Universidad de las Ciencias Informáticas
Facultad 3

Trabajo de diploma para optar por el tíulo de Ingeniero en Ciencias Informáticas

Título:
Herramienta para obtener estadísticas del sistema gestor de base de datos PostgreSQL.

Autores: Anaily Navia López
Yariel Alberto Vivero Díaz

Tutor: Ing. Liannis Soria Barreda

Ciudad de La Habana, Cuba. Noviembre, 2014

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

Ernesto Che Guevara

Declaración de autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ___ días del mes de _____ del año _____.

Anaily Navia López

_____

Firma del autor

Yariel Alberto Vivero Díaz

_____

Firma del autor

Ing. Liannis Soria Barreda

_____

Firma del tutor

Agradecimientos

El agradecimiento más importante de todos es para mis padres, Maira y Enrique, por impulsarme a llegar hasta aquí, por darme su apoyo incondicional, por ayudarme en los momentos buenos y malos, por preocuparse y sacrificarse tanto por mí.

A mi hermana Iliana que me ha ayudado siempre en todas las tareas que le he pedido.

A Gabriel por estar siempre que lo he necesitado y aconsejarme para superarme cada día más.

A toda mi familia y todos los de mi barrio.

A mi tutora Liannis por ser una persona ejemplar, por sus críticas, por sus recomendaciones, por su gran conocimiento que nos ayudó a lograr este trabajo.

A mi novio Yariel que ha estado a mi lado en todo momento, trabajando siempre juntos, por soportarme cuando no salían las cosas bien, por darme ánimo a seguir adelante.

A Yoyi por ser una suegra magnífica, a Lily y a toda su familia.

A mis amigos Liset y Michel que han sido incondicionales, que han sabido escucharme, ayudarme y entenderme en todo momento.

A todos mis compañeros de grupo, pues hemos vivido momentos muy buenos juntos, nos hemos ayudado unos a otros desde cosas pequeñitas hasta grandes proyectos.

Muchísimos son los nombres de las personas que durante la carrera han brindado su granito de arena para poder llegar hasta donde estoy hoy y no quisiera dejar de mencionar alguno, ellos saben que les estoy agradecida por siempre. A todos de corazón muchas gracias.

Anaily

Agradecimientos

A mis padres, hermanos, hermanas y familiares en general, en especial a mi madre Georgina y mi hermana jimagua Lily.

A mi novia Anaily por todos los momentos especiales que hemos pasado juntos. A sus padres Maira y Enrique así como a su hermana lliana y su esposo Gabriel. En general a toda su familia.

A mi tutora Liannis por ayudarnos y apoyarnos durante el desarrollo de la tesis.

A todos mis amistades que me ayudaron en estos cinco años.

Yariel

Dedicatoria

A mis padres, a mi familia y en general a todos mis amigos.

Anaily y Yariel

Resumen

En un proyecto de desarrollo de software los administradores de bases de datos deben mantener funcionando correctamente el Sistema Gestor de Bases de Datos (SGBD) utilizado. Para lograrlo los administradores deben tener a su disposición la mayor cantidad de información posible sobre su funcionamiento. El SGBD PostgreSQL, a través de diferentes consultas, permite obtener información referente al desempeño y estadísticas del servidor, las cuales deben ser consultadas individualmente; esta situación implica que los administradores deban poseer altos conocimientos sobre la sintaxis de las mismas y que no cuenten con toda la información de forma rápida e integrada. Además, con las consultas no se obtienen toda la información necesaria, lo que obliga a los administradores a ejecutar diferentes comandos en terminales del sistema. El presente trabajo de diploma tiene como objetivo el desarrollo de una herramienta para obtener estadísticas del SGBD PostgreSQL llamada MaGeoLoDi, software que permite a los administradores de bases de datos contar con toda la información sobre el funcionamiento de PostgreSQL de forma rápida e integrada.

Palabras clave: administradores, estadísticas, PostgreSQL, rendimiento.

Índice de contenido

Índice de figuras

Índice de tablas

Introducción

Cada día son generados grandes volúmenes de información y de datos para apoyar la toma de decisiones en las organizaciones, por lo que es necesario que sean almacenados y gestionados para su posterior análisis. Con este propósito surgen los sistemas gestores de bases de datos, los cuales permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada convirtiéndose en una herramienta imprescindible para la gestión y almacenamiento de la información en las organizaciones.

Una base de datos (BD) es un conjunto de información o colección de datos agrupados con un propósito específico en una o más entidades lógicas (archivos). En términos estrictos, "base de datos", hace referencia solo al lugar donde está depositada la información desde la perspectiva del software como del hardware. Dicha información generalmente está soportada por un sistema para su gestión. [1]

A nivel mundial han surgido una diversidad de sistemas gestores de bases de datos con características específicas, dando paso a la elección de uno u otro según las necesidades y políticas de la organización. Entre los SGBD más utilizados en la actualidad se encuentran Oracle, SQLServer, MySQL y PostgreSQL, siendo los dos primeros software privados con licencias de alto costo, por lo que cada vez más las empresas se inclinan por PostgreSQL o MySQL, dependiendo del volumen de datos a manejar.

Una de las principales metas que Cuba se ha propuesto dentro del proceso de desarrollo socio-económico es llevar a cabo el proceso de informatización de la sociedad, comenzando fundamentalmente por las empresas y organizaciones hasta llegar a todos los hogares, lo que conlleva la aplicación exhaustiva de gestores de bases de datos adaptables a las necesidades existentes. La soberanía tecnológica es un elemento fundamental a tener en cuenta en este proceso, por lo que Cuba adoptó PostgreSQL como gestor de bases de datos de código abierto por ser el más avanzado de su tipo [2].

La Universidad de las Ciencias Informáticas (UCI) es una de las principales instituciones en el país que contribuye al proceso de informatización de la sociedad. Para esto, se han creado en todas sus facultades diferentes centros de desarrollo los cuales tienen como objetivo desarrollar software. Entre estos centros se encuentra el Centro de Informatización de Gestión de Entidades (CEIGE) y el Centro de Gobierno Electrónico (CEGEL) de la Facultad 3, los cuales usan el SGBD PostgreSQL para gestionar y almacenar la información de sistemas tales como Fiscalía, Tribunales, Cedrux y Sipac.

Las personas encargadas de mantener el correcto funcionamiento de los SGBD son los administradores de bases de datos, los cuales tienen como tareas definir y controlar las bases de datos corporativas; además de ser responsables de asegurar que el sistema opere con el rendimiento adecuado. Se requiere también que los administradores tengan experiencia en el diseño de bases de datos, sistemas operativos, hardware y programación. [3]

Para lograr resultados eficientes en un servidor de base de datos, el administrador del mismo debe mantenerlo monitoreado verificando su rendimiento y funcionamiento. Para esto, PostgreSQL brinda la posibilidad, a través de diferentes consultas, de obtener información referente al desempeño y estadísticas del servidor; las cuales son consultadas individualmente debido a que cada una retorna diferentes datos que varían en dependencia de la utilización del servidor, por lo que resulta incómodo ejecutarlas repetidamente dada la gran cantidad que existen. Estas consultas requieren que los administradores posean altos conocimientos de las mismas, por lo que su desconocimiento puede provocar que en ocasiones problemas fáciles de resolver conlleven a una sobrecarga o lentitud del servidor.

Por otro lado existen diferentes comandos que muestran reportes estadísticos del procesador y de la memoria virtual del sistema, así como el consumo de los recursos del sistema de los procesos activos de PostgreSQL, los cuales son ejecutados en terminales y poseen una serie de parámetros convirtiéndolos en instancias difíciles de recordar. Además para visualizar varias salidas de comandos a la vez, se deben utilizar diversas terminales, las cuales muestran el nombre de los parámetros mediante siglas lo que hace difícil su entendimiento e interpretación, provocando pérdida de tiempo en momentos donde se necesiten los datos de forma inmediata.

Las situaciones anteriormente mencionadas traen consigo que los administradores de bases de datos de un proyecto de desarrollo de software no puedan contar con la información necesaria de forma rápida e integrada para la toma de decisiones que permitan mejorar el rendimiento de un SGBD. Para este trabajo se definió el siguiente **problema de la investigación:** ¿Cómo obtener las estadísticas del rendimiento de un sistema gestor de bases de datos PostgreSQL de manera rápida e integrada?

El **objeto de estudio** del presente trabajo está enmarcado en los Sistemas Gestores de Bases de Datos, definiendo como **campo de acción** las herramientas tuning para el sistema gestor de bases de datos PostgreSQL.

Para dar solución al problema se define como **objetivo general:** Desarrollar una herramienta que permita la obtención de las estadísticas del rendimiento del sistema gestor de bases de datos PostgreSQL de manera rápida e integrada.

Definiéndose los siguientes **objetivos específicos**:

1- Elaborar el marco teórico de la investigación para detallar los antecedentes de las herramientas tuning que existen para el sistema gestor de bases de datos PostgreSQL.
2- Obtener los artefactos de análisis y diseño para una herramienta que permita obtener las estadísticas de rendimiento de un sistema gestor de bases de datos PostgreSQL.
3- Implementar los requisitos descritos para una herramienta que permita obtener las estadísticas de rendimiento de un sistema gestor de bases de datos PostgreSQL.
4- Validar la herramienta implementada para determinar el grado de cumplimiento del objetivo planteado en la investigación.

Para darle cumplimiento a los objetivos específicos, se plantean las siguientes **tareas de investigación**:

1- Análisis bibliográfico de las herramientas de monitoreo existentes.
2- Selección de las métricas a evaluar en la solución propuesta.
3- Identificación de los requisitos de la herramienta de monitoreo.
4- Realización de análisis de la propuesta que permita generar los artefactos correspondientes.
5- Diseño de la solución propuesta.
6- Implementación de la herramienta de monitoreo.
7- Definición de las pruebas a realizar a la herramienta.
8- Realización de pruebas funcionales para verificar el funcionamiento de la herramienta de monitoreo.

En la realización de esta investigación se utilizan los siguientes **métodos científicos**:

Analítico-Sintético: este método permitió analizar y estudiar la bibliografía consultada para extraer lo esencial, lo que permitió comprender las características generales de los sistemas de monitoreo en tiempo real tras sintetizar éstas como un todo, concretando sus elementos más importantes.

Histórico-Lógico: este método permitió realizar un estudio detallado para conocer las tendencias actuales de los sistemas de monitoreo existentes, su empleo en el mundo y los aspectos más importantes desde su surgimiento hasta la actualidad. También permitió recopilar información para describir las tendencias tecnológicas a utilizar, aprovechando elementos teóricos esenciales para el desarrollo de la herramienta.

Inductivo-deductivo: este método permitió realizar un análisis de elementos generales a elementos particulares, o sea, se dispone de un grupo de conocimientos generales sobre las tendencias actuales y características generales de los sistemas de monitoreo y se pretende llegar a particularizar lo conocido en función de la situación problemática presentada.

La presente investigación se estructura en tres capítulos:

En el Capítulo 1 se elabora un marco teórico sobre las herramientas de monitoreo que brindan estadísticas sobre el SGBD PostgreSQL. Además se describe el entorno de desarrollo y la metodología utilizada para el desarrollo del sistema de monitoreo propuesto en la investigación.

En el Capítulo 2 se describen los procesos que intervienen en la solución, los cuales permiten describir las principales características del sistema de monitoreo propuesto en la investigación. Además se definen los requisitos funcionales y no funcionales, así como los diagramas de clases persistentes para lograr un mejor entendimiento del componente a desarrollar.

En el Capítulo 3 se refleja la evaluación técnica del componente propuesto en la investigación. Para esto se valida que los requisitos definidos sean correctos y que el diseño realizado tenga la calidad requerida. Además se realizan validaciones a los resultados mostrados por la herramienta propuesta en la investigación que permiten demostrar su correcto funcionamiento.

Capítulo 1: Fundamentación teórica.

Introducción.

En el presente capítulo se exponen los conceptos relacionados con el tema de las bases de datos, como servidores de gestión, monitoreo, centrándose específicamente en el SGBD PostgreSQL. También se describen los aspectos fundamentales relacionados a las herramientas y la metodología a utilizar para el desarrollo del componente de monitoreo.

1.1. Servidores de Gestión.

Un servidor, como la misma palabra indica, es un ordenador o máquina informática que está al "servicio" de otras máquinas, ordenadores o personas llamadas clientes y que le suministran todo tipo de información [4]. Por regla general, las máquinas servidoras suelen ser algo más potentes que un ordenador normal y suelen tener más capacidad tanto de almacenamiento de información como de memoria principal, ya que tienen que dar servicio a muchos clientes. Existen diferentes tipos de servidores, dependiendo de su utilidad, por ejemplo, se suele llamar servidor web a la computadora cuya actividad principal es enviar páginas web a los usuarios que las solicitan cuando se conectan a internet; un servidor de correo es el que almacena, envía, recibe y realiza todas las operaciones relacionadas con el correo de sus clientes, así como un servidor de base de datos es aquel que da servicios de almacenamiento y gestión de bases de datos a sus clientes. [5]

1.2. Sistemas Gestores de Bases de Datos.

Un sistema gestor de bases de datos es una aplicación que permite a los usuarios definir, crear y mantener la BD y proporciona un acceso controlado a la misma. Un SGBD debe prestar los siguientes servicios [6]:

- ✓ Definición y creación de la base de datos.
- ✓ Manipulación de los datos realizando consultas, inserciones y actualizaciones.
- ✓ Acceso controlado a los datos mediante mecanismos de seguridad de acceso a los usuarios.
- ✓ Mantener la integridad de los datos.
- ✓ Controlar la concurrencia a la base de datos.
- ✓ Mecanismos de copias de respaldo y recuperación para restablecer la información en caso de fallos en el sistema.

