Universidad de las Ciencias Informáticas

Facultad 6

Plugin de monitorización para la herramienta de administración de base de datos HABD

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

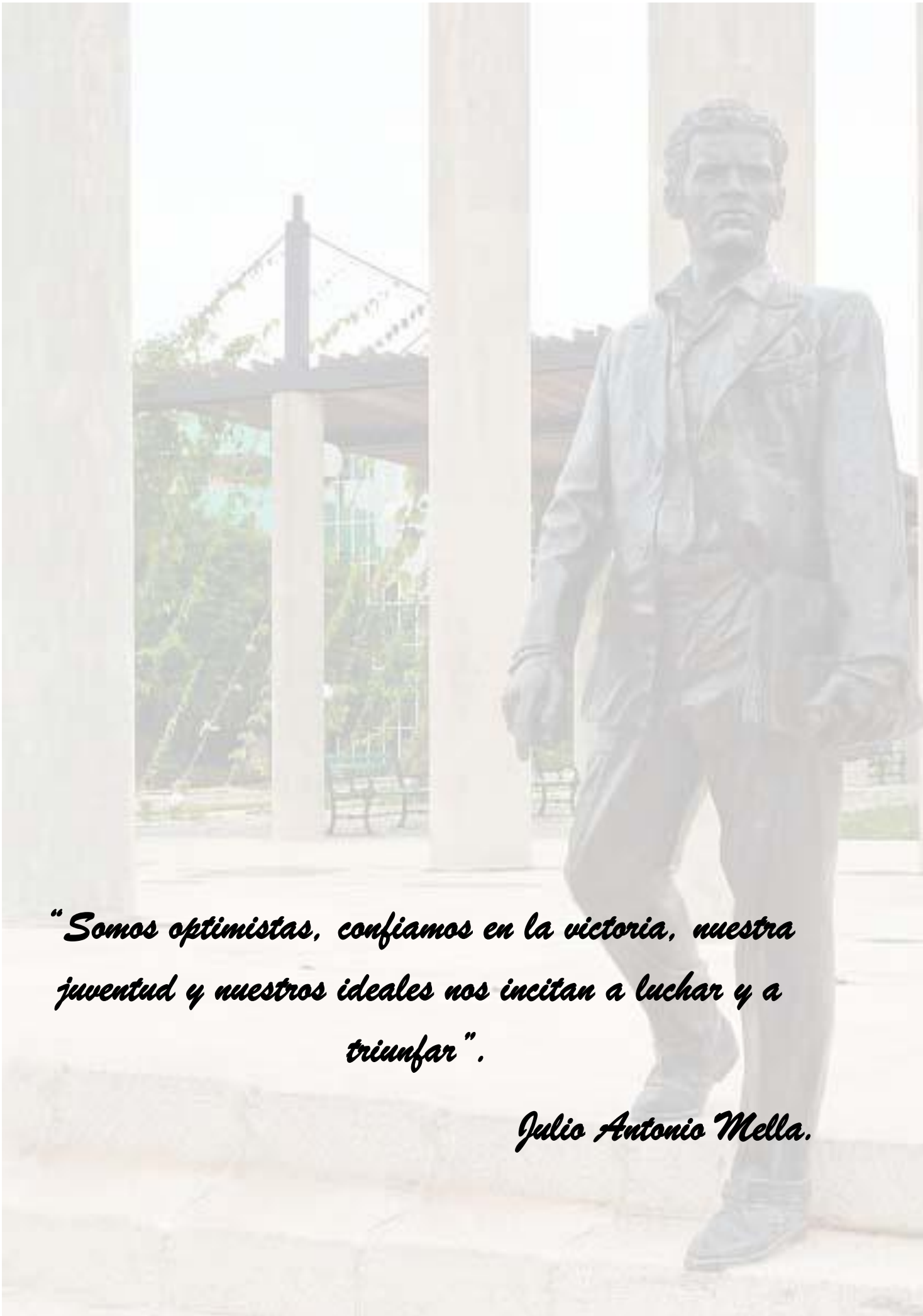
Autor: Beatriz Piñeiro Veitía

Tutores: Ing. Marianela Gutiérrez Rodríguez

Ing. Daymel Bonne Solis

Ciudad de la Habana, Junio 2012

"Año 54 de la Revolución"



"Somos optimistas, confiamos en la victoria, nuestra juventud y nuestros ideales nos incitan a luchar y a triunfar".

Julio Antonio Mella.

Declaración de autoría

DECLARACIÓN DE AUTORÍA

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

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

Beatriz Piñeiro Veitía

_____

[Firma Autor]

Ing. Marianela Gutiérrez Rodríguez

_____

[Firma Tutor]

Ing. Daymel Bonne Solis

_____

[Firma Tutor]

Datos de contacto

DATOS DE CONTACTO

Tutora:
Ing. Marianela Gutiérrez Domínguez
Universidad de las Ciencias Informáticas, La Habana, Cuba
e-mail: mgrodriguez@uci.cu

Tutor:
Ing. Daymel Bonne Solis
Universidad de las Ciencias Informáticas, La Habana, Cuba
e-mail: dbonne@uci.cu

Agradecimientos

AGRADECIMIENTOS

A mi madre Miladys, mamita, como de cariño te digo, gracias por confiar en mí, por brindarme seguridad, amor, cariño y comprensión. Vivo orgullosa de ti, me demostraste que sabes ser padre, madre y amiga. Todas las cosas que hemos vivido juntas y que hemos sufrido han hecho que salga adelante y me creciera ante esas dificultades, pero más que esas cosas quien me inspiraba a seguir eras tú. Te quiero, te adoro, te amo.

A mi padre Genelio (geno), gracias por la educación que me has dado, hemos tenido nuestras diferencias en ocasiones, pero mi amor por ti no ha cambiado, te sigo queriendo como mi padre y te perdono todas las cosas, tal vez, eso ayudó a esforzarme más y a luchar por tener mis propias cosas, creo que hoy es el primer paso, ya soy ingeniera. Te quiero, un beso grande para ti.

Hace cuatro años que conozco a una persona y en ese corto tiempo se ha ganado mi aprecio, cariño y respeto. Con sus virtudes y defectos siempre ha dado su paso al frente para ayudarme, nunca ha existido un no para mí, no existe diferencia entre sus hijos y yo. Siempre atento a mis problemas y buscando solución a ellos. A esa persona la quiero como un padre, cariñosamente lo llamo moyita, para ti, un te quiero bien grande y un beso en la moyerita.

Les agradezco a mis hermanos José Ramón, Andrey, Leticia, Lais y Sary, por darme su apoyo, por confiar en mí, y haber compartido momentos muy bonitos con ustedes, los quiero mucho.

A mis tíos Mirelys, Yamile, Manolito y Reynaldo, a mi abuela Zenaida y a mi prima Lucy, gracias por su preocupación y su apoyo incondicional. Lola, gracias por todo ese cariño y ternura que desde chiquita recibo de ti.

Creo que es un buen momento para agradecer a una persona que me demostró ser un gran amigo, siempre me supo guiar por los buenos caminos, me enseñó muchas cosas tanto profesionales como de la propia vida. Ha estado conmigo en las buenas y en las malas. Esperando un momento para agradecerle se fue entre las manos la oportunidad, por eso quiero darte las gracias por todo, a ti Yoemir.

Agradecimientos

A mis tutores Nela y Bonne, gracias por confiar en mí, por guiarme en el transcurso del trabajo, por brindarme sus conocimientos y por estar disponibles en todos los momentos que necesité de ustedes.

A todas estas personas que mencionaré a continuación quiero darle las más profundas y sinceras gracias por todo el tiempo que me dedicaron, toda la paciencia que tuvieron conmigo y sus buenas intenciones de ayudar. Le agradezco mucho al profe Anthony (Antoniquillo, como yo le digo), a Flavio, Mikel, al profe Tony, Ismel, Denilillo, Andry, Yaciel y a Emmanuel, gracias por todo.

Agradecer a dos personas que vienen conmigo desde muy pequeñas y han estado pendientes de mí, gracias por su amistad Dunia y Arletis.

Y como olvidar al elenco de artistas famosas, a Gina Cabrera (Tahimé) y a Susana Pérez (Ivette) gracias por todo el apoyo, el ánimo y la ayuda brindada para la realización de esta tesis. De ustedes me llevo lo bueno y lo malo, alegrías y tristezas, risas y llanto y la interrogante de que si Este es el mundo real ?

Agradecer a Teresa, Aleydis, karlingirl, Romy, Rebeca, Yeneysi, Yumisiscalla, Yadirosqui, Angel y a Wilber, que a lo largo de la carrera compartimos momentos de alegría, tristeza y preocupaciones.

No porque vaya de último este agradecimiento es menos importante que los demás, esta persona me ayuda a tener más confianza en mí, confío y tengo mucha fe en él, gracias a ti, Aisaidel.

Dedicatoria

DEDICATORIA

De manera muy especial dedico este Trabajo de Diploma a mis abuelos Mireya y Reinaldo (mami y papi).

Papi, aunque ya no estés físicamente conmigo, sé que me cuidas y estas pendiente de mí. Siempre te recuerdo y te tengo presente, ya tu nilinda es hoy ingeniera, parte de este resultado es para ti, sé que estarías orgulloso de mí, te extraño.

Mami, sé que en esto momentos estas tan nerviosa como yo y ansiosa por saber mi resultado, me hubiese gustado mucho que estuvieras aquí para darte un fuerte abrazo, pero eso no importa ya tendremos tiempo. Estoy eternamente agradecida por todo tu apoyo, consejos, cariño, por ser una de las personas que ha estado siempre conmigo en los buenos y los malos momentos, por darme fuerzas constantemente, e incentivarme a salir adelante, por ayudarme a luchar para obtener buenos resultados y ser una mejor persona, por ser una voz digna de escuchar que me ha guiado en muchísimas más ocasiones de las que puedo recordar, tu significas para mí, más de lo que las palabras pudieran expresar, por todo eso, muchas gracias.

Resumen

RESUMEN

En el departamento PostgreSQL se implementó en el curso 2010-2011 una herramienta de administración de base de datos para el gestor PostgreSQL llamada HABD. Su arquitectura basada en Plugin no posee aún las funcionalidades necesarias para brindar un servicio de monitorización. En el presente trabajo de diploma se desarrolló un componente de monitorización, el cual se integró a la herramienta de administración de base de datos HABD. Este Plugin permite un control específico del funcionamiento y rendimiento de cada base de datos. Para cumplir con el objetivo propuesto se realizó un estudio de varias herramientas de monitorización definiendo las métricas a implementar en dicho Plugin. El componente posibilita que los administradores de base de datos puedan visualizar mediante gráficas de barras y descripciones en forma de texto el estado de las bases de datos en un instante de tiempo determinado.

Palabras claves: monitorización, Plugin, PostgreSQL.

Índice

Índice

Índice de figuras y tablas

Índice de figuras

Índice de tablas

Introducción

Introducción

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

La Universidad de las Ciencias Informáticas (UCI), forma parte del motor impulsor en el desarrollo de las tecnologías digitales, la misma cuenta con múltiples proyectos encargados de la confección de productos dirigidos tanto a empresas nacionales, como internacionales. Tiene implementada una infraestructura compuesta por varios Centros de Desarrollos, enfocados en diferentes líneas de producción, siendo uno de estos, el Centro de Tecnologías de Gestión de Datos (DATEC). La línea de desarrollo PostgreSQL está dirigida a la implementación y desarrollo del sistema gestor de base de datos PostgreSQL.

Gran parte de los proyectos desarrollados en este departamento hacen uso del gestor de base de datos PostgreSQL, que como otros proyectos de código abierto, no es desarrollado por una sola empresa, sino es dirigido por una Comunidad de desarrolladores y organizaciones comerciales.

La administración de los servidores de base de datos PostgreSQL, tanto en el entorno de desarrollo, como en el entorno de producción, se realiza mediante la utilización de herramientas, muchas de ellas privativas lo que dificulta su adquisición, estando sujetas a licencias restringidas y costos elevados.

Las herramientas libres existentes presentan un pequeño número de funcionalidades, provocando a los administradores pérdida de tiempo, trabajo ineficiente con las bases de datos y utilización de varias herramientas para poder obtener un único resultado.

Introducción

En el curso 2010-2011 se desarrolló en el departamento de PostgreSQL una herramienta de administración de base de datos para PostgreSQL llamada HABD, la cual provee una interfaz amigable y una arquitectura que permite a los desarrolladores incorporar nuevos servicios de manera sencilla por la flexibilidad que posee.

