** son las propiedades o cualidades que el producto debe tener. Estos requisitos solo describen los atributos con que debe contar el sistema. Debe pensarse en estas propiedades como las características que hacen que el producto sea atractivo, fácil de usar, rápido y confiable.

Existen múltiples categorías para clasificar a los requerimientos no funcionales, algunos ejemplos de estos requisitos son los de Software, de Hardware, de Seguridad, de Fiabilidad, de Soporte, de Funcionalidad, entre otros (34). A partir de estos conceptos, se definieron y clasificaron las principales características con que cuenta el sistema:

Requerimientos de Software: El sistema será desarrollado sobre plataforma Web, por lo que para una mejor accesibilidad al mismo deberá ser compatible con los navegadores Mozilla Firefox, Internet Explorer, Google Chrome, entre otros de los navegadores más usados.

Requerimientos de Hardware para el Servidor: El equipo de cómputo deberá contar con una memoria RAM de 512 MB como mínimo (1 GB o más recomendado), un microprocesador Pentium superior a 3.0 GHz y deberá tener una capacidad de almacenamiento mínimo de 80 GB de espacio en disco.

Requerimientos de Hardware para el Cliente: El equipo de cómputo deberá contar con una conexión con el servidor a través de un módem o tarjeta de red, un Procesador Pentium III o

superior y contar con la instalación completa de Mozilla Firefox, Google Chrome o Internet Explorer.

Requerimientos de Seguridad: El sistema tendrá implementado mecanismos de autenticación reforzada de usuarios de acuerdo al rol que desempeñe, además de que tendrá implementado un sistema interno de registro y control de trazas.

Requerimientos de Rendimiento: la velocidad de procesamiento y la capacidad de respuesta de la aplicación web deben ser rápidas, en dependencia del tipo de conexión que esté establecida con el servidor y las características de la red. Las páginas deberán ser lo más ligeras posibles, con pocas imágenes de calidad aceptable y el sistema debe requerir un consumo mínimo de recursos.

Requerimientos de Soporte: El sistema deberá ser probado, instalado y configurado por los especialistas que también se ocuparán de su mantenimiento.

Restricciones de Diseño e implementación: Para implementar el sistema se usará el lenguaje Python, como gestor de bases de datos PostgreSQL, y para el desarrollo de la interfaz de usuario se utilizará tecnología AJAX con el objetivo de realizar conexiones asíncronas en demandas.

Requerimientos de Portabilidad: El componente al ser desarrollado sobre plataforma Web, podrá ser utilizado por gran variedad de sistemas operativos como es el caso de Linux, Windows Server 2003 o superior, Solaris 10 o superior, Mac OS X (Intel x86) y la salida producida debe ser soportada por los navegadores más usados.

Requerimientos de Disponibilidad: El sistema deberá permitir al administrador conectarse desde cualquier computadora en una red LAN y el mismo estar disponible los 7 días de la semana las 24 horas al día, los 365 días del año, y recuperarse rápidamente ante cualquier tipo de fallo.

Requerimientos de uso: El sistema debe ser fácil para aprender, para ello contendrá un manual de usuario y una ayuda que instruirá al encargado del sistema, sobre el trabajo en el mismo. La interfaz debe ser sencilla y amigable de manera que potencie la comodidad del usuario para su trabajo además de que las opciones más usadas presentarán vías rápidas y cómodas. Debe ser capaz de tener una fácil navegación por todas las funcionalidades del sistema, desde cualquier página. Además que el sistema debe cumplir con las heurísticas que define Jacob Nielsen:

- Visibilidad del estado del sistema.

- Control y libertad del usuario.
- Consistencia y estándares.
- Prevención de errores.
- Correspondencia entre el sistema y el mundo real.
- Reconocer antes que recordar.
- Flexibilidad y eficiencia de uso.
- Estética y diseño minimalista.
- Ayudar a los usuarios a reconocer, diagnosticar y recuperación de errores.
- Ayuda y documentación.

2.3 Historias de Usuarios

Las historias de usuarios son las técnicas utilizadas para especificar los requisitos del software, básicamente se pueden definir como la descripción del sistema desde el punto de vista del usuario. Estas son descompuestas en diferentes tareas de programación que deben ser asignadas a cada programador para ser implementadas durante una iteración (36). A continuación se mostrarán las historias de usuarios más significativas para el desarrollo del sistema.

TABLA 1: H.U BUSCAR SERVIDORES A MONITORIZAR

	Historia de Usuario
Número: 15	**Nombre de la Historia de Usuario:** Buscar servidor a Monitorizar.
Cantidad de modificaciones a la Historia de Usuario: 2	
Usuario: Administrador	**Iteración asignada:**1
Prioridad en negocio: Muy alta	**Puntos estimados:** 1
Riesgo en desarrollo: Muy alta	**Puntos reales:** 1
Descripción: realiza una búsqueda en las colecciones de la base de datos MongoDB que esté activa,	

para la obtención de los servidores que tienen contenido el agente de monitorización. Obteniendo así el nombre del servidor a monitorizar.

TABLA 2: H.U BUSCAR BASES DE DATOS A MONITORIZAR.

Historia de Usuario	
Número: 16	**Nombre de la Historia de Usuario:** Buscar Bases de Datos a monitorizar
Cantidad de modificaciones a la Historia de Usuario: 1	
Usuario:	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 1
Riesgo en desarrollo: Alta	**Puntos reales:** 1
Descripción: permite obtener una lista de todas las bases de datos que contengan un servidor determinado a la hora de monitorizarlo, con el objetivo de generar reportes por base de datos.	

TABLA 3: H.U GENERAR ALERTAS DEL SISTEMA.

Historia de Usuario	
Número: 19	**Nombre de la Historia de Usuario:** Generar alertas del sistema.
Cantidad de modificaciones a la Historia de Usuario: 2	
Usuario: Administrador	**Iteración asignada:** 2
Prioridad en negocio: Muy Alta	**Puntos estimados:** 0.25
Riesgo en desarrollo: Alta	**Puntos reales:** 0.25
Descripción: permite realizar el registro en el sistema de alertas configuradas por el usuario para medir el comportamiento crítico (el administrador define cual sería el estado crítico de una métrica según su perspectiva.) de las métricas en los servidores para tener un control del estado del mismo, brindará un mensaje para resaltar el error en cuestión.	

Observaciones: las alertas se van a guardar en el sistema de forma automática, para un análisis de las mismas.

TABLA 4: H.U ADICIONAR LOS PARÁMETROS DE CONFIGURACIÓN DEL SERVIDOR SMTP.

Historia de Usuario	
Número: 27	**Nombre de la Historia de Usuario:** Adicionar los parámetros de configuración del servidor SMTP
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Administrador	**Iteración asignada:** 2
Prioridad en negocio: Muy Alta	**Puntos estimados:** 0.25
Riesgo en desarrollo: Muy Alta	**Puntos reales:** 0.25
Descripción: esta funcionalidad brinda un formulario para la configuración del servidor de correo donde se especificará además el administrador que recibirá las alertas por el correo del servidor SMTP adicionado.	

TABLA 5: H.U CONFIGURAR PARÁMETROS DE ALERTAS DEL SISTEMA.

Historia de Usuario	
Número: 17	**Nombre de la Historia de Usuario:** Configurar parámetros de alertas del sistema.
Cantidad de modificaciones a la Historia de Usuario: 1	
Usuario: Administrador	**Iteración asignada:** 2
Prioridad en negocio: Alta	**Puntos estimados:** 0.25
Riesgo en desarrollo: Alta	**Puntos reales:** 0.25
Descripción: dado un formulario se permite entrar los valores para la configuración del sistema de	

alertas, esto se realizará por cada una de las métricas de manera independiente, para poder brindar alertas a conveniencia de los administradores.

TABLA 6: H.U NOTIFICAR ALERTAS DEL SISTEMA POR CORREO ELECTRÓNICO.

Historia de Usuario	
Número: 20	**Nombre de la Historia de Usuario:** Notificar Alertas del Sistema por Correo Electrónico.
Cantidad de modificaciones a la Historia de Usuario: 3	
Usuario: Administrador	**Iteración asignada:** 2
Prioridad en negocio: Muy alta	**Puntos estimados:** 4
Riesgo en desarrollo: Muy alta	**Puntos reales:** 4
Descripción: esta funcionalidad envía a través de correo electrónico información sobre el comportamiento de las métricas, cuando ocurra una alerta en el sistema, éste lo notifica al correo de los administradores del sistema.	

TABLA 7: H.U GENERAR HISTÓRICO DE VALORES DE UNA MÉTRICA EN UN SERVIDOR

Historia de Usuario	
Número: 23	**Nombre de la Historia de Usuario:** Generar histórico de valores de una métrica en un servidor.
Cantidad de modificaciones a la Historia de Usuario: 1	
Usuario: Administrador, usuario avanzado	**Iteración asignada: 2**
Prioridad en negocio: Muy alta	**Puntos estimados:** 4
Riesgo en desarrollo: Muy alta	**Puntos reales:** 4
Descripción: dado un servidor, una métrica y un periodo de tiempo se genera un listado sobre el	

comportamiento de dicha métrica en el tiempo y servidor determinado.
Observaciones: tiene que elegirse los tres parámetros fundamentales sino el sistema mostrara un mensaje señalando el error.

TABLA 8: H.U GENERAR REPORTE DE MÉTRICAS POR BASES DE DATOS.

	Historia de Usuario
Número: 18	**Nombre de la Historia de Usuario:** Generar reporte de métricas por Bases de Datos.
Cantidad de modificaciones a la Historia de Usuario: 1	
Usuario: Administrador, usuario avanzado	**Iteración asignada: 2**
Prioridad en negocio: Muy alta	**Puntos estimados:** 4
Riesgo en desarrollo: Muy alta	**Puntos reales:** 4
Descripción: la generación de reportes de métricas a nivel de bases de datos, es una funcionalidad general que tiene sus particularidades, en cuanto al tipo de información y el formato de presentación de ésta, en cada una de las métricas a generar. Esta funcionalidad la realiza cualquier usuario autenticado en el sistema, y consiste en generar un reporte (gráfica, tabla) sobre el estado de los indicadores de una determinada métrica en cada base de datos.	

TABLA 9: H.U GENERAR REPORTE DE MÉTRICAS POR SERVIDOR.

