UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 6

PLUGIN DE MANTENIMIENTO PARA LA HERRAMIENTA DE ADMINISTRACIÓN DE BASES DE DATOS (HABD)

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

Autoras:

Rosa Elvira Morales Martínez

Darlys Milagros Hernández Mitchell

Tutores:

Ing. Marianela Gutiérrez Rodríguez

Ing. Flavio Enrique Roche Rodríguez

La Habana, 2012

"Año 54 de la Revolución."

"El futuro de nuestra patria tiene que ser necesariamente un futuro de hombres de ciencia, tiene que ser un futuro de hombres de pensamiento, porque precisamente es lo que estamos sembrando; lo que más estamos sembrando son oportunidades a la inteligencia".

Fidel Castro Ruz

DECLARACIÓN DE AUTORÍA

Declaramos que somos las únicas autoras de este trabajo y autorizamos a la Facultad 6 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Rosa Elvira Morales Martínez

_____
Firma del autor

Darlys Milagros Hernández Mitchell

_____
Firma del autor

Ing. Marianela Gutiérrez Rodríguez

_____
Firma del tutor

Ing. Flavio Enrique Roche Rodríguez

_____
Firma del tutor

DATOS DE CONTACTO

Tutora:

Ing. Marianela Gutiérrez Rodríguez

Universidad de las Ciencias Informáticas, La Habana, Cuba

Correo electrónico: mgrodriguez@uci.cu

Tutor:

Ing. Flavio Enrique Roche Rodríguez

Universidad de las Ciencias Informáticas, La Habana, Cuba

Correo electrónico: feroche@uci.cu

De: Rosa Elvira

Hoy es un día muy especial, porque se cierra un círculo de mi vida y se abre otro totalmente desconocido. Para lograr cerrarlo existen muchas personas que me han apoyado y brindado su ayuda, este es el momento de agradecerles a todos:

En primer lugar quiero agradecerle a mi padre aunque no se encuentra físicamente entre nosotros, para mí siempre ha sido el hombre perfecto que me da fuerzas para conseguir todo lo que quiero.

A mi madre por ser esa personita que desde que nací me ha dado todo su amor y cariño, me ha apoyado en todas mis decisiones y es la que me da fuerzas cada día para seguir adelante, este sueño hecho realidad va dedicado especialmente a ti.

A mi tía Esther y mi tío Raciel por convertirse en mis segundos padres. Tía quiero agradecerte por enseñarme a ser fuerte y saber que en la vida nunca podemos detenernos. Siempre serás mi ejemplo a seguir.

A mis hermanos por siempre estar ahí cuando los necesito, por confiar en mí y hacerme reír en los momentos más difíciles. Los quiero.

A Carlos y sus padres a los cuales considero parte de mi familia. Carlos este es el mejor momento para agradecerte, lo bueno y dedicado que has sido conmigo, me has demostrado que eres un buen amigo en los momentos buenos y malos, gracias por tu comprensión y ayuda incondicional. "Nunca te olvidaré".

Agradezco a mis tutores y compañera de tesis, sin ustedes este sueño no se hubiese completado.

Y por último quiero agradecerle a Roselys por siempre escucharme y tener una respuesta a mis problemas, por cuidar de mí como una hermana y saber calmarme cuando pienso que todo está perdido. Sé que hoy te vas a sentir muy orgullosa de mí.

AGRADECIMIENTOS

De: Darlys Milagros

Agradezco a mis tíos y mi mamá los cuales me guiaron y apoyaron en el transcurso de toda mi carrera.

Gracias a la ing. Marianela Gutiérrez Rodríguez, el ing. Flavio Enrique Roche Rodríguez y el compañero Carlos Miguel López Durañona por su ayuda, sin la cual no hubiera sido posible la culminación de este trabajo.

En fin a todos aquellos que de una forma u otra hayan colaborado en este trabajo.

RESUMEN

La Universidad de las Ciencias Informáticas promueve el desarrollo de la ciencia y la tecnología mediante la realización de centros productivos. Como parte del desarrollo y soporte a las tecnologías de bases de datos surge el Centro de Tecnologías de Gestión de Datos, el cual está compuesto por varios proyectos, uno de ellos es PostgreSQL, en el que se desarrollan herramientas que facilitan el trabajo con el gestor del mismo nombre. En el año 2010 se implementó la Herramienta de Administración de Bases de Datos, a la cual se le pueden agregar nuevas funcionalidades al ser su arquitectura basada en componentes, actualmente esta herramienta no permite realizar operaciones de salva, restauración y optimización de las bases de datos. En el presente trabajo de diploma se desarrolló un componente de mantenimiento, el cual se integró a la herramienta HABD. Este Plugin permite optimizar las bases de datos, hacer copias de seguridad con regularidad y restaurarlas en cualquier momento. Para cumplir con el objetivo propuesto se realizó un estudio de las diferentes formas de realizar mantenimiento y de algunas herramientas de administración de bases de datos existentes en el mundo que realizan estas tareas, definiendo las funcionalidades a implementar en el Plugin.

Palabras clave: bases de datos, mantenimiento, Plugin.

Tabla de Contenidos

Índice de Tablas

Índice de Figuras

INTRODUCCIÓN

Con las transformaciones que se realizan en la informática se modifica el tratamiento, el almacenamiento, y la transmisión de la información, influyendo en todas las esferas de la actividad humana, el modo de hacer y de pensar del hombre. Las mismas se han desarrollado de manera continua y con un avance exponencial en los últimos años. Debido a esto surge en la sociedad, la necesidad de simplificar y agilizar las tareas relacionadas a la gestión de la información, donde debe garantizarse la calidad y la velocidad de acceso a la misma.

Con el desarrollo de las tecnologías de la información y las comunicaciones (TICs) han aumentado los volúmenes de información. Estas tecnologías se han extendido a todas las esferas de la sociedad, incrementándose considerablemente los datos que necesitan ser almacenados, modificados y accedidos de manera rápida y eficiente, así se masificó el uso de las bases de datos al igual que sus respectivos Sistemas Gestores de Bases de Datos (SGBD). Estos permiten manejar los datos almacenados que posteriormente se convertirán en información, y garantizan que las bases de datos sean más seguras, flexibles y manejables.

En Cuba existen varios centros destinados a impulsar el desarrollo tecnológico del país como es el caso de Desoft, el Centro de desarrollo de software de Villa Clara-Universidad de las Ciencias Informáticas, Transoft y la Universidad de Ciencias Informáticas (UCI). La UCI es un centro de estudios que tiene como propósitos formar profesionales comprometidos con su Patria y altamente calificados en la rama de la Informática, producir aplicaciones y servicios informáticos a partir de la vinculación estudio-trabajo como modelo de formación, servir de soporte a la industria cubana de la informática a través de la producción de software y servicios informáticos que se basan en la integración de los procesos de formación, investigación y producción en torno a una temática para convertirla en una rama productiva. (1)

La UCI está conformada por centros de desarrollo que facilitan la producción de software y servicios informáticos, siendo uno de estos el Centro de Tecnologías de Gestión de Datos (DATEC), que tiene como misión, contribuir al desarrollo de tecnologías de bases de datos. Este centro tiene varios departamentos de producción asociados entre los que se encuentra PostgreSQL, el cual tiene como objetivo desarrollar herramientas que potencien el uso del SGBD PostgreSQL y está dirigido a la

implementación y desarrollo de este gestor como base para sus proyectos. El mismo hace uso de una licencia pública, y no es desarrollado por una sola empresa sino que es dirigido por una comunidad de desarrolladores. (2)

En el departamento PostgreSQL se está llevando a cabo el desarrollo e implementación de funcionalidades para integrarlas a la Herramienta de Administración de Bases de Datos (HABD), la cual surge a partir de investigaciones realizadas sobre los inconvenientes encontrados en las principales herramientas de administración de bases de datos en el mundo. Dicho software posee gran flexibilidad y brinda la posibilidad de incrementar funcionalidades haciendo más factible el uso para los usuarios.

La herramienta HABD tiene como característica fundamental que posee una arquitectura basada en Plugin, lo que facilita a los desarrolladores incorporar nuevos servicios de manera sencilla por la flexibilidad que esto permite. De esta forma se le pueden agregar un gran número de funcionalidades evitando así que los usuarios necesiten de varias aplicaciones para resolver un único problema. El principal inconveniente de esta herramienta es que no posee opciones de mantenimiento como salvar y restaurar, elementos fundamentales para el buen funcionamiento de cualquier herramienta de administración de bases de datos. Después de utilizar de forma intensiva una base de datos esta queda fragmentada, y como consecuencia aumenta su tamaño, además los registros eliminados y la información innecesaria puede ocupar mucho espacio en disco, por lo que contar con algún componente que permita realizar tareas de mantenimiento se convierte en un punto crítico para cualquier herramienta de administración de bases de datos. El mantenimiento para una base de datos siempre es muy importante, si la herramienta HABD no posee estas opciones se privaría de mantener la base de datos en un nivel óptimo y de realizar copias de seguridad con regularidad. Sin embargo, esta frecuencia podría variar de acuerdo con el entorno y con la actividad que recibe cada base de datos.

Teniendo en cuenta lo antes descrito, se plantea como **problema de la investigación:** ¿Cómo contribuir al mantenimiento de las bases de datos PostgreSQL en la Herramienta de Administración de Bases de Datos HABD?

INTRODUCCIÓN

A partir del problema planteado, se define como **objeto de estudio** de la investigación: El proceso de administración de bases de datos en PostgreSQL.

El **campo de acción** se enmarca en las herramientas para el mantenimiento de las bases de datos en PostgreSQL.

Como **objetivo general** de la presente investigación se ha planteado desarrollar un Plugin para la Herramienta de Administración de Bases de Datos, que permita el mantenimiento de las bases de datos en PostgreSQL, del cual se desglosan los siguientes **objetivos específicos:**

1. Analizar las herramientas basadas en el mantenimiento de las bases de datos.
2. Diseñar el Plugin de mantenimiento para la herramienta HABD.
3. Implementar el Plugin de mantenimiento para la herramienta HABD.
4. Probar el Plugin de mantenimiento implementado.

Para dar cumplimiento a los objetivos planteados se proponen las siguientes **tareas de la investigación:**

1. Revisión bibliográfica de las herramientas de administración basadas en el mantenimiento de las bases de datos.
2. Caracterización de las herramientas de administración basadas en el mantenimiento de las bases de datos.
3. Identificación de las funcionalidades del Plugin de mantenimiento.
4. Diseño de la solución a partir de las funcionalidades identificadas.
5. Implementación de las funcionalidades identificadas.
6. Integración del Plugin de mantenimiento a la herramienta HABD.
7. Definición de las estrategias, técnicas y métodos que probarán la solución desarrollada.
8. Ejecución de las estrategias, técnicas y métodos definidos que probarán las funcionalidades implementadas.

Con las tareas de la investigación antes expuestas, se obtendrá como resultado un Plugin de mantenimiento para la herramienta HABD.

El documento está estructurado en introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas, bibliografías, glosario de términos y anexos. A continuación se muestra una breve descripción de cada capítulo.

Capítulo 1. Fundamento Teórico. En este capítulo se describe el marco teórico de trabajo, realizando un profundo análisis de varias herramientas de administración de SGBD. Se muestran definiciones importantes para lograr un mejor entendimiento de la investigación. Se brinda una panorámica de las características fundamentales de cada una de estas herramientas, para así poder definir la metodología y las tecnologías a utilizar para el desarrollo del Plugin de mantenimiento para las herramientas de administración de bases de datos.

Capítulo 2: Descripción de la solución. En el capítulo se muestran las diferentes características del sistema propuesto. Se define el modelo de dominio para lograr un mejor entendimiento del sistema, la arquitectura del sistema, los requerimientos funcionales e historias de usuario. Se realiza un diagrama de clases del diseño que se utiliza como complemento de la metodología definida, para lograr la comprensión del componente.

Capítulo 3: Validación y prueba. En el capítulo se realiza la validación del componente propuesto anteriormente. Se selecciona la estrategia de prueba a realizar, así como el enfoque de pruebas más apropiado de acuerdo a la metodología de desarrollo utilizada, además se selecciona el método de validación de las pruebas definidas por la metodología *Extreme Programming* (XP), que permite verificar el correcto funcionamiento del componente. Se muestran también los resultados de estas últimas en cada una de las iteraciones del proceso de desarrollo de software.

CAPÍTULO 1. FUNDAMENTO TEÓRICO

Introducción

En la actualidad con el desarrollo tecnológico de la informática, los SGBD ocupan un lugar importante dentro de los sistemas de información, muchas aplicaciones o soluciones informáticas hacen uso del mismo. La tecnología de las bases de datos demuestra los cambios ocurridos en el transcurso del tiempo por las necesidades del procesamiento y la gestión de la información, por lo tanto en este capítulo se analizan conceptos teóricos asociados a la investigación como por ejemplo: bases de datos, Sistemas Gestores de Bases de Datos y Plugin. Además de realizar un estudio de las principales herramientas de administración existentes en el mundo y se muestran las características definidas de las herramientas, tecnologías y metodología a utilizar durante el desarrollo de la investigación.

1.1 Conceptos asociados a la investigación

Seguidamente se describirán diferentes conceptos que son muy importantes en el proceso de entendimiento del desarrollo del presente trabajo.

1.1.1 Base de datos

Una base de datos es una colección de información organizada de forma que un programa de ordenador pueda seleccionar rápidamente los fragmentos de datos que necesite. Una base de datos es un sistema de archivos electrónico. Las bases de datos tradicionales se organizan por campos, registros y archivos. Un campo es una pieza única de información; un registro es un sistema completo de campos y un archivo es una colección de registros. Por ejemplo, una guía de teléfono es análoga a un archivo. Contiene una lista de registros, cada uno de los cuales consiste en tres campos: nombre, dirección, y número de teléfono. (3)

Una base de datos consiste de una colección de datos interrelacionados y un conjunto de programas que permiten acceder esos datos. Su objetivo primordial es proporcionar un medio ambiente que sea conveniente y eficiente tanto al extraer como al almacenar datos. (4)

La utilización de las bases de datos ofrece como ventaja consistencia, integridad, seguridad, flexibilidad y rapidez al obtener datos. Tiene entre sus principales características independencia lógica y física de los datos, redundancia mínima, acceso concurrente por parte de múltiples usuarios, consultas complejas optimizadas y acceso a través de lenguajes de programación estándar.