El usuario que interactúe con la herramienta le resulta difícil conocer el estado de las bases de datos. Esta tarea permite evitar o encontrar el momento en que se producen los problemas. Vigila los servicios de base de datos, como uso de espacio en disco, procesos activos, estadísticas del clúster y las bases de datos en general. El desconocimiento de estos parámetros dificulta a los administradores de base de datos llevar el control específico del funcionamiento y rendimiento de los servidores PostgreSQL.

Lo antes descrito conlleva a la siguiente interrogante: ¿Cómo contribuir al control de las bases de datos PostgreSQL en la herramienta de administración de bases de datos HABD?

Con el desarrollo de este trabajo se pretende dar solución al problema antes citado. Para la obtención de estos resultados se plantea como **objeto de estudio**: El Proceso de administración de base de datos en PostgreSQL, enmarcado en el **campo de acción:** Herramientas para la monitorización de las bases de datos en PostgreSQL.

Planteándose lo anterior, el **objetivo de la investigación** es desarrollar un Plugin para la herramienta de administración de bases de datos HABD, que permita la monitorización de las bases de datos en PostgreSQL.

En correspondencia con el objetivo de la investigación se traza una serie de objetivos específicos:

1. Explorar las herramientas basadas en la monitorización de las bases de datos.
2. Diseñar el Plugin de monitorización para la herramienta HABD.
3. Implementar el Plugin de monitorización para la herramienta HABD.
4. Validar el producto implementado.

Para dar cumplimiento a los anteriores objetivos se han definido un grupo de tareas que encaminarán el transcurso de la investigación:

1. Revisión bibliográfica de las herramientas de administración basadas en la monitorización de las bases de datos.
2. Identificación y descripción de las funcionalidades del Plugin.
3. Elaboración del modelo de diseño del Plugin.
4. Implementación de las funcionalidades identificadas.
5. Integración del Plugin monitorización a la herramienta HABD.
6. Identificación de los métodos de prueba para la validación del Plugin.
7. Elaboración de los casos de pruebas basados en historias de usuarios.

Como resultado del presente trabajo se espera obtener la integración satisfactoria del Plugin en la herramienta de administración de bases de datos HABD.

El documento está estructurado en introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas y bibliografía.

Capítulo1. Fundamento Teórico. La investigación realizada para este capítulo contiene un análisis de herramientas de monitorización de base de datos. Muestra definiciones importantes para lograr un mejor entendimiento de la investigación. Se realiza un estudio para definir las herramientas, metodología y tecnologías a utilizar para el desarrollo del Plugin de monitorización a base de datos.

Capítulo 2. Características del sistema propuesto. Se define la descripción del flujo actual de los procesos involucrados. Se describen las principales características del Plugin, los requerimientos e historias de usuario. Para lograr un mejor entendimiento del Plugin se realiza un diagrama de clases del diseño y un modelo de dominio, los cuales se utilizan como complemento de la metodología definida.

Capítulo 3: Validación del sistema propuesto. Para verificar si el sistema cumple con los requisitos descritos por el cliente, se realiza un estudio para definir bajo que pruebas y técnicas se valida el componente de monitorización de base de datos. Se muestran los resultados luego de haber realizado los casos de pruebas por cada una de las funcionalidades del Plugin.

Capítulo 1. Fundamento Teórico

Capítulo1. Fundamento Teórico

1.1 Introducción

En este capítulo se presentan un grupo de conceptos teóricos a los que se hará referencia en el resto del trabajo. Comprende un estudio de definiciones fundamentales para desarrollar la investigación, tales como: base de datos, Sistemas Gestores de Bases de Datos (SGBD) y Plugin. Se realiza un estudio a diferentes herramientas de monitorización existentes. Se muestran las características definidas de las herramientas, tecnologías y metodología a utilizar durante el desarrollo de la investigación.

1.2 Base de datos

Una base de datos es un conjunto de información estructurada en registros y almacenada en un soporte electrónico legible desde un ordenador. Cada registro constituye una unidad autónoma de información que puede estar a su vez estructurada en diferentes campos o tipos de datos que se recogen en dicha base de datos. Por ejemplo, en un directorio de miembros de una asociación, un registro será la ficha completa de cada uno de los socios. En cada registro se recogerán determinados datos, como el nombre, la profesión, la dirección o el teléfono, cada uno de los cuáles constituye un campo. (1)

Las bases de datos son un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. Entre las características de las bases de datos se encuentra: independencia lógica y física de los datos, redundancia mínima, acceso concurrente por parte de múltiples usuarios, integridad de los datos, consultas complejas optimizadas y seguridad de acceso y auditoría.

En la actualidad, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos están en formato digital, solucionando los problemas que existían en el almacenamiento de los datos. Existen programas denominados sistemas gestores de bases de datos que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada.

Capítulo 1. Fundamento Teórico

1.3 Sistema gestor de base de datos

Un sistema gestor de base de datos es una colección de programas que permiten a los usuarios crear y mantener una base de datos. Sistema software de propósito general que facilita los procesos de definición, construcción y manipulación de la base de datos para distintas aplicaciones. (2)

Se trata de un conjunto de programas no visibles al usuario final que se encargan de la privacidad, integridad, seguridad de los datos e interacción con el sistema operativo. Proporciona una interfaz entre los datos, los programas que los manejan y los usuarios finales. Cualquier operación que el usuario hace con la base de datos está controlada por el gestor.

El propósito general de los sistemas de gestión de bases de datos es el manejo de forma clara, sencilla y ordenada de un conjunto de datos, que posteriormente se convertirán en información relevante para una organización. Proveen facilidades para la manipulación de grandes volúmenes de datos, entre las que se encuentran:

- Simplifican la programación de equipos de consistencia.
- Manejo de las políticas de respaldo adecuadas, garantizan que los cambios de la base de datos sean siempre consistentes sin importar si hay errores.
- Organizan los datos con un impacto mínimo en el código de los programas.
- Disminuyen drásticamente los tiempos de desarrollo y aumentan la calidad del sistema desarrollado, si son bien explotados por los desarrolladores.

Con el excesivo cúmulo de información, la centralización de la mayoría de las aplicaciones que contienen base de datos y la cantidad de personas que concurren en dichos sistemas; surge la necesidad de emplear un sistema de administración el cual controle tanto a los datos como a sus usuarios.

Estos sistemas de administración proporcionan a los usuarios la gestión del acceso a la base de datos de manera simultánea y garantizan que no sucedan inconvenientes con la integridad de los mismos. Existen *SGBD* muy potentes que proporcionan muchas de las funciones estándar que el

programador necesita para lograr grandes avances, entre los que se encuentran: Oracle, DB2, PostgreSQL, SQL-Server, Fox Pro, Access y SQLite.

Debido a la migración que está llevando actualmente la universidad, vinculada a las tendencias tecnológicas, que consiste en la utilización de herramientas netamente libres, el empleo del sistema gestor de base de datos PostgreSQL es inminente.

1.3.1 PostgreSQL

PostgreSQL es el gestor de base de datos de código abierto más avanzada del mundo. Distribuido bajo licencia BSD (del inglés, *Berkeley Software Distribution*), lleva más de 15 años desarrollándose y su arquitectura goza de una excelente reputación por su fiabilidad, integridad de datos y estabilidad. (3)

Dispone de versiones para prácticamente todos los sistemas operativos. Tiene soporte para llaves foráneas, *joins*, vistas y disparadores. Incluye la mayoría de los tipos de datos de SQL92 y SQL99 y, así mismo, soporta el almacenamiento de grandes objetos binarios, como imágenes, sonidos y vídeos. Tiene interfaces de programación nativas para C/C++, *Java*, *.Net*, *Perl*, Php, *Python*, *Ruby*, Tcl y ODBC, entre otros, y una excepcional documentación.

PostgreSQL ofrece sofisticadas características tales como control concurrente multiversión (MVCC), *point in time recovery* (PITR), *tablespaces*, replicación asíncrona, transacciones anidadas (*savepoints*), copias de seguridad en caliente/en línea, un sofisticado planificador/optimizador de consultas y *write ahead logging* para ser tolerante a fallos de *hardware*. Soporta juegos de caracteres internacionales, codificaciones de caracteres *multibyte*, *Unicode* y ordena dependiendo de la configuración del idioma local, la diferenciación de mayúsculas y minúsculas, y el formato. Es altamente escalable tanto en la cantidad bruta de datos que puede manejar como en el número de usuarios concurrentes que puede atender. Hay sistemas activos en producción con PostgreSQL que manejan más de cuatro *terabytes* de datos.

Por las características que presenta este gestor de base de datos la Universidad de las Ciencias Informáticas hace uso del mismo. Para su administración se necesitan herramientas que facilite el trabajo de los usuarios y cumplan con las tendencias tecnológicas del país.

Capítulo 1. Fundamento Teórico

1.4 Herramienta de administración de base de datos HABD

Para evitar la utilización de varias herramientas de administración de base de datos y facilitar el trabajo de los usuarios, en el departamento PostgreSQL se desarrolló en el curso 2010-2011 la herramienta HABD. Su arquitectura basada en Plugin contribuye al intercambio entre el usuario y el gestor PostgreSQL, permitiendo agregar las funcionalidades que el usuario necesite. Este trabajo de diploma se enfoca en la investigación de cómo se realiza la monitorización de base de datos para poder desarrollar un Plugin y lograr la integración en la herramienta HABD.

1.5 Monitorización de base de datos

Monitorizar significa controlar el funcionamiento de un sistema, servicio o actividad en un tiempo acordado. Una de las funciones más importantes de un administrador de base de datos es la monitorización de los sistemas a su cargo, para de esa forma saber cómo están funcionando y hacer futuras modificaciones además de las actualizaciones de las mismas.

Ad Hoc: Monitorización específica en caso de problemas o pruebas. Se utiliza generalmente para investigar una situación puntual para intentar encontrar una explicación a un suceso, cambio o problema. Además de utilizar herramientas para monitorizar el funcionamiento de las aplicaciones se utilizan los siguientes comandos para la monitorización de los recursos del sistema operativo: vmstat[1], iostat[2], sar[3], ps[4], top, iotop, htop.

Preventiva: Detecta interrupciones de servicios, alerta sobre posibles problemas y crea gráficos con tendencias y datos históricos sobre el sistema. Este tipo de monitorización ayuda a descubrir cambios en el sistema, que en un tiempo determinado pueden provocar dificultades al mismo.

Los elementos a monitorizar son todos aquellos que brinden información sobre cómo el sistema está funcionando y comportándose. El software para monitorizar servidores de base de datos debe:

[1] Se usa para obtener información sobre el uso de memoria, swap, I/O total y CPU del sistema.
[2] Se utiliza para obtener información sobre la entrada/salida de datos de todos los dispositivos, particiones y sistemas de ficheros NFS.
[3] Este programa se suele utilizar para recolectar y grabar información sobre nuestro sistema durante un período de tiempo.
[4] Este programa brinda información sobre los procesos que se están ejecutando en nuestro sistema.

- Controlar el tamaño de las bases de datos, del buffer caché y del tiempo de conexión de la base de datos.
- Analizar el número de conexiones de usuarios a las base de datos en distintos momentos.
- Analizar las tendencias de uso.
- Ayudar a actuar de manera proactiva antes de que se produzcan incidentes graves.

Existen diversos software que favorecen a la monitorización de base de datos orientados a PostgreSQL, todos pertenecientes a sistemas operativos libres. Nagios y Munin constituyen ejemplos de este software, el primero detecta y alerta sobre posibles problemas, el segundo crea gráficas históricas con los datos recopilados y uso del sistema. (4)

1.6 Herramientas de monitorización de base de datos.

En este epígrafe se realiza un estudio de cuatro herramientas que realizan monitorización a base de datos. Se realiza una selección de las funcionalidades comunes que presentan para colaborar con la implementación del Plugin de monitorización de base de datos.

1.6.1 Applications manager

Es una herramienta de monitorización de servidores para base de datos Oracle, MS SQL, Sybase, IBM DB2 y MySQL. También es de utilidad para los Administradores de Base de Datos (DBA) y para los administradores de sistemas ya que notifica sobre potenciales problemas en el rendimiento de las bases de datos.

Applications Manager funciona de manera activa y se encarga de verificar que todo funciona correctamente, notificando los problemas que surjan a través del correo. Es una aplicación muy completa de la que se puede extraer gran cantidad de información. El programa se encarga de recopilar toda la información del sistema y mostrarla en forma de gráficos, números y luces de colores que indican el estado de cada uno de los sistemas analizados. (5)

Para llevar a cabo la monitorización, Applications Manager, se conecta a la base de datos y garantiza que está funcionando, esta herramienta no utiliza agentes y ejecuta consultas a base de datos para recolectar estadísticas de rendimiento y enviar alertas en el caso de que la base de datos cruce un umbral determinado.