	Historia de Usuario
Número: 17	**Nombre de la Historia de Usuario:** Generar reporte de métricas por servidor.
Cantidad de modificaciones a la Historia de Usuario: 3	
Usuario: Administrador, usuario avanzado	**Iteración asignada: 2**
Prioridad en negocio: Muy alta	**Puntos estimados:** 4

Riesgo en desarrollo: Muy alta	**Puntos reales:** 4
Descripción: la generación de reportes de métricas es una funcionalidad general que tiene sus particularidades, en cuanto al tipo de información y el formato de presentación de ésta, en cada una de las métricas a generar. Esta funcionalidad la realiza cualquier usuario autenticado en el sistema, y consiste en generar un reporte (gráfica, tabla, informe, etc.) sobre el estado de los indicadores de una determinada métrica.	

2.4 Diseño del sistema

La metodología de desarrollo XP no incorpora los Diagramas de Casos de Uso del Sistema (DCUS) de la metodología Rational Unified Process (RUP) como un artefacto resultante del proceso de diseño de software, en su lugar se utilizan las tarjetas CRC (Contenido, Responsabilidad y Colaboración) y las Historias de Usuarios con el mismo propósito de los casos de uso. Sin embargo el empleo de estos diagramas contribuye considerablemente al mejoramiento de la comunicación entre desarrolladores y clientes (37) y al mismo tiempo proporciona los elementos esenciales para la realización de las pruebas de software (38). En este sentido estos recursos fueron empleados como un complemento de los instrumentos que provee XP para el diseño de sistemas informáticos.

2.4.1 Arquitectura del componente de reportes.

El componente de reportes está estructurado en tres módulos fundamentales: un Módulo de Administración, el que se ocupa de la gestión y administración de usuarios, además de proveer mecanismos de autenticación y autorización, registro de trazas, entre otras funcionalidades a las que solo podrán acceder usuarios con credenciales de administrador, excepto la funcionalidad de autenticación disponible para todo tipo de usuario. El Módulo de Configuración se encarga de la configuración de parámetros para otorgarle flexibilidad al sistema de modo que los usuarios tengan la posibilidad de adaptar, en cierta medida, el sistema a necesidades específicas. Entre otros aspectos, permitirá configurar las alertas del sistema para evaluar los estados críticos de las métricas monitorizadas en cada uno de los servidores, y el modo de autenticación de los usuarios. El Módulo de Reportes se ocupa de la generación, impresión y exportación de los reportes además de proveer un conjunto de instrumentos, entre los que se encuentran gráficas y tablas que brindan información sobre el comportamiento de los diferentes servidores a

monitorizar, como contribución a la toma de decisiones a partir del análisis de las diferentes gráficas y alertas que brinda el sistema (Figura 5).

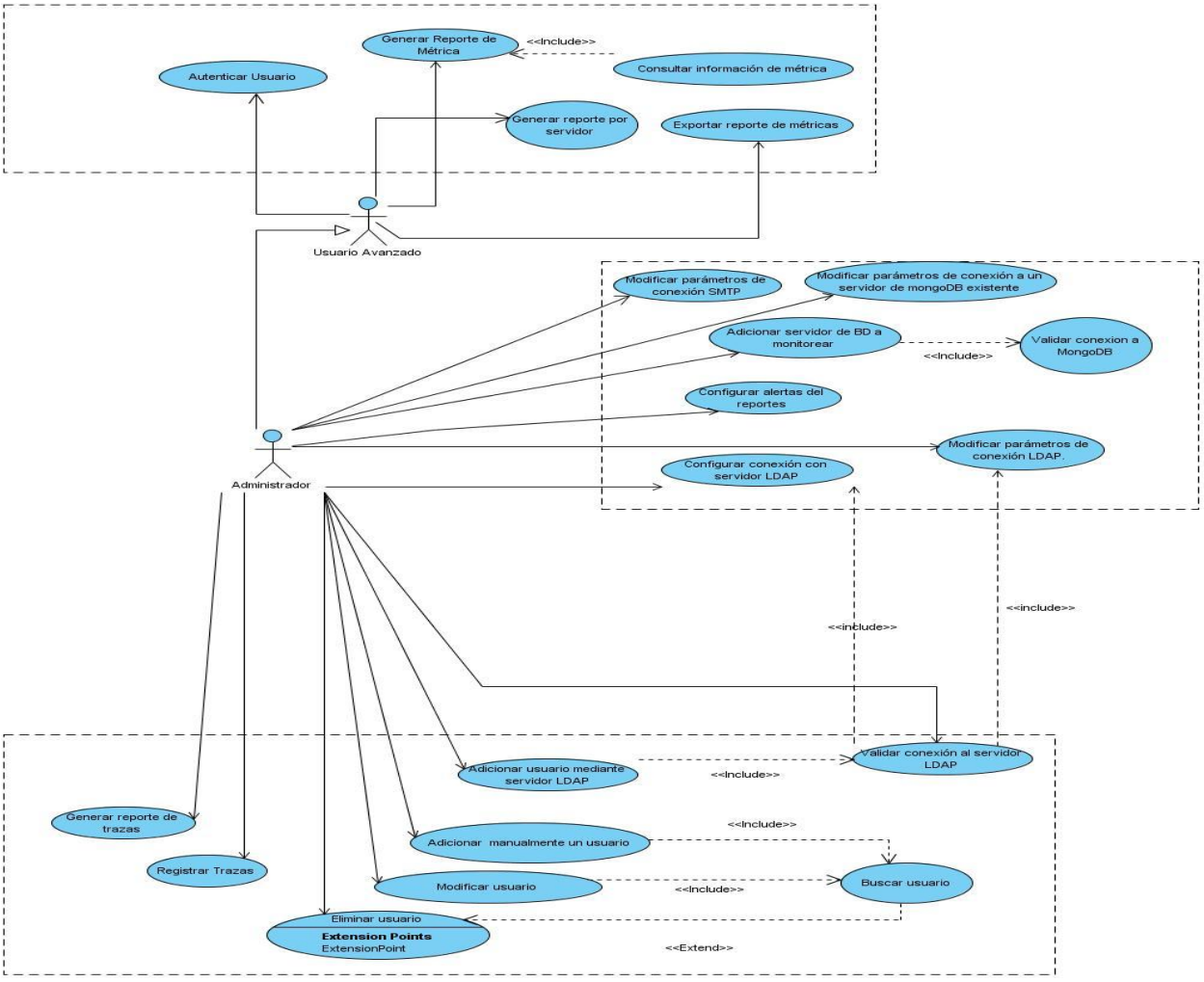

Figura 4: Diagrama UML sobre las principales funcionalidades del sistema.

2.4.2 Comportamiento del Tiempo real en el componente

El tiempo real establecido para el componente de reporte esta dado por el intervalo de tiempo que demora el sistema para consultar la información de los servidores que se monitorizan en la base de datos MongoDB, hasta que la misma es mostrada en el dashboard, este procedimiento debe tener un retardo de 1 segundo para que el sistema sea totalmente factible en cuanto al tiempo real, o sea, debe demorar 0.50 segundos a la hora de consultar la información y 0.50 segundos a la hora de mostrar la misma (Figura 6).

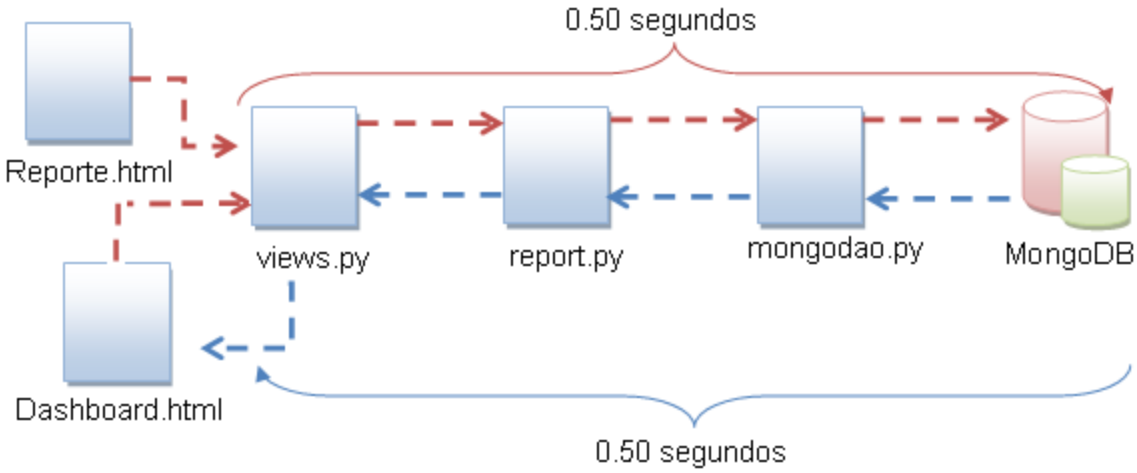

Figura 5: Proceso de consultar información en tiempo real.

2.4.3 Patrón arquitectónico Modelo Vista Controlador

El patrón Modelo Vista Controlador (MVC) describe una forma, muy utilizada en la Web, de organizar el código de una aplicación separando los datos, la interfaz de usuario, y la lógica de control en tres componentes distintos. Se trata de realizar un diseño que desacople la vista del modelo, con la finalidad de mejorar la reusabilidad. De esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos.

Django implementa una variante del patrón MVC denominada Model-Template-View (MTV) (Figura 6). Para Django la vista describe "qué" datos serán presentados y no "cómo" se verán los mismos. Aquí es donde entran en juego las plantillas o *templates*[9], las que se encargan de describir cómo es mostrada la información (39).

[9] Se usarán ambos términos indistintamente.

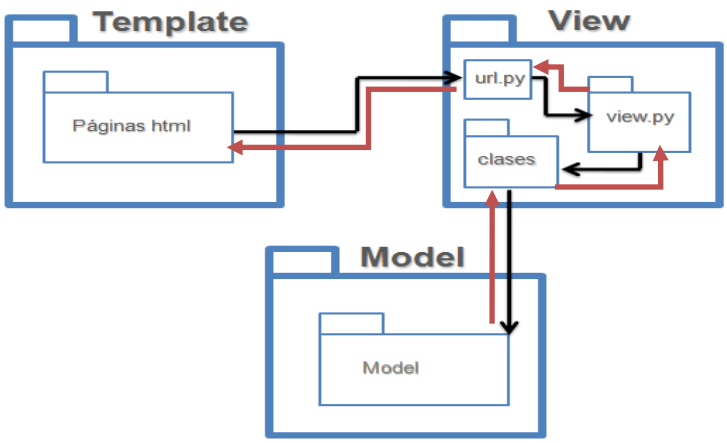

Figura 6: Descripción del patrón arquitectónico Modelo-Vista-Controlador implementado en Django.