1.2.1. El Sistema Gestor de Bases de Datos PostgreSQL.

El sistema gestor de bases de datos PostgreSQL es considerado el más avanzado de los SGBD de código abierto [2] ofreciendo control de concurrencia multi-versión, soportando casi todas las sentencias SQL incluyendo subconsultas, transacciones y funciones definidas por el usuario. Existen disímiles empresas que utilizan este gestor, como la EnterpriseDB, líder proveedor de productos de clase empresarial y servicios basados en PostgreSQL. Esta empresa además de contribuir con el proyecto ofreciendo diversas opciones, brinda soporte técnico para ayudar al desarrollo y despliegue de aplicaciones basadas en PostgreSQL, además de comercializar productos como Postgres Plus Standard Server y Postgres Plus Advanced Server. [7]

PostgreSQL es uno de los sistemas más utilizados de objetos relacionales de gestión de bases de datos en la industria del desarrollo de software y el entorno empresarial. Debido a que los servidores de bases de datos están en el núcleo de muchos servicios esenciales para las empresas, cualquier falla o degradación del rendimiento del SGBD puede impactar negativamente en múltiples servicios. Por tal motivo es de vital importancia realizar el monitoreo de los SGBD. [8]

1.3. Rendimiento de un sistema informático.

Debido al crecimiento vegetativo de la carga de un sistema, cuando las prestaciones del mismo decrecen, es necesario ajustar o cambiar alguno de los parámetros, normalmente para mejorar su comportamiento. En algunos casos si el sistema (hardware más software básico) no se puede variar, hay que intentar mejorar el comportamiento del sistema modificando la carga (programas). El comportamiento de un sistema es muy dependiente de la carga aplicada al mismo; por lo tanto conocer o caracterizar la carga que tiene un sistema y predecir su carga futura es fundamental para poder estudiar el comportamiento del mismo.

Los objetivos de la evaluación del rendimiento son los siguientes:

- ✓ Comparar distintas alternativas.
- ✓ Determinar el impacto de un nuevo elemento o característica en el sistema.
- ✓ Sintonizar el sistema, es decir, hacer que funcione mejor desde algún punto de vista.
- ✓ Medir prestaciones relativas entre diferentes sistemas.
- ✓ Depuración de prestaciones, es decir, identificar los fallos del sistema que hacen que vaya más lento.

Los usuarios o las necesidades exigen que el sistema deba cumplir una serie de requisitos o especificaciones de ejecución para considerar que su funcionamiento es aceptable. Otros de los objetivos que se pretende al evaluar un sistema informático es encontrar los factores que impiden su adecuado funcionamiento. [9]

1.4. Monitoreo a servidores de bases de datos.

Es el proceso sistemático de recolectar, analizar y utilizar información para hacer seguimiento al progreso de un programa en pos de la consecución de sus objetivos, lo que permita guiar las decisiones de gestión de los directivos de una empresa. El monitoreo generalmente se dirige a los procesos en lo que respecta a cómo, cuándo y dónde tienen lugar las actividades, quién las ejecuta y a cuántas personas o entidades beneficia. [10]

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

Para realizar el monitoreo a servidores de aplicaciones y bases de datos es necesario contar con herramientas que permitan obtener estadísticas de los mismos. Dentro de las encargadas de realizar el monitoreo a PostgreSQL se encuentran:

1.5.1. Pgbench.

El pgbench es un programa sencillo para ejecutar las pruebas de referencia en PostgreSQL. Las pruebas las pueden hacer y comparar entre sí todos los usuarios, ya que son siempre las mismas, utilizando los mismos datos. De forma predeterminada, pgbench prueba un escenario, con la participación de cinco selecciones, actualizaciones y comandos INSERT por transacción. Sin embargo, es fácil de probar otros casos pues escribe sus propios archivos script de transacción.

Una limitación de pgbench es que cuando se trata de probar un gran número de sesiones de cliente puede convertirse en sí mismo en cuello de botella, lo que puede ser aliviado mediante la ejecución de la herramienta en una máquina diferente al servidor de base de datos, aunque la latencia baja de red será esencial. Incluso podría ser útil para ejecutar varias instancias simultáneamente, en varias máquinas cliente, contra el mismo servidor de base de datos. Su ejecución es mediante una terminal mostrando salidas en líneas, además con el uso de varias opciones, es posible realizar diferentes pruebas útiles a la base de datos. [2]

A continuación se muestra la sintaxis que se debe utilizar para ejecutar el programa en la Figura 1 y algunas de las diferentes opciones que pueden ser utilizadas en su sintaxis en la Tabla 1, así como una salida típica de la herramienta pgbench en la Figura 2:

```
pgbench [ options ] dbname
```

Figura 1: Sintaxis del pgbench.

Tabla 1: Opciones del pgbench.

Opción	Descripción

-i | --initialize	Se requiere para invocar el modo de inicialización.
-c clients | --client=clients	Número de clientes simulados, es decir, el número de sesiones de base de datos concurrentes.
-C | --connect	Establecer una nueva conexión para cada transacción, en lugar de hacerlo sólo una vez por sesión de cliente.
-d | --debug	Imprimir la salida de depuración.
-l | --log	Escriba el tiempo empleado por cada transacción a un archivo de registro.
-t transactions | --transactions=transactions	Número de transacciones que cada cliente ejecuta.
-h hostname | --host=hostname	Nombre de host del servidor de base de datos
-p port | --port=port	Número de puerto del servidor de base de datos

```
transaction type: TPC-B (sort of)
scaling factor: 10
query mode: simple
number of clients: 10
number of threads: 1
number of transactions per client: 1000
number of transactions actually processed: 10000/10000
tps = 85.184871 (including connections establishing)
tps = 85.296346 (excluding connections establishing)
```

Figura 2: Salida típica del pgbench.

1.5.2. pgBadger.

El pgBadger es un analizador de registros de PostgreSQL para construir a gran velocidad informes detallados y gráficos de los archivos de registro de PostgreSQL. Es un sencillo y pequeño script en Perl que supera a cualquier otro analizador de registro desarrollado para PostgreSQL. Esta herramienta es capaz de detectar automáticamente el formato de archivo de registro (syslog, stderr o csvlog). Está diseñada para analizar los archivos de registro grandes, así como archivo comprimido gzip. También permite el procesamiento paralelo en un único archivo de registro y varios archivos mediante el uso de la opción -j y el número de CPU[1] como valor. Además soporta un modo incremental que permite construir informes incrementales después de sucesivas ejecuciones. En los informes se muestran solamente datos de diferentes consultas, por lo que a través de esta herramienta no se obtiene información alguna de los procesos del sistema.

[1] CPU: Unidad central de procesamiento.

Los comandos de pgBadger son ejecutados mediante una terminal a través de diferentes opciones, permitiendo generar documentos HTML[2] a partir de un archivo de registro de PostgreSQL. Los informes generados se muestran mediante gráficos que son representados utilizando una biblioteca de JavaScript. [11]

A continuación se muestra la sintaxis que se debe utilizar para ejecutar el programa en la Figura 3 y algunas de las diferentes opciones que pueden ser utilizadas en su sintaxis en la Tabla 2, así como una salida típica de la herramienta pgBadger en la Figura 4:

```
pgbadger [opciones] archivo de registro [...]
```

Figura 3: Sintaxis del pgBadger.

Tabla 2: Opciones del pgBadger.

Opción	Descripción
-a | --average minutes	Número de minutos para construir los medios gráficos de consultas y conexiones.
-c | --dbclient host	Sólo informar sobre las entradas para el host del cliente dado.
-d | --dbname database	Sólo informar sobre las entradas de la base de datos determinada.
-G | --nograph	Desactivar gráficos en la salida HTML. Activar por defecto.
-j | --jobs number	Número de archivo de registro para analizar en paralelo. Por defecto es 1, se ejecuta como un solo proceso.
-l | --last-parsed file	Permite el análisis de registro gradual registrando la última fecha y hora y las líneas analizadas.
-S | --select-only	Usarlo si quieres sólo para reportar las consultas de selección.
-v | --verbose	Activar el modo detallado o de depuración. Por defecto está bloqueado.
--include-query regex	Ruta del archivo que contiene toda la expresión regular a utilizar para excluir a las consultas del informe.

[2] HTML: Lenguaje de marcas de hipertexto.

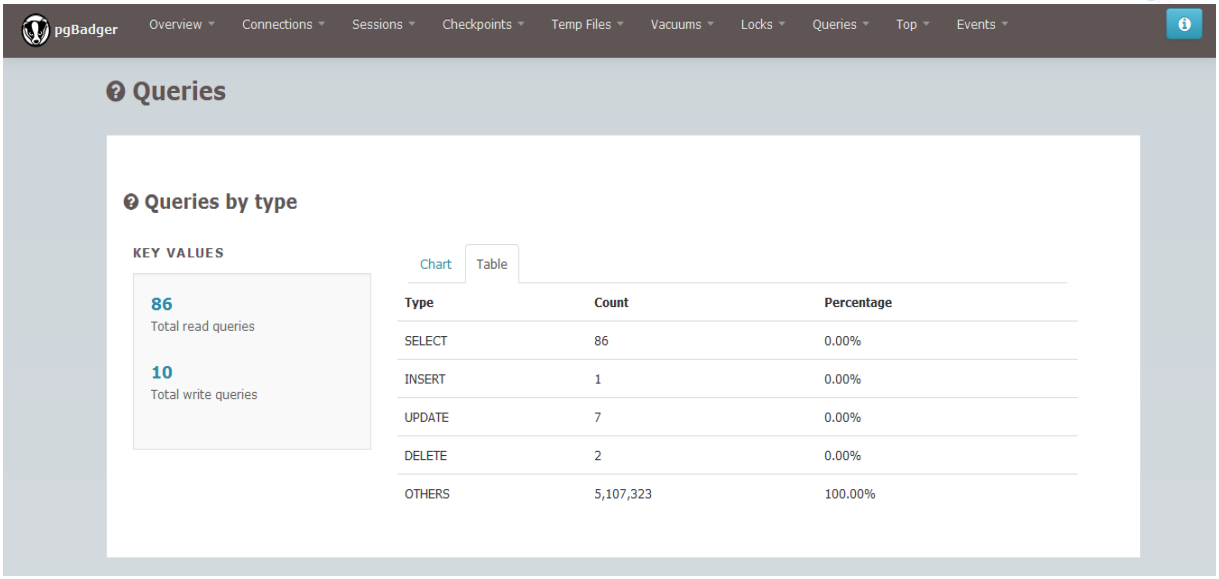

Figura 4: Ejemplo de salida del pgBadger.

1.5.3. Pgpool-II.

La herramienta Pgpool-II es un middleware que funciona entre servidores y clientes de bases de datos PostgreSQL. A continuación se listan sus características. [12]

- ✓ Agrupación de conexiones: Pgpool-II salva conexiones a los servidores PostgreSQL, permite reutilizarlas cada vez que una nueva conexión con las mismas propiedades (es decir, nombre de usuario, bases de datos, la versión de protocolo) entra en juego, reduce el área de conexión y mejora el rendimiento general del sistema.
- ✓ Replicación: a través de esta opción se puede gestionar múltiples servidores PostgreSQL. El uso de la función de replicación permite crear una copia de seguridad en tiempo real en dos o más discos físicos, por lo que el servicio pueda continuar sin detener los servidores en caso de fallo de un disco.
- ✓ Equilibrio de carga: si se replica una base de datos, ejecutar una consulta SELECT en cualquier servidor devolverá el mismo resultado. La herramienta tiene la ventaja de la función de duplicación para reducir la carga en cada servidor PostgreSQL mediante la distribución de consultas SELECT entre varios servidores, mejorando el rendimiento global del sistema. A lo sumo, el rendimiento mejora de forma proporcional al número de servidores PostgreSQL. El equilibrio de carga funciona mejor en una situación en la que hay una gran cantidad de usuarios que ejecutan muchas consultas al mismo tiempo.
- ✓ Exceder límite de conexiones: hay un límite en el número máximo de conexiones simultáneas con PostgreSQL, donde las conexiones son rechazadas después de que la cantidad de conexiones establecidas para este límite sea alcanzada. Una desventaja que presenta es el ajuste del número máximo de conexiones, pues aumenta el consumo de recursos y afecta el

rendimiento del sistema. Pgpool-II también tiene un límite en el número máximo de conexiones, pero las conexiones adicionales se pondrán en cola en lugar de devolver un error de inmediato.

- ✓ Consulta en paralelo: el uso de la función de consulta en paralelo, los datos se puede dividir entre los múltiples servidores, por lo que una consulta puede ejecutarse en todos los servidores a la vez para reducir el tiempo total de ejecución. Consulta en paralelo funciona mejor en la búsqueda de datos a gran escala.

A continuación se muestra la sintaxis que se debe utilizar para ejecutar el programa en la Figura 5 y algunas de las diferentes opciones que pueden ser utilizadas en su sintaxis en la Tabla 3, así como un ejemplo del uso de la función replicación de la herramienta Pgpool-II en la Figura 6:

```
# psql –h pgsql1 –p 9999 –U pgpool2 –d postgres
```

Figura 5: Comando para conectarse a través de Pgpool-II.

Tabla 3: Descripción de las opciones empleadas en la Figura 5.

Opción	Descripción
-h host	Nombre del host al que se va a conectar y en donde se encuentra instalado pgpool-II.
-p port	Puerto definido en el archivo /etc/pgpool2/pgpool.conf.
-U user	Usuario creado en los n nodos del clúster.
-d database	Base de datos a la que se va a conectar.

A continuación, un ejemplo del uso de la función replicación. Se creará una base de datos la cual será replicada a través de Pgpool-II:

```
# createdb –h pgsql1 –p 9999 –U pgpool2 basesdedatosues
```

Figura 6: Creación de una base de datos la cual será replicada a través de Pgpool-II.

1.5.4. Resultados del estudio de las herramientas existentes para el monitoreo.

Después de analizadas las herramientas de monitoreo anteriores se observa que no brindan las estadísticas de rendimiento de un SGBD PostgreSQL de forma integrada. Debido a esta situación el trabajo de los administradores de bases de datos se torna complejo ya que la información que ofrecen las herramientas acerca de transacciones, consultas, conexiones y sesiones se basan en comandos de consola para su ejecución, lo que requiere de conocimientos avanzados en el tema y provoca dificultades a la hora de analizar las estadísticas obtenidas. Por otra parte, las estadísticas

referentes a los procesos activos de PostgreSQL, el procesador y la memoria virtual del sistema, así como los datos del uso del espacio del disco duro y del espacio usado por el servidor PostgreSQL, no son mostrados por ninguna de las herramientas anteriormente analizadas.

1.6. Entorno tecnológico para el desarrollo de la herramienta MaGeoLoDi.

Para el desarrollo de la herramienta se realizó un estudio sobre las posibles herramientas a utilizar en su construcción. Teniendo en cuenta las tendencias actuales que existen en la universidad de migrar a software libre, la mayoría de las herramientas seleccionadas pertenecen a software libre, así como la plataforma de desarrollo.

- **Metodología de desarrollo de software: Extreme Programming (XP).**

Basándose en la simplicidad, la comunicación y la reutilización del código surge la Programación Extrema (XP), siendo así una metodología ligera de desarrollo de software.