1.6.2 MySQL administrator

MySQL Administrator es un software gratuito, editado y distribuido por la empresa MySQL AB, permite administrar, gestionar y monitorizar múltiples instancias de MySQL. Gestiona de forma gráfica un servidor MySQL. Esta aplicación integra administración y mantenimiento de base de datos en una única interfaz. Se trata de un software multiplataforma, que por el momento se encuentra disponible para Linux y Microsoft Windows y que cuenta con un entorno gráfico de usuario muy intuitivo.

Características:

- Administración de usuarios y privilegios.
- Vigilancia de la salud de su espacio de disco del servidor, el número de peticiones y el tamaño del índice.
- Esquema de base de datos de gestión.
- Sistema de copia de seguridad.
- Información clave y detalles sobre el servidor.
- Estado de la conexión.
- Información sobre el estado de la replicación.
- Visualización de registros.

1.6.3 Sybase control center

Sybase Control Center (SCC) es una herramienta para monitorizar servidores Sybase a través de un navegador web estándar con el "plug-in" Flash Player de Adobe. Brinda capacidades de monitorización en tiempo real e histórico. Los recursos que se pueden monitorizar incluyen procesos, base de datos, dispositivos, colas y caminos de replicación. (6) El acceso a éstas estadísticas le permite detectar problemas de aplicación a tiempo y de manera precisa

Algunos de los componentes que son monitorizados en Sybase son:

- Estadísticas de conexión.
- Utilización de la memoria.
- Información de la base de datos.

- Estadísticas actuales.
- Procesos actuales.
- CPU, memoria, utilización de disco.

Capacidades de Monitoreo Sybase:

- Monitorización de las estadísticas del desempeño como las estadísticas de conexión y utilización de la memoria. Las alertas pueden ser configuradas para todos estos parámetros.
- Basado en los umbrales configurados, las notificaciones y alertas son generadas. Las acciones son ejecutadas automáticamente basándose en dichas configuraciones.
- Las gráficas y reportes del desempeño están disponible de manera inmediata. Los reportes pueden ser agrupados y desplegados basándose en la disponibilidad, estado y tiempo de conexión.
- Entrega de métricas del desempeño Sybase tanto actuales como del historial, brindando un entendimiento del desempeño sobre un período determinado de tiempo.

Plataformas que soporta:

- Windows x86/x86-64
- Linux x86/x86-64
- Hp Itanium
- Sun Solaris Sparc 64-bit

1.6.4 Silver sash administrator

Silver Sash Administrator es un programa de administración, desarrollo y monitorización de base de datos. Monitoriza las mediciones de rendimiento de las bases de datos e incluye exportación e importación de volcados de datos.

Funciones detalladas:

- Compatibilidad con Sistemas Operativos.
- Compatible con todas las versiones de Windows.

- Compatibilidad con versiones de Oracle: compatible con Oracle 11g, 10g y 9i.
- Verdadero editor de base de datos.
- Control total de los objetos de la base de datos.
- Apertura simultánea de múltiples bases de datos.
- Realiza operaciones sobre múltiples objetos en una sola pantalla.

Gestiona lo siguiente:

- Tablas.
- Índices.
- Vistas.
- Usuarios.
- Roles.
- Sinónimos.
- Secuencias.
- Clústers.
- Enlaces de base de datos.
- Vistas materializadas, espacios de tablas.
- Particiones.
- Restaura segmentos a estados anteriores.
- Procedimientos.
- Funciones.
- Paquetes.
- Desencadenadores
- Programaciones y trabajos

Conclusiones arribadas en el estudio de las herramientas

El análisis de las herramientas anteriores arroja como resultado que su adquisición no garantiza ni contribuye a alcanzar la independencia tecnológica para Cuba, aunque algunas son de código abierto, reciben soporte de compañías propietarias que impiden la colaboración desde el país con nuevas funcionalidades. Estas herramientas en su conjunto presentan varios indicadores convenientes a utilizar en la investigación, pero por si solas no contienen las características que

se desea para el Plugin de monitorización de base de datos. Se decide escoger las siguientes características que resaltan en cada una de estas herramientas: número de funciones, número de reglas, cantidad de bloqueos, tamaño de las bases de datos y estadísticas de conexión, que comprende, nombre de las bases de datos, dirección ip y usuario.

1.7 Funcionalidades propuestas para el Plugin de monitorización de base de datos

Para la implementación del Plugin de monitorización se escogerán algunas de las funcionalidades presentes en las herramientas estudiadas, agregando otras que fueron consultadas con el equipo administración de base de datos perteneciente al departamento PostgreSQL, que contribuirán a un desarrollo mejor del componente.

- Información de los usuarios conectados.
- Tuplas insertadas por tablas.
- Tuplas eliminadas por tablas.
- Tuplas actualizadas por tablas.
- Tuplas muertas por tablas.
- Tamaño de índices de todas las tablas.
- Tamaño ocupado por las bases de datos.
- Tamaño de todas las tablas.
- Cantidad de bloqueos por base de datos.
- Número de funciones por base de datos.
- Tuplas insertadas por base de datos.
- Tuplas actualizadas por base de datos.

1.8 Plugin

Un Plugin es una aplicación informática que interactuando con otra aplicación le aporta una función específica a esta**.** Permite a los desarrolladores externos colaborar con la aplicación principal dejando extender sus funciones, ayuda a reducir el tamaño de la aplicación, además de separar el código fuente de la misma a causa de las incompatibilidades de las licencias de software libre. Un programa puede tener uno o más conectores. Son muy utilizados en los programas navegadores para ampliar sus funcionalidades.

1.9 Metodología de desarrollo de software

Las metodologías están encaminadas a estructurar, planear y controlar el proceso de desarrollo en sistemas de información. Toda metodología debe estar adecuada a las necesidades del software y cuánto pueda abarcar el mismo. Las metodologías de desarrollo de software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto software. Dichas metodologías pretenden guiar a los desarrolladores al crear un nuevo software, pero los requisitos de un software a otro son tan variados y cambiantes, que ha dado lugar a que exista una gran variedad de metodologías para la creación del mismo.

Estas metodologías se pueden catalogar en dos grupos, las tradicionales (pesadas) y las ágiles (ligeras):

- Las tradicionales se enfocan en el control del proceso, estableciendo estrictamente las actividades involucradas, los artefactos que se deben producir así como las herramientas y notaciones que se usarán.

- Las metodologías ágiles dan protagonismo al individuo, colaboración con el cliente y desarrollo incremental del software con iteraciones cortas. Esta se usa en los proyectos con requisitos cambiantes y de poco complejo su desarrollo, pero manteniendo una alta calidad.

Metodología Extreme Programming

Según un estudio realizado por Patricio Letelier en la Universidad Politécnica de Valencia la metodología Extreme Programming (XP) tiene mayor agilidad con respecto a SCRUM, Crystal Methodologies, Dynamic Systems Development Method (DSDM), Adaptive Software

Development (ASD), Feature-Driven Development (FDD) y Lean Development (LD). La comparación fue establecida bajo las siguientes características: sistema como algo cambiante, colaboración, características metodológicas, resultados, simplicidad, adaptabilidad, excelencia técnica y prácticas de colaboración, alcanzando XP la mayor puntuación.

Se decidió trabajar con la Metodología XP, pues posee un grupo de ventajas que logran que los productos salgan con la mayor calidad posible. Esta metodología se enfoca en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promueve el trabajo en equipo, propicia un ambiente agradable de trabajo. Se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes y simplicidad en las soluciones implementadas. El trabajo con XP ahorra tiempo en documentación y los miembros del equipo se enfocan en poner todo sus esfuerzos en la realización del producto, los cuales son más fiables y robustos contra los fallos gracias al diseño de las pruebas de forma previa a la codificación.

1.10 Herramientas y tecnologías a utilizar

El uso de tecnologías y herramientas es fundamental para la implementación de aplicaciones. Para el desarrollo del Plugin de monitorización para base de datos se utilizarán las siguientes tecnologías.

1.10.1 Framework de desarrollo: QT

Un framework, en el desarrollo de software es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

El framework QT es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario y también para el desarrollo de programas sin interfaz gráfica, como herramientas de la consola y servidores. Utiliza el lenguaje de programación C++ de forma nativa, adicionalmente puede ser utilizado en varios lenguajes de programación a través de bindings. Funciona en todas las principales plataformas, y tiene un amplio apoyo. La API (La interfaz de programación de aplicaciones) de la biblioteca cuenta con métodos para acceder a base de datos mediante SQL,

así como uso de XML, gestión de hilos, soporte de red y una API multiplataforma unificada para la manipulación de archivos, además de estructuras de datos tradicionales. Esta biblioteca se encuentra distribuida bajo los términos de GNU Lesser General Public License, Qt es software libre y de código abierto. (7)

1.10.2 Entorno de desarrollo integrado: QT Creator

El entorno de desarrollo integrado QT Creator desarrollar aplicaciones en C++ de manera sencilla y rápida. Como su nombre lo indica, está basado en la librería Qt y cuenta con las siguientes características principales:

- Editor avanzado para C++.
- Ayuda sensible al contexto integrado.
- Soporta los lenguajes: C#/.NET Lenguajes (Mono), Python: PyQt y PySide, Ada, Pascal, Perl, PHP y Ruby.
- Diseñador de formularios (GUI) integrado.
- Herramientas para la administración y construcción de proyectos.
- Completado automático.
- Depurador visual. (8)

Está completamente integrado con toda la documentación Qt y ejemplos a través de la ayuda Qt Plugin. Para obtener ayuda sensible al contexto, se puede mover el cursor de texto a una clase o función Qt y pulsar F1. Qt proporciona a los usuarios que trabajan más cómodo con el teclado que con el ratón, una amplia gama de atajos en el teclado.

Este IDE es distribuido bajo tres tipos de licencias: Qt Commercial Developer License, Qt GNU LGPL v. 2.1, Qt GNU GPL v. 3.0 y está disponible para las plataformas: Linux, Mac OSX; Windows, Windows CE, Symbian y Maemo.

1.10.3 Lenguaje de Programación: C++

En la actualidad, el C++ es un lenguaje versátil, potente y general. Su éxito entre los programadores profesionales le ha llevado a ocupar el primer puesto como herramienta de desarrollo de aplicaciones. El lenguaje C++ mantiene las ventajas del C en cuanto a riqueza de

operadores y expresiones, flexibilidad, concisión y eficiencia. Además, ha eliminado algunas de las dificultades y limitaciones del C original. (9)

Es un lenguaje procedural (orientado a algoritmos) y orientado a objetos, como lenguaje procedural se asemeja al C y es compatible con él. Como lenguaje orientado a objetos se basa en una filosofía completamente diferente, que exige del programador un completo cambio de mentalidad. El lenguaje C++ depende del hardware, es uno de los más potentes porque permite programar a alto y a bajo nivel.

1.10.4 Herramienta de modelado: Visual Paradigm for UML3.4

Visual Paradigm es una herramienta que permite realizar modelado UML siguiendo el estándar UML 2.1. Esta herramienta tiene unas características gráficas muy cómodas que facilitan la realización de los diagramas de modelado que sigue el estándar de UML, que son: Diagramas de clase, Casos de Uso, Comunicación, Secuencia, Estado, Actividad, Componentes.

El Visual Paradigm ofrece:

- Entorno de creación de diagramas para UML 2.1.
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales IDEs.
- Disponibilidad en múltiples plataformas. (10)

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Unified Modling Languaje (UML): Es un lenguaje para el desarrollo de software orientado a objetos, su propósito es visualizar, especificar, construir y documentar proyectos de software. La herramienta UML CASE proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML.

1.10.5 Sistema de control de versiones: Subversion

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

1.11 Conclusiones parciales

En este capítulo se realizó un estudio de las herramientas de monitorización a base de datos, sus características servirán de apoyo para cumplir con el objetivo de la investigación. Fue seleccionada la metodología Extreme Programing, como guía en el proceso de desarrollo del software. Se realizó un estudio de las herramientas y tecnologías existentes con el propósito de seleccionar las más adecuadas para la solución de la investigación, determinando utilizar Visual Paradigm 6.4 como herramienta CASE, empleando, el lenguaje de modelado UML, el framework QT 4.8, el IDE de desarrollo QT Creator 2.3.1, como lenguaje de programación C++, utilizando el controlador de versiones Subversión.