La capa de **Modelo** es una de las partes más potentes de Django. Cada uno de los modelos creados se mapean en diferentes tablas en la Base de Datos. Esto permite aislar la Base de Datos del código y olvidarse de las acciones de seleccionar y actualizar. El **Template**, que en el framework MVC sería la vista, es la capa de presentación y se basa en plantillas HTML. Django presenta un *template engine* y un *template loader,* que permiten presentar al usuario diversas páginas HTML usando una base como plantilla. Esto es posible porque en cada una de las plantillas se pueden introducir determinadas etiquetas Django que el *template loader* se encargará de interpretar. La **Vista** es lo que en Django se denomina *views,* las que actúan como controlador; que escritas en puro código Python cada *view* atenderá una petición HTML según el mapeo de la URL.

2.4.4 Patrones de diseño

Los principales diseñadores de software expertos en la materia, van creando un amplio repertorio de principios generales y de expresiones a tener en cuenta para la creación de un software, a ellos se les puede llamar con el nombre de patrones, si se codifican en un formato estructurado que describen el nombre del problema y su solución. Éstos se pueden definir como soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos.

En otras palabras se puede decir que un patrón es una pareja de problema/solución con un nombre, que codifica buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades (40).

Patrones GRASP aplicados

GRASP es un acrónimo que significa General Responsibility Asignment Software Patterns (patrones generales de software para asignar responsabilidades). El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos (39), aunque se considera que más que patrones se puede decir que son una serie de buenas prácticas de aplicación recomendables en el diseño de software.

- **Experto:** En cuanto al diseño de la aplicación y la correcta asignación de responsabilidades, se utilizó este patrón para garantizar que las clases contuvieran la información necesaria para las operaciones que implementan (Figura 7). El uso de este patrón contribuye a un adecuado encapsulamiento, lo que favorece la robustez y fácil mantenimiento del sistema.

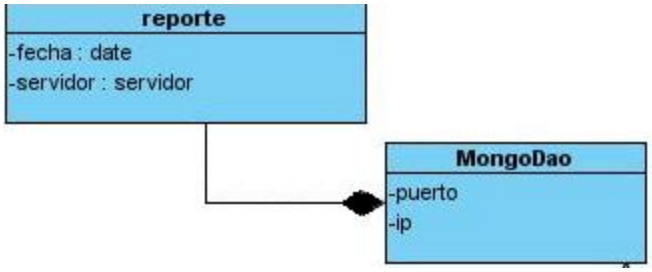

Figura 7: Ejemplo del uso del patrón Experto.

- **Creador:** Para la correcta asignación de responsabilidades orientada a la creación de objetos se utilizó el patrón creador, que sirvió para determinar las clases encargadas de asumir la responsabilidad de la creación de instancias. El uso de este patrón tiene la ventaja de soportar una mayor claridad, encapsulamiento y reusabilidad. (Figura 8).

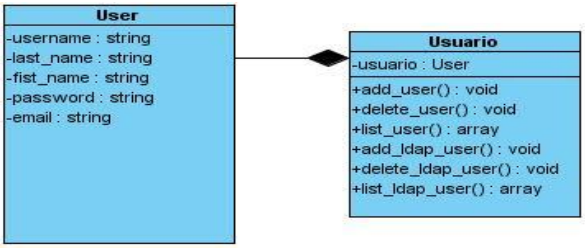

Figura 8: Ejemplo del uso del patrón Creador.

- **Controlador:** Con el fin de una correcta asignación de responsabilidades para el manejo de eventos de un sistema, se utiliza el patrón controlador, con el propósito de utilizar la misma clase controladora con todos los eventos del sistema.
- **Bajo acoplamiento:** Este patrón estimula asignar una responsabilidad de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento[10]. El bajo acoplamiento soporta el diseño de clases más independientes, que reducen el impacto de los cambios y también reutilizables, que acrecientan la oportunidad de una mayor productividad (39).

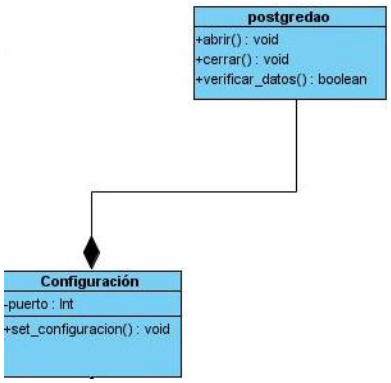

Figura 9: Ejemplo del uso del patrón Bajo Acoplamiento.

- **Alta Cohesión:** Una clase de alta cohesión posee un número relativamente pequeño de responsabilidades, con una importante funcionalidad relacionada y poco trabajo por hacer. Colabora con otros objetos para compartir el esfuerzo si la tarea es grande. Este patrón mejora la calidad y facilidad con que se puede entender el diseño, se genera un bajo acoplamiento y permite fomentar la reutilización.

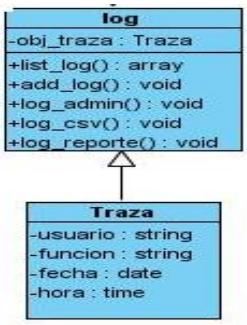

Figura 10: Ejemplo del uso del Patrón Alta Cohesión

[10] Medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas.

Patrones de diseños en Django

Los desarrolladores de Django utilizan comúnmente un conjunto de patrones, con el fin de hacer más simple y más fácil de mantener la codificación y el código. A continuación se muestran algunos de los patrones utilizados en el desarrollo del componente.

- **Projects and apps (Proyectos y Aplicaciones) a nivel de URL:** Este patrón plantea la definición de un archivo *urls.py* responsable del re-direccionamiento de las peticiones HTTP, creado tanto a nivel de proyecto como en cada una de las aplicaciones de éste. Desde el archivo *url.py* del directorio raíz del proyecto se pueden establecer enlaces con el resto de este tipo de archivo en cada una de las aplicaciones, en virtud de un prefijo que describe el inicio de la URL del directorio donde se encuentra el archivo *urls.py* al que se hace referencia (Figura 9).

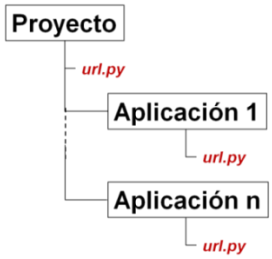

Figura 11: Ejemplo del uso del patrón Projects and apps a nivel de URL.

- **Projects and apps (Proyectos y Aplicaciones) a nivel de plantilla:** Este patrón plantea que debe haber un archivo *base.html* a nivel de proyecto, y un archivo *base.html* en cada uno de los niveles de aplicación. El nivel de aplicación *base.html*, debería ampliar el archivo a nivel de proyecto.
- **Logging:** Este patrón se utiliza con el objetivo de asegurarse que, cuando un objeto sea adicionado, editado o eliminado, haya un registro de cada una de las operaciones realizadas.

2.4.5 Diseño de clases del componente

Para el diseño del componente se emplearon las tarjetas CRC que provee la metodología XP, a partir de las cuales se definen las clases y las relaciones que entre ellas se establecen. Independientemente que un objetivo de estas tarjetas es hacer más entendible las funcionalidades del sistema y servir así de puente de comunicación entre los usuarios y desarrolladores. Las tarjetas CRC no especifican qué tipo de relación se establecen entre las

clases ni permiten comprender el diseño del sistema desde una óptica general y completa. Estos inconvenientes provocan que no se puedan identificar detalles de las clases ni las relaciones, como por ejemplo los atributos, visibilidad de los miembros de clases, generalizaciones y especializaciones, entre otros elementos. Por esta razón es conveniente representar el diseño de las clases mediante un estándar que resuelva estos inconvenientes y para ello UML aparece como una excelente alternativa. El diagrama UML de la Figura 10 muestra las clases utilizadas para el desarrollo del componente y las relaciones que entre ellas se establecen.

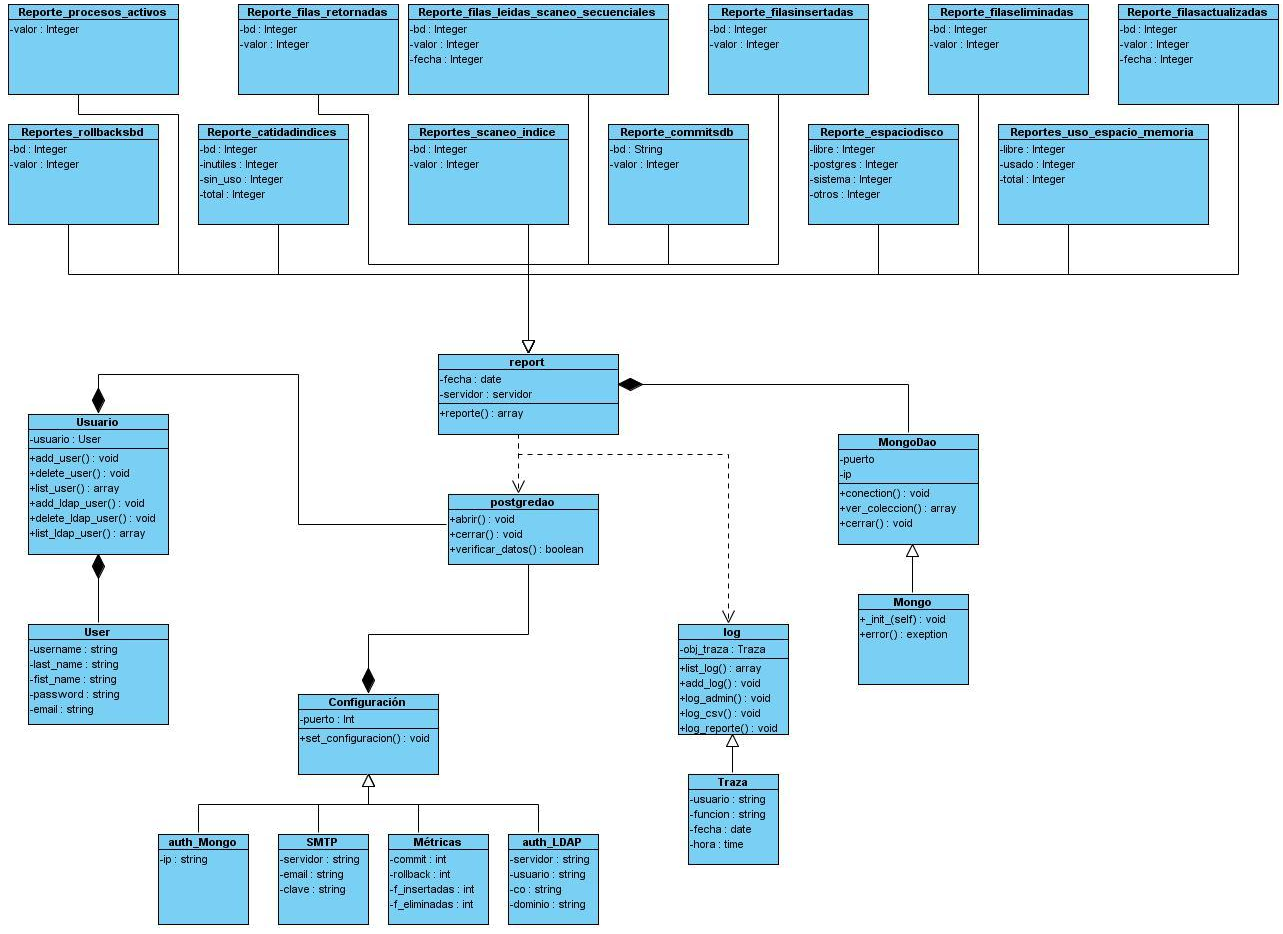

Figura 12: Diagrama con las clases utilizadas en el desarrollo del componente.