1.1.2 Sistema Gestor de Base de Datos

Un Sistema Gestor de Base de Datos es un sistema de software que permite la definición de bases de datos; así como la elección de las estructuras de datos necesarios para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación. (5)

Los SGBD son el 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. Proporcionan 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. (6)

Existen SGBD que proporcionan muchas de las funciones estándares que el programador necesita para lograr grandes avances en el desarrollo de su trabajo, entre los existentes pueden ser mencionados los siguientes:

Ejemplos de gestores de bases de datos libres

- MySQL
- PostgreSQL

Ejemplos de gestores de bases de datos propietarios

- Fox Pro
- IBM Informix
- Microsoft SQL Server
- Open Access
- Oracle
- Paradox

- PervasiveSQL

1.1.3 PostgreSQL

PostgreSQL es un Sistema Gestor de Bases de Datos objeto-relacional, bajo licencia BSD (del inglés, *Berkeley Software Distribution*). Se ejecuta en los sistemas operativos: Linux, Unix, BSDs, Mac OS X, Beos y Windows. Documentación muy bien organizada, pública y libre, con comentarios de los propios usuarios, comunidades muy activas y varias en castellano. Soporte nativo para los lenguajes: PHP, C, C++, Perl, Python. Soporte de todas las características de una base de datos profesional (disparadores, procedimientos almacenados, funciones, secuencias, relaciones, reglas, tipos de datos definidos por usuarios, vistas, vistas materializadas). Altamente adaptable a las necesidades del cliente. (7)

La universidad se encuentra inmersa en un proceso de migración, vinculada principalmente a las nuevas tendencias tecnológicas, que consiste en la utilización de herramientas netamente libres, por lo que se seleccionó PostgreSQL como gestor de bases de datos, por el hecho de que existe una amplia comunidad colaborativa perteneciente al departamento DATEC, en la cual se trabaja de forma desinteresada, altruista y libre. Las características antes mencionadas de este gestor hacen que sea flexible, potente y profesional.

1.1.4 Plugin

Un Plugin en una aplicación informática que se utiliza para la integración de otras aplicaciones aportándole una nueva función, le permite a los desarrolladores interactuar con la aplicación y así aumentar la cantidad de funcionalidades que estos puedan realizar. Brinda la posibilidad de separar el código fuente de la aplicación debido a cualquier incompatibilidad que exista con respecto a las licencias. (8)

Los Plugins no suelen funcionar independientes de la aplicación principal y dependen de los servicios prestados por esta, por el contrario el software hospedero puede funcionar sin la utilización obligatoria de los complementos, lo que permite a los usuarios actualizar y añadir los Plugins de forma dinámica sin tener que hacer modificaciones en la aplicación principal.

1.2 Herramientas de administración de bases de datos

Las herramientas de administración de bases de datos son la base para el desarrollo de los proyectos y para la calidad de las aplicaciones que se realicen. Estas proveen facilidades para la manipulación de grandes volúmenes de datos, son el eje principal de un SGBD, actúa de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan; controlan la creación, el mantenimiento y el uso de las bases de datos de una organización y de sus usuarios finales. A continuación se argumentará sobre la herramienta HABD creada en el proyecto de PostgreSQL en el año 2010.

1.2.1 HABD

Los usuarios que actualmente trabajan con el gestor PostgreSQL deben realizar un gran esfuerzo para explotar sus funcionalidades, además carecen de una variedad de servicios que les pueden facilitar el trabajo a las personas que interactúan con el gestor. Incorporarles nuevas funcionalidades a estos clientes libres resulta complicado por la extensión y poca flexibilidad que poseen; sin embargo las herramientas propietarias son muy potentes pero a la vez son muy costosas, y adquirir las licencias de software privativo se dificulta en Cuba, por pertenecer a la lista de países embargados.

La necesidad del país de contar con una herramienta propia, que le permitiera administrar de forma eficiente el gestor PostgreSQL e independizarse de productos de terceros, propicia la realización de la Herramienta de Administración de Bases de Datos HABD, de ahí su nombre, 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. De esta forma se le pueden agregar gran número de funcionalidades evitando así que los usuarios necesiten de varias aplicaciones para resolver un único problema. La aplicación contribuye al intercambio entre el usuario y el gestor PostgreSQL y facilita el trabajo de éstos.

La herramienta HABD tiene una arquitectura basada en Plugin. Se han realizado otros trabajos de investigación que han sido integrados a la herramienta de administración agregándoles nuevas funcionalidades como es el caso del proceso de normalización de base de datos, particionado de tablas, CRUD-PG, editor de consultas, monitorización, generación de datos, recuperación de estados de entidades y diseñador de base de datos. Todas las funcionalidades antes mencionadas han sido

integradas a la herramienta como Plugin permitiendo que el trabajo sea rápido y menos engorroso. Este trabajo de diploma se enfoca en la investigación de cómo se realiza la funcionalidad de mantenimiento de bases de datos para poder desarrollar un Plugin y lograr su integración en la herramienta HABD.

1.3 Mantenimiento de bases de datos

Las bases de datos utilizadas en forma intensiva se fragmentan e incrementan su tamaño; mucho espacio en el disco puede ser ocupado por registros eliminados e información innecesaria. Esto significa un desperdicio de espacio en el disco. (9)

Un punto crítico en el trabajo con las bases de datos es a la hora de realizar su mantenimiento, para esto se crea un flujo de trabajo de las tareas como por ejemplo: control de integridad, copias de seguridad o compactación de las bases de datos. Las mismas requieren de un mantenimiento adecuado y de una actualización continuada.

En este sentido, existen otras tareas de mantenimiento básicas que no se deben descuidar:

- Agregar y quitar registros cuando cambien los datos.
- Realizar cambios en la información almacenada en la base de datos.
- Revisión de archivos de registro.
- Control del rendimiento del servidor.
- Verificación de que el disco duro no esté sobrecargado.

Las tareas antes mencionadas permiten mantener la base de datos en un nivel óptimo, las operaciones se pueden ejecutar semanalmente, sin embargo, la frecuencia varía de acuerdo con el entorno y con la actividad que recibe cada una de ellas.

1.4 Mantenimiento en bases de datos PostgreSQL

PostgreSQL, como cualquier otro gestor de bases de datos, requiere que ciertas tareas se realicen con regularidad para lograr un rendimiento óptimo. Estas tareas son necesarias, pero de naturaleza repetitiva y pueden fácilmente ser automatizadas usando las herramientas estándares de Unix, tales como scripts

de cron[1] o el Programador de tareas de Windows. Si no, es responsabilidad del administrador de la base crear scripts apropiados. (10)

Una de las tareas de mantenimiento es la creación de copias de seguridad de los datos en un horario regular. Sin una copia de seguridad reciente, no se tiene la posibilidad de recuperación después de una catástrofe (error de disco, incendio, eliminar por error una tabla crítica). Los mecanismos de respaldo y recuperación disponibles en PostgreSQL son:

- SQL dump.
- Sistema de archivos de copia de seguridad de nivel.
- Archivo Continuo. (11)

SQL dump

La idea detrás de este método de descarga es generar un archivo de texto con comandos SQL que cuando retroalimente al servidor recree la base de datos en el mismo estado en que se encontraba en el momento de la descarga. PostgreSQL proporciona el pg_dump, programa de utilidad para este propósito, el cual posee como características realizar salvas consistentes incluso si la base de datos está siendo usada concurrentemente y no bloquea los accesos de otros usuarios que estén accediendo a la misma (lectores o escritores).

Sistema de archivos de copia de seguridad de nivel

Una estrategia de copia de seguridad alternativa es copiar directamente los archivos que PostgreSQL usa para almacenar los datos en la base de datos. Existen dos restricciones que hacen que este método no sea práctico, o al menos inferior al método pg_dump:

1. El servidor de base de datos debe ser cerrado con el fin de obtener una copia de seguridad utilizable.
2. Si se ha excavado en los detalles de la disposición del sistema de archivo de la base de datos, se puede tener la tentación de intentar realizar una copia de seguridad o restaurar sólo algunas tablas o bases de datos individuales de sus respectivos archivos o directorios. Esto no va a funcionar porque

[1] nombre del programa que permite a usuarios Linux/Unix ejecutar automáticamente comandos o scripts (grupos de comandos) a una hora o fecha específica.

la información contenida en estos archivos contiene sólo la mitad de la verdad. La otra mitad se encuentra en los archivos de registro de las confirmaciones pg_clog. Un archivo de la tabla sólo se puede utilizar con esta información. Por supuesto, es también imposible recuperar sólo una tabla y los datos asociados pg_clog porque eso haría que todas las otras tablas de la base de datos del clúster inútil. Así que las copias de seguridad del sistema de archivos sólo trabajan para una completa copia de seguridad y restauración de un clúster de base de datos.

Archivo Continuo

El registro pg_xlog describe cada cambio realizado en los archivos de datos de la base de datos. Este registro existe principalmente para fines de seguridad de choque: si el sistema se bloquea, la base de datos se puede restaurar con consistencia por "reproducir" las entradas del registro realizados desde el último control. Sin embargo, la existencia del log hace posible el uso de una tercera estrategia para realizar copias de bases de datos: se puede combinar un sistema de archivos “Nivel de copia de seguridad” con “backup de los archivos *WAL*”.

1.5 Herramientas de administración que realizan mantenimientos de bases de datos

Durante la investigación se consultó información de varias herramientas de administración de bases de datos, de las cuales se escogieron para realizar un estudio tres, una de código abierto PgAdmin y dos de licencia propietaria Navicat y EMS, por presentar características que conllevaron a su elección como fueron: son líderes en el mercado, brindan una gama de servicios de soporte muy amplia y productos a varios SGBD, además están asociados a empresas de renombre. De dichas herramientas se seleccionan funcionalidades comunes para estudiarlas y posteriormente decidir cuáles utilizar en la implementación del Plugin de mantenimiento de bases de datos, con el objetivo de integrarlo a la herramienta HABD.

1.5.1 PgAdmin

PgAdmin es una aplicación gráfica para trabajar con el gestor de bases de datos PostgreSQL, con licencia *Open Source* (Código Abierto). Está programada en C++ usando la librería gráfica multiplataforma wxWidgets. Se encuentra en los repositorios de las distribuciones de Ubuntu, aunque se puede descargar el código fuente de su última versión en su sitio oficial (www.pgadmin.org). La aplicación también incluye

un editor de resultado de sintaxis SQL, la conexión al servidor se puede realizar a través de TCP/IP o *Unix Domain Sockets*. PgAdmin es desarrollado por una comunidad de expertos de PostgreSQL en todo el mundo y está disponible en más de una docena de idiomas. (12)

PgAdmin brinda la posibilidad de utilizar opciones de mantenimiento de forma visual que le provee el gestor PostgreSQL, dentro de las que se encuentran: Vacuum: limpieza de las tuplas muertas, Analyze: analizar los datos para calcular estadísticas, Reindex: reorganizar los índices. (13)

La opción Vacuum recupera el espacio ocupado por tuplas muertas. En la operación normal de PostgreSQL, las tuplas que son eliminadas o caducadas por una actualización no son físicamente removidas de la tabla; permanecen presentes hasta que se hace un Vacuum. Por lo tanto, es necesario hacerlo periódicamente, especialmente sobre tablas actualizadas frecuentemente. Un Vacuum plano (sin Full) simplemente reclama el espacio y lo hace disponible para su reutilización. Esta forma del comando puede operar en paralelo con lecturas y escrituras normales a la tabla, al no obtener un bloqueo exclusivo. Vacuum Full hace un procesamiento más extensivo, incluyendo mover las tuplas entre bloques para tratar de compactar la tabla a un número mínimo de bloques de disco. Esta forma es mucho más lenta y requiere un bloqueo exclusivo en cada tabla que está siendo procesada. (14)

Por otro lado Analyze recolecta estadísticas sobre los contenidos de las tablas en la base de datos, y almacena los resultados en el catálogo del sistema pg_statistic. Posteriormente, el planificador de consultas usa estas estadísticas para ayudar a determinar el plan de ejecución más eficiente para las consultas.
Sin parámetros, Analyze examina cada tabla en la base de datos actual. Con un parámetro, examina solo una tabla. Además, se puede dar una lista de columnas en cuyo caso sólo las estadísticas para dichas columnas son recuperadas. (15)

La opción Reindex reconstruye un índice utilizando la información almacenada en el índice de la tabla, reemplazando la vieja copia del mismo. Existen muchos escenarios en los que es útil el Reindex:

- Un índice que se ha corrompido, y que no contiene datos válidos.
- Un índice que se ha “hinchado” (contiene muchas páginas vacías o casi vacías).

- Se ha alterado un parámetro de almacenamiento (como un factor de llenado) para un índice, y se desea asegurar que el cambio ha tenido un efecto completo.

1.5.2 Navicat

Navicat es una serie de administradores gráficos de bases de datos y software de desarrollo para MySQL, SQL Server, Oracle, SQLite y PostgreSQL, soporta la mayoría de los objetos de bases de datos. Con su bien diseñada Interfaz Gráfica de Usuario (GUI), los usuarios pueden fácilmente crear, organizar, acceder y compartir información en forma segura. (16)

Está disponible para las plataformas Microsoft Windows, Mac OS X y Linux. Puede conectar usuarios a servidores locales/remotos, proveyendo muchas utilidades y herramientas tales como Modelado de Datos, Transferencia de Datos, Sincronización de Datos/Estructuras, Importar/Exportar, Salva/Restauración, Constructor de Reportes y Planificación para facilitar el proceso de mantenimiento de datos. (16)

Navicat ofrece la posibilidad de utilizar tareas de mantenimiento de forma visual al igual que el PgAdmin, dentro de las que se encuentran: Vacuum, Analyze y Reindex, las cuales fueron explicadas en el sub epígrafe anterior.

1.5.3 EMS SQL Manager for PostgreSQL Lite

EMS PostgreSQL Manager es una poderosa herramienta gráfica para la administración y desarrollo de bases de datos PostgreSQL, funciona con cualquier versión de PostgreSQL, y soporta todas sus nuevas características. (17)

EMS SQL Manager para PostgreSQL está integrado por un conjunto de funcionalidades entre las que se pueden encontrar el diseñador visual de base de datos, constructor visual o editor profesional de texto de SQL con la capacidad de crear y ejecutar consultas avanzadas, así como creación de bases de datos instantáneas en forma de SQL script. También da la posibilidad de ver, buscar y editar el contenido de la base de datos visualmente para que se puedan recuperar los datos relevantes. Además tiene una

impresionante exportación de los datos y capacidades de importación. Puede exportar datos a 17 formatos de archivo e importarlos. (12)

También provee la interfaz gráfica para el mantenimiento de la mayoría de los servicios de PostgreSQL. Las siguientes operaciones para el mantenimiento de bases de datos PostgreSQL están disponibles en el EMS SQL Manager:

Estado del servidor: El visor de estado del servidor permite obtener información común en las actuales conexiones al servidor, cierres, transacciones preparadas, y ver archivos de trazas del servidor.

Analizador de Trazas SQL del Servidor: El analizador de trazas SQL del servidor permite analizar trazas del servidor PostgreSQL en una forma simple brindando la habilidad para ordenar, filtrar y agrupar declaraciones de trazas.