Capítulo 2. Características del sistema propuestos

Capítulo 2: Características del sistema propuesto

2.1 Introducción

En el siguiente capítulo se presenta la propuesta de un Plugin para la monitorización de base de datos. Se hace una especificación de las características que tendrá el mismo, además de sus requisitos funcionales y no funcionales. Se definen las historias de usuario y se realiza un diagrama de clases del diseño que se utiliza como complemento de la metodología definida, para lograr un mejor entendimiento del componente.

2.2 Identificación del problema

En el Departamento de PostgreSQL, en el curso 2010-2011, se desarrolló una herramienta de administración de base de datos llamada HABD para el gestor de base de datos PostgreSQL. El objetivo de la herramienta es brindar a los usuarios tantas funcionalidades necesiten para administrar una base de datos de manera sencilla. Su arquitectura está basada en Plugin, proporcionando la integración de varios de estos, con funcionalidades que aporten a mantener la base de datos en correcto funcionamiento y que los usuarios puedan interactuar con la misma cómodamente.

Una de las tareas más importantes que debe de realizar el administrador de base de datos es vigilar el funcionamiento de la misma. HABD, debe visualizar información sobre el rendimiento de las bases de datos y tablas que el usuario desee conocer.

Para desarrollar un Plugin que influya positivamente en la integración de HABD se realizó un estudio de varias herramientas de monitorización que existen. De estas se toman las características comunes y diferentes, y se conformará el componente de monitorización.

En los epígrafes posteriores se abordarán todos los temas referentes al desarrollo del Plugin, los requisitos y su arquitectura.

2.3 Modelo de dominio

Capítulo 2. Características del sistema propuestos

En la metodología XP no está definido una técnica para determinar el negocio, por lo que es necesario realizar este proceso de una manera entendible para llevar a cabo el desarrollo del Plugin. Una de las técnicas más utilizadas es el modelo de dominio, ayudando a facilitar el entendimiento del software que se desea desarrollar. Es considerado uno de los artefactos más necesarios que debe contener cualquier proyecto.

El Modelo de Dominio es una representación visual estática del entorno, un diagrama con los objetos que existen (reales) relacionados con el proyecto. Ayuda a comprender los conceptos que utilizan los usuarios, conceptos con los que trabajan y con los que deberá trabajar nuestra aplicación.

La figura 1 muestra como el usuario interactúa con el Front_End y configura las opciones de apariencias que tiene el mismo. En el Front_End va integrado el Plugin de monitorización de base de datos, el cual muestra datos a través de gráficas.

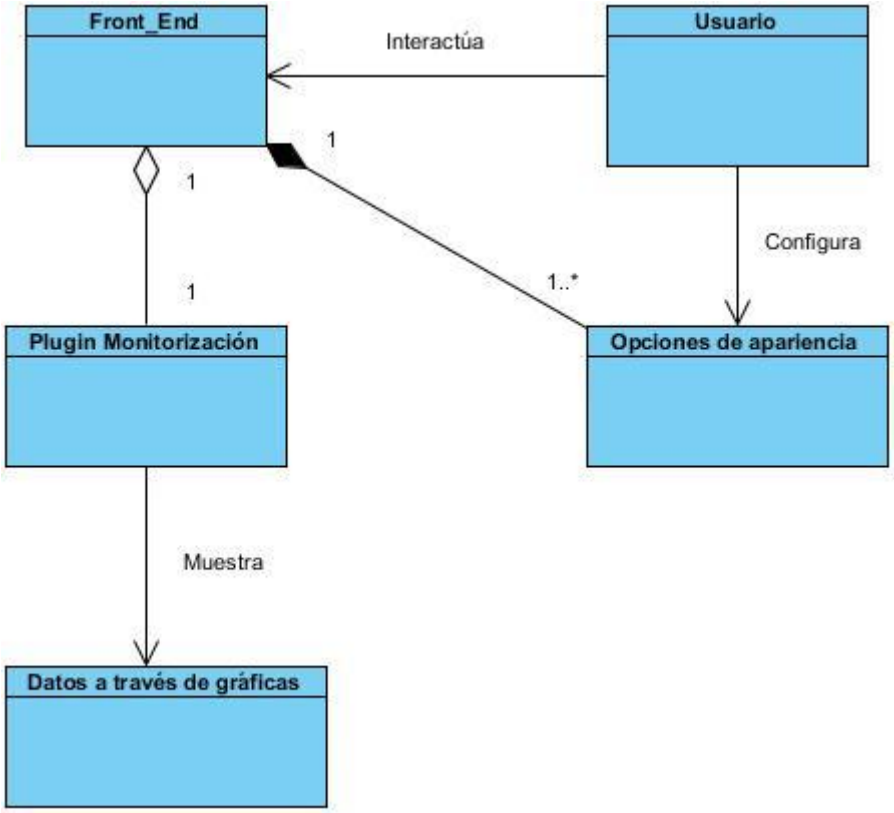


Figura 1 Diagrama Modelo de Dominio

Capítulo 2. Características del sistema propuestos

Usuario: Persona que interactúa con el Front-End.

Front_End: Armazón que carga y manipula los plugins que sean adicionados por el usuario.

Plugin Monitorización: Módulo que añade la nueva funcionalidad al Front-End de la monitorización a base de datos.

Datos a través de gráficas: Muestra gráficas de barras con la información de las bases de datos.

Opciones de apariencia: Opciones funcionales que le permite al usuario hacer cambios en la apariencia del Front-End.

2.4 Descripción del Plugin de monitorización

El Plugin de monitorización a base de datos para integrarse a HABD, presenta una interfaz visual sencilla y amigable para quienes interactúen con la herramienta. El usuario puede conocer de todas las bases de datos que existen en el clúster, la cantidad de tuplas actualizadas, insertadas y retornadas por cada una de ellas. Brinda la información del espacio que ocupan las mismas en PostgreSQL, teniendo la opción de mostrar o no, las bases de datos propias del sistema, el tamaño de las bases de datos se muestra en la unidad de medida mega byte (MB). El resultado de las funcionalidades antes mencionadas se visualiza en la página Clúster mediante gráficas de barras. En esta misma página el usuario puede conocer a través de una tabla, información de la base de datos activa, mostrándose el nombre de la base de datos, usuario conectado, dirección ip, aplicación ejecutada y la consulta que está realizando.

En la página Base de Datos, el usuario puede conocer la cantidad de funciones, bloqueos, índices y reglas que tiene la base de datos activa. El tamaño de índice por tablas y el tamaño de cada tabla se muestran en gráficas de barras, cada barra representa una tabla dentro de la base de datos. Los tamaños de estas dos últimas funcionalidades se muestran en MB.

La página Tabla, posibilita conocer estadísticas, como: tuplas actualizadas, eliminadas, muertas e insertadas por tablas. La navegación del usuario con la aplicación puede ser tanto por una de las tres páginas (Clúster, Base de Datos o Tabla) o yendo a la funcionalidad específica que desee mediante los botones que existen en la barra de menú.

Capítulo 2. Características del sistema propuestos

2.5 Historias de usuario

Las historias de usuarios son artefactos generados en XP, tienen la misma finalidad que los casos de uso, pero con algunas diferencias: constan de tres o cuatro líneas escritas por el cliente en un lenguaje no técnico, no se debe hablar ni de posibles algoritmos para su implementación ni de diseños de base de datos adecuados. Son usadas para estimar tiempos de desarrollo de la parte de la aplicación que describen. También se utilizan en la fase de pruebas, para verificar si el programa cumple con lo que especifica la historia de usuario. Para el presente trabajo se obtienen un total de seis historias de usuarios (HU) que serán realizadas en tres iteraciones, se muestra la HU Mostrar estadísticas de base de datos (Ver tabla 1), y el resto en el documento Plantilla de Historias de Usuarios del Expediente de proyecto.

Tabla 1 Historia de Usuario "Mostrar Estadísticas de base de datos"

	Historia de Usuario
Número:1	**Nombre de la Historia de Usuario:** Mostrar estadísticas de base de datos.
Cantidad de modificaciones a la Historia de Usuario: ninguna	
Usuario: Beatriz Piñeiro Veitía	**Iteración asignada:** 1Iteración
Prioridad en negocio: Alta	**Puntos estimados:** 3 semanas
Riesgo en desarrollo: Medio	**Puntos reales:** 3 semanas
Descripción: En esta HU se mostrará una descripción en forma de texto correspondiente a la base de datos activa con los siguientes datos: número de funciones, número de tablas, número de reglas, cantidad de índices y total de bloqueos.	
Observaciones: ninguna	

2.6 Lista de reserva del producto

Capítulo 2. Características del sistema propuestos

La lista de reserva del producto es una tabla(Ver tabla 2) que contiene los requisitos funcionales que debe cumplir la aplicación que se desea realizar, ordenados según la prioridad de implementación, ubicados en Muy Alta, Alta, Media y Baja, en esta última aparecen los requisitos de menor complejidad, además de los requisitos no funcionales del sistema a desarrollar. Indica la estimación de cada uno de ellos y su implementación por semanas y quien hizo la estimación.

Tabla 2 Lista reserva del producto

Ítem *	Descripción	Estimación	Estimado por
Prioridad: Muy Alta			
1	Mostrar cantidad de conexiones por base de datos.	0.6 semanas	Analista
2	Mostrar tuplas eliminadas por base de datos.	0.6 semanas	Analista Analista
3	Mostrar tuplas insertadas por base de datos.	0.6 semanas	Analista
4	Mostrar tuplas actualizadas por base de datos.	0.6 semanas	Analista
5	Mostrar tuplas retornadas por base de datos.	0.6 semanas	Analista
6	Mostrar cantidad de funciones.	0.5 semanas	
7	Mostrar total de bloqueos.	0.5 semanas	Analista
8	Mostrar cantidad de reglas.	0.5 semanas	Analista
9	Mostrar cantidad de índices.	0.5 semanas	Analista
10	Mostrar cantidad de tablas.	0.5 semanas	Analista
11	Mostrar suma de tuplas muertas.	0.5 semanas	Analista
Prioridad: Alta			
12	Mostrar tuplas actualizadas por tablas.	0.5 semanas	Analista Analista
13	Mostrar tuplas insertadas por tablas.	0.5 semanas	Analista
14	Mostrar tuplas eliminadas por tablas.	0.5 semanas	Analista
15	Mostrar tuplas muertas por tablas.	0.5 semanas	
Prioridad: Media			

16	Obtener tamaño de todas las base de datos.	0.5 semanas	Analista
17	Mostrar tamaño que ocupan las base de datos en el clúster.	0.5 semanas	Analista
18	Obtener tamaño de todas las tablas de una base de datos.	0.5 semanas	Analista
19	Graficar tamaño de todas las tablas.	0.5 semanas	Analista
20	Obtener tamaño de índices de todas las tablas.	0.5 semanas	Analista
21	Mostrar tamaño de índice de todas las tablas.	0.5 semanas	Analista
Prioridad: Baja			
	Usabilidad. La aplicación podrá ser usada por cualquier usuario que desee monitorizar su base de datos. **Portabilidad.** Será un sistema multiplataforma, lo que permitirá poder disponer del mismo en cualquier sistema operativo. **Software.** Para poder hacer uso de este Plugin se debe tener instalado las librerías de QT y el compilador de C++. **Apariencia o interfaz externa.** La aplicación al integrarse con la herramienta HABD, tendrá una		

	interfaz amigable y fácil de interactuar para hacer más factible el uso para los usuarios. **Disponibilidad.** Se podrá hacer uso de la aplicación siempre que esté instalado todas las librerías necesarias. **Restricciones de diseño e implementación.** Para el desarrollo del Plugin se usará el lenguaje de programación C++ con el framework QT.		

2.7 Tarea de la ingeniería

El equipo de desarrollo evalúa cada historia de usuario y lo divide en tareas, donde cada una de estas representa una característica del sistema. A continuación aparece una tarea de la ingeniería (TI) (Ver tabla 3) donde se muestra el encargado de programarla y una descripción breve de lo que se necesita. La siguiente tabla muestra la TI Mostrar número de funciones y el resto de ellas se encuentran en el documento correspondiente a las TI del expediente de proyecto.

Tabla 3 Tarea de la ingeniería "Mostrar número de funciones"

Tarea de Ingeniería	
Número Tarea:1	**Número Historia de Usuario:** 1
Nombre Tarea: Mostrar número de funciones	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.75

Fecha Inicio:9/1/2012	**Fecha Fin:**13/1/2012
Programador Responsable: Beatriz Piñeiro Veitía	
Descripción: Mostrará en forma de texto el número de funciones que tiene la base de datos activa.	