2.5 Diseño de la base de datos del componente

La mayoría de los productos informáticos que se desarrollan hoy día en el mundo, requieren de almacenamiento y gestión de grandes volúmenes de datos. Para ello, se hace necesario el uso de estructuras de datos y con esto el empleo de una base de datos para almacenar la información (41).

El componente de reportes consulta la base de datos documental MongoDB para extraer de ésta los datos de las diferentes métricas de rendimiento y funcionamiento. Pero independientemente de eso, el hecho de elaborar el sistema en sí, implica la creación de una nueva base de datos propia para la gestión interna de la información.

Es por ello que se ha utilizado una base de datos PostgreSQL para almacenar los datos referentes a la configuración y administración del sistema (Figura 11), además de todos los

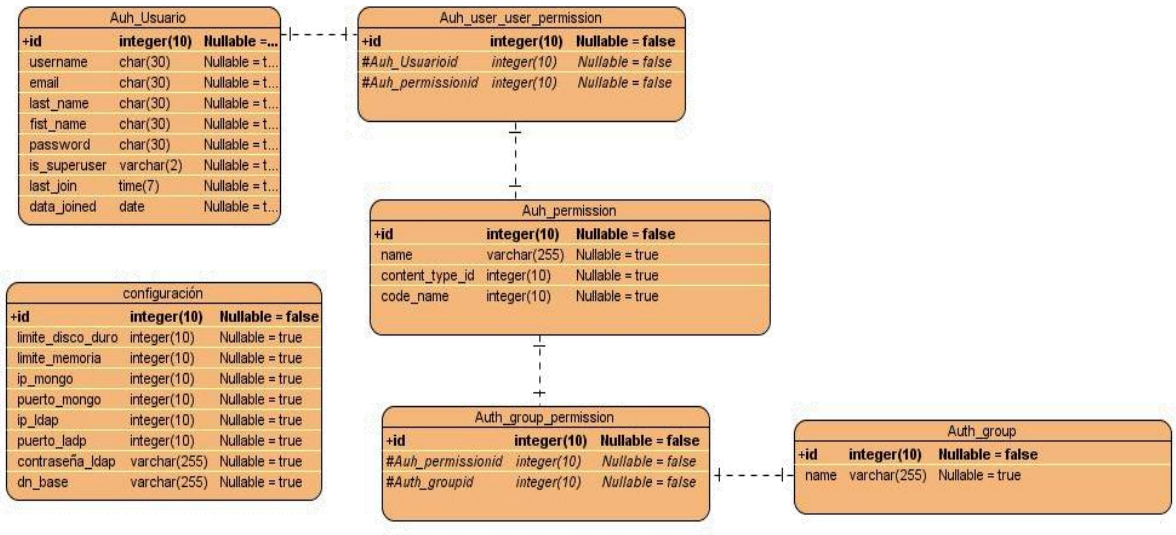

Figura 13: Modelo Entidad Relación sobre la gestión y autenticación de usuarios.

reportes y estados de aleratas que se generan tras el analisis de cada unos de los servidores que se monitorizan (Figura 12).

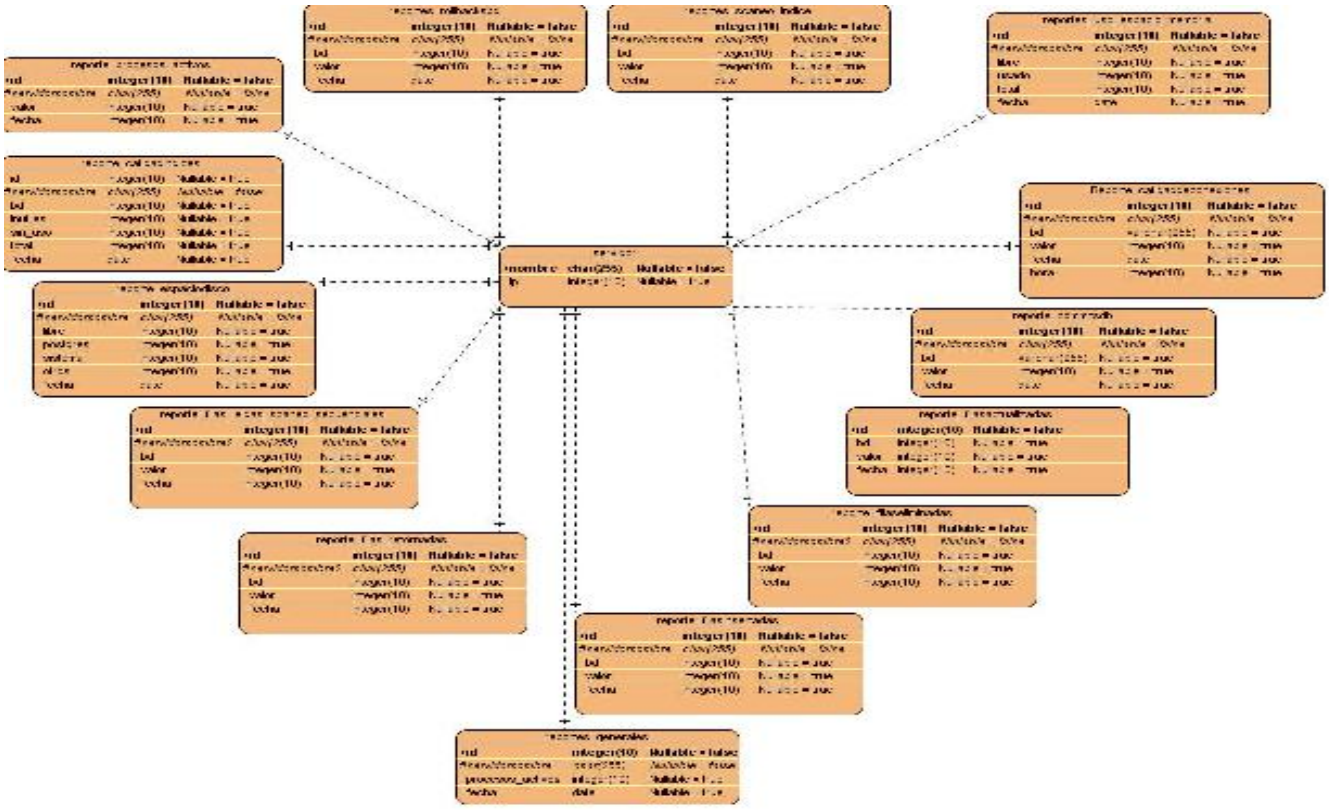

Figura 14: Modelo Entidad Relación sobre la gestión de las métricas.

Conclusiones parciales

A partir de las necesidades existentes en el proyecto PostgreSQL Empresarial se definió las principales funcionalidades a implementar, y luego se precisó la prioridad de cada una de estas funcionalidades y la iteración en que éstas serían implementadas. Se realizó el diseño del sistema. Se identificaron además todos los patrones de diseño e implementación aplicados en el desarrollo del componente como un conjunto de buenas prácticas que son recomendables. Además, se implementó la aplicación capaz de dar los reportes requeridos por los usuarios.

“Validación”

CAPÍTULO 3

INTRODUCCIÓN

Las pruebas de software representan una revisión final de las especificaciones, diseño y codificación de un sistema. En este sentido existen diferentes técnicas y métodos que son utilizadas para este propósito, como es el caso de las pruebas de caja negra, pruebas de caja blanca, pruebas de unidad, de integración, pruebas de arquitectura y aplicaciones, pruebas del sistema, entre otras.

El objetivo general establece el desarrollo de un componente de reportes que contribuya a mejorar la facilidad de uso, efectividad y eficiencia en el proceso de monitorización a servidores de bases de datos PostgreSQL. Para medir el nivel de la facilidad de uso alcanzada se aplicó en el componente de reportes un modelo matemático basado en las heurísticas de Jacob Nielsen, por otra parte se emplearon pruebas de caja negra para evaluar el cumplimiento de los requisitos del sistema. En este capítulo se analizan además los resultados obtenidos por las pruebas aplicadas y por el modelo matemático utilizado, además de una comprobación del cumplimiento del tiempo real en el sistema.

3.1 Pruebas de caja negra

Según Pressman *“(…) La validación del software se consigue mediante una serie de pruebas de caja negra que demuestran la conformidad con los requisitos (…)”.* Precisamente las pruebas de caja negra son uno de los instrumentos aplicados para la evaluación del cumplimiento del objetivo general. Este tipo de prueba se centra específicamente en el cumplimiento de los requisitos funcionales del software sin fijarse en el funcionamiento interno del programa, y con ellas se demuestran que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. Este tipo de prueba permite obtener un conjunto de condiciones de entrada que ejerciten de forma completa todos los requisitos funcionales de un programa (42).

3.1.1 Diseño de los Casos de Pruebas Funcionales

Tabla 10 : AUTENTICAR USUARIO.