XP consiste en una programación extrema (rápida). Como requisito para alcanzar el éxito del proyecto se tiene al usuario final como parte del equipo. Es una metodología con reconocido éxito y se utiliza en proyectos con entregas a cortos plazos. [13]

Las características fundamentales de XP son:

- ✓ Desarrollo iterativo e incremental: lo que permite pequeñas mejoras, unas tras otras consecutivamente.
- ✓ Pruebas unitarias: son continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión.
- ✓ Programación en parejas: recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Permitiendo que el código sea revisado al mismo tiempo que se escribe.
- ✓ Frecuente interacción del equipo de programación con el cliente o usuario. Recomendando que un representante del cliente trabaje junto al equipo de desarrollo.
- ✓ Corrección de todos los errores antes de añadir nueva funcionalidad haciendo entregas frecuentes.
- ✓ Refactorización del código: reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad.
- ✓ Propiedad del código compartida: no se dividen las responsabilidades en el desarrollo de cada módulo. Este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.

- ✓ Simplicidad en el código: es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

Lo esencial en el proceso de desarrollo es lograr la comunicación entre desarrolladores, la retroalimentación entre ellos y la simplicidad en el código. Teniendo en cuenta las particularidades de la herramienta a desarrollar se hace uso de la metodología XP con adaptaciones que respondan a determinadas características. Para lograr la solución se realiza un proceso completo sin iteraciones, utilizando un modelo de desarrollo en cascada. Otro elemento fundamental es el cliente, pues en este caso no forma parte del equipo de desarrollo.

Esta metodología propone cuatro fases, planificación, diseño, desarrollo y pruebas. Dado las características específicas de la herramienta no es necesario generar todos los artefactos de cada fase, por lo que solo se necesitan los requisitos funcionales y no funcionales, las historias de usuarios, las tarjetas CRC y las tareas de ingeniería.

- **Lenguaje de Programación: Java.**

Como cualquier lenguaje de programación, el lenguaje Java tiene su propia estructura, reglas de sintaxis y paradigma de programación. A continuación se listan las ventajas del lenguaje: [14]

- ✓ Manejo automático de la memoria, lo que ahorra significante tiempo de programación.
- ✓ Lenguaje Multi-plataforma: el código escrito en java es leído por un intérprete, por lo que su programa andará en cualquier plataforma.
- ✓ Programación Orientada a Objetos: paradigma muy utilizado que facilita y organiza mucho la programación.
- ✓ Sintaxis similar a C/C++, pero más simple.
- ✓ Fácil de aprender.

- **Entorno de desarrollo: Netbeans 8.0**

NetBeans IDE es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso. [15]

Las ventajas que ofrece Netbeans se muestran a continuación: [15]

- ✓ Soporta el desarrollo de cualquier tipo de aplicación.
- ✓ Reutilización de módulos.
- ✓ Permite el uso de la herramienta Update Center Module.

- ✓ Instalación y actualización simple.
- ✓ Incluye Templates y Wizards.
- ✓ Posee soporte para Php.

- **Sistema Operativo: Ubuntu 14.04**

Ubuntu 14.04, es un software libre de código abierto. El escritorio predeterminado en Ubuntu es GNOME, líder como escritorio y como plataforma de desarrollo. Ubuntu proporciona un sistema basado en Debian, con frecuentes lanzamientos basados en el tiempo, responsabilidad empresarial, y una interfaz de escritorio más considerada. Ubuntu concentra su objetivo en facilidad de uso, la libertad de uso, lanzamientos regulares, la facilidad en la instalación y proporciona un entorno robusto. [16]

- **Lenguaje de modelado: UML**

El lenguaje unificado de modelación (UML, por sus siglas en inglés, Unified Modeling Language) prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos. Es capaz de describir la semántica general de los diagramas y los significados de los símbolos utilizados. Actualmente es el más conocido y utilizado. UML desarrolla un lenguaje gráfico para visualizar, especificar y documentar las variadas partes que incluye el desarrollo de software. [17]

Las principales características de UML son: [17]

- ✓ Tecnología de orientación a objetos.
- ✓ Viabilidad en la corrección de errores.
- ✓ Desarrollo incremental e iterativo.
- ✓ Mejor comunicación con el cliente.

- **Herramienta de Modelado: Visual Paradigm for UML 8.0 Enterprise Edition.**

Visual Paradigm for UML Enterprise Edition (VP-UML EE) es una herramienta que permite el modelado de datos orientado a objetos, hace posible la documentación del proyecto y el mapeo relacional de objetos para Java, .NET y PHP, además permite al equipo de desarrollo de software realizar el análisis y diseño de los sistemas con mayor eficacia, reduciendo costos y aumentando su productividad. Como lenguaje de modelado utiliza UML. [18]

1.7. Conclusiones parciales.

- Del estudio realizado sobre el rendimiento de bases de datos en PostgreSQL, se puede concluir que su correcto desempeño depende en gran medida del seguimiento que realicen los

administradores de bases de datos a las estadísticas de su funcionamiento. El seguimiento debe ser realizado a través de constantes monitoreos a su funcionamiento con las herramientas existentes, lo que permita tomar las decisiones adecuadas en el momento indicado.

- Las herramientas que existen para el monitoreo de un servidor de bases de datos PostgreSQL, proporcionan toda la información asociada a su rendimiento de manera fragmentada. Así como la manera en que se visualizan las informaciones a los usuarios dificulta su entendimiento.
- Además se describieron las principales características del entorno de trabajo seleccionado para el desarrollo del componente de monitoreo para PostgreSQL. Este entorno se encuentra compuesto por XP como metodología de desarrollo y UML como lenguaje de modelado. Se seleccionó también el Visual Paradigm para desarrollar los diagramas UML, PostgreSQL como gestor de bases de datos y Java como lenguaje de programación para la implementación del sistema.

Capítulo 2: Descripción de la solución.

Introducción.

En el presente capítulo se describe la herramienta propuesta en la investigación para obtener las estadísticas del rendimiento del SGBD PostgreSQL. También se describen los artefactos generados en el proceso de desarrollo siguiendo la metodología seleccionada en el presente trabajo, además se especifican los requisitos funcionales y no funcionales de la herramienta, las historias de usuarios, tarjetas CRC y el uso de patrones de diseño en la solución propuesta.

2.1. Descripción de la herramienta MaGeoLoDi.

El componente a desarrollar es el encargado de la obtención de estadísticas referentes al rendimiento de un SGBD PostgreSQL. El administrador debe establecer la conexión con el servidor que desea monitorear, introduciendo los campos servidor, puerto, usuario y contraseña o seleccionando una conexión guardada donde solamente tendrá que introducir la contraseña. Una vez establecida la conexión se muestra la interfaz principal a través de la cual se pueden visualizar las estadísticas referentes a los índices, triggers, secuencias, tabla-tupla, tabla-bloque, tabla-vaccum, disco duro, transacción de bloques, tiempo temporal de las bases de datos, conflictos, bgwriter, replicaciones, la actividad en servidor y los usuarios con permiso de autenticación en este.

En otra de las interfaces se pueden obtener estadísticas acerca de los procesos activos de PostgreSQL, reportes estadísticos del procesador y de la memoria virtual del sistema. Las estadísticas se encuentran agrupadas por secciones según el tipo de información de forma tal que permita un mayor entendimiento. Para realizar un mejor análisis de forma tal que puedan realizarse comparaciones entre los datos mostrados con mayor facilidad, las informaciones son visualizadas a través de tablas, gráficos de barras y circulares. Además para contribuir a una mayor legibilidad de los resultados mostrados el componente cuenta con leyendas asociadas a las siglas de los atributos[3].

La principal ventaja de la solución propuesta consiste en la visualización de la información de forma integrada, ya que brinda la posibilidad de integrar un conjunto de estadísticas en la misma pantalla, sin necesidad de ejecutar comandos en terminales así como consultas sql[4] en un servidor de base de datos PostgreSQL. Además con la utilización de esta herramienta los administradores de un SGBD PostgreSQL logran un significativo ahorro de tiempo para obtener las estadísticas del rendimiento del servidor, lo que permite agilizar la toma decisiones. Su mayor demora estaría asociada al

[3] Atributos: nombre de las columnas de cada tabla.
[4] sql: lenguaje de consulta estructurado.

establecimiento de una conexión a un servidor de base de datos, lo cual se realiza en un segundo por lo que se considera un tiempo insignificante respecto al tiempo que se toma la generación de la obtención de cada una de las estadísticas a través de la ejecución de comandos en terminales y consultas sql u otras de las herramientas existentes. La herramienta además permite establecer la conexión con un servidor tanto de forma local como remota.

2.2. Requisitos de software.

Una de las principales tareas en el ciclo de desarrollo de un software es la de determinar los requisitos del sistema a desarrollar (las condiciones o capacidades que el sistema debe cumplir). Los requisitos deben definirse con claridad y sin ambigüedades, así como lograr que el desarrollador conozca sobre qué debe y qué no debe hacer el sistema. Los requisitos se clasifican en funcionales y no funcionales.

2.2.1. Requisitos funcionales.

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Su meta principal es identificar y documentar las acciones que debe ejecutar el sistema para que cumpla con los objetivos propuestos [19]. De acuerdo a los objetivos planteados en esta investigación el sistema a desarrollar debe ser capaz de:

1- Establecer conexión a un servidor de base de datos PostgreSQL.
2- Cerrar conexión a un servidor de base de datos PostgreSQL.
3- Crear ficheros para guardar conexiones establecidas.
4- Capturar el uso del espacio en disco duro.
5- Capturar el uso de las base de datos.
6- Capturar el uso de los índices en una base de datos.
7- Capturar el uso de las secuencias en una base de datos.
8- Capturar el uso de los triggers en una base de datos.
9- Capturar el uso de las tuplas de las tablas de una base de datos.
10- Capturar el uso de los bloques de las tablas de una base de datos.
11- Capturar el uso de los vacuum de las tablas de una base de datos.
12- Capturar el uso de las transacciones-bloques de las bases de datos en el servidor.
13- Capturar el uso del tiempo-temporal de las bases de datos en el servidor.
14- Capturar el uso de los conflictos de las bases de datos en el servidor.
15- Capturar el uso de las replicaciones en el servidor.
16- Capturar el uso de las actividades del sistema en el servidor.
17- Capturar el uso del Bgwriter en el servidor.

18- Capturar usuarios autenticados en el servidor.
19- Capturar el uso de los procesos activos de PostgreSQL.
20- Capturar el uso del procesador.
21- Capturar el uso de la memoria virtual del sistema.
22- Mostrar gráfico de la cantidad de índices por tabla.
23- Mostrar gráfico de la cantidad de scan por índices.
24- Mostrar gráfico de la cantidad de tuplas leídas por índices.
25- Mostrar gráfico de la cantidad de tuplas devueltas por índices.
26- Mostrar gráfico de la cantidad de bloques del disco leídos desde índices.
27- Mostrar gráfico de la cantidad de accesos al buffer en índices.
28- Mostrar gráfico de la cantidad de bloques del disco leídos desde secuencias.
29- Mostrar gráfico de la cantidad de accesos al buffer leídos en secuencias.
30- Mostrar gráfico de los tipos de manipulación de eventos de los triggers.
31- Mostrar gráfico de los tipos de sincronización de eventos de los triggers.
32- Mostrar gráfico de la cantidad de trigger por tabla.
33- Mostrar gráfico de la cantidad de tuplas insertadas por tablas.
34- Mostrar gráfico de la cantidad de tuplas actualizadas por tablas.
35- Mostrar gráfico de la cantidad de tuplas eliminadas por tablas.
36- Mostrar gráfico de la cantidad de tuplas vivas por tablas.
37- Mostrar gráfico de la cantidad de tuplas muertas por tablas.
38- Mostrar gráfico de la cantidad de tuplas HOT actualizadas por tablas.
39- Mostrar gráfico de la cantidad de escaneos secuenciales por tablas.
40- Mostrar gráfico de la cantidad de tuplas vivas leídas por escaneo secuencial en tablas.
41- Mostrar gráfico de la cantidad de tuplas vivas leídas por escaneo de índices por tablas.
42- Mostrar gráfico de la cantidad de escaneos de índices por tabla.
43- Mostrar gráfico de la cantidad de bloques del disco leídos por tablas.
44- Mostrar gráfico de la cantidad de accesos al buffer por tablas.
45- Mostrar gráfico de la cantidad de bloques del disco leídos en todos los índices por tablas.
46- Mostrar gráfico de la cantidad de accesos al buffer en todos los índices por tablas.
47- Mostrar gráfico de la cantidad de vacuum realizados manualmente por tablas.
48- Mostrar gráfico de la cantidad de vacuums realizados por autovacuum deamon por tablas.
49- Mostrar gráfico de la cantidad de análisis realizados manualmente por tablas.
50- Mostrar gráfico de la cantidad de análisis realizados por autovacuum deamon por tablas.
51- Mostrar gráfico de la cantidad de backends actualmente conectados por base de datos.
52- Mostrar gráfico de la cantidad de transacciones cometidas por base de datos.

53- Mostrar gráfico de la cantidad de transacciones revertidas por base de datos.
54- Mostrar gráfico de la cantidad de bloques del disco leídos por base de datos.
55- Mostrar gráfico de la cantidad de veces que los bloques del disco se encontraron en la caché del buffer por base de datos.
56- Mostrar gráfico de la cantidad de consultas canceladas debido a conflictos con la recuperación por base de datos.
57- Mostrar gráfico de la cantidad de archivos temporales creados por consultas por base de datos.
58- Mostrar gráfico del importe total de los datos escritos en los archivos temporales por las consultas por base de datos.
59- Mostrar gráfico de la cantidad de interbloqueos detectados por base de datos.
60- Mostrar gráfico del tiempo dedicado a la lectura de bloques de archivos de datos del backend (en milisegundos) por base de datos.
61- Mostrar gráfico del tiempo dedicado a escribir bloques de archivos de datos del backend (en milisegundos) por base de datos.
62- Mostrar gráfico de la cantidad de consultas canceladas debido a la reducción de espacio de tabla por base de datos.
63- Mostrar gráfico de la cantidad de consultas canceladas debido al tiempo de espera de bloqueo por base de datos.
64- Mostrar gráfico de la cantidad de consultas canceladas debido a las instantáneas antiguas por base de datos.
65- Mostrar gráfico de la cantidad de consultas canceladas debido a buffers cubiertos por base de datos.
66- Mostrar gráfico de la cantidad de consultas canceladas debido a los puntos muertos por base de datos.
67- Mostrar gráfico de la cantidad de bases de datos conectados al backend.
68- Mostrar gráfico de la cantidad de usuarios conectados al backend.
69- Mostrar gráfico de cantidad de aplicaciones conectadas al backend.
70- Mostrar gráfico de la cantidad de direcciones IP conectadas al backend.
71- Mostrar gráfico de la cantidad de hosts conectadas al backend.
72- Mostrar gráfico de la cantidad de puertos TCP conectadas al backend.
73- Mostrar gráfico de la cantidad de privilegios por tipos.