2.8 Plan de Iteración

Luego de tener definidas las historias de usuarios se debe crear un plan de iteraciones, indicando cuáles se desarrollarán en cada iteración del programa. El obejtivo de este artefacto es tener una planificación del trabajo, donde los desarrolladores y el cliente establecen los tiempos de implementación de las historias de usuarios y la prioridad con que serán desarrolladas. La siguiente tabla(Ver tabla 4) muestra el plan de iteraciones para el Plugin de monitorización para la herramienta de administración de base de datos HABD.

Tabla 4 Plan de iteración

Release	Descripción de la iteración	Orden de la HU a implementar	Duración total

1	En esta iteración se implementarán las historias de usuarios correspondientes a las estadísticas de base de datos y clúster.	2 y 4	6 semanas
2	En esta iteración se implementará la historia de usuario correspondiente a estadísticas de tablas.	6	2 semanas
3	En esta iteración se implementarán las historias de usuario correspondiente a mostrar tamaño de todas las bases de datos y tamaño de todas las tablas.	1, 3 y 5	3 semanas

2.9 Modelo de diseño

Capítulo 2. Características del sistema propuestos

En la fase de diseño de la metodología XP se sugiere tener diseños simples y sencillos, para conseguir un mejor entendimiento e implementación consiguiendo reducir el tiempo y esfuerzo a la hora de desarrollar. Esta fase incluye las tarjetas clase-responsabilidades-colaboración (CRC) y diagrama de clases.

2.9.1 Diagrama de clases

Los diagramas de clases se utilizan para representar la estructura estática de un sistema incluyendo una colección de elementos, tales como, clases y relaciones. (11) Un diagrama de clases está compuesto por los siguientes elementos:

- Clase: atributos, métodos y visibilidad.
- Relaciones: herencia, composición, agregación, asociación y uso.

A continuación (ver figura 2) se observa el diagrama de clases para el Plugin de monitorización, presentando un total de ocho clases.

IConsulta: Interfaz que contiene todos los métodos para conocer las estadísticas de las tablas, base de datos y clúster.

Consulta8_4: Clase que hereda de IConsulta, implementando los métodos para la versión 8.4 de PostgreSQL.

Consulta9_0: Clase que hereda de consulta8_4, para redefinir los métodos de acuerdo a la versión 9.0.

Consulta9_1: Clase que hereda de consulta9_0, para redefinir los métodos de acuerdo a la versión 9.1.

Control: Clase controladora encargada de manejar el negocio de la aplicación.

Fabricacion: Clase encargada de crear objetos de tipo consulta según la versión de PostgreSQL en la que se establezca la conexión.

Mainwindows: Clase que crea la interfaz de la aplicación.

Dibujar: Se encarga de crear las gráficas con los datos obtenidos de las consultas realizadas a la base de datos.

Capítulo 2. Características del sistema propuestos

Capítulo 2. Características del sistema propuestos

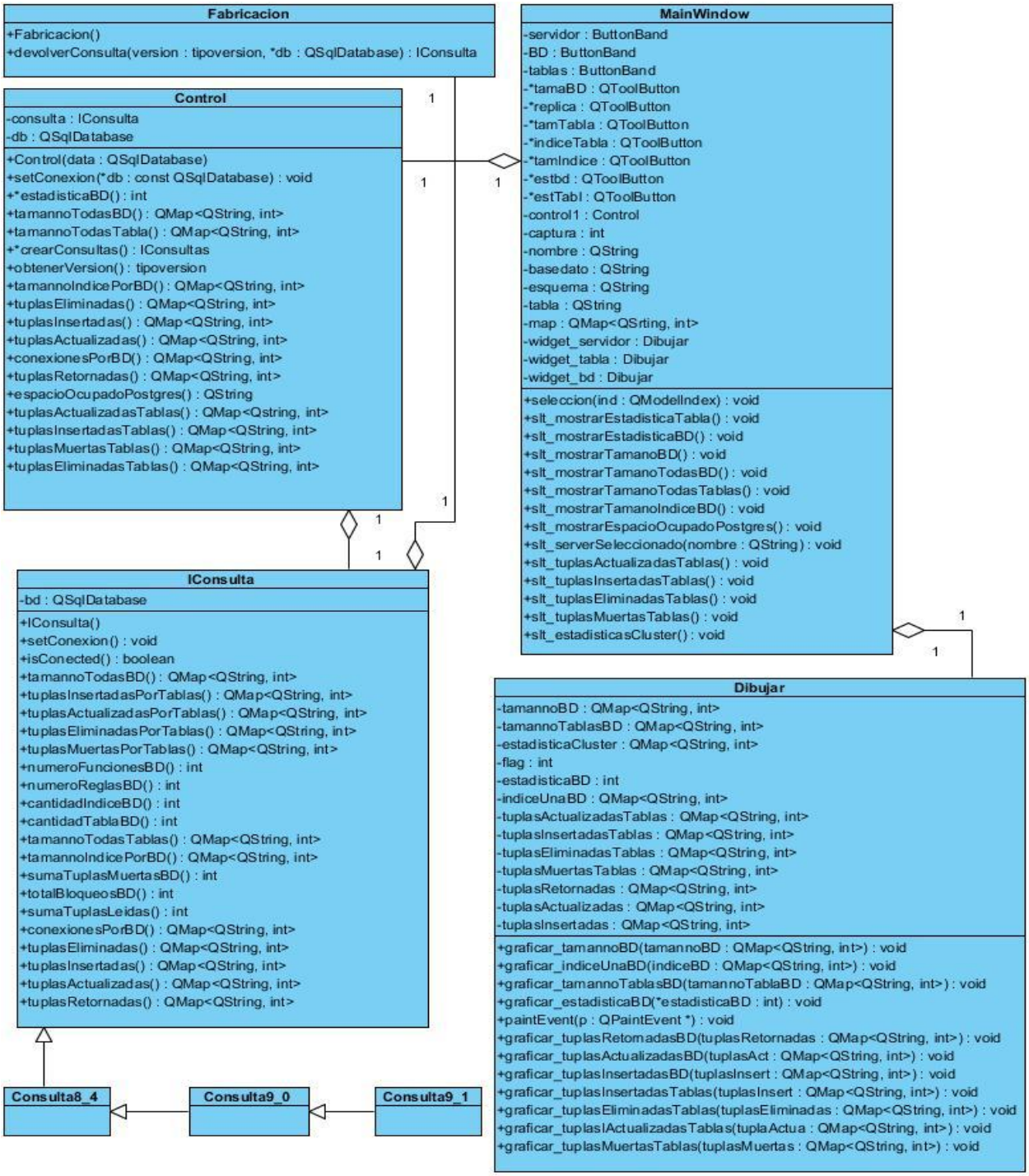


Figura 2 Diagrama de clases del Plugin monitorización

2.9.2 Tarjeta clase-responsabilidades-colaboración

El uso de las tarjetas CRC permite al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de la programación procedural clásica. Las tarjetas CRC están divididas en tres partes:

Clase: representa una colección de objetos similares.

Responsabilidades: es aquello que la clase sabe o hace. En ocasiones la misma no tiene toda la información para hacerlo. Esto hace que deba interactuar con otras clases.

Colaboración: toman dos formas, un pedido de información o un pedido a que se realice una acción.

Tabla 5 Tarjeta CRC para la clase Estadística

Tarjeta CRC	
Clase: Estadísticas	
Responsabilidades	**Colaboraciones**
Mostrar cantidad de conexiones por base de datos. Mostrar tuplas eliminadas por base de datos. Mostrar tuplas insertadas por base de datos. Mostrar tuplas actualizadas por base de datos. Mostrar tuplas retornadas por base de datos. Mostrar número de funciones. Mostrar total de bloqueos. Mostrar cantidad de índices. Mostrar número de reglas. Mostrar cantidad de tablas. Mostrar cantidad de tuplas muertas. Mostrar tuplas actualizadas por tablas. Mostrar tuplas insertadas por tablas. Mostrar tuplas eliminadas por tablas. Mostrar tuplas muertas por tablas.	

Capítulo 2. Características del sistema propuestos

2.10 Arquitectura de software

En los inicios de la informática, se desconocía una serie de formas o guías para facilitar la programación, con el objetivo de simplificar el trabajo que pasaban los desarrolladores en aquel entonces surge la Arquitectura de software, que indica la estructura y funcionamiento entre las partes del software.

2.10.1 Patrón de arquitectura para el desarrollo del Plugin

En el desarrollo del Plugin de monitorización se utiliza una arquitectura en tres capas, empleando el patrón modelo vista controlador (MVC). Este patrón separa la lógica de negocio de la interfaz de usuario, facilita la evolución por separado de ambos aspectos, incrementa reutilización y flexibilidad. Para tener una visión más detallada, se explicará en que consiste cada una de las capas.

Modelo: Esta es la representación específica de la información con la cual el sistema opera. El modelo se limita a lo relativo de la vista y su controlador facilitando las presentaciones visuales complejas. El sistema también puede operar con más datos no relativos a la presentación, haciendo uso integrado de otras lógicas de negocio y de datos afines con el sistema modelado.

Vista: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

Controlador: Este responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelo y, probablemente, a la vista. (12)

En la siguiente figura (ver figura 3) se muestran las clases que se agrupan en cada una de las capas del patrón de arquitectura escogido. En la capa modelo se agrupan las clases que van a tener acceso a los datos, se encuentran todas las consultas realizadas a la base de datos para obtener toda la información que se desea mostrar al usuario mediante gráficas. En la capa contolador, la clase Control se encarga de recopilar toda la información de la capa presentada anteriormente y le envía los datos a las clases de la capa Vista. En esta última capa las clases Mainwindow y Dibujar son las encargadas de graficar los datos que reciben y mostrarlos al usuario.

Capítulo 2. Características del sistema propuestos

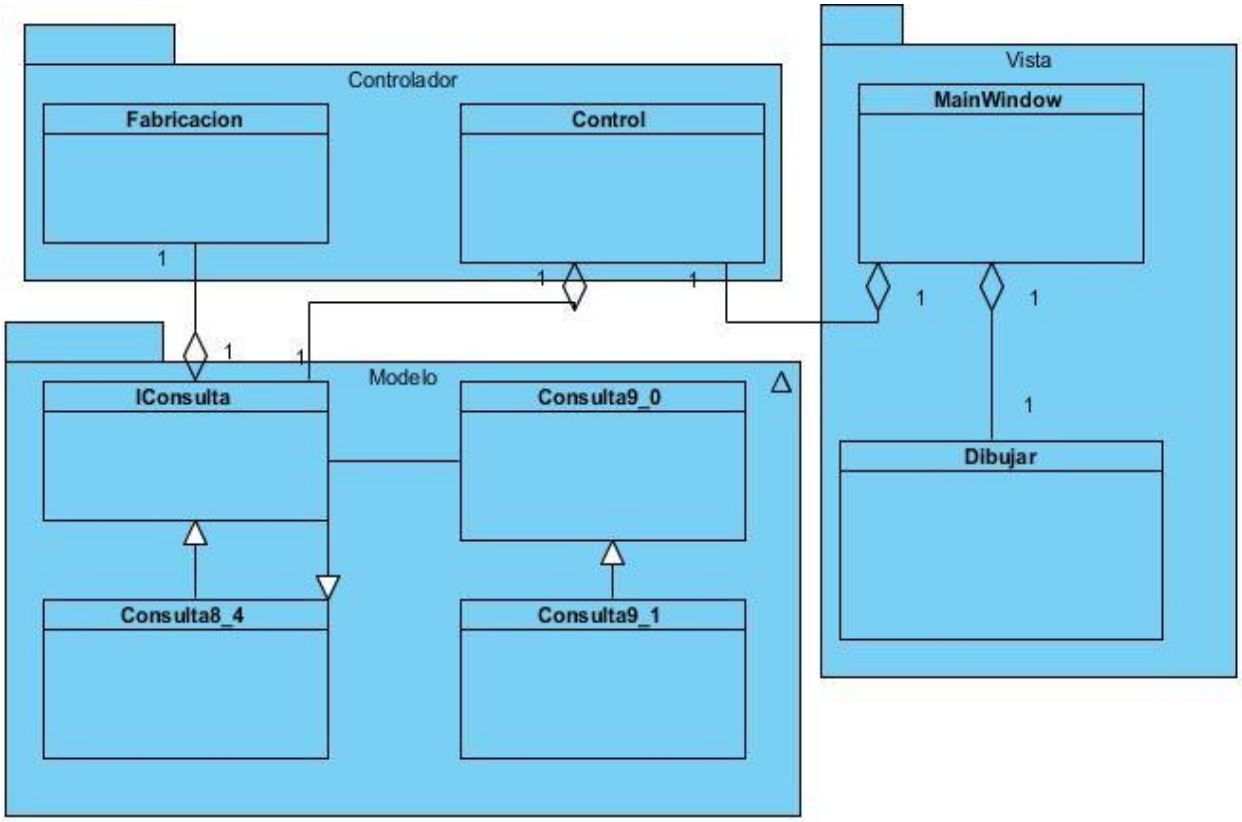


Figura 3 Patrón Modelo Vista Controlador

2.10.2 Patrones de diseño empleado

Los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan.