Configuración del Servidor: EMS SQL Manager para PostgreSQL provee un útil y efectivo servicio: la habilidad de ver y cambiar la configuración del servidor. Existen muchos parámetros de configuración que afectan el comportamiento del sistema de bases de datos, el cual puede ser configurado a través del Administrador de Configuración. Usando las pestañas del Administrador de Configuración del Servidor se puede ver/editar un número de parámetros y opciones del servidor los cuales pueden ser cambiados para optimizar la ejecución del servidor PostgreSQL.

Conclusiones arribadas en el estudio de las herramientas de administración de bases de datos

Con el estudio realizado anteriormente a las herramientas de administración de bases de datos, se pudo conocer cuáles son las principales tareas de mantenimiento que realizan, las mismas se tendrán en cuenta a la hora de implementar el Plugin de mantenimiento de HABD. De dichas herramientas se decide escoger las siguientes tareas de mantenimiento para implementarlas en el nuevo Plugin: Vacuum, Analyze y Reindex.

1.6 Funcionalidades propuestas para el Plugin de mantenimiento de bases de datos

Para la implementación del Plugin de mantenimiento se escogieron algunas de las funcionalidades descritas anteriormente: Vacuum, Analyze y Reindex, y se agregarán otras que incrementarán el número de funcionalidades del Plugin: Salvar y Restaurar.

A continuación se listan las funcionalidades a implementar:

- ✓ Salvar
  - Clúster
  - Completa
  - Comprimida
  - Por Fragmentos
- ✓ Restaurar
  - Clúster
  - Completa
  - Comprimida
  - Por Fragmentos
- ✓ Vacuum
  - FULL
  - FREEZE
  - Analyze
- ✓ Analyze
- ✓ Reindex

1.7 Metodología de desarrollo de software

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software. Es como un libro de recetas de cocina, en el que se van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Además detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla. Las metodologías surgen ante la necesidad de utilizar una serie

de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto de software. Las mismas pretenden guiar a los desarrolladores al crear un nuevo software. (18)

Las metodologías se dividen en dos grupos, tradicionales (pesadas) y ágiles (ligeras):

- Las metodologías pesadas son las más tradicionales, se centran en la definición detallada de los procesos y tareas a realizar, herramientas a utilizar, y requiere una extensa documentación, porque pretenden prever todo de antemano. (19)
- Las metodologías ágiles dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones cortas. Este enfoque se utiliza en proyectos con requisitos cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad. (20)

Metodología Extreme Programming

La metodología *Extreme Programming* (XP) se basa en la suposición de que es posible desarrollar software de gran calidad a pesar, del cambio continuo. Su principal concepción es que con un poco de planificación, un poco de codificación y unas pocas pruebas se puede decidir si se está siguiendo un camino acertado o equivocado, evitando así tener que echar marcha atrás demasiado tarde. (21)

Para el desarrollo de la herramienta HABD se determina continuar, al igual que en el trabajo de diploma **"Exploración y diseño de la Herramienta de Administración de Bases de Datos para PostgreSQL (HABD)"**, con el uso de la Metodología *Extreme Programming* (XP) pues sus características se ajustan a las necesidades de la investigación antes mencionada. Esta metodología posee un grupo de ventajas, dentro de las cuales se puede mencionar, que satisface al cliente a través de la entrega temprana y continua de las aplicaciones, y son aceptados los requisitos que puedan tener algún cambio durante el desarrollo del proyecto, por las continuas versiones que se ofrecen al usuario se atienden las necesidades del mismo con mayor exactitud y además el uso de esta metodología reduce el número de errores debido a que los requisitos del software son fáciles de modificar obteniéndose el código más simple y entendible.

El trabajo con la metodología XP ahorra tiempo en documentación y los miembros del equipo se enfocan en poner todos 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. Además de potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promover el trabajo en equipo y propiciar un ambiente agradable de trabajo.

1.8 Tecnologías y herramientas a utilizar

Para el desarrollo del Plugin de mantenimiento de bases de datos es necesario tener en cuenta las herramientas con las que se va a realizar el diseño y la implementación del mismo, las que se mencionan a continuación.

1.8.1 Framework de desarrollo

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.

Los framework son la piedra angular de la moderna ingeniería del software. Un framework no tiene funcionalidades de una aplicación específica, sino que las aplicaciones se construyen sobre ellos. Seguidamente se mencionarán algunas características del framework QT 4.7.

QT 4.7

El framework QT 4.7 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. 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. (22)

En el presente trabajo se decide, al igual que en el desarrollo de otros Plugin para la herramienta HABD, continuar haciendo uso de QT 4.7, el cual presenta un buen diseño orientado a objetos, permite una mayor reutilización de código para agilizar el proceso de desarrollo de aplicaciones visuales, lo que trae consigo que sea más fácil el trabajo diario del programador, responde con una rápida velocidad al estar escrito en el lenguaje C++.

1.8.2 Entorno de desarrollo integrado

Un Entorno de Desarrollo Integrado o IDE (por sus siglas en inglés), es una herramienta que permite a los desarrolladores de software escribir sus programas en uno o más lenguajes. Consiste básicamente en una plataforma en la que se integran un editor de código, un compilador, un depurador y una interfaz gráfica de usuario.

Qt Creator 2.5 es un entorno multi-plataforma de desarrollo integrado (IDE) adaptado a las necesidades de los desarrolladores de Qt. Está distribuido bajo tres tipos de licencias: Qt *Commercial Developer License*, Qt GNU LGPL v. 2.1, Qt GNU GPL v. 3.0 y es disponible para las plataformas: Linux, Mac OS X; Windows, Windows CE, Symbian y Maemo. Tiene como características fundamentales: (23)

- Editor de código C++.
- Soporta los lenguajes: C#/.NET Languages (Mono), Python: PyQt y PySide, Ada, Pascal, Perl, PHP y Ruby.
- Posee una GUI integrada y diseñador de formularios.
- Herramienta para proyectos y administración.
- Ayuda sensible al contexto integrada.
- Depurador visual.
- Resaltado y auto-completado de código.
- Soporte para refactorización de código.

Las características antes mencionadas conllevaron a la selección del QT Creator para la elaboración del Plugin de mantenimiento, además este IDE fue diseñado para hacer que el desarrollo de aplicaciones sea más rápido y fácil al poseer un editor de código C++. Presenta una ayuda para los desarrolladores muy

bien redactada, argumentada y escrita de manera clara y sencilla. Es el IDE idóneo para trabajar con el framework QT, al ser realizados por la misma compañía permitiendo una mejor integración.

1.8.3 Lenguaje de Programación

El lenguaje de programación es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. (24)

Un lenguaje de programación está formado de un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Una característica relevante de los lenguajes de programación es precisamente que más de un programador pueda usar un conjunto común de instrucciones que sean comprendidas entre ellos para realizar la construcción del programa de forma colaborativa. (25)

C++

En la actualidad, el C++ es un lenguaje versátil, potente y general. Su éxito entre los programadores profesionales le ha llevado a ocupar un alto puesto como herramienta de desarrollo de aplicaciones. El C++ mantiene las ventajas del C en cuanto a riqueza de operadores y expresiones, flexibilidad, concisión y eficiencia. (26)

C++ es un lenguaje procedural (orientado a algoritmos) y orientado a objetos. Se puede decir que 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 C++ depende del hardware y permite programar a alto y a bajo nivel. Además, ha eliminado algunas de las dificultades y limitaciones del lenguaje C original.

Se decide para la realización de este trabajo escoger este lenguaje de programación, producto a que es un lenguaje que permite que los niveles de velocidad de ejecución de los programas sean más altos, el

consumo de memoria es más pequeño que otros lenguajes y PostgreSQL tiene sus librerías programadas en este lenguaje.

1.8.4 Herramientas CASE (Computer-Aided Software Engineering) para el modelado

Las herramientas CASE son diversas aplicaciones informáticas, destinadas a aumentar la productividad en el desarrollo de software, reduciendo el costo de las mismas, en términos de tiempo y dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software. Además de ofrecer muchos beneficios para todos los involucrados en un proyecto, permiten aplicar la metodología de análisis y diseño orientados a objetos y abstraerse del código fuente, en un nivel donde la arquitectura y el diseño se tornan más obvios y más fáciles de entender y modificar. En la actualidad, posibilitan la creación y modificación de diagramas con gran facilidad, además de automatizar varias actividades como generación de código para el desarrollo de software, lo cual mejora considerablemente la calidad, el rendimiento, la utilidad y fiabilidad de las herramientas CASE.

Visual Paradigm 8.0

Visual Paradigm for UML (Unified Modeling Language) es una herramienta CASE, multiplataforma de modelado UML. Posee una licencia *Freeware*. Aporta a los desarrolladores de software una plataforma de desarrollo para construir aplicaciones de gran calidad y rapidez. Permite dibujar todo tipo de diagramas UML, revertir código fuente a modelos UML y generar código fuente desde dichos diagramas. (12)

Soporta el ciclo de vida completo de desarrollo de un software, desde la fase de análisis hasta el despliegue del mismo. Además de la generación automática de informes en formato PDF, Word o HTML. Esta herramienta además se puede integrar con diversos IDE como NetBeans (de Sun), Developer (de Oracle), Eclipse (de IBM), JBuilder (de Borland). (12)

Se decide utilizar esta herramienta porque tiene la capacidad de ejecutarse sobre diferentes sistemas operativos lo que le confiere la característica de ser multiplataforma. Es fácil de instalar, actualizar y compatible entre ediciones. Visual Paradigm diseña y desarrolla productos que eliminan la complejidad, mejoran la productividad, y comprimen el software en los plazos de desarrollo.

1.8.5 Lenguaje de Modelado

El lenguaje de modelado está formado por un conjunto de símbolos. Se utiliza extensivamente en combinación con una metodología de desarrollo de software para avanzar de una especificación inicial a un plan de implementación y para mostrar y comunicar dicho plan a todo un equipo de desarrolladores.

Lenguaje Unificado de Modelado (Unified Modeling Language o UML)

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como: procesos de negocio y funciones del sistema. Además, presenta aspectos concretos como: expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. (27)

En el presente trabajo se hará uso de este lenguaje. UML ofrece un estándar para describir un modelo del sistema, incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

1.8.6 Sistema de control de versiones

Un sistema de control de versiones es un sistema centralizado que se utiliza generalmente para compartir información. Permite la gestión de archivos y directorios, y sus cambios a través del tiempo, además de la conexión de múltiples usuarios al repositorio para leer o escribir. Se encarga de administrar el acceso a un conjunto de ficheros, y mantiene un historial de cambios realizados. Un controlador de versiones es útil porque permite explorar los cambios que dio lugar a cada una de esas versiones y facilita el retiro arbitrario de la misma.

Subversion

Subversion (SVN) 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. SVN ha alcanzado gran popularidad en las comunidades de desarrolladores de software pues es distribuido bajo licencia libre de tipo Apache/BSD. (6)

En el presente trabajo se hará uso de este sistema de control de versiones, que es el utilizado en el departamento PostgreSQL.

1.9 Conclusiones del capítulo

En el presente capítulo se realizó un estudio de varias herramientas de administración de bases de datos, de las cuales se escogieron para profundizar tres de las más utilizadas, una de código abierto PgAdmin y las otras dos de licencia propietaria Navicat y EMS. De estas herramientas se seleccionaron como funcionalidades a implementar en el Plugin de mantenimiento: Salvar, Restaurar, Vacuum, Analyze y Reindex. Como guía en el proceso de desarrollo del software, se seleccionó la metodología Extreme Programing. Se realizó un estudio de las herramientas y tecnologías existentes con el propósito de seleccionar las más adecuadas para la solución de la investigación, partiendo del anterior análisis se determinó utilizar como herramienta CASE Visual Paradigm 8.0 con el lenguaje de modelado UML, el framework QT 4.7, el IDE de desarrollo QT Creator, empleando el lenguaje de programación C++, utilizando el controlador de versiones Subversion.

CAPÍTULO 2. DESCRIPCIÓN DE LA SOLUCIÓN

Introducción

En el siguiente capítulo se presenta la propuesta del Plugin de mantenimiento de bases de datos para HABD. Son definidas 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 el negocio se establece un modelo de dominio a través del cual se puede tener un mejor entendimiento del problema a resolver. Además se exponen las tareas de ingeniería y el plan de iteraciones en las cuales serán implementadas cada historia de usuario y se realiza una estimación de la duración de cada tarea de la ingeniería para de esta forma lograr una implementación correcta y con éxito.

2.1 Identificación del problema

En el departamento de PostgreSQL, se creó en el año 2010 la Herramienta de Administración de Bases de Datos llamada HABD, para el gestor de bases de datos PostgreSQL. Su objetivo es brindarle al usuario las funcionalidades para administrar una base de datos de manera sencilla. Posee una arquitectura basada en Plugin, proporcionando la integración de funciones, lo que permite el buen funcionamiento de la herramienta y la comodidad de los usuarios a la hora de interactuar con la misma.

Una de las funcionalidades fundamentales con la que debe contar toda herramienta de administración de bases de datos es realizar tareas mantenimiento. Actualmente HABD no cuenta con ninguna opción de mantenimiento, por lo que no se pueden realizar salvas, recuperaciones u optimizaciones a las bases de datos. Para influir positivamente en la integración de estas funcionalidades a HABD se realizó un estudio de varias herramientas de administración existentes. Se analizaron sus características comunes y otras diferentes, referente a la forma en que realizan el mantenimiento de las bases de datos. El presente trabajo de diploma arrojará la solución a dicho problema con la creación de un Plugin de mantenimiento a bases de datos para HABD.

2.2 Modelo de dominio

La metodología XP no define un modo específico para definir el negocio, por lo que se decide utilizar el modelo de dominio, mediante el cual se realiza el proceso de comprensión y entendimiento de la manera más cómoda para aquellas personas que van a trabajar con la aplicación. El modelo del dominio es una representación de los conceptos u objetos del mundo real, significativos para un problema. (28)

En la Figura 1 se muestra el modelo de dominio conformado, el cual representa las clases conceptuales relacionadas en el Plugin.