2.2.2. Requisitos no funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en las propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Los requisitos no funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto y de esta forma poder decir si el producto tiene la calidad requerida. [19]

Requisitos de Hardware:

- ✓ Para el cliente (componente de monitoreo):
  - Microprocesador Pentium 4 o superior.
  - 512 Mega Bytes de RAM o superior.
  - 100 Mega Bytes de espacio libre en disco.
- ✓ Para el servidor (ninguno):
  - El componente de monitoreo no requiere de ningún servidor para su ejecución.

Requisitos de Software:

- ✓ Para el cliente:
  - Sistema gestor de bases de datos PostgreSQL.
  - Se aconseja tener instalado como sistema operativo Ubuntu 10.04 o superior.
  - Instalados los comandos who, ps, mpstat y vmstat para obtener datos del sistema operativo.
  - Instalada la máquina virtual de java (jdk8) para Linux o superior.
  - El usuario debe tener permiso de súper usuario para ejecutar consultas al catálogo.
  - El SGBD PostgreSQL debe permitir conexiones externas.
- ✓ Para el servidor (ninguno):
  - El componente de monitoreo no requiere de ningún servidor para su ejecución.

2.3. Fase de planificación de la metodología XP.

La metodología XP define como primera fase la planificación. En esta etapa se lleva a cabo el trabajo de identificación y descripción de las historias de usuario, así como la familiarización del equipo de trabajo con las tecnologías y herramientas seleccionadas para el desarrollo del software. También el equipo de desarrollo en conjunto con el cliente especifica la prioridad en que se deben implementar las historias de usuario, así como una estimación del esfuerzo que costará implementar cada una. El resultado de la fase es un plan de entregas donde se realiza una estimación de las versiones que tendrá el producto en su realización, de manera tal que guíe el desarrollo del mismo. [20]

2.3.1. Historias de usuarios.

La identificación de las historias de usuario (HU) es la técnica utilizada para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las HU es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. Se definen para el desarrollo de la herramienta de monitoreo 73 historias de usuarios.

A continuación se muestran dos de las HU de la solución propuesta, las demás se pueden encontrar en el Anexo 1.

Tabla 4: Historia de usuario número 1: Establecer conexión a un servidor de base de datos PostgreSQL.

Historia de usuario	
Número: 1	**Usuario:** usuario.
Nombre: Establecer conexión a un servidor de base de datos PostgreSQL.	
Prioridad en negocio: Alta.	**Puntos estimados:** 2 días.
Riesgo en desarrollo: Medio.	**Iteración asignada:** 1
Descripción: Permite establecer la conexión al servidor de base de datos PostgreSQL. Para realizar la conexión es necesario llenar una serie de parámetros, entre los que se encuentran, nombre del servidor a monitorear, puerto de escucha, usuario y contraseña del administrador del servidor de base de datos PostgreSQL.	
Observaciones:	

Tabla 5: Historia de usuario número 5: Capturar uso de los índices en una base de datos.

Historia de usuario	
Número: 5	**Usuario:** usuario.
Nombre: Capturar uso de los índices en una base de datos.	
Prioridad en negocio: Alta.	**Puntos estimados:** 2 días.
Riesgo en desarrollo: Medio.	**Iteración asignada:** 1
Descripción:	
Observaciones: Permite capturar los valores del uso de los índices, teniendo en cuenta el nombre de la tabla indexada, nombre del índice, cantidad de scan con un índice, cantidad de tuplas leídas con un índice, cantidad de tuplas devueltas, número de bloques del disco leídos por un índice, accesos al buffer por un índice y definición del índice. Todos los valores son extraídos de una base de datos seleccionada.	

2.4. Fase de diseño de la metodología XP.

La metodología de desarrollo XP plantea prácticas especializadas que reaccionan directamente en la realización del diseño para lograr un sistema robusto y reutilizable, tratando de conservar en todo momento su simplicidad, es decir, crear un diseño evolutivo que va mejorando incrementalmente y que permite hacer entregas pequeñas y frecuentes de valor para el cliente. A la hora de darle cumplimiento a la actividad de diseñar, XP no especifica ninguna técnica de modelado. Pueden utilizarse indistintamente sencillos esquemas en una pizarra, diagramas de clases utilizando UML o tarjetas CRC (Clase, Responsabilidad y Colaboración) siempre que sean útiles, tributen a la comprensión y no requieran mucho tiempo en su creación.

2.4.1. Descripción de la arquitectura de software.

El patrón de arquitectura de software empleado para el desarrollo de la herramienta es el Modelo-Vista-Controlador (MVC). Este separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información y por otro lado para la interacción del usuario. Este patrón de diseño se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

De manera genérica, el patrón MVC está formado por tres niveles:

- ✓ El modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio. Integra las clases que contienen las consultas a la base de datos, así como la definición de las tablas, sus relaciones y atributos.
- ✓ La vista transforma el modelo en una interfaz que se muestra finalmente al cliente para interactuar con la aplicación.
- ✓ El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista. Gestiona todas las peticiones del usuario y se encarga de darle respuesta.

En la Figura 7 se evidencia la utilización del patrón Modelo-Vista-Controlador (MVC) para la obtención de las estadísticas asociadas a los procesos activos de PostgreSQL, reportes estadísticos del procesador y de la memoria virtual del sistema.

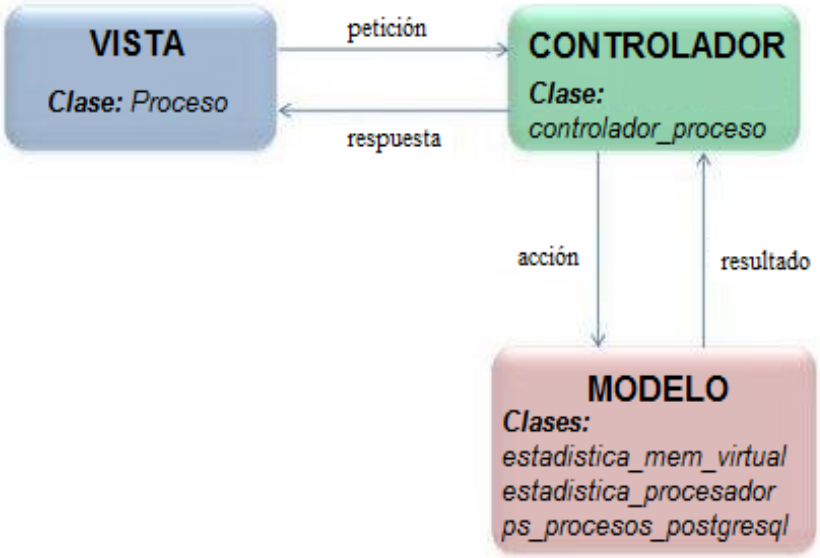

Figura 7: Patrón Modelo-Vista-Controlador.

2.4.2. Diagrama de clases persistente.

Para el diseño de las aplicaciones, la metodología XP no requiere la presentación del sistema mediante diagramas de clases utilizando notación UML, en su lugar se usan otras técnicas como las tarjetas CRC. No obstante, el uso de diagramas puede aplicarse siempre, no como un artefacto de la metodología, pero se pueden utilizar siempre y cuando influyan en el mejoramiento de la comunicación, no sea un peso su mantenimiento, no sean extensos y se enfoquen en la información importante. A continuación, en la Figura 8, se representa el diagrama de clases del diseño definido para el desarrollo de la herramienta de monitoreo, apoyándose en UML como lenguaje de modelado para facilitar el entendimiento de la herramienta a desarrollar.

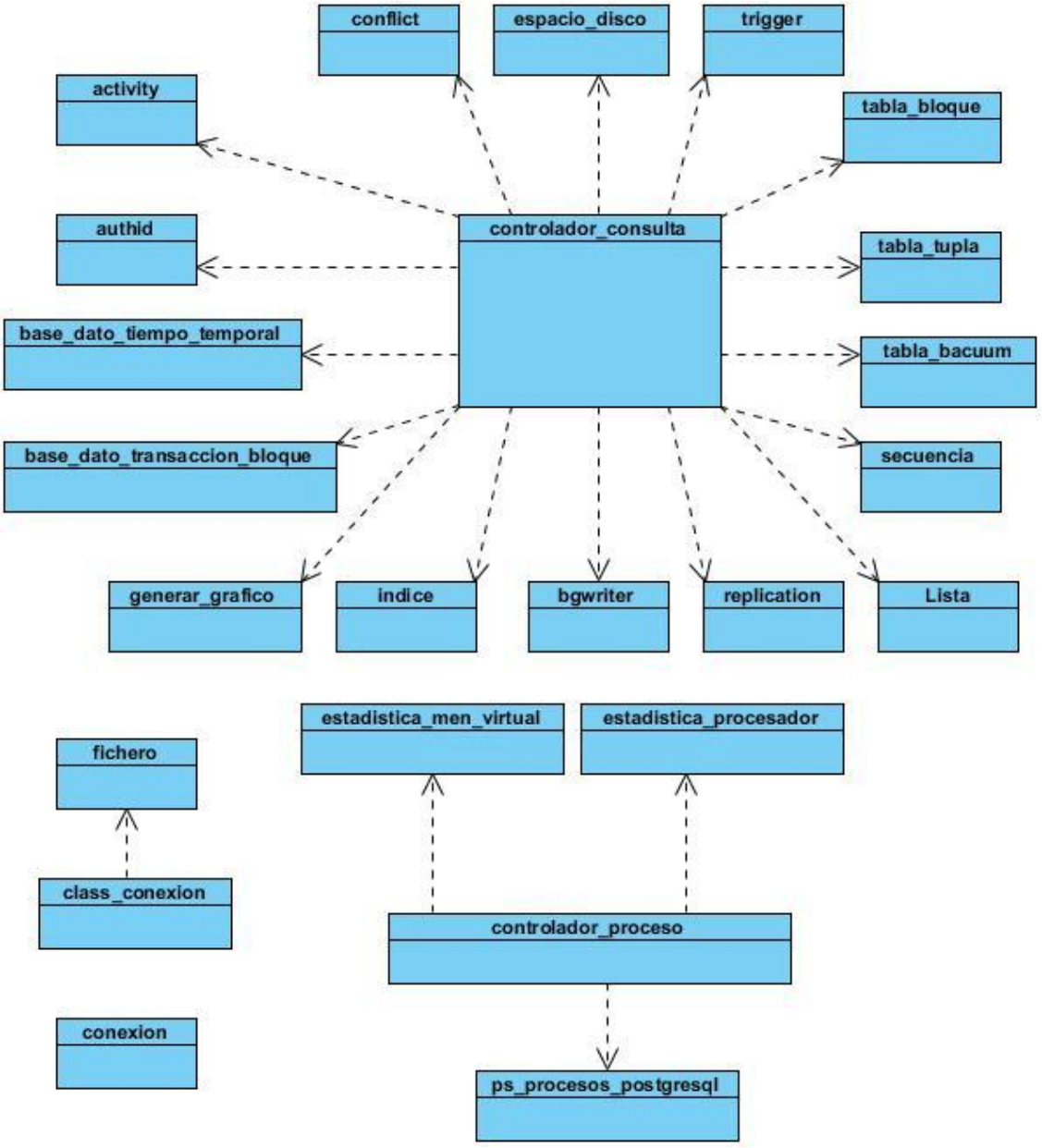

Figura 8: Diagrama de clases del diseño.

2.4.3. Tarjetas CRC.

La metodología XP, como se ha mencionado anteriormente en lugar de utilizar diagramas para desarrollar modelos representa las clases mediante tarjetas. Las tarjetas CRC son utilizadas para representar las responsabilidades de las clases y sus interacciones. Además, permiten trabajar con una metodología basada en objetos, permitiendo que el equipo de desarrollo completo contribuya en la fase del diseño. El nombre de la clase se coloca a modo de título de la tarjeta, las responsabilidades se colocan a la izquierda y las clases que se implican en cada responsabilidad a la derecha, en la misma línea que su requisito correspondiente. Las tarjetas determinan el

comportamiento de cada actividad. En el trabajo realizado se definieron un total de 27 tarjetas CRC. A continuación se muestran las tarjetas controlador_consulta y controlador_proceso de la solución propuesta, las demás se pueden encontrar en el Anexo 2.

Tabla 6: Tarjeta CRC controlador_consulta.

Tarjeta CRC	
Clase: controlador_consulta	
Responsabilidades:	**Colaboraciones:**
Mostrar el espacio usado y libre del disco duro.	espacio_disco
Mostrar el espacio usado por el servidor PostgreSQL en el disco duro.	
Mostrar lista de índices.	índice
Adicionar índice.	
Eliminar datos de la lista de índices.	
Mostrar lista de secuencias.	secuencia
Adicionar secuencia.	
Eliminar datos de la lista de secuencias.	
Mostrar lista de triggers.	trigger
Adicionar trigger.	
Eliminar datos de la lista de triggers.	
Mostrar lista de tabla-tupla.	tabla_tupla
Adicionar tabla-tupla.	
Eliminar datos de la lista de tabla-tupla.	
Mostrar lista de tabla-bloque.	tabla_bloque
Adicionar tabla-bloque.	
Eliminar datos de la lista de tabla-bloque.	
Mostrar lista de tabla-vacuum.	tabla_vacuum
Adicionar tabla-vacuum.	
Eliminar datos de la lista de tabla-vaccum.	
Mostrar lista del tiempo temporal de la base de datos.	base_dato_tiempo_temporal
Adicionar tiempo temporal de las base de datos.	
Eliminar datos de la lista tiempo temporal de la base de datos.	
Mostrar lista de la transacción de bloques de las base de datos.	base_dato_transaccion_bloque
Adicionar transacción de bloques de la base de datos.	

Eliminar datos de la lista de transacción de bloques de las base de datos.	
Mostrar lista de conflictos.	conflict
Adicionar conflicto.	
Eliminar datos de la lista de conflictos.	
Mostrar lista de replicaciones.	replication
Adicionar replicación.	
Eliminar datos de la lista de replicaciones.	
Mostrar lista de Bgwriter.	bgwriter
Adicionar Bgwriter.	
Eliminar datos de la lista Bgwriter.	
Mostrar lista de las actividades del sistema.	activity
Adicionar actividad del sistema.	
Eliminar datos de la lista de actividades del sistema.	
Mostrar lista de usuarios autenticados.	authid
Adicionar usuario autenticado.	
Eliminar datos de la lista de usuarios autenticados.	