Los patrones de diseños brindan gran ayuda a la hora de resolver problemas de diseños ya que sugieren clases y objetos. Proponen además interfaces entre objeto, ejemplo de esto son las clases y operaciones abstractas. Permiten la reutilización del código. Los patrones tienen la facilidad de implementarse de varias formas siempre que se mantengan estables las interfaces. (13)A continuación se presentan los patrones empleados en el Plugin de monitorización.

Fábrica

Se utiliza para crear distintos objetos pertenecientes a una misma familia. Este patrón se evidencia en la clase Fabricación que es la encargada de crear objetos de tipo IConsulta según la versión de PostgreSQL.

Observer

La idea principal detrás del patrón observer es que existe una entidad con estados cambiantes y una o más entidades observándola. Los observadores esperan a que la entidad observada les informe de un cambio de estado a través de un evento que puede ser de su interés, por lo que los observadores se registran con la entidad observada. Este patrón se evidencia en la clase Mainwindows, cuando los componentes visuales emiten eventos al usuario interactuar con la aplicación.

Controlador

El patrón creador nos ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. Este patrón se evidencia en la clase Control, donde se maneja el negocio de la aplicación.

2.11 Estándares de codificación

Usar técnicas de codificación y realizar buenas prácticas de programación con vista a generar un código de alta calidad, es de gran importancia para el software que se desarrolla y para obtener un buen rendimiento del mismo. Además, si se aplica de forma correcta el estándar definido facilita añadir nuevas características, modificar las ya existentes, depurar errores, o mejorar su rendimiento. El propósito de las siguientes reglas y recomendaciones es lograr que los programadores del proyecto PostgreSQL tengan un estilo de código común**.**

Identación

La unidad de identado es de cuatro espacios. El uso de la tabulación debe ser evitado porque (tal como se escribía en el siglo pasado) no existe un estándar que determine con precisión el ancho que va a producir la tabulación.

Longitud de la Línea

Evitar líneas con más de ochenta caracteres. Cuando una sentencia sobrepase una línea simple del editor, esta deber ser separada en otras. Realice la separación después de un operador, preferentemente luego de una coma. Realizar la ruptura después de un operador disminuye la probabilidad de que al realizar un copiado del código se cometa un error por olvido de la inserción del punto y coma. La siguiente línea debe ser identada con cinco espacios.

Comentarios

Es conveniente dejar información que pueda ser leída tiempo después por personas (posiblemente usted mismo) que necesitan entender que fue lo que se hizo en el fragmento de código. Los comentarios deben ser escritos correctamente y claros.

Generalmente deben usarse comentarios de una sola línea. Reserve los comentarios de bloques para la documentación formal o para comentar porciones de código.

Declaración de variables

Cada variable debe de ser declarada en una línea y comentada. También deben aparecer ordenadas alfabéticamente:

QMap<DependenceContainer*,bool>installedInstances; //plugins instalados

QList<DependenceContainer*>loadedInstancesLst; //plugins cargados

El nombre de las variables debe de comenzar con letras minúsculas y cada palabra relevante por la que esté compuesta debe ser con letra mayúscula.

Ejemplo: loadedInstancesLst, installedInstances.

Declaración de funciones

No debe haber espacio entre el nombre de la función y el paréntesis izquierdo, ni entre este y la lista de parámetros. Debe haber un espacio entre el paréntesis derecho y la llave de comienzo del cuerpo de la función. Las sentencias del cuerpo deben estar en la línea siguiente. La llave que cierra debe estar alineada con la línea de declaración de la función. Los nombres de las funciones se rigen por las mismas características que el de las variables.

functionejemploPrincipal(c, d)

```
{

var e = c * d;

return inner(0, 1);

}
```

Identificadores

Los identificadores pueden estar formados por cualesquiera de las 26 letras minúsculas o mayúsculas (A... Z, a... z), los 10 dígitos (0... 9) y el caracter subrayado "_". Debe evitarse el uso de caracteres internacionales (ej.: ñ, ü) porque no siempre pueden ser leídos o entendidos correctamente en todos los lugares. No se debe usar el símbolo dólar "$" o la barra invertida "\" en los identificadores.

Sentencias

Sentencias Simples

Cada línea debe contener como máximo una sentencia. Debe poner un punto y coma ";" al final de cada sentencia simple. Tenga en cuenta que una sentencia de asignación puede resultar en la asignación de una función o de un objeto como literal y en todos los casos como sentencia de asignación debe estar finalizada con un punto y coma.

Sentencias Compuestas

Las sentencias compuestas son aquellas sentencias que contienen una lista de sentencias encerradas entre llaves:

- Las sentencias encerradas deben ser identadas a cuatro espacios.
- La llave que inicia la lista de sentencias debe estar al final de línea de la sentencia compuesta.
- La llave que termina la lista de sentencias debe estar al comienzo de una línea y guardar la misma Identación que la sentencia compuesta en correspondencia con la llave que inicia.

- Las llaves siempre serán usadas para listar todas las sentencias, aunque se trate de una sola, cuando son parte de una estructura de control como if o for. Ello facilita agregar nuevas sentencias sin la introducción accidental de errores.

Etiquetas

Las sentencias etiquetadas son opcionales. Solo estas sentencias deben ser etiquetadas: while, do, for, foreach, switch.

Sentencia return

Una sentencia return no debe utilizar paréntesis "()" alrededor del valor que se retorna. La expresión cuyo valor se retorna debe comenzar en la misma línea que la palabra reservada return, terminada con un punto y coma.

Sentencia if

La sentencia if debe ser escrita de esta manera:

```
if (condition)
{
statements
}

if (condition)
{
statements
} else
{
statements
}
if (condition)
 {
```

```
statements
}
else if (condition)
{
statements
}
else
{
statements
}
```

Estructuras repetitivas

Las estructuras repetitivas deben ser escritas de esta manera:

```
for (initialization; condition; update)
{
statements
}

while (condition)
{
statements
}

foreach (valor1, valor2)
{
statements
}
```

Sentencia switch

La sentencia switch debe ser escrita de la siguiente forma:

```
switch (expression)
{
case expression:
statements
case expression:
statements
default:
statements
}
```

Espacios en blanco

Las líneas en blanco facilitan la lectura determinando secciones de código lógicamente relacionada. Los espacios en blanco pueden ser (o no deben ser) utilizados en las siguientes circunstancias:

- No se debe utilizar un espacio en blanco entre el identificador de una función y el paréntesis que abre a la lista de parámetros. Ello ayuda a distinguir entre palabras reservadas y llamadas a funciones.
- Para cualquier operador binario excepto el punto ".", el paréntesis que abre "(" y el corchete que abre "[" todos deben ser separados por un espacio entre operando y operador.
- No se debe utilizar el espacio para separar un operador unario de su operando excepto cuando ese operador es una palabra como type of.
- Cada punto y coma ";" en una sentencia de control debe ser seguido por un espacio.
- Debe dejarse un espacio luego de cada coma ",".

Declaraciones de clases

Solo debe existir un fichero con más de una clase declarada.

Ejemplo incorrecto:

fichero.h

```
class A
{
}

class B
{
}
```

Ejemplo correcto:

ficheroA.h

```
class A
{
}
```

ficheroB.h

```
class B
{
}
```

2.12 Interfaces de la aplicación

A continuación se presentarán una serie de imágenes donde se evidencia la integración del Plugin en la herramienta de administración de base de datos HABD, así como varias funcionalidades: estadísticas del clúster, estadísticas de la base de datos y tamaño de todas las tablas.

En la siguiente figura (ver figura 4) se muestra la cantidad de tuplas retornadas por base de datos, formando parte de las estadísticas del clúster. Se evidencia además el tamaño ocupado por las base de datos en PostgreSQL.

Capítulo 2. Características del sistema propuestos

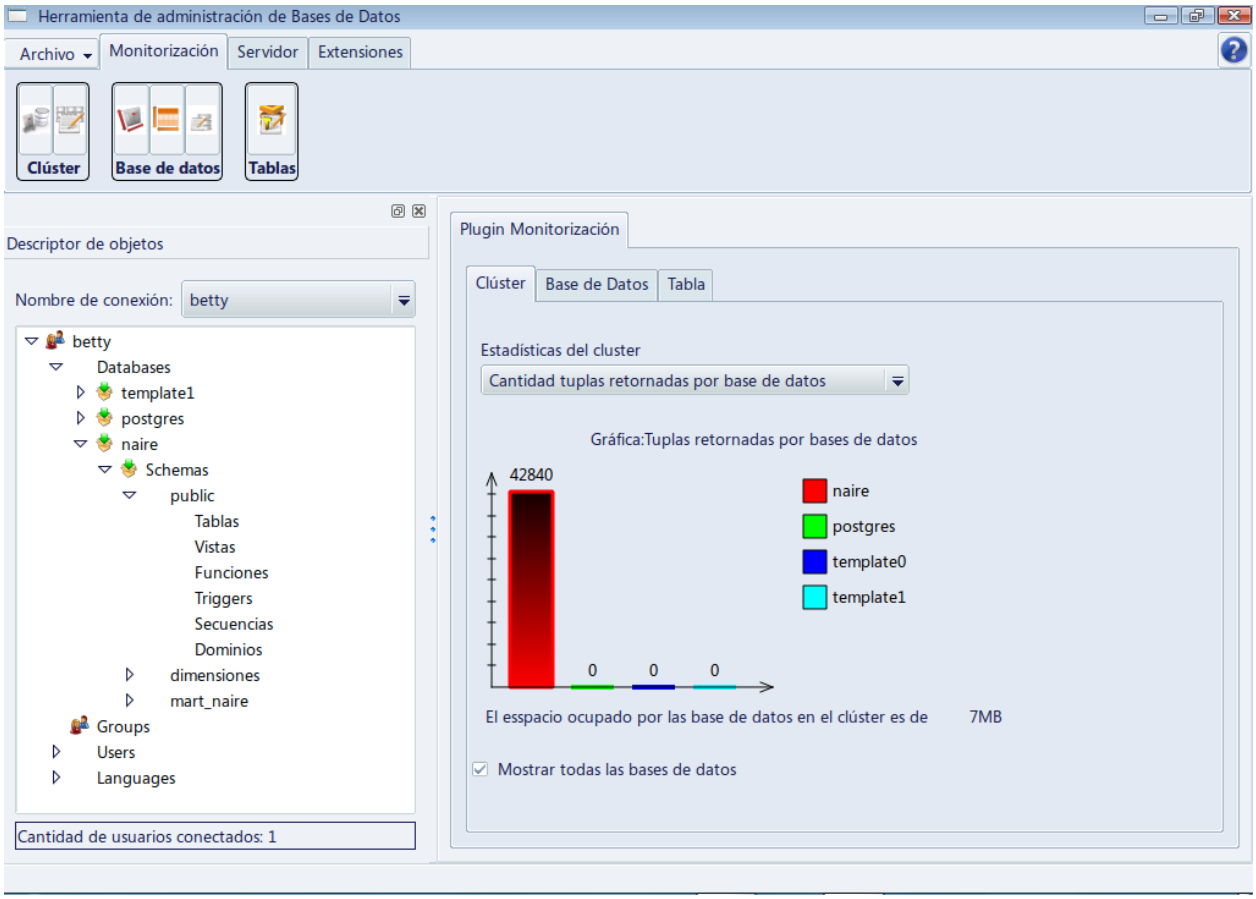


Figura 4 Interfaz cantidad de tuplas retornadas por base de datos

En la siguiente imagen (ver figura 5) se muestra la funcionalidad tamaño de todas las tablas para la base de datos activa. En esta misma pestaña de base de datos se muestran las estadísticas de base de datos y el tamaño de índice de todas las tablas.