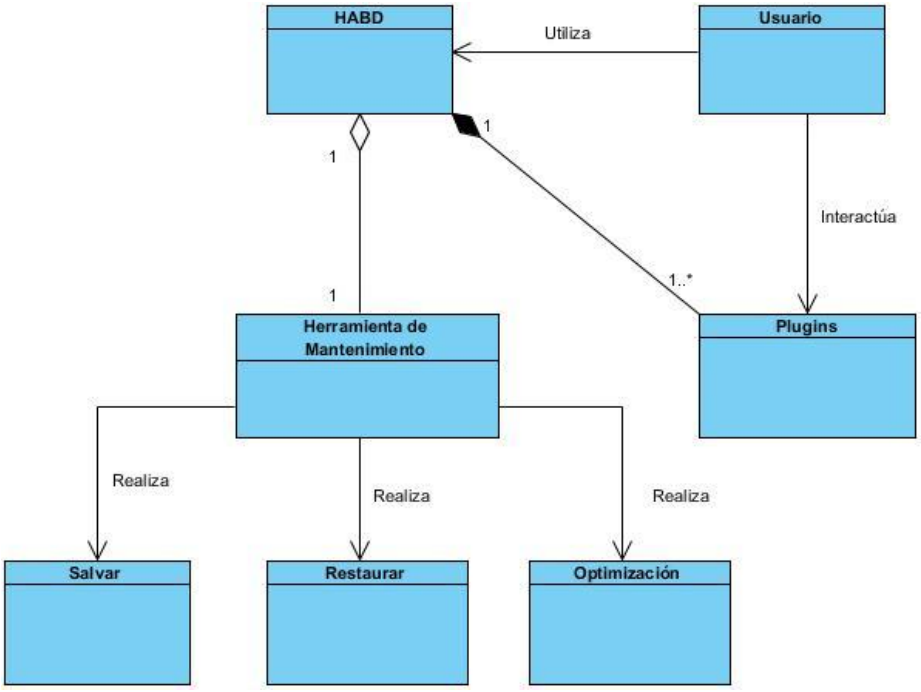


Figura 1: Diagrama Modelo de Dominio.

Usuario: Persona que interactúa con la Herramienta de Administración de Bases de Datos.

HABD: Herramienta de administración de bases de datos capaz de cargar y manipular los Plugins que sean incorporados.

Plugins: Conjunto de Plugins que extienden las funcionalidades del sistema de administración de bases de datos.

Herramienta de mantenimiento: Módulo que añade la funcionalidad de mantenimiento a base de datos a la herramienta HABD.

Salvar: Permite realizar salvas a las bases de datos del gestor PostgreSQL.

Restaurar: Se encarga de restaurar salvas realizadas a las bases de datos del gestor PostgreSQL.

Optimizar: Conjunto de opciones que permite realizar Vacuum, Analyze y Reindex a las bases de datos del gestor PostgreSQL.

2.3 Propuesta del componente a desarrollar

El sistema propuesto es un Plugin de mantenimiento a bases de datos para integrarse a la herramienta HABD. El mismo brindará la posibilidad de realizar diferentes tipos de copias de seguridad, al realizar el proceso frecuentemente se cuenta con la posibilidad de recuperar datos después de una catástrofe. Permitirá además calcular las estadísticas de las tablas, organizar los índices y diferentes opciones para eliminar el espacio ocupado por las tuplas muertas. Al salvar o restaurar una base de datos, el usuario podrá escoger la forma más conveniente de realizar el proceso, según las características de la base de datos que se estén utilizando. El usuario obtendrá notificaciones de las acciones realizadas a través de la opción Mensajes detallados. Además el Plugin contará con una ayuda para facilitar la navegación del usuario por la aplicación, explicando el uso de cada una de las funcionalidades, en caso de que el mismo presente dudas.

2.4 Historias de usuario

Las historias de usuario son un artefacto generado por la metodología XP. Las escriben los propios clientes, tal y como ven ellos las necesidades del sistema, por tanto son descripciones cortas y escritas en el lenguaje del usuario sin terminología técnica. Las historias de usuario proporcionan los detalles sobre la estimación del riesgo y cuánto tiempo conllevará la implementación de dicha historia de usuario. (29)

Para el presente trabajo se generan un total de 13 historias de usuario (HU). Estas alcanzan un grado de prioridad muy alta, alta y media, puesto que es de suma importancia la realización exitosa de cada una de ellas. A continuación se listan según su categoría.

Prioridad muy alta

- Salvar base de datos completa.
- Salvar clúster de bases de datos.
- Salvar base de datos por fragmentos.
- Salvar base de datos comprimida.

Prioridad alta

- Restaurar base de datos completa.
- Restaurar clúster de bases de datos.
- Restaurar base de datos por fragmentos.
- Restaurar base de datos comprimida.
- Realizar Vacuum.
- Realizar Analyze.
- Realizar Reindex.

Prioridad media

- Integrar Plugin de mantenimiento a la herramienta HABD.
- Mostrar manual de usuario.

En las Tablas 1 y 2 se muestran algunas de las historias de usuario generadas, en las que se especifican diferentes datos sobre las mismas, tales como el número y nombre de la HU, responsable, duración y descripción.

Tabla 1. Historia de usuario: Salvar base de datos completa.

	Historia de Usuario
Número: 1	**Nombre de la Historia de Usuario:** Salvar base de datos completa.

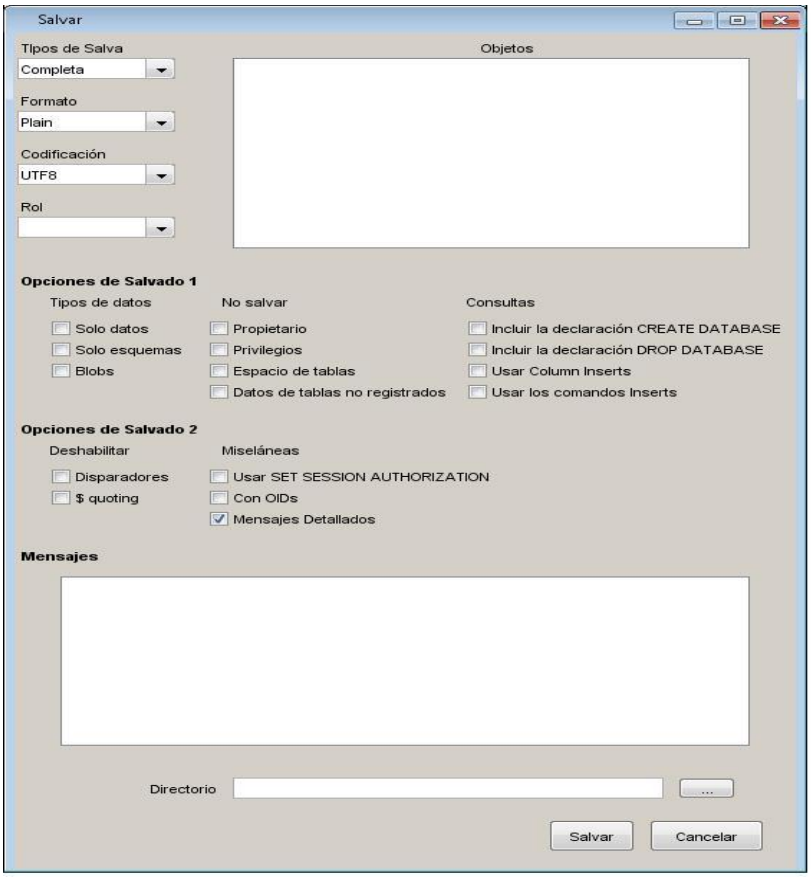
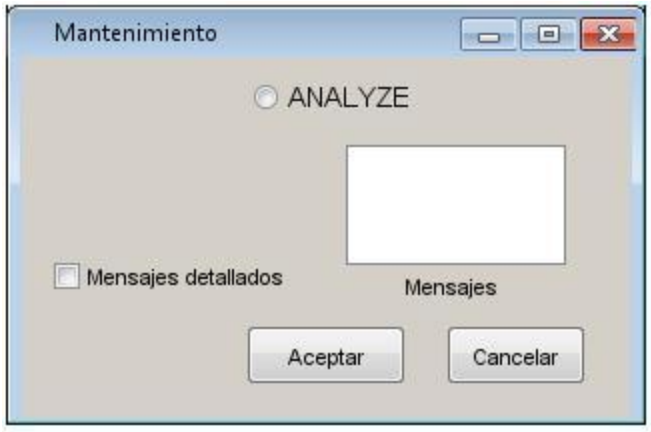
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Darlys Milagros Hernández Mitchell	**Iteración asignada:** 1
Prioridad en negocio: Muy alta	**Puntos estimados:** 0.8 semanas
Riesgo en desarrollo: Alto	**Puntos reales:** 0.8 semanas
Descripción: Salva la base de datos seleccionada por el usuario.	
Observaciones: Ninguna	
Prototipo de interfaces: 	

Tabla 2. Historia de usuario: Realizar Analyze.

Historia de Usuario	
Número: 12	**Nombre de la Historia de Usuario:** Realizar Analyze.
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Rosa Elvira Morales Martínez	**Iteración asignada:** 2
Prioridad en negocio: Alta	**Puntos estimados:** 0.5 semanas
Riesgo en desarrollo: Alto	**Puntos reales:** 0.5 semanas
Descripción: Recolecta estadísticas sobre los contenidos de las tablas en la base de datos, y almacena los resultados en el catálogo del sistema pg_statistic. Permite seleccionar opcionalmente Mensajes detallados, habilita la visualización de mensajes de progreso.	
Observaciones: Ninguna	
Prototipo de interfaces: 	

El resto de las historias de usuario se encuentran en la Planilla de Historias de Usuario del expediente de proyecto.

2.5 Lista de reserva del producto

En la metodología XP la lista de reserva del producto (ver Tabla 3) representa los requisitos que debe contener la aplicación tanto funcionales como no funcionales, ordenados según la prioridad, muy alta, alta, media y baja. Además aparece la estimación para su implementación por semanas y el rol del que realizó la estimación del requisito.

Tabla 3. Lista de reserva del producto.

Ítem *	Descripción	Estimación	Estimado por
Prioridad: Muy Alta			
1	Salvar base de datos completa.	0.8	Analistas
2	Salvar clúster de bases de datos.	0.8	Analistas
3	Salvar base de datos por fragmentos.	0.8	Analistas
4	Salvar base de datos comprimida.	0.8	Analistas
Prioridad: Alta			
5	Restaurar base de datos completa.	0.5	Analistas
6	Restaurar clúster de bases de datos.	0.5	Analistas
7	Restaurar base de datos por fragmentos.	0.5	Analistas
8	Restaurar base de datos comprimida.	0.5	Analistas
9	Realizar Vacuum.	0.5	Analistas
10	Realizar Analyze.	0.5	Analistas
11	Realizar Reindex.	0.5	Analistas
Prioridad: Media			
12	Integrar Plugin de mantenimiento a la herramienta HABD.	0.4	Analistas

13	Mostrar manual de usuario.	0.4	Analistas
Prioridad: Baja			
1	**Usabilidad.** Para utilizar el sistema es necesario poseer conocimientos elementales de computación, así como de base de datos.		
2	**Soporte.** Se dispondrá de documentación técnica que describa todas las funcionalidades del sistema, de modo que existirá un manual para el uso de la aplicación orientada a clientes y usuarios finales.		
3	**Software.** Para poder hacer uso de este Plugin se debe tener instalado las librerías de QT, el compilador de C++ y PostgreSQL.		
4	**Hardware.** Se necesita 100 MB de memoria RAM mínimo, 1GB de espacio libre en el disco duro para su instalación y un micro a 300 MHz.		
5	**Disponibilidad.** Se podrá hacer uso de la aplicación, siempre que estén instaladas todas las librerías necesarias.		

2.6 Tareas de la ingeniería

Una vez descritas las historias de usuario, se procede a describir las tareas de ingeniería pertenecientes a cada historia de usuario (ver Tablas 4, 5, 6, 7, 8 y 9). Estas tareas son las entradas de trabajo para el equipo de programadores, cada una de estas presenta una característica del sistema, a continuación se muestra un ejemplo de las tareas de ingeniería donde aparece el número identificador de la tarea, el identificador de la historia de usuario con la que está relacionada, el nombre de la tarea, la fecha de inicio, la fecha de fin, el equipo responsable y la descripción.

Tabla 4. HU 1: Tarea de la ingeniería: Seleccionar base de datos.

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 1
Nombre Tarea: Seleccionar base de datos.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio: 7/01/2013	**Fecha Fin:** 7/01/2013
Programador Responsable: Darlys Milagros Hernández Mitchell	
Descripción: Es la encargada de elegir la base de datos que se le desea realizar la copia de seguridad.	

Tabla 5. HU 1: Tarea de la ingeniería: Visualizar directorio de salva.

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 1
Nombre Tarea: Visualizar directorio de salva.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio: 7/01/2013	**Fecha Fin:** 7/01/2013
Programador Responsable: Darlys Milagros Hernández Mitchell	
Descripción: El usuario debe seleccionar el directorio donde desea que se salve la base de datos.	

Tabla 6. HU 1: Tarea de la ingeniería: Seleccionar los parámetros.

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** 1
Nombre Tarea: Seleccionar los parámetros.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio: 8/01/2013	**Fecha Fin:** 8/01/2013
Programador Responsable: Darlys Milagros Hernández Mitchell	
Descripción: Se seleccionan los parámetros necesarios para realizar el proceso de salva.	

Tabla 7. HU 1: Tarea de la ingeniería: Presionar botón Salvar.

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** 1
Nombre Tarea: Presionar botón Salvar.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio: 8/01/2013	**Fecha Fin:** 8/01/2013
Programador Responsable: Darlys Milagros Hernández Mitchell	
Descripción: Presionar el botón Salvar	

Tabla 8. HU 1: Tarea de la ingeniería: Capturar los parámetros.

Tarea de Ingeniería	
Número Tarea: 5	**Número Historia de Usuario:** 1
Nombre Tarea: Capturar los parámetros.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 9/01/2013	**Fecha Fin:** 10/01/2013
Programador Responsable: Darlys Milagros Hernández Mitchell	

Descripción: Se capturan los parámetros que el usuario seleccionó.

Tabla 9. HU 1: Tarea de la ingeniería: Ejecutar proceso de salva.

Tarea de Ingeniería	
Número Tarea: 6	**Número Historia de Usuario:** 1
Nombre Tarea: Ejecutar proceso de salva.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 10/01/2013	**Fecha Fin:** 11/01/2013
Programador Responsable: Darlys Milagros Hernández Mitchell	
Descripción: Ejecuta el proceso que realiza la salva pasando los parámetros capturados.	

El resto de las tareas de ingeniería se encuentran en la Planilla Tareas de Ingeniería del expediente de proyecto.

2.7 Plan de iteraciones

El objetivo de este artefacto es tener una planificación del trabajo, donde los desarrolladores y el cliente establecen los tiempos de implementación de las historias de usuario y la prioridad con que serán desarrolladas.

En la Tabla 10 se muestra el plan de iteraciones del Plugin de mantenimiento de la herramienta HABD, en el que se indican las historias de usuario a desarrollar en cada iteración y su duración. En la primera iteración serán implementadas las funcionalidades con una prioridad muy alta, descritas en las HU 1, 2, 3 y 4, con una duración total de tres semanas y un día. Durante la segunda iteración se desarrollarán las funciones con una prioridad alta, descritas en las HU 5, 6, 7, 8, 9, 10 y 11, con una duración total de tres semanas y dos días. En la tercera iteración serán desarrolladas las funcionalidades con prioridad media, descritas en las HU 12 y 13, con una duración de cuatro días. La duración total de la etapa de implementación será de siete semanas y dos días.