Tabla 7: Tarjeta CRC controlador_proceso.

Tarjeta CRC	
Clase: controlador_proceso	
Responsabilidades:	**Colaboraciones:**
Mostrar lista de procesos activos de PostgreSQL.	ps_proceso_postgresql
Adicionar proceso activo de PostgreSQL.	
Modificar proceso activo de PostgreSQL.	
Eliminar datos de la lista de procesos activos de PostgreSQL.	
Mostrar lista de estadísticas del procesador.	estadistica_procesador
Adicionar estadística del procesador.	
Modificar estadística del procesador.	
Eliminar datos de la lista de estadísticas del procesador.	
Mostrar lista de estadísticas de la memoria virtual.	estaditica_mem_virtual
Adicionar estadística de la memoria virtual.	
Modificar estadística de la memoria virtual.	

Eliminar datos de la lista de estadísticas de la memoria virtual.	

2.4.4. Patrones de diseño.

"Un patrón es una descripción de un problema y la solución a la que se da un nombre, y que se puede aplicar a nuevos contextos; idealmente, proporciona consejos sobre el modo de aplicarlo en varias circunstancias" [21].

Durante el diseño del software pueden utilizarse los patrones de diseño. Una vez que se ha desarrollado el modelo de análisis, el diseñador puede examinar una representación detallada del problema que debe resolver y las restricciones que impone el problema. [22]

Patrones generales de software para asignación de responsabilidades (GRASP)[5]

Los GRASP son patrones para la asignación de responsabilidades, que ayudan a entender el diseño de objetos. Incluso cuando se utilizan metodologías ágiles como XP, es necesario elegir cuidadosamente las clases adecuadas y decidir cómo deben interactuar [21]. Es por ello que resulta de vital importancia el uso de patrones GRASP.

Durante la elaboración de las clases del diseño surgieron diferentes problemas a la hora de representar sus relaciones, por lo que se hizo necesario el uso de los patrones de asignación de responsabilidades para darle solución. A continuación se muestran los problemas surgidos y los patrones empleados para darle solución a los mismos.

- ✓ Experto: para elegir el principio fundamental para realizar la asignación de las responsabilidades a los objetos se emplea el patrón experto. Con este patrón se garantiza que una clase contenga toda la información necesaria para realizar la labor que tiene encomendada. Este patrón lo podemos ver claramente evidenciado en la clase "conexion" de la solución propuesta.
- ✓ Creador: para definir el responsable de crear una nueva instancia de alguna clase se emplea el patrón creador para darle solución. Con este patrón se crean las instancias de una clase guiando la asignación de responsabilidades relacionadas con la creación de objetos. Este patrón se ve en las clases "controlador_consulta" y "controlador_proceso" de la solución, pues en estas se realizan instancias de objetos para ser creados.
- ✓ Alta cohesión: para mantener la complejidad manejable de una clase se emplea el patrón alta cohesión. El empleo de este patrón proporciona refactorización en el sistema ya que permite que cada elemento del diseño realice una labor única. Está evidenciado en la solución propuesta pues cada clase realiza solamente las funciones para las que fueron diseñadas y no para

[5]GRASP: General Responsibility Assignment Software Patterns.

funciones ajenas a estas. Un ejemplo del uso de este patrón en la solución propuesta se encuentra en las clases “fichero” y “generar_grafico”.

- ✓ Controlador: para definir la clase encargada de atender un evento del sistema se emplea el patrón controlador. Para darle solución plantea que la responsabilidad de controlar el flujo de eventos del sistema se debe asignar a clases específicas facilitando la centralización de actividades en el sistema. El siguiente patrón se evidencia en las clases “controlador_consulta” y “controlador_proceso”, pues estas son las encargadas de controlar la mayoría de las operaciones realizadas en la solución propuesta.

2.5. Fase de desarrollo de la metodología XP.

La etapa de desarrollo del software es lo que constituye la fase principal de la metodología XP, en esta fase se obtiene una primera versión del producto deseado por el cliente. Además se propone un conjunto de prácticas que sirven para el desarrollo exitoso del mismo. Al inicio se crean tareas de ingeniería para ayudar a organizar la implementación exitosa de las HU.

2.5.1. Tareas de ingeniería.

Para lograr mejor entendimiento de las HU, son creadas las tareas de ingeniería, estas no son más que pasos lógicos a seguir por el programador para realizar la implementación de una HU. A continuación se muestran algunas de las tareas a desarrollar por cada HU, las demás se pueden encontrar en el Anexo 3.

Tabla 8: Tarea de ingeniería 1: Establecer conexión a un servidor de base de datos PostgreSQL.

Tarea de ingeniería	
Número de tarea: 1	**Nombre de la historia de usuario:** Establecer conexión a un servidor de base de datos PostgreSQL.
Nombre de la tarea: Establecer conexión a un servidor de base de datos PostgreSQL.	
Tipo de tarea: Desarrollo	**Puntos estimados (días):** 2 días.
Fecha inicio: 14/01/2015	**Fecha fin:** 15/01/2015
Programador responsable: Yariel A. Vivero Díaz.	
Descripción: Esta tarea facilita la creación de una interfaz que le permite al usuario establecer la conexión con el servidor de base de datos PostgreSQL introduciendo usuario, contraseña, puerto y nombre del servidor. Luego de establecer la conexión se muestra la interfaz principal con las estadísticas.	

Tabla 9: Tarea de ingeniería 2: Crear ficheros para guardar conexiones establecidas.

Tarea de ingeniería	
Número de tarea: 2	**Nombre de la historia de usuario:** Crear ficheros para guardar conexiones establecidas.
Nombre de la tarea: Crear ficheros para guardar conexiones establecidas.	
Tipo de tarea: Desarrollo	**Puntos estimados (días):** 1 día.
Fecha inicio: 16/01/2015	**Fecha fin:** 16/01/2015
Programador responsable: Anaily Navia López.	
Descripción: Esta tarea facilita al usuario la creación de un fichero para almacenar la dirección donde se guardará un segundo fichero que contiene los datos (servidor, puerto y usuario) de las conexiones establecidas.	

Tabla 10: Tarea de ingeniería 5: Capturar uso de los índices en una base de datos.

Tarea de ingeniería	
Número de tarea: 5	**Nombre de la historia de usuario:** Capturar uso de los índices en una base de datos.
Nombre de la tarea: Capturar uso de los índices en una base de datos.	
Tipo de tarea: Desarrollo	**Puntos estimados (días):** 2 días.
Fecha inicio: 23/01/2015	**Fecha fin:** 26/01/2015
Programador responsable: Yariel A. Vivero Díaz.	
Descripción: Esta tarea facilita la creación de una interfaz que permite capturar y mostrarle al usuario los datos del uso de los índices. Los datos son mostrados en una tabla luego de seleccionada una base de datos. En la parte inferior de la tabla se representa con una leyenda el significado de los datos mostrados con siglas.	

Tabla 11: Tarea de ingeniería 59: Mostrar gráfico de la cantidad de consultas canceladas debido a la reducción de espacio de tabla por base de datos.

Tarea de ingeniería	
Número de tarea: 59	**Nombre de la historia de usuario:** Mostrar gráfico de la cantidad de consultas canceladas debido a la reducción de espacio de tabla por base de datos.
Nombre de la tarea: Mostrar gráfico de la cantidad de consultas canceladas debido a la	

reducción de espacio de tabla por base de datos.	
Tipo de tarea: Desarrollo	**Puntos estimados (días):** 1 día.
Fecha inicio: 30/04/2015	**Fecha fin:** 30/04/2015
Programador responsable: Yariel A. Vivero Díaz.	
Descripción: Esta tarea facilita la creación de un gráfico de barra y uno circular, los cuales representan la cantidad de consultas canceladas debido a la reducción de espacio de tabla por base de datos. Cada color representa una base de datos y el valor depende de la cantidad de consultas canceladas debido a la reducción de espacio de tabla por cada una.	

2.6. Conclusiones parciales.

- En el presente capítulo se definieron los requisitos funcionales y no funcionales, lo que permitió conocer las acciones y cualidades que el sistema debía cumplir y tener. Además se crearon las HU, permitiendo especificar los requisitos de software anteriormente definidos.
- Teniendo en cuenta los requisitos definidos para la solución se definió la arquitectura del sistema a desarrollar, la cual se regirá por el patrón MVC. Para esto se generaron las tarjetas CRC donde se reflejan las diferentes clases que intervendrán en la herramienta con sus responsabilidades e interacciones, por lo que se sentaron las bases para comenzar con la fase de desarrollo.
- Con el uso de patrones de diseño se logró asignar correctamente las responsabilidades a las clases definidas para la solución propuesta en esta investigación. Asimismo permitió la creación de un código más fácil de comprender, mantener y extender.
- Además para guiar la implementación de la solución propuesta se crearon las tareas de ingeniería y el diagrama de clases del diseño.
- La herramienta desarrollada permite visualizar los datos de forma integrada, con un ahorro significativo de tiempo en la obtención de los datos, así como una mejor visualización de las estadísticas a través del uso de los gráficos de barra y circulares para su análisis, además de contribuir a un mayor entendimiento de los atributos a través de una leyenda con el significado de las siglas y lograr el establecimiento de la conexión a un servidor de base de datos PostgreSQL de forma remota.

Capítulo 3: Validación de la solución.

Introducción.

En el presente capítulo se realiza la validación de la solución propuesta. Por tal motivo fueron empleadas diferentes técnicas para realizar validaciones, de manera que se pueda comprobar la eficiencia de las clases u operaciones utilizadas para dar respuesta a las necesidades de los clientes. Además de validarse los requisitos, el diseño y finalmente la herramienta.

3.1. Validación de requisitos.

Los requisitos una vez definidos necesitan ser validados. Dicha validación tiene como misión demostrar que la definición de los requisitos define realmente el sistema que el usuario necesita o el cliente desea. Es necesario asegurar que el análisis realizado y los resultados obtenidos de la etapa de definición de requisitos son correctos. Durante el proceso se comprueba también que los requisitos de software sean consistentes, completos, precisos, sin ambigüedades y verificables. [23]

3.1.1. Métricas para la validación de requisitos.

Para la validación de los requisitos de la solución propuesta se decidió la utilización de tres métricas principales:

- ✓ Especificidad de los requisitos
- ✓ Estabilidad de los requisitos.
- ✓ Grado de validación de los requisitos.

3.1.1.1. Especificidad de los requisitos.

La especificidad de los requisitos fue medida usando la métrica de la siguiente forma [23]:

$$Q_1 = \frac{n_{ui}}{n_r} = \frac{73}{73} = 1$$

Donde:

Q1: especificidad de los requisitos.

Nr: total de requisitos definidos.

Nui: cantidad de requisitos para los que los desarrolladores no tuvieron interpretaciones idénticas.

Fueron definidos un total de 73 requisitos funcionales de los cuales los desarrolladores no encontraron interpretaciones idénticas. La métrica propone como valor máximo el 1 para asegurar que todos los requisitos definidos fueron especificados correctamente. Luego de aplicada la métrica se puede asegurar que la especificación de los requisitos no posee ambigüedad ya que se obtuvo el mayor valor que puede arrojar como resultado esta técnica. Esta característica asegura mayor calidad en el proceso de especificación.

3.1.1.2. Estabilidad de los requisitos.

La estabilidad de los requisitos fue medida usando la métrica de la siguiente forma [23]:

$$ETR = \left[\frac{RT - RM}{RT}\right] * 100 = \left[\frac{73 - 0}{73}\right] * 100 = 100$$

Donde:

ETR: estabilidad de los requisitos.

RT: total de requisitos definidos.

RM: cantidad de requisitos modificados.

Fueron identificados un total de 73 requisitos funcionales de los cuales ninguno fue modificado. La métrica ofrece valores entre 0 y 100, siendo el mejor valor de ETR el más cercano a 100, lo que representa que no se han realizado cambios sobre los requisitos, son estables y por tanto es confiable trabajar sobre su análisis y diseño. Luego de aplicada la métrica se obtuvo un valor de 100 % por lo que se llegó a la conclusión de que los requisitos son estables.

3.1.1.3. Grado de validación de los requisitos.

El grado de validación de los requisitos fue medido usando la métrica de la siguiente forma [23]:

$$VR = \frac{n_c}{(n_c + n_{nv})} = \frac{73}{(73 + 0)} = 1$$

Donde:

VR: grado de validación de los requisitos.

Nc: número de requisitos que se han validado como correctos.

Nnv: número de requisitos no validados aún.

Esta métrica propone como valor máximo el 1 para asegurar un alto nivel de corrección de los requisitos. De los 73 requisitos definidos todos fueron validados como correctos, por lo que se puede concluir que todos los requisitos definidos poseen valor máximo.

3.2. Validación del diseño.

La validación del diseño verifica su calidad y flexibilidad, lo que permite garantizar una buena base para la implementación. Lorenz y Kidd dividen las métricas basadas en clases en cuatro categorías: tamaño, herencia, valores internos y valores externos. Las métricas orientadas a tamaños para una clase se centran en cálculos de atributos y de operaciones, para luego promediar los valores para el sistema en su totalidad. Las métricas basadas en herencia se centran en la forma en que se reutilizan las operaciones en la jerarquía de clases. Las métricas para valores internos examinan la cohesión y asuntos relacionados con el código así como las métricas orientadas a valores externos examinan el acoplamiento y reutilización. [24]

Del conjunto de métricas planteadas por Lorenz y Kidd se aplicó al diseño propuesto:

- ✓ Tamaño Operacional de Clase.
- ✓ Relaciones entre clases.

3.2.1. Métrica tamaño operacional de clase (TOC).

Esta métrica es muy usada por diseñadores de software, el tamaño general de una clase puede medirse determinando las siguientes medidas:

- ✓ El total de operaciones (operaciones tanto heredadas como privadas de la instancia), que se encapsulan dentro de la clase.
- ✓ El número de atributos (atributos tanto heredados como privados de la instancia), encapsulados por clase.

Si la métrica TOC arroja grandes valores, indican que la clase debe tener bastante responsabilidad, lo cual reducirá la reusabilidad de la misma y complicará su implementación. Cuanto menor sea el valor promedio para el TOC, mayor la posibilidad de reutilizarse. La métrica TOC evalúa los atributos de Responsabilidad, Complejidad de Implementación y Reutilización de la siguiente manera:

Tabla 12: Atributos de calidad de la métrica TOC.