Capítulo 2. Características del sistema propuestos

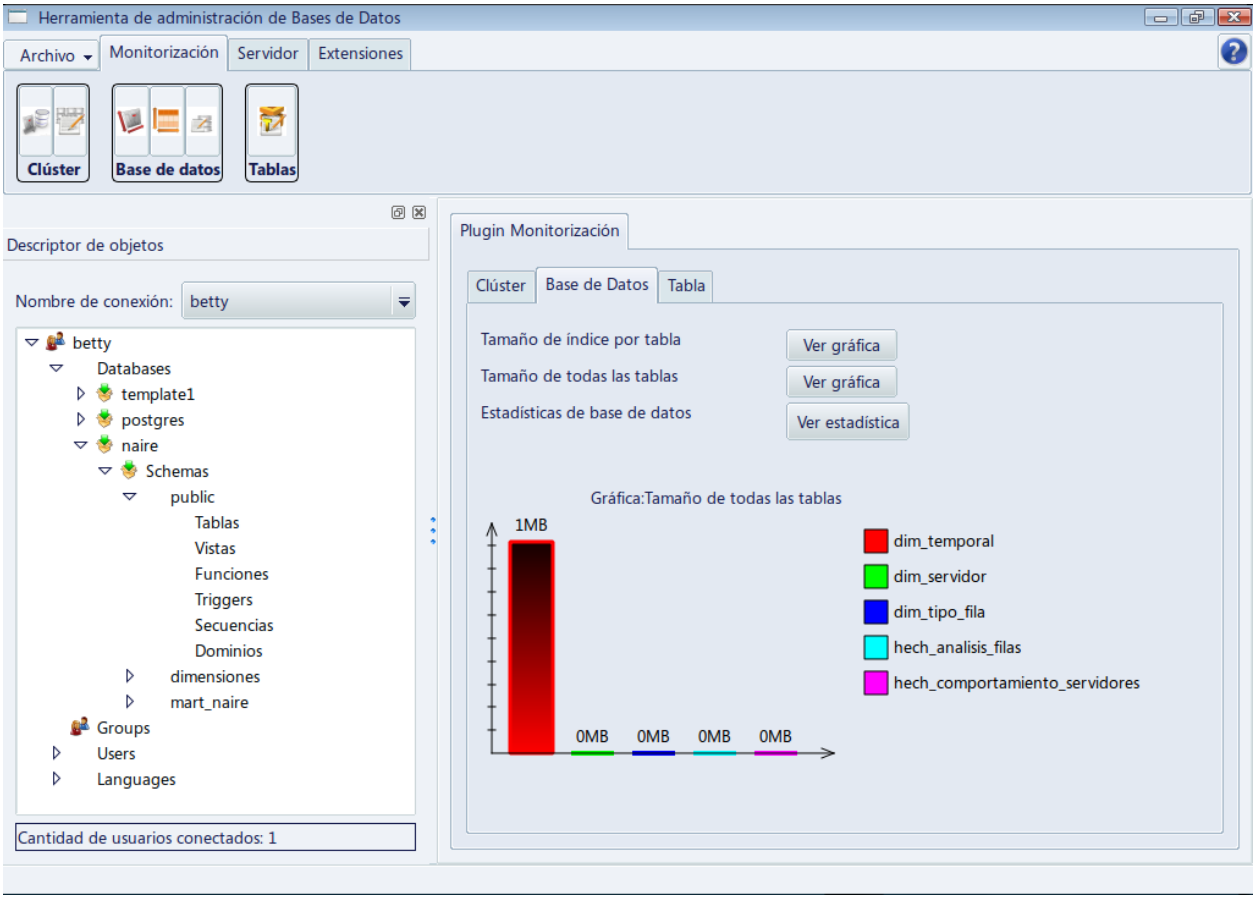


Figura 5 Interfaz Tamaño de todas las tablas

A continuación se muestra (ver figura 6) la interfaz generada para la funcionalidad estadísticas de base de datos, la figura muestra, número de funciones, cantidad de bloqueos, tablas, índices, reglas y tuplas muertas.

Capítulo 2. Características del sistema propuestos

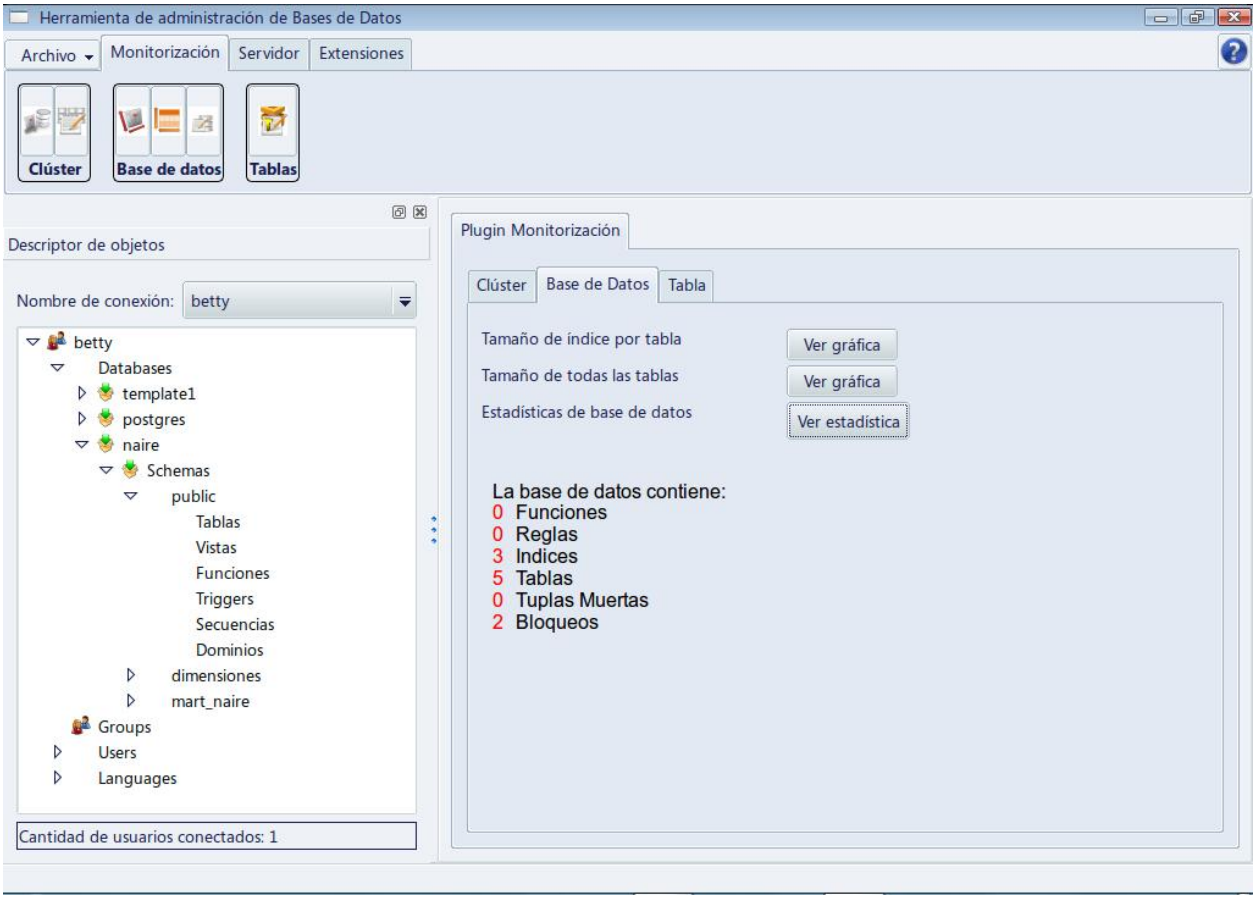


Figura 6 Interfaz Estadísticas de base de datos

2.13 Conclusiones parciales

En este capítulo se definen las características del sistema propuesto para el Plugin. Se identificaron seis historias de usuarios y se estableció en que iteración y prioridad se desarrollarían. Se definió un diagrama de clases del diseño y modelo de dominio, artefactos que no son generados por la metodología utilizada pero es necesario para lograr mejor entendimiento en la implementación. Se identificaron los patrones de diseños y el patrón de arquitectura a utilizar. Se realiza una breve descripción de algunas de las interfaces de la aplicación.

Capítulo 3. Validación del sistema propuesto

Capítulo 3: Validación del Plugin de monitorización

3.1 Introducción

Para determinar si el resultado de un producto es el esperado, se hace necesario someterlo a pruebas, las mismas son las que permiten determinar su calidad. Se integran a las diferentes fases del ciclo de desarrollo dentro de la ingeniería de software, manifestándose mediante la ejecución de un programa o mediante técnicas experimentales con el propósito de descubrir errores. En este capítulo se analizan las estrategias de pruebas definidas por XP, el método que se utilizará para diseñar los casos de pruebas y los resultados arrojados por estas.

3.2 Enfoques de diseños de prueba·

Las metodologías ágiles no consideran las pruebas como un conjunto de niveles que hay que ir superando para alcanzar la validación final del sistema que se está desarrollando. Las metodologías ágiles presentan distintos enfoques del proceso de desarrollo que vienen determinados por los tipos de pruebas que se realizan. (14)

La metodología XP propone para validar las necesidades de los usuarios y dirigir la implementación, las pruebas unitarias y las de aceptación. Las unitarias están encomendadas a verificar el código, son diseñadas por los programadores, garantizan que un determinado módulo cumpla con un comportamiento esperado antes de ser integrado al sistema. Se emplean cuando la implementación es complicada y la interfaz de un método no es clara. (15)

Las pruebas unitarias se realizan para controlar el funcionamiento de pequeñas porciones de código. Generalmente son realizadas por el mismo programador, debido que al conocer con mayor detalle el código, se les simplifica la tarea de elaborar conjuntos de datos de prueba para testearlo. (15)

Las pruebas de aceptación, se realizan sobre el producto terminado e integrado, están concebidas para que sea un usuario final quien detecte los posibles errores. Se clasifican en dos tipos: pruebas Alfa y pruebas Beta.

Capítulo 3. Validación del sistema propuesto

Las pruebas Alfa se realizan por un cliente en un entorno controlado por el equipo de desarrollo. Para que tengan validez, se debe primero crear un ambiente con las mismas condiciones que se encontrarán en las instalaciones del cliente. Una vez logrado esto, se procede a realizar las pruebas y a documentar los resultados. (16)

Las pruebas Beta se realizan en las instalaciones propias de los clientes. Se realizan copias del sistema y es distribuido para cada uno de ellos. Cada usuario realizará sus propias pruebas y documentará los errores que encuentre, así como las sugerencias que crea conveniente realizar para que el equipo de desarrollo tenga en cuenta al momento de analizar las posibles modificaciones. Cuando el sistema tenga un solo cliente, las pruebas de aceptación se hacen de común acuerdo con éste y se definen los aspectos a probar y la forma de informar resultados.

3.2.1 Método seleccionado

El estudio realizado en el epígrafe anterior ayudó a determinar qué estrategia de prueba emplear. Se decide utilizar para el desarrollo del Plugin, la estrategia de prueba aceptación de tipo alfa, con la técnica de caja negra.

Las pruebas de aceptación son más importantes que las unitarias, puesto que representan la satisfacción del cliente con el producto desarrollado. Es más efectivo que el cliente de una aceptación del producto que desea, a que el producto lo pruebe el programador. Estas pruebas conllevan al cliente a precisar lo que la aplicación debe hacer en determinadas circunstancias, por esto, el cliente es la persona adecuada para diseñar las pruebas. Las pruebas unitarias no descubrirán todos los errores del código. Por definición, sólo prueban las unidades por sí solas. Por lo tanto, no descubrirán errores de integración, problemas de rendimiento y otros problemas que afectan a todo el sistema en su conjunto.

3.3 Casos de pruebas basados en historias de usuarios

Los casos de pruebas se realizan con el objetivo de determinar que una funcionalidad ha sido implementada satisfaciendo las necesidades del cliente. Para cada historia de usuario debe estar asociado un caso de prueba, estos se derivan cuando se aplica alguna técnica, en este caso la de caja negra. Los casos de prueba obtenidos al aplicar esta técnica, pretenden demostrar que

Capítulo 3. Validación del sistema propuesto

las funciones del software son operativas, que la entrada se acepta correctamente, y se produce una salida correcta.

Las pruebas de caja negra se aplican a la interfaz de la aplicación, son totalmente indiferentes al comportamiento interno y la estructura del programa. Están centradas en realizar pruebas a las funcionalidades del software.

Para probar las funcionalidades se diseñaron un total de seis casos de pruebas correspondientes a las seis historias de usuarios identificadas y descritas en el capítulo dos. A continuación se muestra el caso de prueba asociado a la historia de usuario Estadísticas del clúster.