Tabla 10. Plan de iteraciones.

Release	Descripción de la iteración	Orden de la HU a implementar	Duración total
1	En esta iteración se realizarán las historias de usuario de prioridad muy alta, estas historias de usuario se encargarán del proceso de salvar la o las bases de datos según el usuario estime conveniente.	1-2-3-4	3.2 semanas
2	El objetivo de esta iteración es que el Plugin de mantenimiento de bases de datos consiga restaurar y realizar las opciones de mantenimiento según el usuario estime conveniente ya que todas las historias de usuario son de prioridad alta.	5-6-7-8-9-10-11	3.5 semanas
3	En esta iteración se implementarán las historias de usuario de prioridad media, con esta iteración se logrará integrar el Plugin de mantenimiento a la herramienta HABD y facilitarle al usuario el trabajo con el mismo a través de una ayuda.	12-13	0.8 semanas

2.8 Diseño del sistema

El desarrollo de cualquier proyecto requiere de un buen diseño de sus clases para de esta forma realizarlo con la mejor calidad posible y así el cliente quede satisfecho. En la metodología XP el diseño de las clases se realiza a través de las tarjetas clase-responsabilidades-colaboración (CRC), para de esta forma ayudar al refinamiento de las clases, estructurando el conjunto de las mismas.

2.8.1 Tarjetas CRC

Las tarjetas CRC proponen una forma de trabajo, preferentemente grupal, para encontrar los objetos del dominio de la aplicación, sus responsabilidades y cómo colaboran con otros para realizar tareas. Las mismas registran el nombre de las clases, sus responsabilidades y las otras clases con la que colaboran. Sus principales características son:

- Identificación de clases y asociaciones que participan del diseño del sistema.
- Obtención de las responsabilidades que debe cumplir cada clase.
- Establecimiento de cómo una clase colabora con otras clases para cumplir con sus responsabilidades.

Para el presente trabajo se identificaron un total de 5 tarjetas CRC: 1- Salvar, 2- Restaurar, 3- Optimizar, 4- MainWindow, 5- Consultas.

En las Tablas 11 y 12 se muestran las tarjetas CRC Restaurar y Optimizar generadas para el diseño del Plugin el cual debe realizar varias funciones entre las que se encuentran: restaurar base de datos completa, restaurar clúster, restaurar por fragmentos y restaurar base de datos comprimida.

Tabla 11. Tarjeta CRC "Restaurar"

Tarjeta CRC	
Clase: Restaurar	
Responsabilidades	**Colaboraciones**
Restaurar base de datos completa Restaurar clúster	

Restaurar por fragmentos Restaurar comprimida	

Tabla 12. Tarjeta CRC "Optimizar"

Tarjeta CRC	
Clase: Optimizar	
Responsabilidades	**Colaboraciones**
Vacuum Reindex Analyze	Consultas

El resto de las tarjetas CRC se encuentran en la Planilla Modelo de Diseño del expediente de proyecto.

2.8.2 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. (30)

El diagrama de clases es para el análisis y diseño. Representa las clases del sistema con sus relaciones estructurales. La definición de clase incluye definiciones para atributos y operaciones. Aunque la metodología XP no define un diagrama de clases del sistema, se presenta a continuación el realizado para un mejor entendimiento de la aplicación, puesto que éste se puede utilizar siempre y cuando contribuya para el mejoramiento de la comunicación y comprensión del mismo, ya que es una guía esencial para los desarrolladores a la hora de implementar. El diagrama propuesto contiene 5 clases, vinculadas mediante relaciones de composición y agregación. A continuación se detallan algunas de las clases principales del diagrama que se muestra en la Figura 2.

Consultas: Clase que contiene todas las consultas que se realizan a la base de datos, y que serán llamadas en el momento de realizar una tarea de mantenimiento específica. El tener agrupadas todas las consultas en una misma clase facilita el trabajo de futuras actualizaciones y mejoras de las mismas.

MainWindow: Es la interfaz gráfica de la aplicación. Permite seleccionar el tipo de mantenimiento a realizar así como sus parámetros. Contiene los botones y controles con los que interactúa el usuario y genera los eventos con los que funciona el sistema.

Restaurar: Clase que se encarga de realizar operaciones de restauración sobre las bases de datos. El restaurado de una base de datos puede ser completo, de clúster, por fragmentos o comprimido.

Salvar: Es la encargada de realizar diferentes tipos de salvas sobre las bases de datos, entre las que se encuentran el salvado completo, de clúster, por fragmentos o comprimido.

Optimizar: Esta clase contiene funciones que culminan en la eliminación de tuplas muertas, en el análisis de las estadísticas y en la reorganización de los índices de las tablas de la bases de datos.

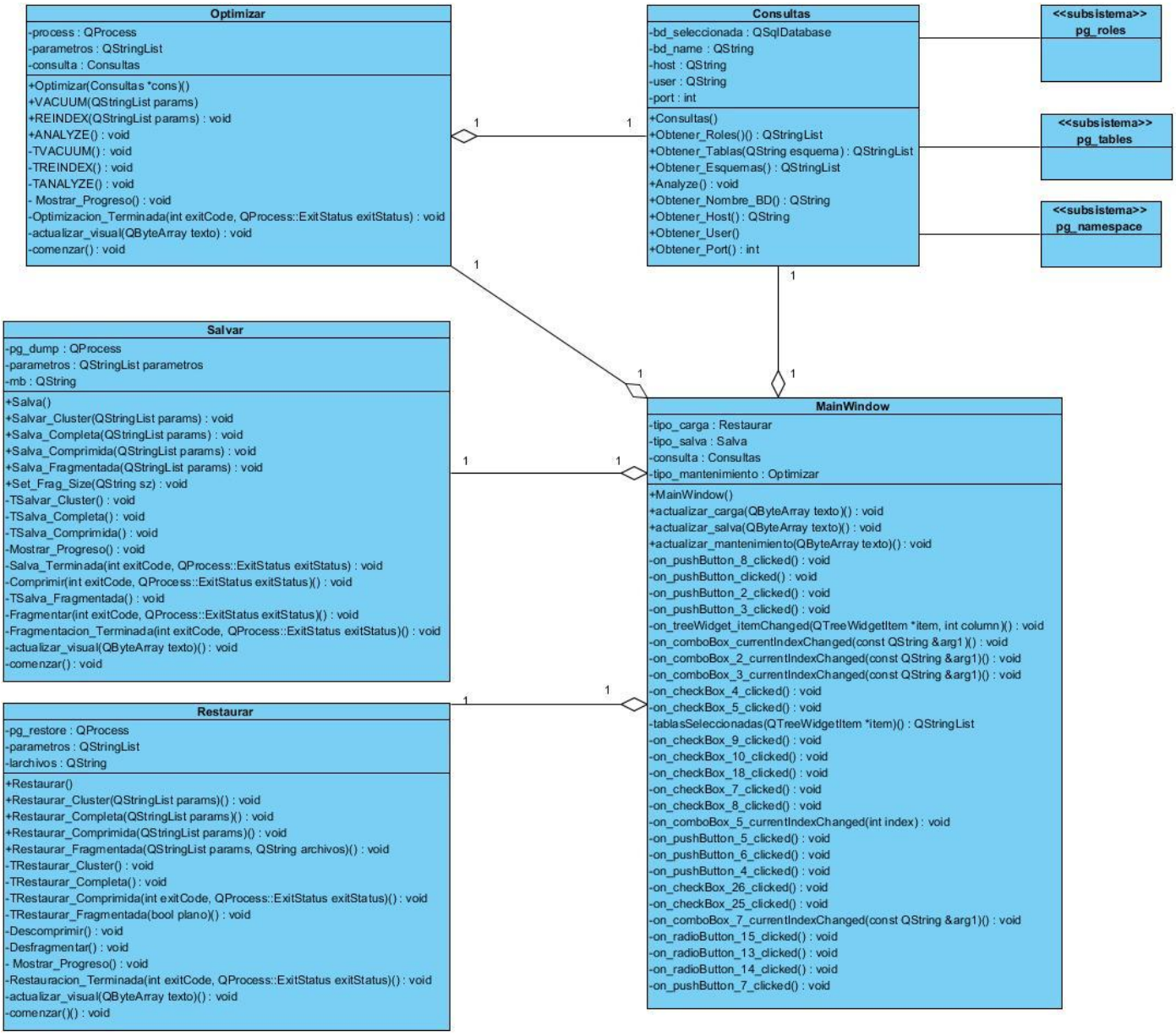


Figura 2. Diagrama de clases del Plugin de mantenimiento.

2.9 Arquitectura de software

La arquitectura del software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución. (31)

2.10 Estilos arquitectónicos. Patrones de arquitectura

Involucrados en una arquitectura se encuentran los estilos arquitectónicos. Los diferentes estilos tienen sus fortalezas y debilidades, y ciertos estilos hacen que sea más fácil o más difícil trabajar con diferentes obstáculos.

Un estilo es un concepto descriptivo que define una forma de articulación u organización arquitectónica. El conjunto de los estilos cataloga las formas básicas posibles de estructuras de software. (32)

Existen numerosos estilos arquitectónicos entre los que se encuentra el Estilo de llamada y retorno y dentro de él, el patrón de arquitectura Modelo-Vista-Controlador (MVC). Éste separa la lógica de control, la interfaz de usuario y los datos de una aplicación en tres componentes distintos. Se utiliza puesto que se tiene una capa de acceso a datos donde van a estar todas las clases que de una forma u otra piden información a una base de datos.

El paquete **Modelo** contiene todas las clases que tienen el código relacionado con el acceso a datos, para que este sea lo más genérico posible y se pueda reutilizar en otras situaciones y proyectos. Se incluirán consultas a las bases de datos y validaciones de entrada de datos. Se evidencia dentro del modelo la clase Consultas, la cual contiene varios métodos que envían a ejecutar consultas SQL al gestor obteniendo y devolviendo los resultados de las mismas para luego ser procesados.

El paquete **Vista** contiene todas las clases que poseen el código representando la parte que será visualizada en pantalla por el usuario. Se evidencia dentro de este paquete la clase Mainwindow, la misma es una interfaz visual que se le muestra al usuario para que el mismo interactúe con la aplicación.

El paquete **Controlador** contiene las clases que ejecutan la lógica de la aplicación, realizan llamadas al modelo para obtener los datos y se los pasa a la vista para que los muestre al usuario. Responden a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. Se evidencia en este paquete varias clases; Salvar, Restaurar y Optimizar, las cuales se encargan de recopilar toda la información de la capa Modelo y le envía los datos a las clases de la capa Vista.

Con el uso de este patrón se persigue mejorar la reusabilidad y que las modificaciones en las vistas impacten en menor medida en la lógica de negocio o de datos. En la Figura 3 se muestra el patrón MVC realizado.

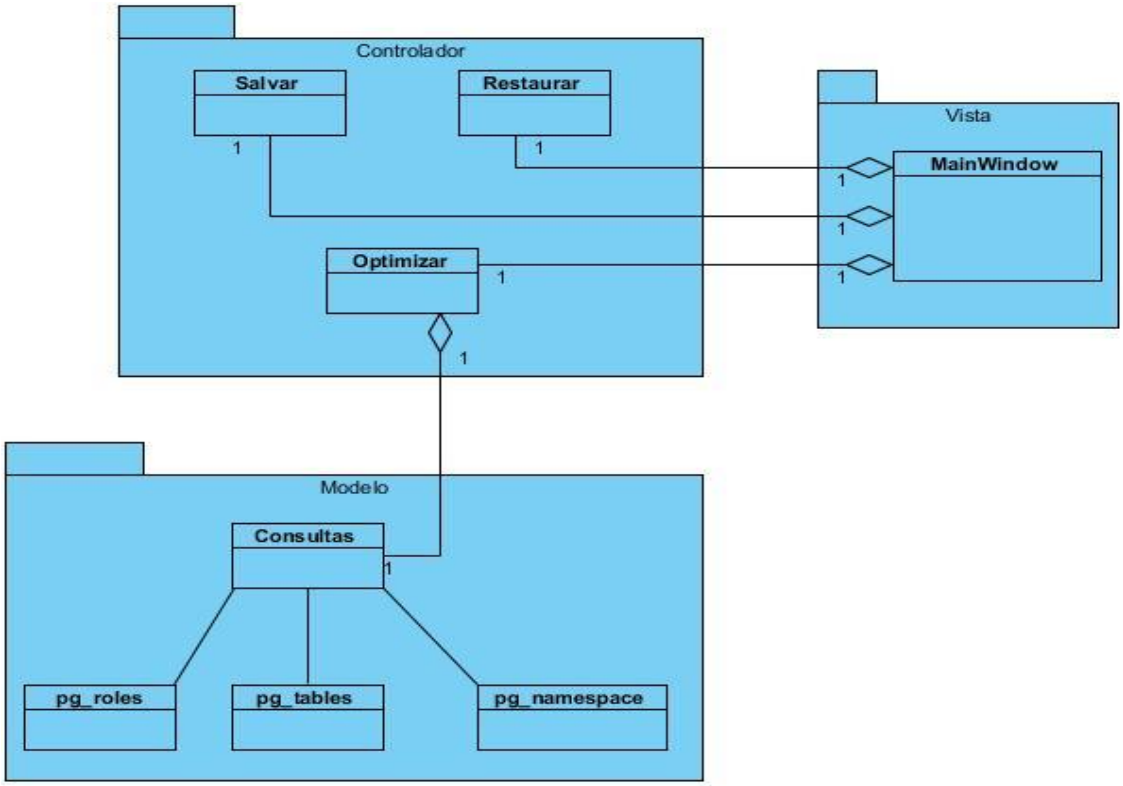


Figura 3. Patrón Modelo Vista Controlador.

2.11 Patrones de Diseño

Son un conjunto de prácticas de óptimo diseño que se utilizan para abordar problemas recurrentes en la programación orientada a objetos. Proponen además interfaces entre objetos, 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. Cabe destacar que existen muchas familias de patrones de diseño, en función del objetivo que se ha planteado en este trabajo, se describirán los mismos.

Los patrones GRASP (Patrones de Software para la Asignación General de Responsabilidad) y los patrones GoF (*Gang of Four*).

Patrones GRASP: Acrónimo que significa *General Responsability Assignment Software Patterns* en español significa Patrones de Software para la Asignación General de Responsabilidad. Las patrones GRASP constituyen la base del cómo se diseñará el sistema. Describen los principios fundamentales de diseño de objetos y la asignación de responsabilidades, expresados como patrones. (33)

Dentro de los patrones **GRASP** se encuentran; Experto, Creador, Controlador, Bajo Acoplamiento y Alta Cohesión.