Atributos de calidad	Definición
Responsabilidad	Consiste en agregarle una responsabilidad a una clase en un marco de modelado de dominio.
Complejidad de implementación	Consiste en el grado de dificultad que tiene implementar un

	diseño de clases determinado.
Reutilización	Consiste en el grado de reutilización presente en una clase o estructura de clase.

La siguiente tabla muestra el rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC.

Tabla 13: Rango de valores para la evaluación técnica de los atributos de la métrica TOC.

Atributos	Categorías	Criterios
Responsabilidad	Baja	TOC< = Promedio
	Media	Promedio < TOC < 2* Promedio
	Alta	TOC > 2* Promedio
Complejidad de implementación	Baja	TOC <= Promedio
	Media	Promedio < TOC < 2* Promedio
	Alta	TOC > 2* Promedio
Reutilización	Baja	TOC > 2* Promedio
	Media	Promedio < TOC < 2* Promedio
	Alta	TOC <= Promedio

Al analizar los resultados obtenidos luego de aplicar la métrica TOC, los cuales se muestran en la Figura 9, se llega a la conclusión de que el diseño propuesto para el desarrollo de la herramienta cumple con los requerimientos para tener buena calidad. Los resultados arrojados demuestran que existe baja responsabilidad y complejidad por lo que se puede observar una alta reutilización, permitiendo que el sistema no tenga una compleja implementación, promoviendo así el bajo acoplamiento entre las clases.

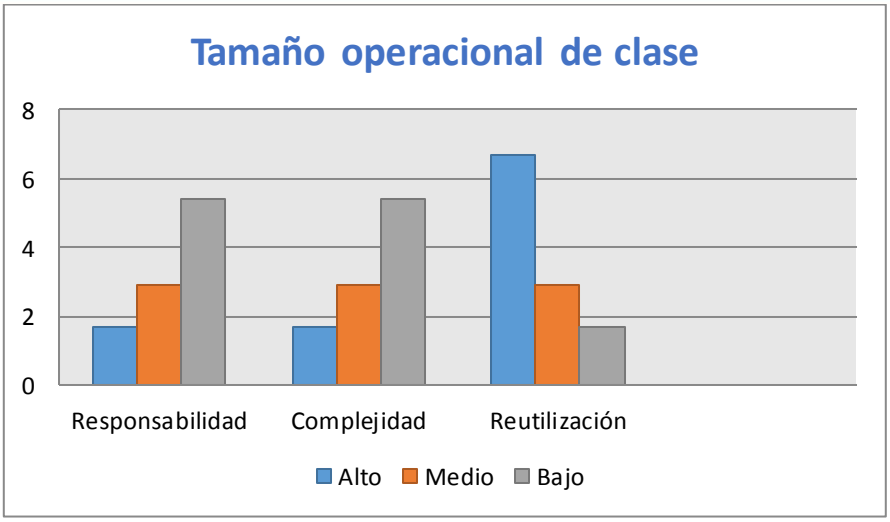

Figura 9: Gráfica de la métrica TOC.

3.2.2. Métrica relación entre clase (RC).

Esta métrica está dada por el número de relaciones de uso de una clase con otra. Para aplicar la métrica RC se tendrá en cuenta un conjunto de atributos de calidad como los que se muestran a continuación:

Tabla 14: Atributos de calidad de la métrica RC.

Atributos de calidad	Definición
Acoplamiento	Consiste en el grado de dependencia o interconexión de una clase con otras.
Complejidad de mantenimiento	Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software.
Reutilización	Consiste en el grado de reutilización presente en una clase o estructura de clase.
Cantidad de pruebas	Consiste en el número o grado de esfuerzo necesario para realizar las pruebas.

A continuación se muestra una tabla con el rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica RC.

Tabla 15: Rango de valores para la evaluación técnica de los atributos de la métrica RC.

Atributos	Categorías	Criterios
	Ninguno	RC=0

Acoplamiento	Bajo	RC=1
	Medio	RC=2
	Alto	RC>2
Complejidad de mantenimiento	Baja	RC <= Promedio
	Media	Promedio < RC < 2* Promedio
	Alta	RC > 2* Promedio
Reutilización	Baja	RC > 2* Promedio
	Media	Promedio < RC < 2* Promedio
	Alta	RC <= Promedio
Cantidad de pruebas	Baja	RC <= Promedio
	Media	Promedio < RC < 2* Promedio
	Alta	RC > 2* Promedio

De los resultados obtenidos durante la evaluación de la métrica RC, los cuales se pueden observar en la Figura 10, se puede decir que el diseño de la herramienta tiene una calidad aceptable. Los atributos de calidad fueron evaluados satisfactoriamente demostrando que las clases están diseñadas correctamente, ya que el 71% de las clases presentan bajo acoplamiento, el 92% representan baja complejidad de mantenimiento y de cantidad de pruebas, mientras que la reutilización generó un valor alto con 92%. Todos los resultados obtenidos sustentan la calidad del diseño ya que existe elevada reutilización, bajo acoplamiento, poca complejidad de mantenimiento y poca cantidad de pruebas.

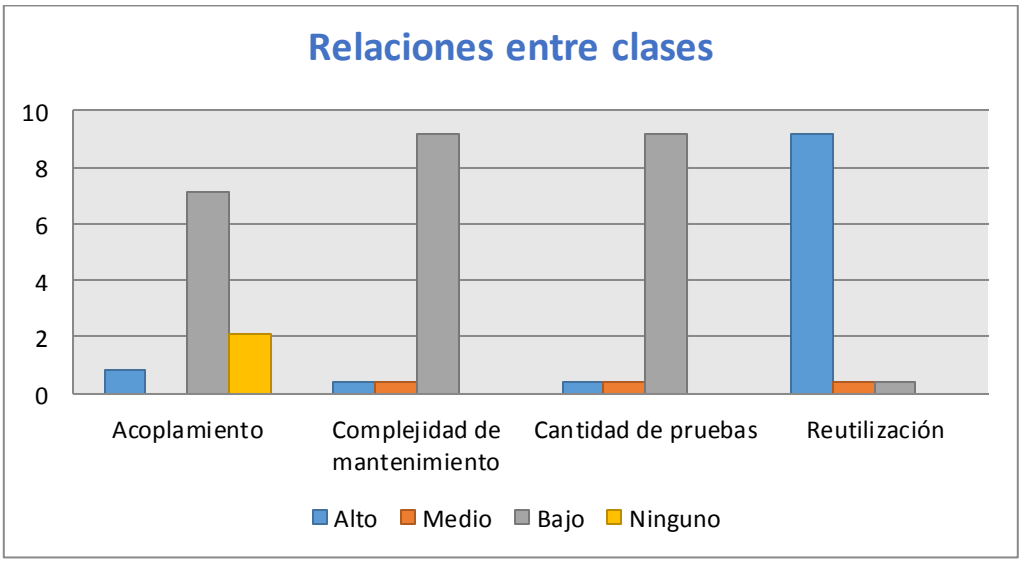

Figura 10: Gráfica de la métrica RC.

3.3. Validación de la solución.

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

3.3.1. Validación para establecer conexión.

Paso 1.- Al ejecutar la herramienta debe aparecer la interfaz que se muestra a continuación.

Figura 11: Interfaz para conectarse a un servidor PostgreSQL.

Paso 2.- Al introducir los datos; servidor, puerto, usuario y contraseña debe presionarse el botón “Conectarse”. Si los datos introducidos son válidos el sistema debe establecer la conexión con el servidor de base de datos PostgreSQL mostrando la siguiente interfaz principal, por lo que queda demostrado que la herramienta permite conectarse correctamente al servidor de base de datos especificado.

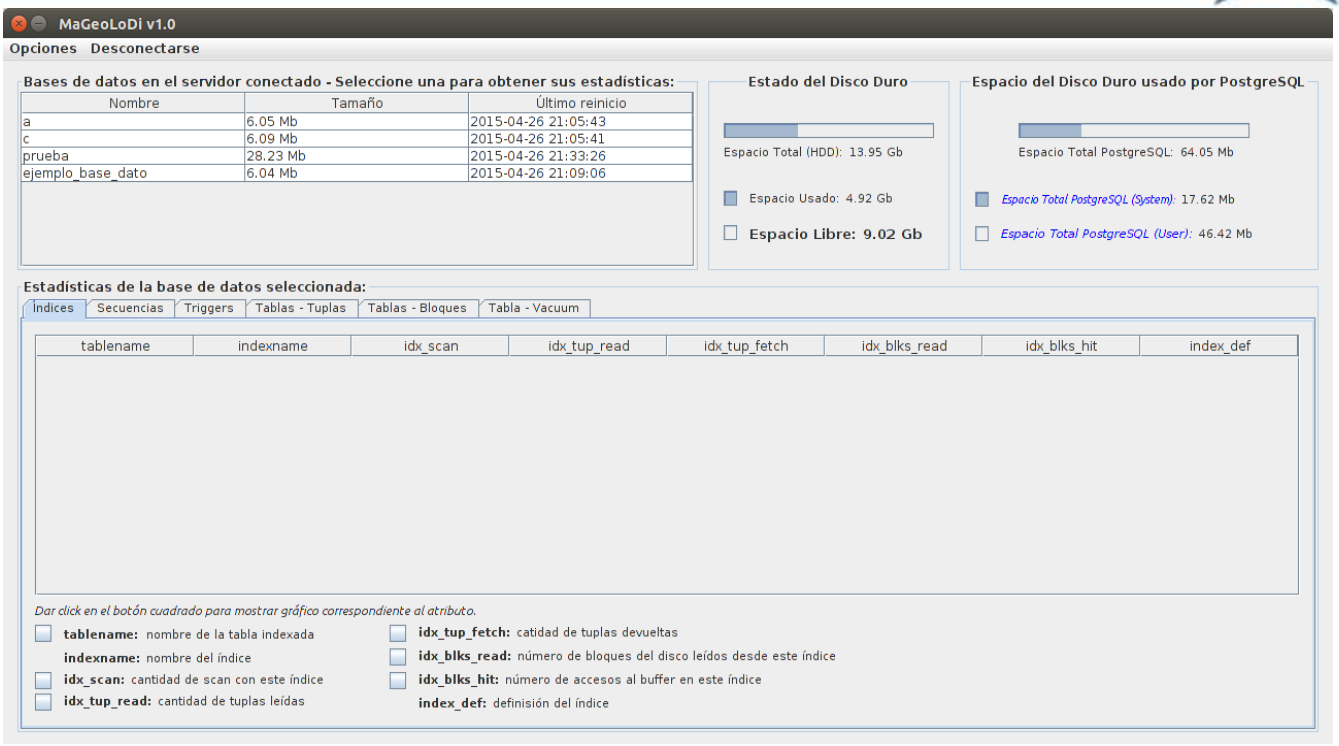

Figura 12: Interfaz principal.

3.3.2. Validación de estadísticas referentes a las Tablas - Tuplas.

Luego de establecida la conexión se realizan los siguientes pasos:

Paso 1.- Seleccionar la base de datos a la que se le van a obtener las estadísticas, en este caso fue seleccionada la base de datos *ejemplo_base_dato*. Para esto se debe mostrar la interfaz siguiente:

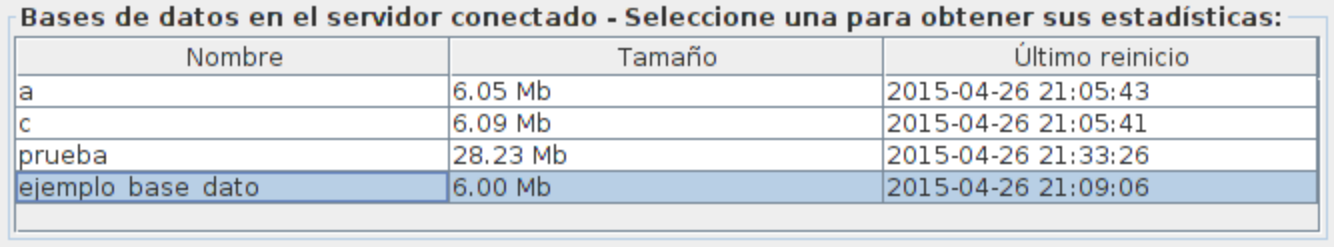

Figura 13: Interfaz para seleccionar una base de datos.

Paso 2.- Una vez seleccionada la base de datos, se elige el tipo de estadísticas, en este caso fue elegida la estadística referente a las *Tablas - Tuplas*. Para esto se debe mostrar la interfaz siguiente:

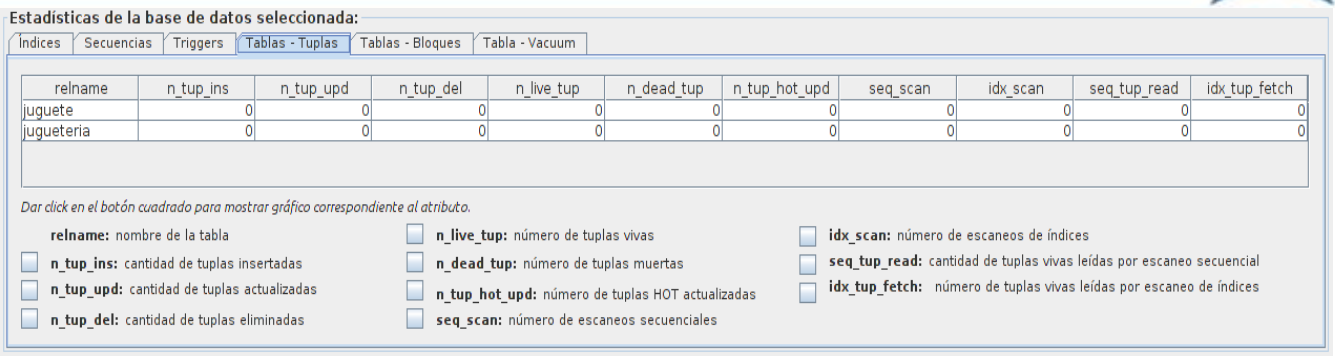

Figura 14: Interfaz que muestra las estadísticas referentes a las Tablas - Tuplas.