Capítulo 3. Validación del sistema propuesto

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Cantidad de conexiones	Se muestra una tabla con el ip,la base de datos,aplicación y usuarios conectados	Se muestra una tabla con el ip,la base de datos,aplicación y usuarios conectados	1. El usuario despliega el combobox y selecciona la opción "Cantidad de conexiones por base de datos".
EC 1.2 Tuplas retornadas por base de datos	Se muestra una gráfica en forma de barras con todas las base de datos y la cantidad de tuplas que tiene retornadas.	Se muestra una gráfica en forma de barras con todas las base de datos y la cantidad de tuplas que tiene retornadas.	1. El usuario despliega el combobox y selecciona la opción "Cantidad de tuplas retornadas por base de datos".
EC 1.3 Tuplas actualizadas por base de datos	Se muestra una gráfica en forma de barras con todas las base de datos y la cantidad de tuplas que tiene actualizadas	Se muestra una gráfica en forma de barras con todas las base de datos y la cantidad de tuplas que tiene actualizadas	1. El usuario despliega el combobox y selecciona la opción "Cantidad de tuplas actualizadas por base de datos".
EC 1.4 Tuplas insertadas por base de datos	Se muestra una gráfica en forma de barras con todas las base de datos y la cantidad de tuplas que tiene insertadas.	Se muestra una gráfica en forma de barras con todas las base de datos y la cantidad de tuplas que tiene insertadas.	1. El usuario despliega el combobox y selecciona la opción "Cantidad de tuplas insertadas por base de datos".

Figura 7 Caso de prueba Estadísticas del clúster.

Después de haber sido aplicados los casos de pruebas al Plugin de monitorización se detectaron en una primera iteración cinco no conformidades, las mismas fueron resueltas en período de tiempo de cinco días. Se realizó una segunda iteración y se detectaron dos no conformidades siendo solucionadas en un día. En una tercera iteración no se detectaron no conformidades siendo eliminadas en su totalidad en las dos iteraciones anteriores. En la siguiente figura se muestra una gráfica con la cantidad de no conformidades que se detectaron por cada iteración.

Capítulo 3. Validación del sistema propuesto

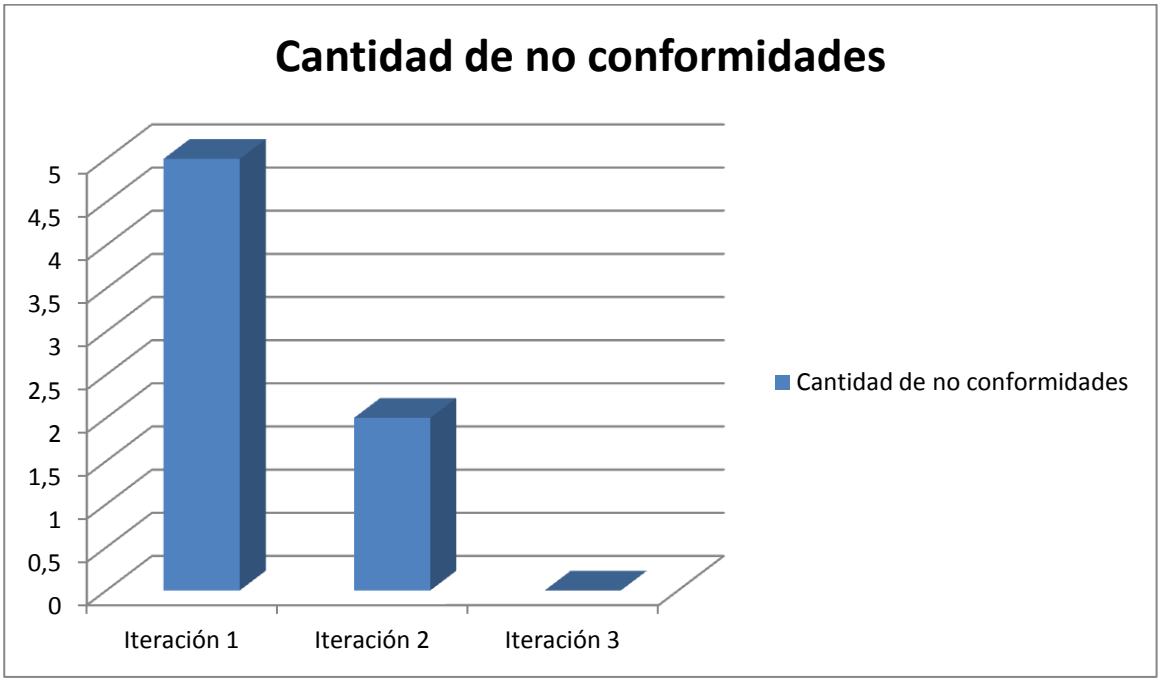


Figura 8 Gráfica: resultados de iteraciones de prueba.

3.4 Conclusiones parciales

Se realizó un estudio para determinar que estrategias de pruebas emplear, siendo las pruebas de aceptación las más convenientes para el Plugin de monitorización, realizándose el diseño de seis casos de pruebas uno por cada Historia de Usuario, utilizando la técnica de caja negra. Se obtuvo un total de siete no conformidades las cuales fueron solucionadas en tres iteraciones.

Conclusiones generales

Conclusiones generales

Se identificaron las características comunes y específicas de cada una de las herramientas de monitorización estudiadas, las cuales determinaron los requerimientos del Plugin.

Mediante el modelo de diseño se determinaron las ocho clases del Plugin.

El Plugin cuenta con 21 requisitos funcionales, se implementaron ocho métricas de monitorización y fue integrada a la herramienta HABD, permitiéndole a la misma monitorizar los servidores PostgreSQL para apoyar a la toma de decisiones.

Mediante los casos de pruebas basados en historias de usuarios se validaron las seis historias de usuario del Plugin verificando el correcto funcionamiento de la aplicación partiendo del diseño propuesto.

Recomendaciones

Para extender la investigación presentada en este trabajo de diploma se recomienda:

- ✓ Modificar las gráficas para que sean interactivas.
- ✓ Continuar con la implementación de otras métricas de monitorización con el objetivo de aportar más información de las base de datos.

Referencias bibliográficas

1. **Yunta, Luis Rodríguez.** *BASES DE DATOS DOCUMENTALES: ESTRUCTURA Y PRINCIPIOS .* MALDONADO,Angeles : s.n., 2001.

2. **Pérez, Teresa Garzón.** *Sistemas Gestores de Bases de Datos.* 2010. 1988-6047.

3. **Martinez, Rafael.** Postgresql.org.es. [En línea] 17 de julio de 2009. [Citado el: 10 de 10 de 2011.] http://www.postgresql.org.es/node/313.

4. —. PostgreSQL-es. [En línea] 11 de febrero de 2011. http://www.postgresql.org.es/node/582.html.

5. ManageEngine. [En línea] [Citado el: 14 de 3 de 2012.] http://me.zma.com.ar/caracteristicas-de-producto-me-applications-manager-9-32.html.

6. MTbase. [En línea] [Citado el: 12 de 3 de 2012.] http://www.mtbase.com/productos/gestionbasesdedatos/scc.

7. **Garrido, Salvador Alemany.** [En línea] 21 de noviembre de 2009. [Citado el: 12 de 10 de 2012.] http://www.polinux.upv.es/drupal/files/IntroduccionQt.pdf.

8. [En línea] 15 de mayo de 2009. [Citado el: 1 de 11 de 20012.] http://www.glatelier.org/2009/05/qt-creator-desarrollando-aplicaciones-rapidamente/.

9. **Jalón, Javier García de.** [En línea] [Citado el: 23 de 11 de 2011.] http://mat21.etsii.upm.es/ayudainf/aprendainf/Cpp/manualcpp.pdf.

10. **Sierra, María.** [En línea] [Citado el: 24 de enero de 2012.] http://personales.unican.es/ruizfr/is1/doc/lab/01/is1-p01-trans.pdf.

11. **Zapata, Mª Antonia.** [En línea] 2006. [Citado el: 2 de 11 de 2012.] http://ocw.unizar.es/ciencias-experimentales/modelos-matematicos-en-bases-de-datos/uml/03UML_DiagramaClases.pdf.

12. **Mestras, Juan Pavón.** [En línea] 2009. [Citado el: 24 de marzo de 2011.] http://www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf.

13. **Laman, Craig.** *Introducción al análisis y diseño orientado a objeto .* PRENTICE HALL, MCxico : s.n., 1999. 970-17-0261-.

14. **Riva, Claudio de la.** *Taller sobre Pruebas en Ingeniería del Software.* 2008. 1988-3455.

15. **Pablo Suárez, Carlos Fontela.** [En línea] 2003. http://materias.fi.uba.ar/7507/content/20101/lecturas/documentacion_pruebas.pdf.

16. **J. J. Gutiérrez, M. J. Escalona, M. Mejías, J. Torres.** [En línea] [Citado el: 8 de mayo de 2012.] http://www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf.

Bibliografía

1. **Yunta, Luis Rodríguez.** *BASES DE DATOS DOCUMENTALES: ESTRUCTURA Y PRINCIPIOS .* MALDONADO,Angeles : s.n., 2001.

2. **Pérez, Teresa Garzón.** *Sistemas Gestores de Bases de Datos.* 2010. 1988-6047.

Bibliografía

3. **Martinez, Rafael.** Postgresql.org.es. [En línea] 17 de julio de 2009. [Citado el: 10 de 10 de 2011.] http://www.postgresql.org.es/node/313.

4. —. PostgreSQL-es. [En línea] 11 de febrero de 2011. http://www.postgresql.org.es/node/582.html.

5. ManageEngine. [En línea] [Citado el: 14 de 3 de 2012.] http://me.zma.com.ar/caracteristicas-de-producto-me-applications-manager-9-32.html.

6. MTbase. [En línea] [Citado el: 12 de 3 de 2012.] http://www.mtbase.com/productos/gestionbasesdedatos/scc.

7. **Garrido, Salvador Alemany.** [En línea] 21 de noviembre de 2009. [Citado el: 12 de 10 de 2012.] http://www.polinux.upv.es/drupal/files/IntroduccionQt.pdf.

8. [En línea] 15 de mayo de 2009. [Citado el: 1 de 11 de 20012.] http://www.glatelier.org/2009/05/qt-creator-desarrollando-aplicaciones-rapidamente/.

9. **Jalón, Javier García de.** [En línea] [Citado el: 23 de 11 de 2011.] http://mat21.etsii.upm.es/ayudainf/aprendainf/Cpp/manualcpp.pdf.

10. **Sierra, María.** [En línea] [Citado el: 24 de enero de 2012.] http://personales.unican.es/ruizfr/is1/doc/lab/01/is1-p01-trans.pdf.

11. **Zapata, Mª Antonia.** [En línea] 2006. [Citado el: 2 de 11 de 2012.] http://ocw.unizar.es/ciencias-experimentales/modelos-matematicos-en-bases-de-datos/uml/03UML_DiagramaClases.pdf.

12. **Mestras, Juan Pavón.** [En línea] 2009. [Citado el: 24 de marzo de 2011.] http://www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf.

13. **Laman, Craig.** *Introducción al análisis y diseño orientado a objeto .* PRENTICE HALL, MCxico : s.n., 1999. 970-17-0261-.

14. **Riva, Claudio de la.** *Taller sobre Pruebas en Ingeniería del Software.* 2008. 1988-3455.

15. **Pablo Suárez, Carlos Fontela.** [En línea] 2003. http://materias.fi.uba.ar/7507/content/20101/lecturas/documentacion_pruebas.pdf.

16. **J. J. Gutiérrez, M. J. Escalona, M. Mejías, J. Torres.** [En línea] [Citado el: 8 de mayo de 2012.] http://www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf.

17. Masadelante.com. [En línea] [Citado el: 10 de 10 de 2012.] http://www.masadelante.com/faqs/plug-in.

Bibliografía

18. manageengine.com. [En línea] [Citado el: 20 de 11 de 2011.] http://www.manageengine.com.mx/products/applications_manager/sybase-monitoring.html.

19. *MySQL Administrator Manual.* s.l. : Sun Microsystems, Inc., 2010.

20. **LÓPEZ, ANDREA SIERRA LOUZÁN y ALBERTO SUÁREZ.** *Bases de Datos.* 2008.

21. **Escofet, Carme Martín.** *El lenguaje SQL.*

22. **A., Ernesto Quiñones.** http://www.apesol.org. [En línea] http://www.apesol.org.

23. *Monitoreando y midiendo el rendimiento de un servidor Postgresql en Sistemas Windows, Linux y Solaris. Parte 1.*

24. **Pressman.** Estrategias de prueba del software cap13.

25. —. Tecnicas de prueba del software cap 14.

26. *Sybase Control Center.* s.l. : 2010 by Sybase, Inc., 2010.

27. **Smith, Gregory.** *PostgreSQL 9.0 High Performance.* 2010. 978-1-849510-30-1.

28. **Simon Riggs, Hannu Krosing.** *Solve real-world PostgreSQL problems with over 100 simple, yet incredibly effective recipes.* 2010. 978-1-849510-28-8.