Experto: Indica que la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo, es aquella cuya responsabilidad es llevar consigo toda la información y ser experta en el tema para no depender de ninguna otra clase. (33)
En este trabajo se evidencia este patrón en la clase Salvar (ver Figura 4), la cual tiene la responsabilidad de realizar todos los tipos de salvas que se le brinda a los usuarios.

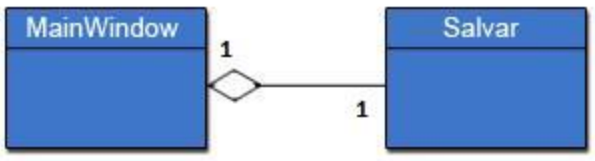


Figura 4. Patrón Experto.

Creador: Guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. (33)

Un ejemplo de este patrón en este trabajo se muestra en la clase MainWindow (ver Figura 5) debido a que se encarga de crear los objetos de las clases utilizadas para realizar las operaciones de mantenimiento.

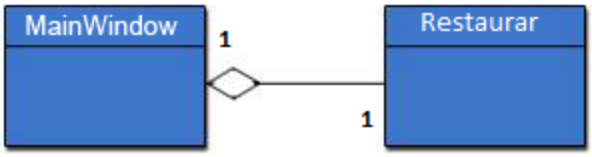


Figura 5. Patrón Creador.

Controlador: Es el encargado de controlar un evento del sistema, es aquel que sirve para intermediar entre la interfaz y el algoritmo. En este trabajo se refleja este patrón en la clase Restaurar (ver Figura 6), la cual recibe datos de la Interfaz de Usuario después de un evento y ejecuta una serie de algoritmos para realizar el restaurado de la base de datos.

Figura 6. Patrón Controlador.

Alta Cohesión: La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. El alto nivel de cohesión, es presentado por las clases que tienen responsabilidades y colaboran con otras para llevar a cabo las tareas. (33)

Un ejemplo de este patrón se evidencia en las clases Consultas y Optimizar (ver Figura 7) ya que la información que almacenan es coherente y relacionada con las tareas que deben desempeñar.

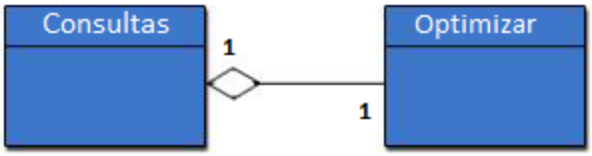


Figura 7. Patrón Alta Cohesión.

Bajo Acoplamiento: Es una medida de la fuerza con que una clase está conectada a otra. El bajo acoplamiento estimula la asignación de responsabilidades de forma tal que la inclusión de éstas no incremente el acoplamiento, es decir, significa asignar una responsabilidad para mantener pocas dependencias entre las clases. (33)

Se puede evidenciar el uso de este patrón en la clase Restaurar (ver Figura 8) puesto que Restaurar tiene poca dependencia con otras clases.

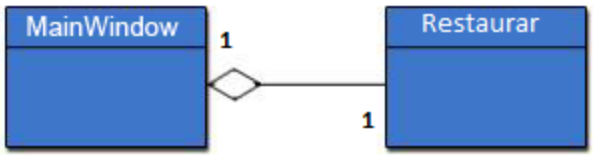


Figura 8. Patrón Bajo Acoplamiento.

Patrones GoF: Los patrones GoF (*Gang of Four*, "Pandilla de los Cuatro") recopilan una serie de patrones de diseños, agrupados en tres categorías: de creación, de estructura y de comportamiento. De los diferentes patrones que ofrece GoF se ha tenido en cuenta para la modelación del Plugin los siguientes:

Estructurales

Definen cómo las clases y objetos pueden unirse para formar grandes estructuras, para de esta forma proporcionar nuevas funcionalidades. A continuación se menciona el patrón que se pone de manifiesto en la implementación:

Puente (Bridge): Desacopla una abstracción de su implementación de modo que los dos puedan variar por separado. Este patrón se evidencia a la hora de conectar el Plugin con la herramienta de administración HABD y se muestra en la clase IPlugin (ver Figura 9) la cual permite la integración del Plugin de mantenimiento a la herramienta HABD.

Figura 9. Patrón Puente.

Comportamiento

Los patrones de comportamiento facilitan y definen la comunicación e iteración entre los objetos de un sistema y que éstos estén lo menos mezclados posible.

Observador: Define una dependencia de uno a muchos objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos. En la programación de Interfaces Gráficas de Usuario (GUI) se necesita que los cambios o eventos producidos sobre un objeto sean comunicados al resto de ellos. Qt posee el mecanismo de Signals y Slot, el cual posibilita entre estos comunicación segura, flexible y orienta a objetos. Básicamente, al producirse un evento sobre un objeto se emite una señal (Signal), que permite que se ejecute una determinada función (Slot) en respuesta a la señal emitida, debido a ello el patrón observador se encuentra inherente en Qt.

El uso de este patrón se evidencia en la clase Optimizar, donde al emitirse una señal (*SIGNAL*) se inicia automáticamente el procedimiento declarado como *SLOT* conectado previamente a la señal. En la Figura 10 se muestra un ejemplo de código donde se utiliza el sistema de señales y eventos.

```
connect(this,SIGNAL(comenzar()),this,SLOT(TVACUUM()));
emit comenzar();
```

Figura 10. Patrón Observador.

2.12 Estándares de codificación

La metodología XP promueve la programación basada en estándares, de manera que sea fácilmente entendible por todo el equipo, y que facilite la recodificación. Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. El propósito de las siguientes reglas y

recomendaciones es lograr que los programadores del proyecto PostgreSQL tengan un estilo de código común.

Comentarios

Es conveniente dejar información que pueda ser leída tiempo después por personas (posiblemente por la persona que programó) que necesitan entender qué fue lo que se hizo en el fragmento de código. Los comentarios deben ser escritos correctamente y claros, generalmente deben usarse de una sola línea. Se debe reservar los comentarios de bloques para la documentación formal o para comentar porciones de código (ver Figura 11).

```
//Marcar/Desmarcar los nodos del arbol
void checkSubTree(QTreeWidgetItem *item, Qt::CheckState st){
item->setCheckState(0, st);
for(int i=0;i<item->childCount();i++)
checkSubTree(item->child(i), st);
}
```

Figura 11. Ejemplo de estándar de comentarios.

Declaración de variables

Cada variable debe de ser declarada en una línea y comentada. También deben aparecer ordenadas alfabéticamente: El nombre de las variables debe comenzar con letras minúsculas y cada palabra relevante por la que esté compuesta debe ser con letra mayúscula (ver Figura 12).

```
private:
    Restaurar *tipoCarga; //Variable instancia de la clase Restaurar

    Salva *tipoSalva; //Variable instancia de la clase Salvar
```

Figura 12. Ejemplo de estándar de declaración de variables.

Declaración de funciones

No debe haber espacio entre el nombre de la función y el paréntesis izquierdo, ni entre éste 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 (ver Figura 13).

```
//obtener tablas seleccionadas
QStringList MainWindow::tablasSeleccionadas(QTreeWidgetItem *item) {
    QStringList tablas;
    for(int i = 0; i < item->childCount(); i++)
    {
        QTreeWidgetItem *child = item->child(i);
        QString schema = child->text(0);
        for(int j = 0; j < child->childCount(); j++)
```

Figura 13. Ejemplo de estándar de funciones.

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 (ver Figura 14).

```
return tablas;
```

Figura 14. Ejemplo de estándar de sentencia return.

Sentencia if

La sentencia if debe ser escrita de esta manera (ver Figura 15).

```
if(ui->comboBox_3->currentText()=="Plain")
{
    params << "--format=plain";
}
else if(ui->comboBox_3->currentText()=="Custom")
{
    params << "--format=custom";
}
```

Figura 15. Ejemplo de estándar sentencia if.

El resto de los estándares de codificación se encuentran en la Planilla Estándar de Codificación del expediente de proyecto.

2.13 Interfaces principales de la aplicación

La aplicación desarrollada se compone de diferentes interfaces para permitirle al usuario interactuar con las funciones del Plugin de mantenimiento. A continuación se describen dichas interfaces.

Área de salva de bases de datos

En la Figura 16 se muestra la ventana mediante la que se realiza el salvado de las bases de datos, la cual está compuesta por diferentes opciones para configurar los parámetros de las salvas, tales como el tipo de salva, el formato y la codificación.

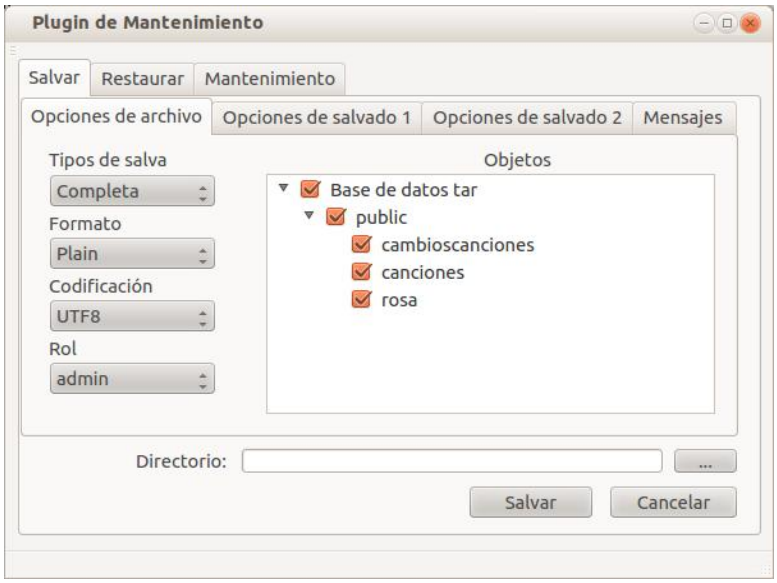


Figura 16. Ventana con las diferentes opciones para salvar bases de datos.

Área de restaurar bases de datos

En la Figura 17 se muestra la ventana mediante la que se realiza el restaurado de las bases de datos, la cual está compuesta por diferentes opciones para configurar los parámetros de restauración, tales como el tipo de restaurado, el formato y el rol.

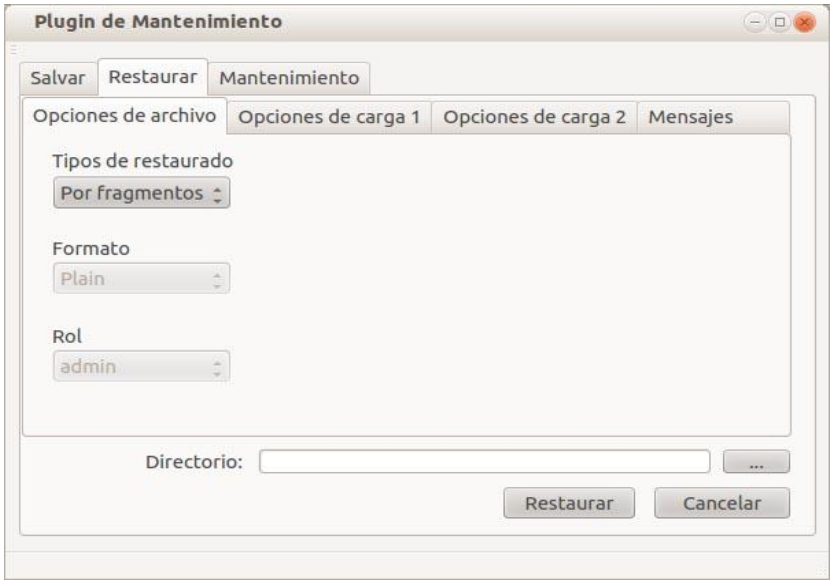


Figura 17. Ventana con las diferentes opciones para restaurar bases de datos.

Área de mantenimiento de bases de datos

En la Figura 18 se muestra la ventana mediante la que se realizan las tareas de mantenimiento a las bases de datos, como por ejemplo Vacuum Y Analyze.

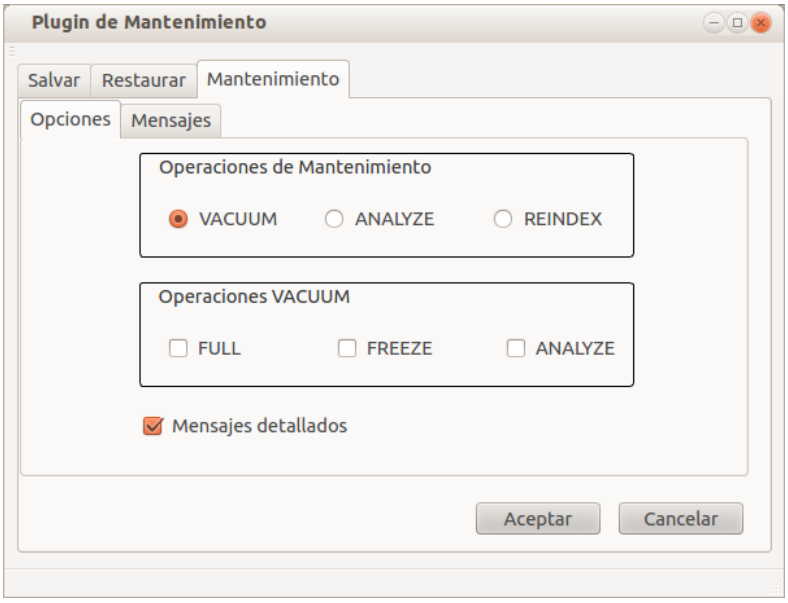


Figura 18. Ventana con las diferentes tareas de mantenimiento a bases de datos.

El Plugin de mantenimiento integrado en la herramienta HABD y los diferentes procesos de las tareas de mantenimiento concluidos se pueden observar en los ANEXOS.

2.14 Conclusiones del capítulo

En el presente capítulo han sido analizados todos los elementos que describen las características y diseño del Plugin de mantenimiento. 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 elaboró una propuesta con las perspectivas de solucionar el problema identificado, para lo cual fueron desarrolladas las fases de exploración y diseño en las que se definieron 13 historias de usuario y se estableció en que iteración y prioridad se desarrollarían. Se identificaron los patrones de diseños y el patrón de arquitectura a utilizar. Se realizó una breve descripción de algunas de las interfaces de la aplicación.

CAPÍTULO 3. VALIDACIÓN Y PRUEBA

Introducción

El desarrollo de un producto de software implica la realización de una serie de actividades encaminadas a encontrar errores. Incorporar acciones que evalúen la aplicación que se está desarrollando es de vital importancia para lograr un producto de software con la calidad requerida. En este capítulo se analizan las estrategias de pruebas que define la metodología XP y la técnica que se utilizará para diseñar los casos de pruebas (CP), además de mostrarse los resultados arrojados por las mismas.