Paso 3.- Para comprobar que la herramienta muestra los datos actuales de las estadísticas y funciona correctamente, en la base de datos *ejemplo_base_dato* se ejecutan diferentes consultas que modifiquen los datos mostrados como se muestra a continuación:

```
INSERT INTO jugueteria(id, nombre) VALUES (1, 'Pinocho');
INSERT INTO jugueteria(id, nombre) VALUES (2, 'Juguetes voladores');
INSERT INTO jugueteria(id, nombre) VALUES (3, 'Princesas');

INSERT INTO juguete(id, nombre, tipo, id_jugeteria)  VALUES (1, 'Avion Galaxy', 'Volador', 2);

UPDATE jugueteria SET nombre='Carros' WHERE id=3;

DELETE FROM jugueteria WHERE id=1;
```

Figura 15: Consultas sql.

Paso 4.- Luego se vuelven a realizar los pasos 1 y 2 para analizar que los datos expuestos contengan las modificaciones realizadas.

En este caso se puede observar que los datos mostrados variaron, en un inicio estaban en cero todos los atributos y luego de realizadas las consultas aumentaron sus valores, señalados con un cuadro rojo como se muestra en la Figura 16, por lo que queda demostrado que la herramienta muestra la información actual y funciona correctamente.

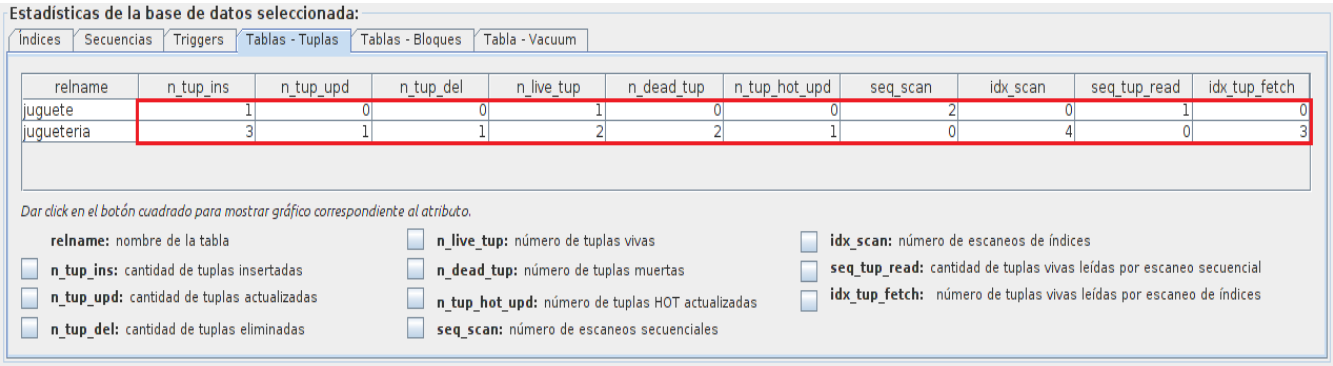

Figura 16: Interfaz que muestra las estadísticas modificadas referentes a las Tablas – Tuplas.

3.3.3. Validación de generar gráfico de la cantidad de tuplas insertadas referente a las estadísticas Tablas – Tuplas.

Luego de establecida la conexión se realizan los siguientes pasos:

Paso 1.- Se realizan los pasos 1 y 2 de la validación anterior (3.3.2).

Paso 2.- Se presiona el botón asociado a la cantidad de tuplas insertadas (n_tup_ins: cantidad de tuplas insertadas). Para esto se debe utilizar la interfaz siguiente en la cual se encuentra señalado el botón referente a la opción mencionada:

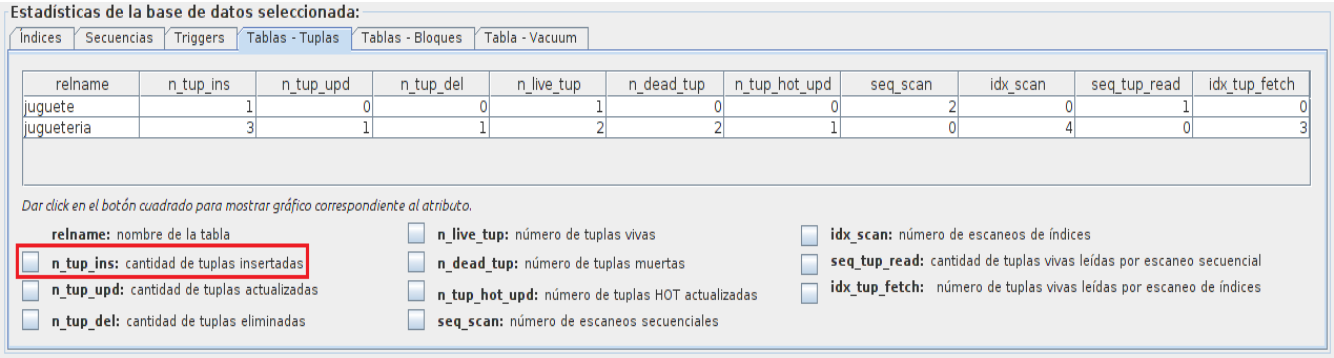

Figura 17: Interfaz que muestra el botón para graficar la cantidad de tuplas insertadas.

Paso 3.- Seleccionar el tipo de gráfico que desea generar, en este caso fue seleccionado de tipo barra según la interfaz de la Figura 18 para luego mostrarse el gráfico con los datos obtenidos en la Figura 19:

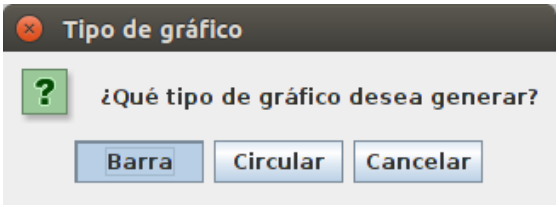

Figura 18: Interfaz que permite la elección del tipo de gráfico a generar.

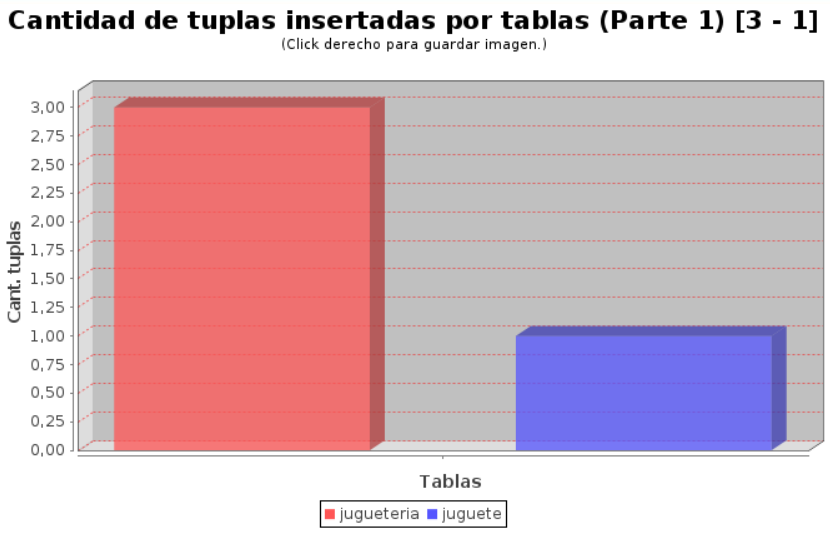

Figura 19: Gráfico de barra de la cantidad de tuplas insertadas.

Paso 4.- Para comprobar que la herramienta grafica correctamente los datos actuales de las estadísticas referentes a la cantidad de tuplas insertadas, en la base de datos *ejemplo_base_dato* se ejecutan diferentes consultas que modifiquen los datos mostrados, como se muestra a continuación:

```
INSERT INTO jugueteria(id, nombre) VALUES (5, 'Gran Rey');
INSERT INTO jugueteria(id, nombre) VALUES (6, 'Copo de nieve');
INSERT INTO jugueteria(id, nombre) VALUES (7, 'Principe azul');

INSERT INTO juguete(id, nombre, tipo, id_jugeteria) VALUES (4, 'Rey oso', 'Rey', 5);
INSERT INTO juguete(id, nombre, tipo, id_jugeteria) VALUES (5, 'Casa nevada', 'Casa', 6);
```

Figura 20: Consultas sql.

Paso 5.- Luego se vuelven a realizar los pasos 1, 2 y 3 para analizar que los datos expuestos contengan las modificaciones realizadas.

Se puede comprobar que en el gráfico mostrado varían los valores de la cantidad de tuplas insertadas en cada tabla con respecto al gráfico de la Figura 19. Los valores están en correspondencia con las consultas ejecutadas en el paso 4, por lo que queda demostrado en la Figura 21 que la herramienta grafica la información actual y funciona correctamente.

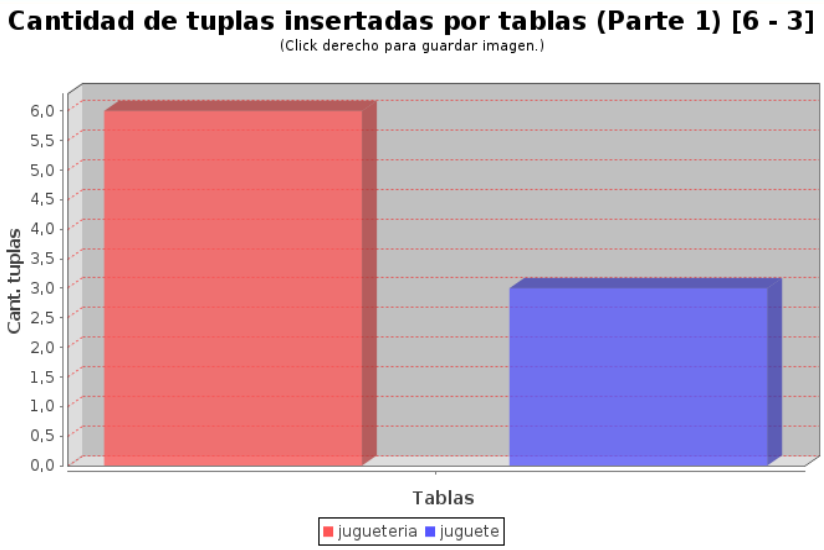

Figura 21: Gráfico de barra de la cantidad de tuplas insertadas una vez ejecutadas consultas.

3.3.4. Validación del uso del disco duro (carpeta /home).

En la interfaz principal, luego de establecida la conexión, se pueden observar los datos del uso del disco duro, mostrados a través de una barra de estado que representa el espacio usado, libre y total de la carpeta home, la cual contiene las carpetas de cada uno de los usuarios con permiso de autenticación al sistema operativo. La siguiente imagen muestra la interfaz principal en la cual se encuentra señalado con un cuadro rojo los datos referentes al uso del disco duro.

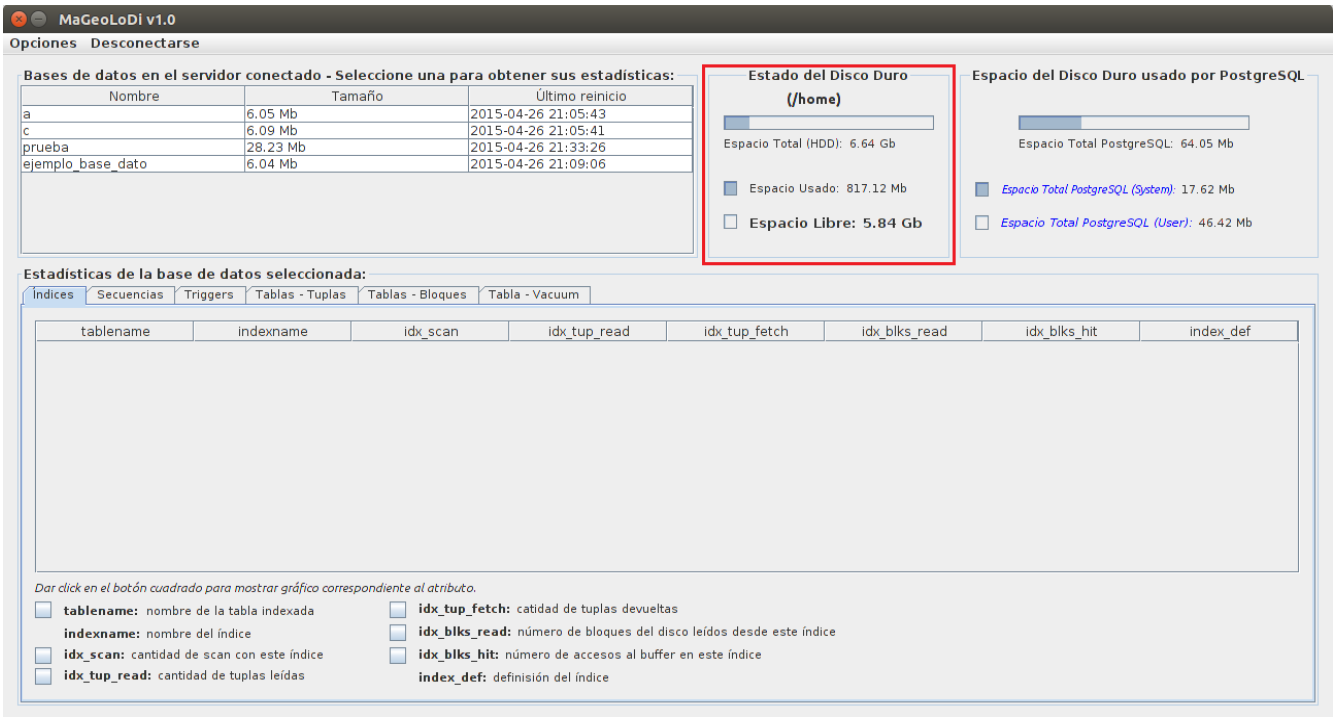

Figura 22: Interfaz principal en la cual se señala los datos del uso del disco duro.

A continuación se muestran los datos en una imagen ampliada.

Figura 23: Datos del uso del disco duro.

Para comprobar que la herramienta funciona correctamente, se copió un archivo de 1,12 Gigabytes (Gb) para la carpeta ubicada en la dirección /home/usuario01. Luego se analizan los datos expuestos en la Figura 24, en este caso se observa que disminuyó el espacio libre del disco duro pues inicialmente era de 5.84 Gb y al finalizar la copia era de 4.72 GB, por lo que queda demostrado que la herramienta obtiene la información actual y funciona correctamente.

Figura 24: Datos del uso del disco duro.