Escenario	Descripción	Variable 1	Variable 2	Respuesta del sistema	Flujo central
EC 1.1 Autenticar usuario correctamente	Procedimiento mediante el cual el usuario presenta sus credenciales.	V	V	Se direcciona al usuario a su ambiente de trabajo por defecto, de acuerdo a los permisos que le corresponden.	1. El usuario introduce sus credenciales en el sistema (*usuario, contraseña*). 2. Clic en el botón *Entrar.*
EC 1.2 Autenticar usuario con campos incorrectos	El usuario no puede entrar al sistema.	I	V	El sistema muestra un mensaje de error en las credenciales.	
		V	I		
		I	I		

Tabla 11: VARIABLES DE LA HU AUTENTICAR USUARIO.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
Variable 1	Usuario	Campo de texto	No	Usuario con permiso en el sistema.
Variable 2	Contraseña	Campo de texto	No	Contraseña.

Tabla 12: EXPORTAR REPORTE DE TRAZAS.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Exportar reporte de trazas.	Esta funcionalidad que la realiza el administrador del sistema consiste en exportar las trazas de un usuario	El sistema brinda la posibilidad de exportar el reporte generado en formato PDF.	1. Entrar al módulo de *Administración*. 2. Elegir la pestaña *Trazas*.

	determinado o de todos los usuarios en PDF.		3. Clic en el ícono de PDF.

Tabla 13 : NOTIFICAR AUTENTICACIÓN DE USUARIO MEDIANTE CORREO ELECTRÓNICO.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Notificar autenticación de usuarios mediante correo electrónico.	Esta funcionalidad le envía al usuario registrado un correo, con sus credenciales así como otros datos de su perfil.	El sistema le envía un correo electrónico al usuario registrado con sus credenciales.	1. Entrar a la pestaña de *Administración* 2. Nuevo Usuario. 3. Adicionamos el Usuario.

Tabla 14: NOTIFICAR ALERTAS DEL SISTEMA POR CORREO ELECTRÓNICO.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Notificar Alertas por correo electrónico.	Esta funcionalidad envía a través de correo electrónico información sobre el comportamiento de las métricas.	El sistema le envía un correo electrónico al administrador del sistema.	1. Entrar a la pestaña *Reporte.* 2. Escoger el servidor a Monitorizar.

Tabla 15: GENERAR REPORTE DE MÉTRICAS POR SERVIDOR.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1	Muestra toda la información	El sistema muestra un dashboard para	1. Entrar en la pestaña de

Generar reporte de métricas por servidor.	de un servidor en forma de gráficas y tablas.	monitorizar el comportamiento del servidor.	*Reporte*. 2. Elegir el servidor a monitorizar.

Tabla 16: GENERAR REPORTE DE MÉTRICAS POR BASES DE DATOS.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Generar reporte de métricas por Bases de Datos.	Muestra toda la información de una base de datos de servidor en forma de gráficas y tablas.	El sistema muestra un dashboard para monitorizar el comportamiento de una base de datos de un servidor.	1. Entrar en la pestaña de *Reporte*. 2. Elegir el servidor a monitorizar. 3. Elegir la base de datos a monitorizar.

Tabla 17: BUSCAR SERVIDOR DE BASES DE DATOS A MONITORIZAR.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Buscar servidor de Bases de Datos a monitorizar.	Realiza una búsqueda en las colecciones de la base de datos MongoDB para la obtención de los servidores que se van a monitorizar.	El sistema muestra un listado de todos los servidores listos para monitorizar.	1. Entrar en la pestaña de *Reporte*.

3.1.2 No conformidades detectadas

Tabla 18 : NO CONFORMIDADES DETECTADAS EN LA SEGUNDA ITERACIÓN

Elemento	No conformidad	Aspecto correspondiente	Etapa de detección	Signif	No signif
Página Listar_usuarios.html	No se puede editar un usuario	Cuando se va a editar el usuario	Etapa de aplicación de pruebas a la Historia de Usuario Gestionar Usuario.	X	
Página Listar_usuarios.html	No cancela la eliminación de un usuario.	Cuando se elimina el usuario	Etapa de aplicación de pruebas a la Historia de Usuario Gestionar Usuario.	X	
Página Adicionar_usuario.html	No se puede adicionar usuarios mediante conexión LDAP	Cuando se adiciona un usuario de tipo LDAP	Etapa de aplicación de pruebas a la Historia de Usuario Autenticar Usuario mediante servidor LDAP.	X	
Página Dashboard.html	No se puede cambiar de servidor a monitorizar	Cuando se está monitorizando un servidor	Etapa de aplicación de pruebas a la Historia de Usuario Generar reportes de métricas de un servidor.	X	
Página Generar_reporte.html	No se puede realizar reportes por intervalos.	Cuando se genera un reporte.	Etapa de aplicación de pruebas a la Historia de Usuario Generar reportes.	X	
Página Generar_trazas.html	No se exporta el reporte de trazas en formato PDF.	Cuando se generan las trazas de un usuario.	Etapa de aplicación de pruebas a la Historia de Usuario Generar Trazas.		X

3.2 Análisis de los resultados

Después de haber realizado las pruebas funcionales se obtuvo como resultado 15 no conformidades en la primera iteración (Figura 13), de las cuales todas fueron resueltas, en el análisis de la segunda iteración se detectaron un total de 6 no conformidades (Figura 14) las cuales se resolvieron satisfactoriamente. Se ejecutó un Caso de Prueba Funcional por cada una de las Historias de Usuarios para puntear el nivel de errores dando así mayor validez al producto.

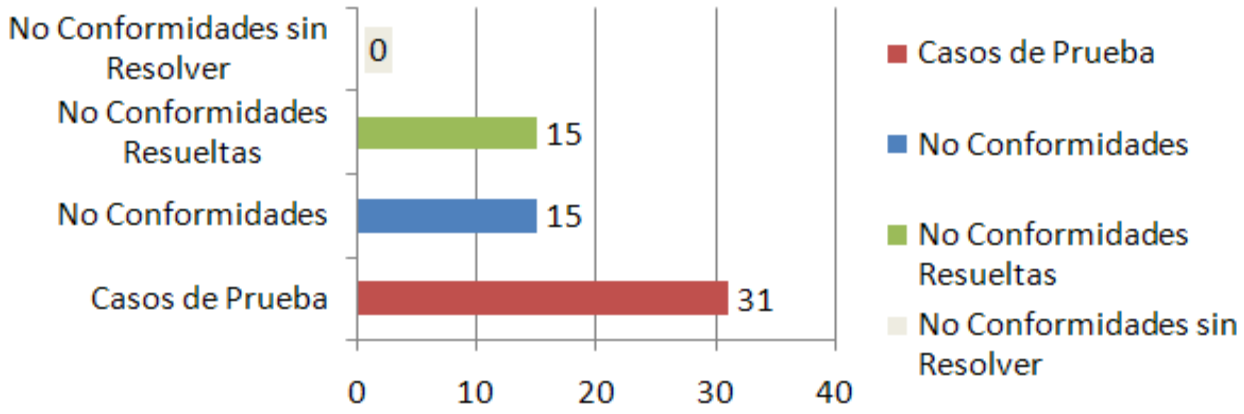

Figura 15: Primera iteración de los Casos de Pruebas.

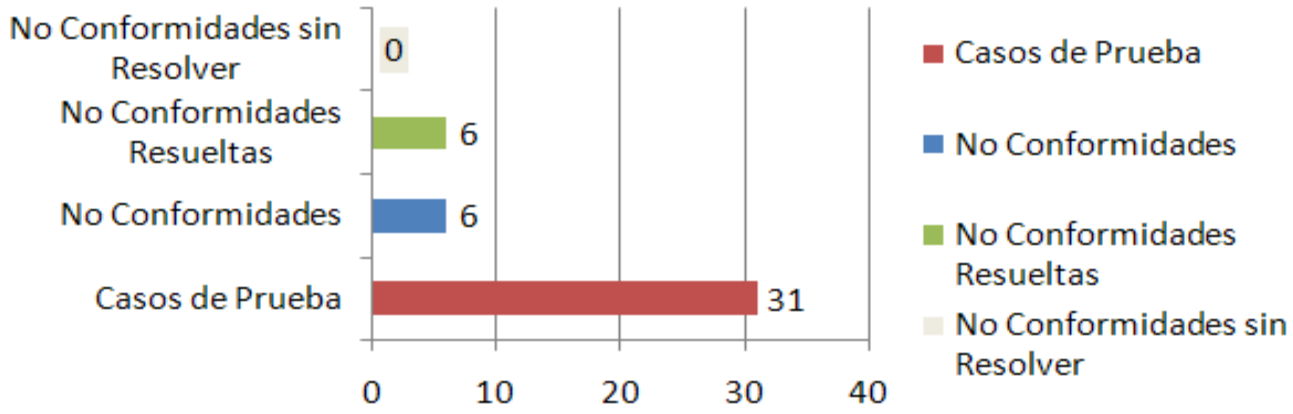

Figura 16: Segunda iteración de los Casos de Pruebas.

3.3 Evaluación de la facilidad de uso

La interfaz de usuario es un componente crítico para el desarrollo de cualquier aplicación informática, debido a que le permite al usuario interactuar con el sistema y éste a su vez debe ser capaz de mostrar de manera rápida y sencilla todo lo que puede hacer. Para los usuarios es de gran importancia el hecho de que los sistemas que se desarrollen hoy día tengan una interfaz simple, de fácil aprendizaje y además de fácil acceso a las funcionalidades que ofrecen. La accesibilidad está muy ligada a la facilidad de uso, que no es más que la facilidad con que las personas pueden usar una herramienta determinada con rapidez y efectividad.

La ISO/IEC 9241-11(Guidance on Usability) define la facilidad de uso como…*"La extensión para la que un producto puede ser usado por usuarios específicos, para lograr metas específicas con efectividad, eficacia y satisfacción en un contexto de uso específico".*

Desde la visión de Jacob Nielsen, la facilidad de uso se define en términos de cinco atributos: aprendizaje, eficiencia, memorización, prevención de error y satisfacción subjetiva, y para él la facilidad de uso es la *"Parte de la utilidad del sistema, la cual es parte de la aceptabilidad práctica y, finalmente parte de la aceptabilidad del sistema".*

Existen en el mundo diferentes formas y métodos de evaluar la facilidad de uso en un sistema, debido que este es un concepto imprescindible para asegurar un buen diseño de la interfaz de usuario. Según un análisis realizado sobre las diferentes clasificaciones establecidas por un conjunto de autores para evaluar la facilidad de uso, se identificó que los métodos más utilizados son (43):

- **Inspección:** Especialistas en software examinan el código en una búsqueda metódica de problemas. Seleccionan un conjunto de tareas representativas y prueban comando a comando, menú por menú, se formulan preguntas que están basadas en la teoría cognoscitiva de las relaciones entre metas, acciones y el estado visible de la interfaz.
- **Evaluación heurística:** Realizada revisando la interfaz del usuario y generando un informe de acuerdo a la propia opinión.
- **Investigación**: Empleada para conocer las opiniones de los usuarios o para entender sus preferencias sobre un producto potencial o uno existente a través de cuestionarios y entrevistas.
- **Empíricos:** Basada en observación del desempeño del usuario del diseño en uso.