3.1 Pruebas

Uno de los pilares de la metodología XP es el proceso de pruebas. Esta metodología anima a probar constantemente, permitiendo aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones. (34)

3.1.1 Estrategias de pruebas

Las estrategias de pruebas proporcionan un mapa que describe los pasos que se darán como parte de una prueba, indican cuando se planean y cuando se dan esos pasos. Por tanto las mismas deben incorporar planificación de pruebas, diseño y ejecución de casos de pruebas, recolección y evaluación de los datos resultantes. (35)

Pruebas unitarias

Las pruebas 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. (35)

Pruebas de aceptación o funcionales

Las pruebas de aceptación son muy importantes, dado que significan la satisfacción del cliente con el producto desarrollado al final de una iteración y al comienzo de la siguiente. Estas pruebas son definidas por el cliente para cada historia de usuario, y tienen como objetivo asegurar que las funcionalidades del sistema cumplen con lo que se espera de ellas. Existen dos tipos de pruebas de aceptación:

La prueba alfa

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

La prueba beta

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

Se decide utilizar para el desarrollo del Plugin la estrategia de prueba aceptación de tipo alfa, al ser más efectivo que el cliente dé una aceptación del producto que desea. Estas pruebas conllevan al cliente a precisar lo que la aplicación debe hacer en determinadas circunstancias, por esto el cliente es la persona adecuada para diseñar las pruebas.

3.1.2 Técnica de prueba y método seleccionado

Todo proyecto puede probarse mediante dos técnicas: conociendo la función específica para el que fue diseñado y conociendo el funcionamiento del producto. El primer enfoque se centra en las llamadas pruebas de caja negra y el segundo en las pruebas de caja blanca. (36)

En la Figura 19 se representa gráficamente la filosofía de las pruebas de caja blanca y caja negra.

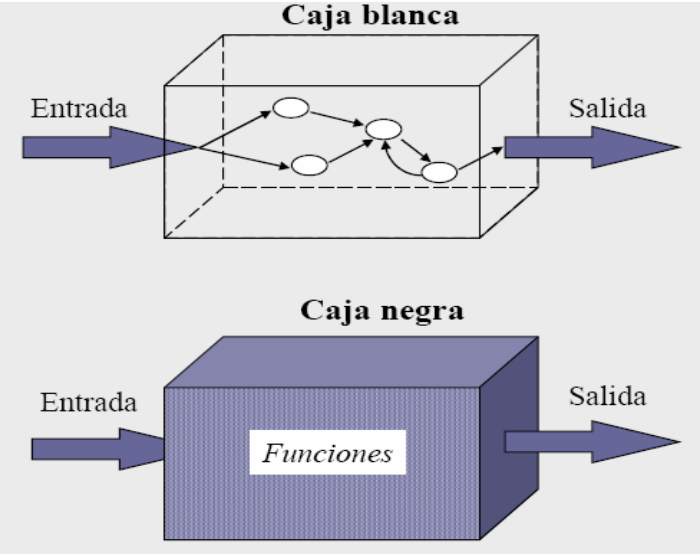


Figura 19. Representación de pruebas de caja blanca y caja negra.

A continuación se realiza una breve caracterización de estas dos técnicas para de ellas seleccionar una y lograr la correcta realización de las pruebas.

Pruebas estructurales o caja blanca

Se centran en la estructura lógica interna del software. Se basa en un examen detallado de los procedimientos y caminos lógicos del sistema.

La prueba de la caja blanca es una técnica de diseño de CP que realiza las pruebas al código de la aplicación y que usa la estructura de control del diseño procedimental para derivar los casos de prueba.

Pruebas de funcionalidad o caja negra

Se llevan a cabo sobre la interfaz del software. Se trata de demostrar que las funciones del software son operativas, que las entradas se manejan de forma adecuada y se produce el resultado esperado.

Las pruebas de caja negra buscan encontrar errores en cinco categorías:

- Funciones incorrecta o ausente.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.

- Errores de rendimiento.
- Errores de inicialización y terminación. (36)

Métodos empleados por la técnica de caja negra

- **Técnica de prueba basada en grafos:** En esta técnica se debe entender los objetos que se modelan en el software y las relaciones que conectan a estos, tales como objetos de datos, objetos de programa como módulos o colecciones de sentencias del lenguaje de programación. (36)

- **Partición equivalente:** La técnica de prueba partición equivalente divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. En otras palabras, este método intenta dividir el dominio de entrada de un programa en un número finito de clases de equivalencia. De tal modo que se pueda asumir razonablemente que una prueba realizada con un valor representativo de cada clase es equivalente, a una prueba realizada con cualquier otro valor de dicha clase. (36)

- **Análisis de Valores Límite:** El análisis de valores límite (AVL) es una técnica de diseño de casos de prueba que completa a la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el AVL lleva a la elección de casos de prueba en los extremos de la clase. (36)

Se decide utilizar para el desarrollo del Plugin, la estrategia de prueba aceptación de tipo alfa, con la técnica de caja negra, empleando el método partición equivalente, al probar si las funcionalidades son operativas a través de los casos de prueba, y principalmente se logra cumplir con las especificaciones tratadas con el cliente.

3.1.3 Casos de pruebas basados en historias de usuario

Un Caso de Prueba (CP) no es más que un conjunto de condiciones, bajo las cuales se introducen datos, con el objetivo de obtener varios resultados, permitiendo determinar si se ha cumplido satisfactoriamente el desarrollo de las funcionalidades que se han estado probando. Se puede saber si un caso de prueba es aceptable, si el mismo presenta una alta probabilidad de detectar un error, que no haya sido encontrado hasta el momento.

En la Tabla 13 se presenta la Sección (SC) “Reindex”, del caso de prueba realizado a la historia de usuario “Realizar Reindex”.

Tabla 13. Caso de prueba aplicado al sistema.

Escenario	Descripción	Variable 1	Variable 2	Respuesta del sistema	Flujo central
EC 1.1 Realizar reindex con mensajes detallados.	*El Plugin realiza reindex correctamente.*	*V(Seleccionado)*	*V(Seleccionado)*	*El sistema muestra el progreso del proceso de reindex y muestra el mensaje: "Proceso de mantenimiento concluido".*	*1- Seleccionar pestaña Mantenimiento/Opciones/Operaciones de Mantenimiento. 2- Seleccionar el radioButtom Reindex. 3- Seleccionar parámetro "Mensajes detallados". 4- Seleccionar botón Aceptar.*
EC 1.2 Realizar reindex sin mensajes detallados.	*El Plugin realiza reindex correctamente.*	*V(Seleccionado)*	*V( )*	*El sistema no muestra el progreso del proceso de reindex y muestra el mensaje: "Proceso de mantenimiento concluido".*	1- Seleccionar pestaña Mantenimiento/Opciones/Operaciones de Mantenimiento. 2- Seleccionar el radioButtom Reindex. 3- Seleccionar botón Aceptar.

La tabla que se muestra a continuación describe las variables que se encuentran asociadas al caso de prueba representado, SC “Reindex”.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
variable 1	Reindex	radioButtom	si	Reorganiza los índices.
variable 2	Mensajes detallados	Chekbox	si	Muestra el progreso del proceso seleccionado.

El resto de los CP se encuentran en la Planilla de Casos de Pruebas del expediente de proyecto.

Las pruebas realizadas proporcionaron resultados satisfactorios en los casos de pruebas efectuados. Los resultados que no fueron satisfactorios pasaron a ser no conformidades y se emitieron en el registro de

dificultades encontradas. En la Tabla 14 se muestra un ejemplo de las no conformidades encontradas, en la HU “Realizar Reindex”.

Tabla 14. Ejemplo de no conformidades encontradas y resueltas en la HU “Realizar Reindex”.

Elemento	No.	No conformidad	Aspecto correspondiente	Etapa de detección	Clasificación	Estado NC	Respuesta del equipo de desarrollo
Aplicación	1	No muestra en el progreso de la acción realizada la cantidad de índices reorganizados.	Realizar REINDEX.	Prueba	S	18/05/13 PD 21/05/13 RA	Se corrigió el error encontrado, ya que se muestran la cantidad de índices reorganizados.
Aplicación	2	No muestra el mensaje "Proceso de mantenimiento concluido".	Realizar REINDEX.	Prueba	NS	20/05/13 PD 21/05/13 RA	Se corrigió el error encontrado ya que se muestra el mensaje de proceso concluido.

El resto de las no conformidades se encuentran en la Planilla de No Conformidades del expediente de proyecto.

3.1.4 Presentación de los resultados de las pruebas funcionales

Con el objetivo de verificar que todos los requisitos funcionales fueron desarrollados correctamente se aplicaron pruebas específicas por cada una de las historias de usuario. La Figura 20 muestra el número de CP realizados, las No Conformidades (NC) detectadas y las No Conformidades Resueltas (NCR)

durante las tres iteraciones realizadas. Fueron encontradas en una primera iteración cinco no conformidades, de un total de cuatro CP realizados, las cuales fueron resueltas en siete días. Para una segunda iteración se encontraron cuatro no conformidades, de un total de siete CP, solucionadas completamente en un período de tres días. Finalmente para una tercera iteración no se encontraron no conformidades.

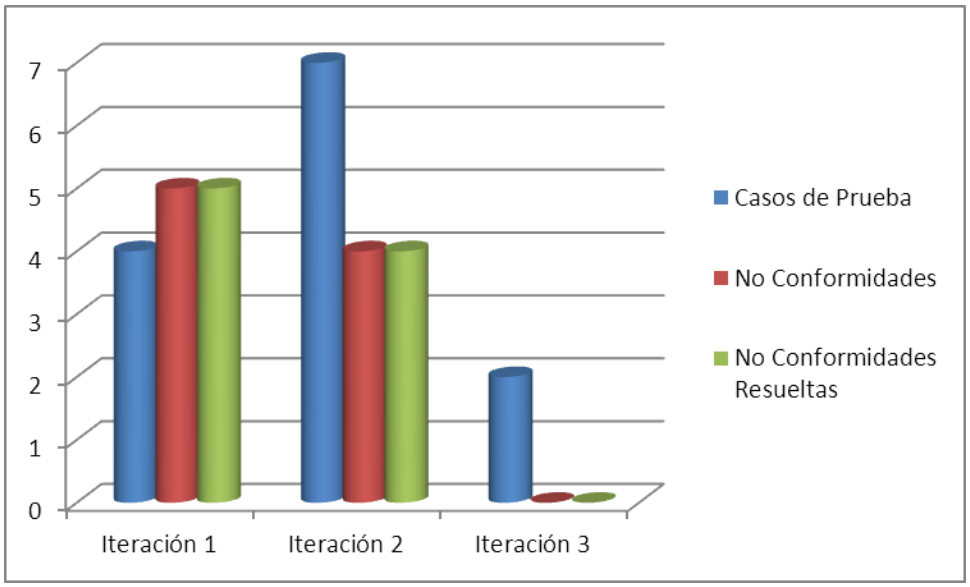


Figura 20. Resultados de las pruebas aplicadas.

3.2 Conclusiones del capítulo

Se realizó un estudio para determinar que estrategias de pruebas emplear, siendo las pruebas de aceptación las más convenientes para el Plugin de mantenimiento, mediante los casos de pruebas se validaron las 13 historias de usuario para verificar su correcto funcionamiento, utilizando la técnica de caja negra. Se logró presentar los resultados arrojados en cada iteración logrando obtener una aplicación que responde a todos los requerimientos funcionales identificados.

CONCLUSIONES GENERALES

Al concluir el desarrollo de la presente investigación, se arriba a las siguientes conclusiones:

- Se identificaron las funcionalidades comunes de cada una de las herramientas de mantenimiento estudiadas, de las cuales se seleccionaron como funcionalidades a implementar en el Plugin de mantenimiento: Salvar, Restaurar, Vacuum, Analyze y Reindex.
- Mediante el modelo de diseño se determinaron cinco clases del Plugin de mantenimiento que guiaron el proceso de implementación.
- Se implementaron los 13 requisitos funcionales identificados para el Plugin de mantenimiento, el cual se integró a la herramienta HABD permitiéndole a la misma realizar mantenimiento de bases de datos en PostgreSQL.
- Se validaron las 13 historias de usuario mediante un número igual de casos de pruebas, verificando el correcto funcionamiento de la aplicación partiendo del diseño propuesto.

RECOMENDACIONES

Con el objetivo de mejorar el funcionamiento y utilidad del Plugin de mantenimiento de la herramienta HABD se recomienda:

- Perfeccionar la funcionalidad Analyze con el objetivo de mostrar las estadísticas almacenadas en el catálogo pg_static, y así comprobar si el planificador de consultas optimizó de forma eficiente el plan de ejecución para las consultas.

REFERENCIAS BIBLIOGRÁFICAS

1. UCI. [En línea] [Citado el: 8 de 11 de 2012.] http://www.uci.cu/mision.

2. PostgreSQL. [En línea] [Citado el: 8 de 11 de 2012.] http://postgresql.uci.cu/.

3. *masadelante.com.* [En línea] [Citado el: 8 de 11 de 2012.] http://www.masadelante.com/faqs/base-de-datos.

4. *Base de Datos.* [En línea] [Citado el: 8 de 11 de 2012.] http://www.uprb.edu/profesor/ntorres/base_de_datosventajasdesventajas.htm.

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

6. **Veitía, Beatriz Piñeiro.** *Plugin de monitorización para la herramienta de administración de base de datos HABD.* La Habana : s.n., 2011.

7. *Comunidad Técnica Cubana de PostgreSQL. PostgreSQL Cuba.* [En línea] [Citado el: 10 de 11 de 2012.] http://postgresql.uci.cu/node/134.

8. [En línea] [Citado el: 11 de 11 de 2012.] http://www.slideshare.net/guacaneme/cris006.

9. *SendBlaster.* [En línea] [Citado el: 17 de 11 de 2012.] http://www.sendblaster.es/database-maintenance/.

10. *Manual de usuario de PostgreSQL.Cap 23.*

11. *Manual de usuario de PostgreSQL.Cap 24.*

12. **Acosta, Ivette Rosa Teodosio.** *Exploración y diseño de la Herramienta de Administración de Bases de Datos para PostgreSQL (HABD).* 2011.

13. *ArPUG grupo de usuarios de Argentina .* [En línea] [Citado el: 18 de 11 de 2012.] http://www.arpug.com.ar/trac/wiki/PgAdmin.

14. *ArPUG grupo de usuarios de Argentina.* [En línea] [Citado el: 18 de 11 de 2012.] http://www.arpug.com.ar/trac/wiki/sql-Vacuum.html.

15. *ArPUG grupo de usuarios de Argentina.* [En línea] [Citado el: 18 de 11 de 2012.] http://www.arpug.com.ar/trac/wiki/sql-Analyze.html.

16. *Navicat.* [En línea] [Citado el: 19 de 11 de 2012.] www.navicat.com.

17. *EMS Manager.* [En línea] [Citado el: 19 de 11 de 2012.] http://www.freedownloadmanager.org/es/downloads/SME_Gerente_de_PostgreSQL_37536_p/.