3.3.5. Validación de la integración.

La forma integrada de mostrar las estadísticas es una de las principales características de la herramienta, pues permite que se puedan observar diversas estadísticas como los datos del uso de los triggers, los índices, las secuencias, el disco duro, entre otros. Para visualizarlos solo es necesario estar situado en la interfaz principal de la herramienta, como se muestra en la Figura 12. A continuación se muestra la forma en la que los administradores de bases de datos obtienen los mismos datos que son mostrados en la Figura 12 por la herramienta desarrollada en este trabajo. Para obtener los datos es necesario utilizar uno o varios editores SQL como se muestra en el primer ejemplo o ejecutar diferentes comandos en terminales como se muestra en el segundo ejemplo.

Ejemplo 1.- Obtener los datos del uso de los índices, tablas-tuplas, tablas-bloques a través de consultas sql.

Consulta #1

```
select i1.tablename, i1.indexname, i2.idx_scan,
       i2.idx_tup_read, i2.idx_tup_fetch,
       i3.idx_blks_read, i3.idx_blks_hit, i1.indexdef
from pg_catalog.pg_indexes i1, pg_stat_user_indexes i2, pg_statio_user_indexes i3
where i1.schemaname='public' and i1.tablename=i2.relname and  i1.indexname=i2.indexrelname
      and i1.tablename=i3.relname and  i1.indexname=i3.indexrelname;
```

	tablename name	indexname name	idx_scan bigint	idx_tup_read bigint	idx_tup_fetch bigint	idx_blks_read bigint	idx_blks_hit bigint	indexdef text
1	jugueteria	primaria j	6	6	5	3	15	CREATE UN
2	juguete	primaria j	0	0	0	3	5	CREATE UN

Consulta #2

```
select relname, n_tup_ins, n_tup_upd, n_tup_del, n_live_tup,
       n_dead_tup, n_tup_hot_upd, seq_scan, idx_scan,
       seq_tup_read, idx_tup_fetch
from pg_stat_user_tables;
```

	relname name	n_tup_ins bigint	n_tup_upd bigint	n_tup_del bigint	n_live_tup bigint	n_dead_tup bigint	n_tup_hot_upd bigint	seq_scan bigint	idx_scan bigint	seq_tup_read bigint	idx_tup_fetch bigint
1	juguete	3	0	0	3	0	0	6	0	13	0
2	jugueter	6	1	1	5	2	1	7	6	30	5

Consulta #3

```
select relname, heap_blks_read, heap_blks_hit, idx_blks_read,
       idx_blks_hit
from pg_statio_user_tables;
```

	relname name	heap_blks_read bigint	heap_blks_hit bigint	idx_blks_read bigint	idx_blks_hit bigint
1	juguete	2	9	3	5
2	jugueteria	2	24	3	15

Figura 25: Ejemplo de consultas sql.

Ejemplo 2.- Obtener los datos de los procesos activos de PostgreSQL, reportes estadísticos del procesador y de la memoria virtual del sistema a través de las siguientes terminales.

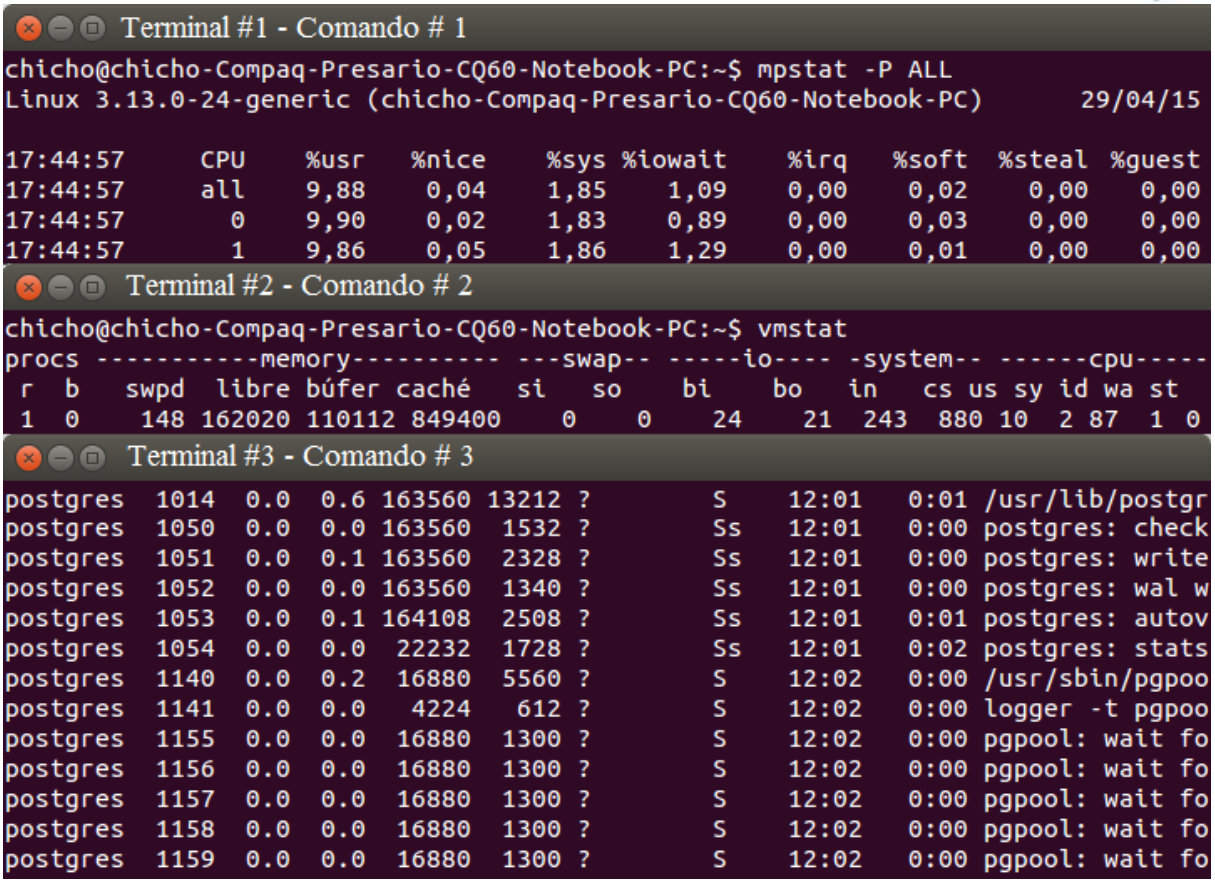

Figura 26: Ejemplo de comandos.

Luego de analizadas las maneras en que se muestran los datos en la herramienta desarrollada en este trabajo y por las dos vías por la que los administradores de bases de datos pueden obtenerlas, se puede llegar a la conclusión de que en la herramienta desarrollada las estadísticas referentes al SGBD PostgreSQL se encuentran integradas.

3.3.6. Validación del tiempo.

En el presente trabajo, el tiempo no deja de ser un elemento fundamental, pues a la hora de tomar decisiones importantes es necesario contar con la información en instantes de tiempos muy rápidos. En esta sección se pretende demostrar que con la utilización de la herramienta desarrollada se ahorra más tiempo que realizando las búsquedas a través de las diferentes vías por las que los administradores de bases de datos obtienen las estadísticas del rendimiento de un servidor de base de datos PostgreSQL. A continuación se muestra una comparación del tiempo utilizado por la herramienta desarrollada y un usuario, el cual utilizará editores sql y terminales para obtener las estadísticas del desempeño de un servidor de base de datos PostgreSQL, así como de los datos del rendimiento del sistema operativo.

Para el caso del usuario se realizó la prueba empleando un archivo sql donde se encontraban escritas las consultas necesarias, permitiendo ahorrar más tiempo pues en caso de no usar un archivo el usuario tendría que escribirlas manualmente en el editor de consultas. Se debe señalar que uno de los elementos que no se tuvo en cuenta para realizar la validación del tiempo fue el análisis de los resultados obtenidos, es decir, no se midió el tiempo empleado para visualizar y comprender las estadísticas. Los datos obtenidos por el usuario siempre se muestran a través de siglas, por lo que se tendría que emplear mayor tiempo en la búsqueda del significado de las mismas para comprender toda la información mostrada; este tiempo en la herramienta desarrollada se ve minimizado debido a que se cuenta con leyendas que permiten entender dichas siglas.

Tabla 16: Comparación de los tiempos estimados para obtener estadísticas.

Operación a realizar	Tiempo estimado por la herramienta	Tiempo estimado por el usuario
Obtener los datos del uso de los índices, las secuencias, los triggers, tablas-tuplas, tablas-bloques y tablas vacuum.	4 segundos	40 segundos
Obtener los datos de las replicaciones, las actividades en el servidor, los usuarios autenticados, el tiempo temporal y las transacciones de bloques de la base de datos, así como los conflictos y Bgwriter.	20 segundos	25 segundos
Obtener el tamaño de cada una de las bases de datos, el espacio del disco duro utilizado por PostgreSQL y los datos del uso del disco duro (libre/usado).	1 segundo	20 segundos
Obtener los datos de los procesos activos de PostgreSQL, reportes estadísticos del procesador y de la memoria virtual del sistema.	2 segundos	35 segundos
Tiempo total estimado.	27 segundos	2 minutos

Luego de analizada la tabla anterior queda demostrado que a través de la herramienta desarrollada se pueden obtener las estadísticas de forma más rápida que haciendo uso de otras vías, lo que le agiliza el trabajo a los administradores en la toma de decisiones.

3.4. Conclusiones parciales.

- El uso de las técnicas para la validación de los requisitos permitió asegurar que fueron definidos correctamente, sin ambigüedades y estables.

- El uso de las métricas Tamaño Operacional de Clases y Relaciones Entre Clases permitió validar el diseño de la herramienta desarrollada concluyéndose que la mayor parte de las clases son reutilizables, contienen pocas dependencias, bajo acoplamiento y bajas responsabilidades.
- Además se realizó la validación de la solución poniendo a prueba el correcto funcionamiento de la herramienta, lo que asegura que muestra los datos correctos de forma rápida e integrada.

Conclusiones

Con la realización del presente trabajo de diploma se arribaron a las siguientes conclusiones:

- Las herramientas que existen en la actualidad para el monitoreo de bases de datos PostgreSQL no son capaces de obtener y mostrar todas las estadísticas necesarias para la toma de decisiones de forma rápida e integrada.
- Para facilitar una descripción detallada de las funcionalidades que la herramienta debía cumplir se confeccionaron 73 historias de usuarios.
- Para lograr mejor implementación de las funcionalidades se crearon 27 tarjetas CRC evidenciando las relaciones entre las clases.
- Para comprobar la calidad y correcto funcionamiento la herramienta desarrollada así como el cumplimiento de los objetivos se aplicaron métricas de calidad que arrojaron resultados positivos.
- Se desarrolló una herramienta que le facilita el trabajo a los administradores de bases de datos a la hora de obtener las estadísticas del rendimiento del SGBD PostgreSQL.

Recomendaciones

A partir de los análisis y criterios que han sido expresados en el presente trabajo y con el propósito de asegurar la posterior ampliación, modificación y mejora de las funcionalidades de la herramienta desarrollada, se expresan las siguientes recomendaciones:

- Registrar los estados de las estadísticas del rendimiento del servidor lo que permita realizar comparaciones y estudios de su comportamiento en determinado momento.
- Implementar un mecanismo de comunicación con otros servidores que se encuentren en la red para poder acceder a sus estadísticas del rendimiento del sistema operativo.
- Implementar un sistema de alertas para los cambios que sean efectuados en los parámetros y datos del SGBD según las necesidades del administrador de bases de datos.

Bibliografía

1 Becerra Zepeda, Sergio Antonio. Bases de datos inteligentes. Tesis de Maestría. Universidad de Colima. Coquimaltán, 1999. 145.

2 PostgreSQL. PostgreSQL, 2015. [Disponible en: http://www. postgresql.org].

3 Vázquez Ortiz, Yudisney. Propuesta del gestor de bases de datos cubano basado en PostgreSQL. Grupo Editorial Ediciones Futuro, 2010, 3 (9): 1-13.

4 Date, C. J. Introducción a los SISTEMAS DE BASES DE DATOS. México, Prentice Hall, 1999. 898.

5 García, Rosa María Mato. Diseño de bases de datos, 1999.

6 Ramírez, Raquel Zambrano. SISTEMAS GESTORES DE BASE DE DATOS, Innovación y experiencias educativas, 2008, 6(14): 1-9.

7 EnterpriseDB. EnterpriseDB, 2015. [Disponible en: http://www.enterprisedb.com].

8 ManageEngine. ManageEngine, 2015. [Disponible en: http://www.manageengine.com].

9 La Red, David L. Martínez. Análisis comparativo del rendimiento, 2011.

10 Entidad de Naciones Unidas para la igualdad de Género y el Empoderamiento de las Mujeres, 2014. [Disponible en: http://www.endvawnow.org/es/articles/330-cual-es-el-monitoreo-y-la-evaluacion.html].

11 pgBadger. pgBadger, 2015. [Disponible en: http://dalibo.github.io/pgbadger].

12 Jaume Sabater. Replicación y alta disponibilidad de PostgreSQL con pgpool-II, 2008.

13 Visual Paradigm. Visual Paradigm for UML 8.0 Released, 2010. [Disponible en: http://www.visual-paradigm.com/aboutus/newsreleases/vpuml80.jsp].

14 Bacerril, Francisco Caballero. Java a su alcance. México, McGraw-Hill, 1999. 287.

15 NetBeans. NetBeans, 2015. [Disponible en: http://www.netbeans.org].

16 Comunidad de Ubuntu. doc.ubuntu-es, 2012. [Disponible en: http://doc.ubuntu-es.org/Sobre_Ubuntu].

17 IBM. Retrieved, 2015. [Disponible en: http://www.ibm.com].

18 Beck, Kent. Extreme Programming Explained. Estados Unidos, Addison-Wesley, 1999. 96.

19 Sommerville, Ian. Ingeniería del software. Madrid, PEARSON EDUCACIÓN. S.A, 2005. 691.

20 Beck, Kent. Una explicación de la programación extrema: aceptar el cambio. Madrid, Addison-Wesley, 2002. 216.

21 Larman, Craig. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. México, Prentice Hall, 1999. 15.

22 Pressman, Roger. Ingeniería de software, un enfoque práctico. Estados Unidos, McGraw-Hill, 2002. 589.

23 Fornaris, Maite Sánchez y Rabí, Dayana Alcantara. Propuesta de una guía de métricas para evaluar el desarrollo de los Sistemas de Información Geográfica. 2010.

24 Lorenz, Mark y Kidd, Jeff. Object-Oriented Software Metrics. Nueva Jersey, Prentice Hall, 1994. 146.