El método que se propone para hallar la medida de la facilidad de uso, que permita decidir cuán eficiente y eficaz es la aplicación desarrollada, es a partir de una evaluación heurística definida por Jacob Nielsen.

3.3.1 Heurísticas de Jacob Nielsen

Jacob Nielsen define un conjunto de heurísticas[11] que deben ser aplicadas al desarrollar las interfaces gráficas de los sistemas informáticos y que éstas a su vez sean efectivas y eficientes.

- **H.1 Visibilidad del estado del sistema:** El sistema debe ser capaz de mantener informado al usuario de lo que está pasando en la aplicación, a través de información adecuada en un plazo razonable.
- **H.2 Control y libertad del usuario**: Los usuarios a menudo eligen funciones del sistema por error y será necesario un marcado claramente como "salida de emergencia" para salir del estado no deseado sin tener que pasar a través de un amplio diálogo. Apoyo de deshacer y rehacer.
- **H.3 Consistencia y estándares:** Los usuarios no deberían tener que preguntarse si diferentes palabras, situaciones o acciones significan lo mismo.
- **H.4 Prevención de errores:** El mejor tratamiento de los errores es prevenirlos con un buen diseño de los diálogos desde el primer momento en que ocurren, minimizando los riesgos de que puedan ocurrir. Se debe realizar un buen diseño de mensajes de error que den la posibilidad al usuario de retraerse antes de que se realice la acción y se comprometan los datos.
- **H.5 Correspondencia entre el sistema y el mundo real:** El sistema debe hablar el lenguaje de los usuarios, con palabras, frases y conceptos familiares al usuario, en lugar de términos orientados al sistema. Seguir las convenciones del mundo real, por lo que la información aparezca en un orden natural y lógico.
- **H.6 Reconocer antes que recordar:** Minimizar la carga del usuario, haciendo memoria de los objetos, acciones y opciones visibles. El usuario no debería tener que recordar la información de una parte del diálogo a otra. Instrucciones de uso del sistema deben ser visibles o fácilmente recuperables cuando sea apropiado.

[11] La heurística es una forma de revisión de los sitios web y que consisten en comprobar diferentes aspectos, desde el diseño y navegación hasta operación y utilidad.

- **H.7 Flexibilidad y eficiencia de uso:** El sistema se debe diseñar para que lo puedan manejar diferentes tipos de usuarios, en función de su experiencia con la aplicación. De esta manera se aumentará la productividad del usuario y se ganará en facilidad de uso.
- **H.8 Estética y diseño minimalista:** Los diálogos no deben contener información que sea irrelevante para la tarea que está realizando el usuario. Debe ser una interfaz simple, fácil de aprender y de usar y con fácil acceso a las funcionalidades que ofrece la aplicación.
- **H.9 Ayudar a los usuarios a reconocer, diagnosticar y recuperación de errores:** Los mensajes de error deben ser expresados en un lenguaje sencillo (sin códigos), indicar con precisión el problema y sugerir una solución constructiva.
- **H.10 Ayuda y documentación:** El mejor sistema es el que no necesita ningún tipo de documentación, pero de todas formas hay que proporcionar al usuario ayuda y documentación. Esta debe ser fácil de encontrar y enfocada a la tarea que el usuario realiza. Se deben listar sólo los pasos necesarios para la realización de la tarea.

3.3.2 Análisis de resultados en la medición de la facilidad de uso

A partir de los resultados obtenidos con la aplicación del modelo matemático que define Jacob Nielsen, como se muestra en la Figura 15, se obtuvo un nivel de facilidad de uso de 7,6 por lo que sitúa a la interfaz del componente en el intervalo de mejorable de tipo menor, con ello se concluye que los usuarios pueden fácilmente trabajar a pesar de presentar algunos problemas, además de que la interfaz de usuario hay que mejorarla para su liberación dentro de la versión actual hasta que su medida resulte aceptable, este modelo matemático se le aplicó además para realizar una comparación a la interfaz que provee el SGBD PostgreSQL y se obtuvo como resultado un nivel de facilidad de uso de 2,2 lo que lo sitúa en el intervalo de interfaz inaceptable de tipo Mayor, lo cual define que los usuarios tienen dificultad, pero son capaces de encontrar la forma de trabajar a pesar de ellas, aunque se puede poner en peligro el éxito de la tarea que se realice, además de que la interfaz de usuario es del todo inaceptable y se debería comenzar totalmente de nuevo su diseño**.**

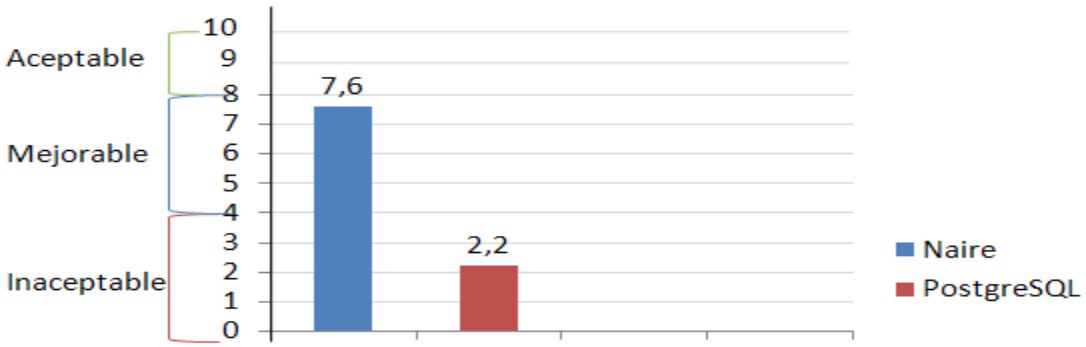

Figura 17: Niveles de facilidad de uso de la interfaz del Servidor de Gestión Naire y de PostgreSQL.

3.4 Evaluación del tiempo real del sistema

En el proceso de actualización y visualización de la información ha sido llevado a cabo en el tiempo requerido, cada una de las métricas son adicionadas y mostradas en un lapso de tiempo homogéneo donde el atributo hora es el encargado de verificar y almacenar el tiempo en que se realiza cada una de las monitorizaciones. La Figura 7 muestra el Estado del disco duro de un servidor en un intervalo de 12 segundos.

	id [PK] serial	servidor character va	libre character va	postgres character va	sistema character va	otros character va	fecha date	hora time withou
524	529	lp00-003-13	93 GB	69 MB	8926 MB	37 GB	2011-06-21	18:40:01
525	530	lp00-003-13	93 GB	69 MB	8926 MB	37 GB	2011-06-21	18:40:02
526	531	lp00-003-13	93 GB	69 MB	8926 MB	37 GB	2011-06-21	18:40:03
527	532	lp00-003-13	93 GB	69 MB	8926 MB	37 GB	2011-06-21	18:40:04
528	533	lp00-003-13	93 GB	69 MB	8926 MB	37 GB	2011-06-21	18:40:05
529	534	lp00-003-13	93 GB	69 MB	8926 MB	37 GB	2011-06-21	18:40:06
530	535	lp00-003-13	93 GB	69 MB	8926 MB	37 GB	2011-06-21	18:40:07
531	536	lp00-003-13	93 GB	69 MB	8926 MB	37 GB	2011-06-21	18:40:08
532	537	lp00-003-13	93 GB	69 MB	8926 MB	37 GB	2011-06-21	18:40:09
533	538	lp00-003-13	93 GB	69 MB	8926 MB	37 GB	2011-06-21	18:40:10
534	539	lp00-003-13	93 GB	69 MB	8926 MB	37 GB	2011-06-21	18:40:11
535	540	lp00-003-13	93 GB	69 MB	8926 MB	37 GB	2011-06-21	18:40:12

Figura 18: Análisis del Tiempo en el proceso de reporte

Conclusiones parciales

Con el objetivo de lograr una mayor calidad en el producto se realizaron pruebas de caja negra, diseñando un total de 31 casos de pruebas, las cuales arrojaron finalmente 21 no conformidades, donde todas fueron resueltas satisfactoriamente. Tambien se realizó un caso de prueba para verificar el cumplimiento del tiempo real en el componente, el cual dio resultados satifactorios. Debido a que la interfaz de usuario es una parte importante en el desarrollo de aplicaciones informáticas y es clave en el éxito o fracaso a la hora de la puesta en producción de la aplicación en el mundo real, se aplicaron métodos para medir el nivel de facilidad de uso de la aplicación, específicamente las Heuristicas definidas por Jacob Nielsen, las cuales arrojaron resultados satisfactorios a pesar de que la interfaz puede ser mejorada.

Conclusiones

El presente trabajo representa, además del comienzo de una nueva etapa donde la puesta en práctica del conocimiento adquirido es la premisa, la culminación de una larga etapa de trabajo que tuvo como resultado final un software capaz de generar reportes sobre el rendimiento general de los servidores de base de datos PostgreSQL, siendo una solución factible para el Servidor de Gestión de PostgreSQL del Centro de Tecnologías de Gestión de Datos (DATEC).

- El estudio de varias sistemas de monitorización a servidores PostgreSQL desarrollado por algunas de las empresas más importante del mercado, permitió identificar los requerimientos necesarios para implementar el servidor NAIRE en función de las necesidades particulares de Cuba y en favor de la soberanía tecnológica y el ahorro de gastos por concepto de soporte en caso de que la decisión hubiese sido adoptar cualquiera de los otros sistemas analizados.

- Las pruebas de caja negra permitieron identificar el correcto funcionamiento del sistema, de acuerdo a los requerimientos definidos. En este sentido se desarrollaron 31 de casos de pruebas que permitieron validar la conformidad de todos los requerimientos y al mismo tiempo identificar las recomendaciones necesarias para dar continuidad al trabajo.

- El modelo matemático aplicado para validar la facilidad de uso del sistema permitió corroborar que la interfaz de usuario, tan importante para este componente de reportes, cumplía los requerimientos mínimos de las meta heurísticas de la facilidad de uso tenidas en cuenta. No obstante se considera que aún es preciso continuar trabajando en la validación de estándares de diseño y la organización de la información para alcanzar niveles superiores en la evaluación de la facilidad de uso.

RECOMENDACIONES

Como derivación del transcurso de la investigación y realización de la aplicación, se seleccionaron un conjunto de mejoras para futuras versiones del software:

- Se propone utilizar el protocolo XMPP (protocolo de mensajería instantánea) como solución para mejorar la comunicación y el intercambio de información entre el componente de reporte y el componente de monitorización.
- Agregar un módulo de estadísticas el cual se encargue de realizar cálculos matemáticos a profundidad para contribuir de manera eficiente a la toma de decisiones.
- Elevar el número de métricas que contribuyan a aumentar el alcance del software.

REFERENCIAS BIBLIOGRAFICAS

1. **Infraestructura Productiva.** *Proyecto técnico PostgreSQL Empresarial.* PostgreSQL Empresarial, Universidad de las Ciencias Informáticas. La Habana : s.n., 2010. Proyecto Técnico. Proyecto Postgre Empresarial.

2. **Feinberg, Donald y Beyer, Mark A.** *Magic Quadrant for Data Warehouse. Database Management Systems.* Gartner RAS Core Research, Gartner. 2010.

3. **Castro Ibarra, Yamile.** Informática 2011. XIV Convención y Feria Internacional. *Informática 2011.* [En línea] 2009. [Citado el: 25 de Septiembre de 2010.] Agencia Internacional de Noticias (AIN). http://www.informaticahabana.cu/node/320.

4. **PostgreSQL Global Development Group.** PostgreSQL. [En línea] PostgreSQL Global Development Group, 2010. [Citado el: 25 de Septiembre de 2010.] Copyright © 1996 – 2010 PostgreSQL Global Development Group . http://www.postgresql.org/.

5. **EnterpriseDB Corporation.** EnterpriseDB. *EnterpriseDB.* [En línea] EnterpriseDB Corporation, 2010. [Citado el: 2 de Octubre de 2010.] http://www.enterprisedb.com/.

6. **Rice University.** ¿Qué son las bases de datos? *Connexions.* [En línea] 2010. [Citado el: 16 de Octubre de 2010.] http://cnx.org/content/m17423/latest/.

7. **Mato García, Rosa Maria.** *Diseño de Bases de Datos.* Habana : s.n., 1999. pág. 91. http://eva.uci.cu/file.php/624/3._Bibliografia/Diseno_de_BD/Libro_de_BD_de_Rosa_Maria.pdf.

8. **IEEE.** IEEE Advancing Technology for Humanity. *About IEEE.* [En línea] IEEE, 2010. [Citado el: 16 de Octubre de 2010.] http://www.ieee.org/.

9. **Date, C. J.** *Introducción a los Sistemas de Bases de Datos.* s.l. : Pearson Educación, 2001. Visitado en : http://books.google.com.cu/books?id=Vhum351T-K8C&dq=introduccion+a+los+sistemas+de+bases+de+datos+c.j.+date+%2B+gratis&printsec=frontcover&source=bn&hl=es&ei=fyKqTLniF8P58Aac7OjZDA&sa=X&oi=book_result&ct=result&resnum=4&ved=0CB8Q6AEwAw#v=o. 9684444192.

10. **PGDG- Wiki Oficial.** PostgreSQL Wiki. *PostgreSQL Wiki.* [En línea] PostgreSQL Global Development Group, 2010. [Citado el: 16 de Octubre de 2010.] http://wiki.postgresql.org/wiki/Preguntas_Frecuentes#.C2.BFQu.C3.A9_es_PostgreSQL.3F_.C2.BFCu.C3.A1l_es_su_pronunciaci.C3.B3n.3F_.C2.BFQu.C3.A9_es_Postgres.3F. Wiki oficial del PostgreSQL Global Development Group..

11. **Cybertec Schönig y Schönig GmbH.** Cybertec The PostgreSQL Database Company. [En línea] Cybertec Schönig y Schönig GmbH, 2010. [Citado el: 17 de Octubre de 2010.] http://www.cybertec.at/en/postgresql_products/cybercluster-2-0-synchronous-postgresql-replication.

12. **Web Commerce Group.** A Web Commerce Group Case Study on PostgreSQL. [En línea] 2002. [Citado el: 17 de Octubre de 2010.] http://www.postgresql.org/files/about/casestudies/wcgcasestudyonpostgresqlv1.2.pdf.

13. **Piñero, Pedro Y. y Arcia Carrazana, Maikel.** *Informe Tecnológico de la Producción.* Dirección Técnica de la UCI, Universidad de las Ciencias Informáticas. La Habana : s.n., 2010. pág. 37, Informe de Diagnóstico.

14. **IDRC.** Centro Internacional de Investigaciones para el Desarrollo (IDRC). [En línea] Parlamento de Canadá, 2010. [Citado el: 12 de Octubre de 2010.] http://www.idrc.ca/es/ev-30231-201-1-DO_TOPIC.html.

15. **Hyperic HQ.** Spring Source:Hyperic. *Spring Source:Hyperic.* [En línea] Hyperic HQ, 24 de Septiembre de 2010. [Citado el: 22 de Noviembre de 2010.] Dirección General :http://support.hyperic.com/dashboard.action. http://support.hyperic.com/display/DOC/ui-Dashboard.

16. **Hester, Blair.** vFabric Hyperic Documentation. *Spring Source:Hyperic.* [En línea] Hyperic HQ, 24 de Septiembre de 2010. [Citado el: 22 de Noviembre de 2010.] Última actualización: Marie McGarry. http://support.hyperic.com/.

17. **vFabric Documentation Team.** Monitor and Manage Resources. *Spring Sources Hyperic.* [En línea] 19 de Noviembre de 2010. [Citado el: 28 de Noviembre de 2010.] http://support.hyperic.com/download/attachments/72450372/Monitor_and_Manage_Resources-111910.pdf.

18. **Bartelt, Meg y McGarry, Marie.** ui-Report.Center . *Sptring Source Hyperic.* [En línea] 28 de Octubre de 2010. [Citado el: 28 de Noviembre de 2010.] http://support.hyperic.com/display/EVO/ui-Report.Center.

19. **McGarry, Marie.** HQ API Command-Line Tools. *Spring Source Hyperic.* [En línea] 6 de Agosto de 2010. [Citado el: 28 de Noviembre de 2010.] http://support.hyperic.com/display/EVO/Web+Services+API.

20. **Bartelt, Meg.** User Interface Reference. *Spring Source Hyperic.* [En línea] 5 de Agosto de 2010. [Citado el: 28 de Noviembre de 2010.] http://support.hyperic.com/display/EVO/User+Interface+Reference.

21.**opengraphprotocol.** [En línea] [Citado el: 20 de 2 de 2011.] http://opengraphprotocol.org/schema/sistemas-de-tiempo-real.htm.

22. **Canonical Ltd.** Ubuntu. *Ubuntu.* [En línea] 2010. [Citado el: 15 de Noviembre de 2010.] http://www.ubuntu.com/.

23. **Centro Europeo de Empresas e Innovación de Navarra.** Centro de excelencia de Osftware. [En línea] Centro Europeo de Empresas e Innovación de Navarra., 2008. [Citado el: 9 de Noviembre de 2010.] © 2008 CEIN - Centro Europeo de Empresas e Innovación de Navarra. http://www.cesnavarra.net/cesdigital/Lists/Noticias%20CESDigital/DispFormCES.aspx?List=5ec0dfc7-7911-470b-8b6b-71ba72783fdd&ID=48.

24. **Sitio Oficial del Redmine.** Redmine. [En línea] Redmine, 2006-2010. [Citado el: 9 de Noviembre de 2010.] Redmine © 2006-2010 Jean-Philippe Lang. http://www.redmine.org.

25. **Alfresco Software.** Alfresco. [En línea] Alfresco Software, 2010. [Citado el: 9 de Noviembre de 2010.] © 2010 Alfresco Software, Inc. Todos los Derechos Reservados.. http://www.alfresco.com.

26. **Wells, Don.** XP:eXtreme Programming. *Extreme Programming:A gentle introduction .* [En línea] 28 de Septiembre de 2009. [Citado el: 30 de Noviembre de 2010.] http://www.extremeprogramming.org/.

27. **Visual Paradigm.** Visual Paradigm. *Visual Paradigm.* [En línea] 2010. [Citado el: 9 de Noviembre de 2010.] //www.visual-paradigm.com/.

28. **Downey, Allen, Elkner, Jeffrey y Meyers, Chris.** *Aprende a pensar como un programador con Pyton.* Wellesley,Massachusetts : Green Tea Press, 2002. 0-9716775-0-6.

29. **Django Software Foundation.** Dango. [En línea] 2005-2010. [Citado el: 15 de Noviembre de 2010.] http://www.djangoproject.com/.

30. **jQuery Project.** jQuery . *jQuery .* [En línea] 2010. [Citado el: 15 de Noviembre de 2010.] http://jquery.com/.

31. **Nginx.** Nginx. [En línea] Nginx, Agosto de 2009. [Citado el: 4 de Abril de 2011.] http://nginx.net/.

32. **GIMP Team .** Gimp:GNU Image Manipulation Program . *Gimp:GNU Image Manipulation Program .* [En línea] 2001-2010. [Citado el: 15 de Noviembre de 2010.] http://www.gimp.org/. .

33. **MongoDB.** MongoDB. [En línea] Atlassian Confluence 3.0.0_01, 2010. [Citado el: 27 de Enero de 2010.] Copyright © 10gen, Inc. All Rights Reserved | Licensed under Creative Commons. http://www.mongodb.org/display/DOCS/Updating.

34. **IEEE Std 1233.** *Guía para el desarrollo de Especificaciones de Requerimientos de Sistemas.* 1998. IEEE Std 1233, Edición 1998.

35. **Rational Software Corporation.** *Rational Unified Process.* s.l. : Rational Software Corporation., 1987-2003. Copyright 1987 - 2003 Rational Software Corporation.All rights reserved..

36. **Leterier, Patricio y Penadés, Maria del Carmen.** *Metodologías ágiles para el desarrollo de Software:eXtreme Programming(XP).* Valencia : s.n. 46022.

37. **Aguilar Leiva, Milenis y Vinent Leyva, Henry.** *Análisis, Diseño e Implementación de un portal para la Escuela Cubana de Judo Femenino.* Universidad de las Ciencias Informáticas. La Habana, : s.n., 2009. pág. 91, Tesis de Pregrado.

38. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Pearson Educación S.A, 2000. ISBN: 0-20157169-2.

39. **Holovaty, Adrian y Most, Jacob Kaplan.** *El libro de Django.* s.l. : Appres, 2007.

40. **Larman, Craig.** *UML y Patrones.* México : Editorial Mexicana, 1999. ISBN 0-13-748880-7.

41. **Hernández González, Anaisa.** Revistas especializadas de prestigio en formato electrónico. e-Journal. . [En línea] 2006. [Citado el: 2011 de Abril de 3.] http://www.ejournal.unam.mx/cys/vol07-04/CYS07402.pdf.

42. **Pressman, Roger S.** Ingeniería de Software,un enfoque práctico. Quinta Edición, 1997, pág. 614.

43. **Alva Obeso, María Elena.** *Metodología de Medición y Evaluación de la Usabilidad en Sitios Web Educativos.* Oviedo : s.n., 2005. Tesis de Doctorado.

BIBLIOGRAFIAS

Aguilar Leiva, Milenis y Vinent Leyva, Henry. 2009. *Análisis, Diseño e Implementación de un portal para la Escuela Cubana de Judo Femenino.* Universidad de las Ciencias Informáticas. La Habana, : s.n., 2009. pág. 91, Tesis de Pregrado.

Almaer, Dion. *The Dojo Toolkit in Practice: An AJAX library for more than ´Prototype-ing´.*

Bartelt, Meg y McGarry, Marie. 2010. ui-Report.Center . *Sptring Source Hyperic.* [En línea] 28 de Octubre de 2010. [Citado el: 28 de Noviembre de 2010.] http://support.hyperic.com/display/EVO/ui-Report.Center.

Brito, Nacho. 2009. *Manual de desarrollo web con grails.* 2009.

Castro Ibarra, Yamile. 2009. Informática 2011. XIV Convención y Feria Internacional. *Informática 2011.* [En línea] 2009. [Citado el: 25 de Septiembre de 2010.] Agencia Internacional de Noticias (AIN). http://www.informaticahabana.cu/node/320.

Consorcio SIU. 2009. Monitoreando y midiendo el rendimiento de un servidor Postgresql en Sistemas Windows, Linux y Solaris. Parte 1. [En línea] 15 de Septiembre de 2009. [Citado el: 5 de Diciembre de 2010.] http://www.siu.edu.ar/documentos/Monitoreando%20Servidor%20Postgresql.pdf.

—. 2010. SIU. [En línea] 2010. [Citado el: 11 de Octubre de 2010.] Powered by SIU-Omaguaca CMS Engine. http://www.siu.edu.ar/documentos/Monitoreando%20Servidor%20Postgresql.pdf.

Date, C. J. 2001. *Introducción a los Sistemas de Bases de Datos.* s.l. : Pearson Educación, 2001. Visitado en : http://books.google.com.cu/books?id=Vhum351T-C&dq=introduccion+a+los+sistemas+de+bases+de+datos+c.j.+date+%2B+gratis&printsec=frontcover&source=bn&hl=es&ei=fyKqTLniF8P58Aac7OjZDA&sa=X&oi=book_result&ct=result&resnum=4&ved=0CB8Q6AEwAw#v=o. 9684444192.

Django Software Foundation. 2005-2010. Dango. [En línea] 2005-2010. [Citado el: 15 de Noviembre de 2010.] http://www.djangoproject.com/.

Downey, Allen, Elkner, Jeffrey y Meyers, Chris. 2002. *Aprende a pensar como un programador con Pyton.* Wellesley,Massachusetts : Green Tea Press, 2002. 0-9716775-0-6.

Draper, Dave. *Dojo concepts for Java Developers.*Embarcadero. [En línea] [Citado el: 12 de 12 de 2010.] www.embarcadero.com.

Feinberg, Donald y Beyer, Mark A. 2010. *Magic Quadrant for Data Warehouse. Database Management Systems.* Gartner RAS Core Research, Gartner. 2010.

GIMP Team . 2001-2010. Gimp:GNU Image Manipulation Program . *Gimp:GNU Image Manipulation Program .* [En línea] 2001-2010. [Citado el: 15 de Noviembre de 2010.] http://www.gimp.org/.

Gobierno de Chile. 2008. Guía para el Desarrollo de Sitios Web. [En línea] Gobierno de Chile, 2008. [Citado el: 7 de Marzo de 2011.] http://www.guiaweb.gob.cl/guia-2/capitulos/05/anexos/pauta-evaluacion-heuristica.pdf.

Hernández González, Anaisa. 2006. Revistas especializadas de prestigio en formato electrónico. e-Journal. . [En línea] 2006. [Citado el: 2011 de Abril de 3.] http://www.ejournal.unam.mx/cys/vol07-04/CYS07402.pdf.

Hester, Blair. 2010. vFabric Hyperic Documentation. *Spring Source:Hyperic.* [En línea] Hyperic HQ, 24 de Septiembre de 2010. [Citado el: 22 de Noviembre de 2010.] Última actualización: Marie McGarry. http://support.hyperic.com/.

Holovaty, Adrian y Most, Jacob Kaplan. 2007. *El libro de Django.* s.l. : Appres, 2007.

IEEE. 2010. IEEE Advancing Technology for Humanity. *About IEEE.* [En línea] IEEE, 2010. [Citado el: 16 de Octubre de 2010.] http://www.ieee.org/.

Infraestructura Productiva. 2010. *Proyecto técnico PostgreSQL Empresarial.* PostgreSQL Empresarial, Universidad de las Ciencias Informáticas. La Habana : s.n., 2010. Proyecto Técnico. Proyecto Postgre Empresarial.

Ingenieros de Software. [En línea] [Citado el: 25 de 1 de 2011.] http://www.ingenierosoftware.com.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid : Pearson Educación S.A, 2000. ISBN: 0-20157169-2.

jQuery Project. 2010. jQuery . *jQuery .* [En línea] 2010. [Citado el: 15 de Noviembre de 2010.] http://jquery.com/.

Kenneth Barclay, John Savage. *Groovy Programming an Introduction for Java Developers.*

L., Craig. 2009. *UML y Patrones.Introducción al Análisis y Diseño Orientado a Objetos.* 2009.

Larman, Craig. 1999. *UML y Patrones.* México : Editorial Mexicana, 1999. ISBN 0-13-748880-7.

Leterier, Patricio y Penadés, Maria del Carmen. *Metodologías ágiles para el desarrollo de Software:eXtreme Programming(XP).* Valencia : s.n. 46022.

Mato García, Rosa Maria. 1999. *Diseño de Bases de Datos.* Habana : s.n., 1999. pág. 91. http://eva.uci.cu/file.php/624/3._Bibliografia/Diseno_de_BD/Libro_de_BD_de_Rosa_Maria.pdf.

McGarry, Marie. 2010. HQ API Command-Line Tools. *Spring Source Hyperic.* [En línea] 6 de Agosto de 2010. [Citado el: 28 de Noviembre de 2010.] http://support.hyperic.com/display/EVO/Web+Services+API.

Microsoft. 2007. *Service_Oriented_Architecture_ (SOA) _in_the_real_World).* 2007.

MongoDB. 2010. Mongo DB. [En línea] Rights Reserved, 2010. [Citado el: 27 de Enero de 2011.] Copyright © 10gen, Inc. All Rights Reserved | Licensed under Creative Commons. http://www.mongodb.org/.

Mora, Sergio Luján. *Programación de aplicaciones web: historia, principios básicos y clientes web.*

Murray, Alex Russell-Greg. *Usisng the Dojo Toolkit to Develop AJAX-Enabled Java EE Wen Aplications.*

Nginx. 2009. Nginx. [En línea] Nginx, Agosto de 2009. [Citado el: 4 de Abril de 2011.] http://nginx.net/.

Nielsen, Jacob. Jacob Nielsen Website. [En línea] [Citado el: 4 de Marzo de 2011.] http://www.useit.com/papers/heuristic/heuristic_list.html.

Orallo, Enrique Hernández. *El lenguaje Unificado de Modelado(UML).* Osiatis. [En línea] [Citado el: 5 de 12 de 2010.] http://itilv3.osiatis.es.

Pérez, Javier Eguíluz. *Introducción a AJAX.*

Piñero, Pedro Y. y Arcia Carrazana, Maikel. 2010. *Informe Tecnológico de la Producción.* Dirección Técnica de la UCI, Universidad de las Ciencias Informáticas. La Habana : s.n., 2010. pág. 37, Informe de Diagnóstico.

PostgreSQL Global Development Group . 2010. PostgreSQL. *PostgreSQL.* [En línea] PostgreSQL Global Development Group, 2010. [Citado el: 21 de Octubre de 2010.] http://www.postgresql.org/about/casestudies/vanten.

PostgreSQL Global Development Group. 2010. Case Studies. *PostgreSQL.* [En línea] 2010. [Citado el: 17 de Octubre de 2010.] Organizaciones que usan PostgreSQL.. http://www.postgresql.org/about/casestudies/.

Pressman. 2005. *Ingenieria del Software. Un enfoque práctico.* 2005.

Pressman, Roger S. 1997. Ingeniería de Software,un enfoque práctico. Quinta Edición, 1997, pág. 614.

Rational Software Corporation. 1987-2003. *Rational Unified Process.* s.l. : Rational Software Corporation., 1987-2003. Copyright 1987 - 2003 Rational Software Corporation.All rights reserved..

Rawld Gill, Craig Rieke, Alex Russell. *Mastering Dojo JavaScript and AJAX Tools for Greats Web Experiences.*

real, La Arquitectura Orientada a Servicios de Microsoft aplicada al mundo. 2006. 2006.

Rice University. 2010. ¿Qué son las bases de datos? *Connexions.* [En línea] 2010. [Citado el: 16 de Octubre de 2010.] http://cnx.org/content/m17423/latest/.

Taylor, Sharon, Iqbal, Majid y Nieves, Michael. 2008. *ITIL v3 - Libro 1 - Service Strategy.* 2008.

Visconti, Marcello y Astudillo, Hernán. 2010. [En línea] 2010.

Visual Paradigm Design Group. visualparadigm. [En línea] http://www.visual-paradigm.com.

Visual Paradigm. 2010. Visual Paradigm. *Visual Paradigm.* [En línea] 2010. [Citado el: 9 de Noviembre de 2010.] //www.visual-paradigm.com/.

Wells, Don. 2009. XP:eXtreme Programming. *Extreme Programming:A gentle introduction .* [En línea] 28 de Septiembre de 2009. [Citado el: 30 de Noviembre de 2010.] http://www.extremeprogramming.org/.