18. *UNIVERSIDAD DE MURCIA.* [En línea] [Citado el: 15 de 11 de 2012.] http://www.um.es/docencia/barzana/IAGP/IAGP2-Metodologias-de-desarrollo.html.

19. *EcuRed.* [En línea] [Citado el: 15 de 11 de 2012.] http://www.ecured.cu/index.php/Metodolog%C3%ADas_de_desarrollo_de_software.

20. **Canós, José H, Letelier, Patricio y Penadés, Carmen.** *Metodologías Ágiles en el Desarrollo de Software.* [En línea] [Citado el: 16 de 11 de 2012.] http://www.willydev.net/descargas/prev/TodoAgil.Pdf .

21. **Rodríguez, Hermes Alexy Marañón y Montero, Pedro David Duharte.** *Planificación y Diseño del Portal para la Comunidad Técnica Cubana de PostgreSQL.* La Habana : s.n., 2010.

22. **Garrido, Salvador Alemany.** [En línea] [Citado el: 16 de 11 de 2012.] http://www.polinux.upv.es/drupal/files/IntroduccionQt.pd.

23. **Meyer y Pérez, Lisandro Damián Nicanor.** *Introducción al desarrollo multiplataforma con Qt® 4.* Septiembre 2007.

24. **Castellanos, Yeneysi Pérez y Martínez, Marlon Alain Morejón.** *Plugin diseñador de bases de datos para la Herramienta de Administración de Bases de Datos HABD.* La Habana : s.n., 2011.

25. *MSDN .* [En línea] [Citado el: 17 de 11 de 2012.] http://msdn.microsoft.com/es-es/library/ms172579%28v=vs.80%29.aspx.

26. **de Jalón, Javier García, Rodríguez , José Ignacio, Sarriegui, José María y Brazález, Alfonso.** *Aprenda C++ como si estuviera en primero.* San Sebastián : s.n., abril 1998.

27. *EcuRed.* [En línea] [Citado el: 19 de 11 de 2012.] http://www.ecured.cu/index.php/Lenguaje_Unificado_de_Modelado.

28. *Entorno Visual de Aprendizaje.* [En línea] [Citado el: 18 de 2 de 2013.] http://eva.uci.cu/mod/resource/view.php?id=9400&subdir=/UML_y_Patrones/Construccion de un modelo conceptual.

29. **Escribano, Gerardo Fernández.** *Introducción a Extreme Programming.* 9-12-2002.

30. **Zapata, Antonia.** *Diseño estructural: Diagrama de clases.*

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

32. **Reynoso, Carlos Billi.** *Introducción a la arquitectura de software. versión 1.0 . Universidad de Buenos Aires. Buenos Aires : s.n.,* 2004.

33. **Visconti, Marcello y Astudilo, Hernán.** *Fundamentos de Ingeniería de Software.*

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

35. **Pressman, Roger S.** *Ingeniería de Software. Un enfoque práctico. 6ta edición, Cap 13. pág. 1.*

36. **Pressman, Roger S.** *Ingeniería de Software. Un enfoque práctico. 6ta edición, Cap 14. pág. 1.*

BIBLIOGRAFÍA

1. **Acosta, Ivette Rosa Teodosio.** *Exploración y diseño de la Herramienta de Administración de Bases de Datos para PostgreSQL (HABD).* 2011.

2. *ArPUG grupo de usuarios de Argentina .* [En línea] [Citado el: 18 de 11 de 2012.] http://www.arpug.com.ar/trac/wiki/PgAdmin.

3. *ArPUG grupo de usuarios de Argentina.* [En línea] [Citado el: 18 de 11 de 2012.] http://www.arpug.com.ar/trac/wiki/sql-Vacuum.html.

4. *ArPUG grupo de usuarios de Argentina.* [En línea] [Citado el: 18 de 11 de 2012.] http://www.arpug.com.ar/trac/wiki/sql-Analyze.html.

5. *Base de Datos.* [En línea] [Citado el: 8 de 11 de 2012.] http://www.uprb.edu/profesor/ntorres/base_de_datosventajasdesventajas.htm.

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

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

8. *Comunidad Técnica Cubana de PostgreSQL. PostgreSQL Cuba.* [En línea] [Citado el: 10 de 11 de 2012.] http://postgresql.uci.cu/node/134.

9. **Canós, José H, Letelier, Patricio y Penadés, Carmen.** *Metodologías Ágiles en el Desarrollo de Software.* [En línea] [Citado el: 16 de 11 de 2012.] http://www.willydev.net/descargas/prev/TodoAgil.Pdf .

10. **Camejo, Onaysi Vasallo y Artigas Yislén Dolores Ramírez.** *Proceso de Pruebas de Liberación al Sistema de Manejo de Datos de Ensayos Clínicos Cubano. Ciudad de la Habana: Universidad de las Ciencias Informáticas., 2009.*

11. **Castellanos, Yeneysi Pérez y Martínez, Marlon Alain Morejón.** *Plugin diseñador de bases de datos para la Herramienta de Administración de Bases de Datos HABD.* La Habana : s.n., 2011.

12. **de Jalón, Javier García, Rodríguez , José Ignacio, Sarriegui, José María y Brazález, Alfonso.** *Aprenda C++ como si estuviera en primero.* San Sebastián : s.n., abril 1998.

13. *Database Administration: The Complete Guide to Practices and Procedures.* June 14, 2002.

14. *EMS Manager.* [En línea] [Citado el: 19 de 11 de 2012.] http://www.freedownloadmanager.org/es/downloads/SME_Gerente_de_PostgreSQL_37536_p/.

15. *EcuRed.* [En línea] [Citado el: 15 de 11 de 2012.] http://www.ecured.cu/index.php/Metodolog%C3%ADas_de_desarrollo_de_software.
16. *EcuRed.* [En línea] [Citado el: 19 de 11 de 2012.] http://www.ecured.cu/index.php/Lenguaje_Unificado_de_Modelado.
17. *Entorno Visual de Aprendizaje.* [En línea] [Citado el: 18 de 2 de 2013.] http://eva.uci.cu/mod/resource/view.php?id=9400&subdir=/UML_y_Patrones/Construccion de un modelo conceptual.
18. **Escribano, Gerardo Fernández.** *Introducción a Extreme Programming.* 9-12-2002.
19. ExplicaXP. [En línea] [Citado el: 10 de Enero de 2012.]www.willydev.net/descargas/prev/ExplicaXP.pdf.
20. **Garrido, Salvador Alemany.** [En línea] [Citado el: 16 de 11 de 2012.] http://www.polinux.upv.es/drupal/files/IntroduccionQt.pd.
21. **Guerrero, Lugo Manuel Barbosa.** *Arquitectura de software como eje temático de investigación*. 2006.
22. **García, Rosa María Mato.** *Diseño de Bases de Datos.* Segunda edición corregida y aumentada. s.l.: Pueblo y educación, septiembre del 2005. p. pp 2. 968-444-419-2. 165.
23. **Harvey, M.Deitel y J.Deitel.** *Como programar en C/C++ y Java.*
24. *masadelante.com.* [En línea] [Citado el: 8 de 11 de 2012.] http://www.masadelante.com/faqs/base-de-datos.
25. *Manual de usuario de PostgreSQL.Cap 23.*
26. *Manual de usuario de PostgreSQL.Cap 24.*
27. **Meyer y Pérez, Lisandro Damián Nicanor.** *Introducción al desarrollo multiplataforma con Qt® 4.* Septiembre 2007.
28. *MSDN .* [En línea] [Citado el: 17 de 11 de 2012.] http://msdn.microsoft.com/es-es/library/ms172579%28v=vs.80%29.aspx.
29. *Navicat.* [En línea] [Citado el: 19 de 11 de 2012.] www.navicat.com.
30. PostgreSQL. [En línea] [Citado el: 8 de 11 de 2012.] http://postgresql.uci.cu/.
31. **Pressman, Roger S.** *Ingeniería de Software. Un enfoque práctico. 6ta edición, Cap 13. pág. 1.*
32. **Pressman, Roger S.** *Ingeniería de Software. Un enfoque práctico. 6ta edición, Cap 14. pág. 1.*
33. **Reynoso, Carlos Billi.** *Introducción a la arquitectura de software. versión 1.0 . Universidad de Buenos Aires. Buenos Aires : s.n.,* 2004.

34. **Rodríguez, Hermes Alexy Marañón y Montero, Pedro David Duharte.** *Planificación y Diseño del Portal para la Comunidad Técnica Cubana de PostgreSQL.* La Habana : s.n., 2010.

35. **Rumbaugh, James.** *El Lenguaje Unificado de Modelado.* California: s.n., 1998.

36. *SendBlaster.* [En línea] [Citado el: 17 de 11 de 2012.] http://www.sendblaster.es/database-maintenance/.

37. **Schildt's, Herb.** C++ Programming CookBook. New York, Chicago, San Francisco: s.n., 2008, ISBN: 0-07-164385-0.

38. **Serradilla, Juan Luis.** Control de Versiones con Subversion y TortoiseSVN. Sección de Metodología, Normalización y Calidad del Software ATICA, Universidad de Murcia, Vicerrectorado de Investigación y Nuevas Tecnologías. Murcia: s.n., 2007.

39. Sommerville, **Ian.** *Ingeniería de Software, séptima edición.*

40. UCI. [En línea] [Citado el: 8 de 11 de 2012.] http://www.uci.cu/mision.

41. *UNIVERSIDAD DE MURCIA.* [En línea] [Citado el: 15 de 11 de 2012.] http://www.um.es/docencia/barzana/IAGP/IAGP2-Metodologias-de-desarrollo.html.

42. **Veitía, Beatriz Piñeiro.** *Plugin de monitorización para la herramienta de administración de base de datos HABD.* La Habana : s.n., 2011.

43. **Visconti, Marcello y Astudilo, Hernán.** *Fundamentos de Ingeniería de Software.*

44. **Wesley, Addison.** *Comparative Programming Languages.* s.n., *1993*.

45. **Zapata, Antonia.** *Diseño estructural: Diagrama de clases.*

GLOSARIO DE TÉRMINOS

Disparadores: Es el procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación.

GNU: El proyecto GNU fue iniciado por Richard Stallman con el objetivo de crear un sistema operativo completamente libre.

Lenguaje de consulta estructurado (SQL): Es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas.

Lenguaje procedural: Es un lenguaje donde el usuario especifica que datos se necesitan y como obtenerlos.

Licencia BSD: La licencia BSD (Berkeley Software Distribution) es una licencia de software libre permisiva. Esta licencia tiene pocas restricciones y permite el uso del código fuente en software no libre.

Licencia GPL: La licencia GPL (General Public License) obliga a incluir el código fuente en su distribución, siendo imposible cambiar la licencia al programa, al distribuirlo tal cual o modificado.

Script: Contiene las descripciones de las instrucciones utilizadas para crear una base de datos.

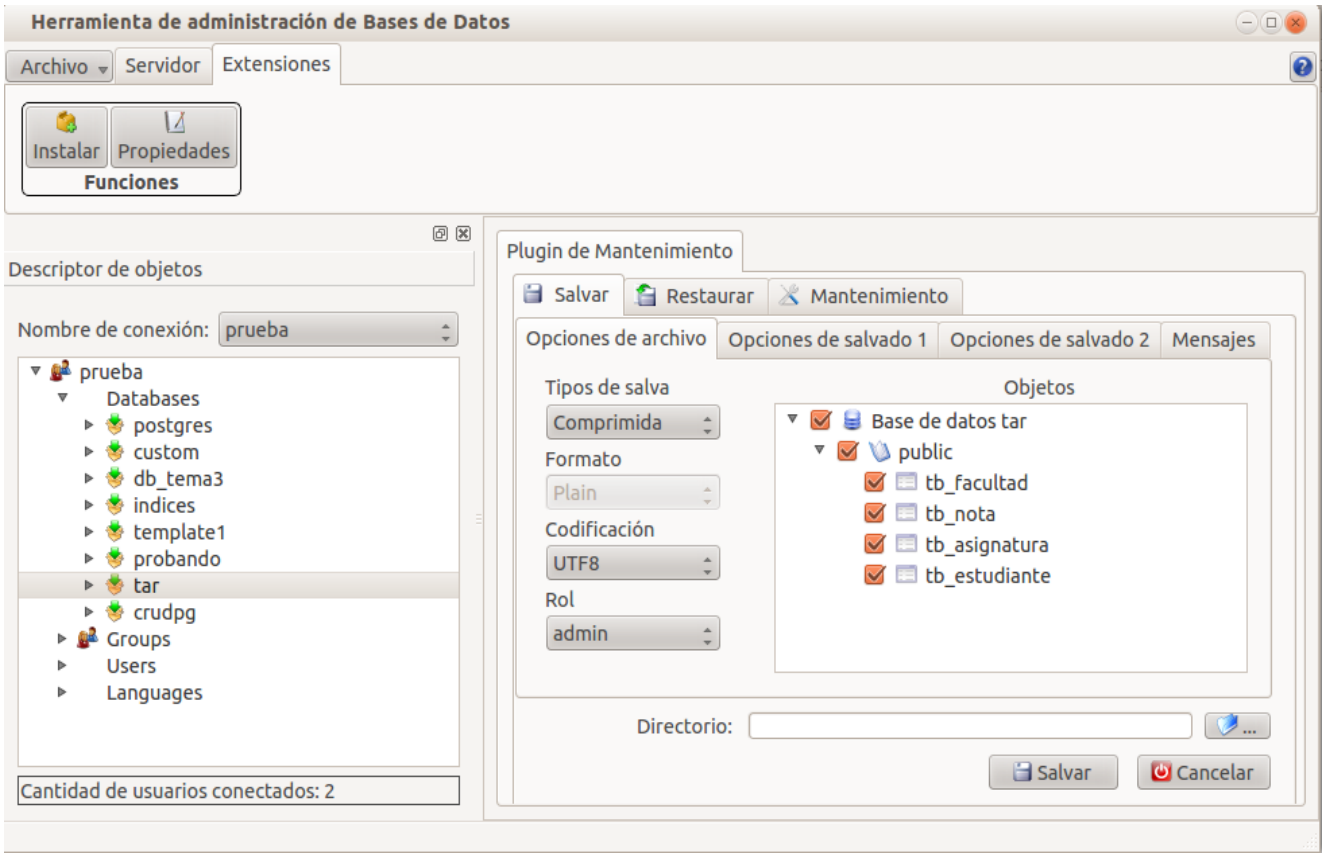
TCP/IP: Es la base de Internet, y sirve para enlazar computadoras que utilizan diferentes sistemas operativos.

Vistas materializadas: En un sistema de gestión de base de datos que siga el modelo relacional, una vista es una tabla virtual, que representa el resultado de una consulta. Una vista materializada utiliza una aproximación diferente: el resultado de la consulta se almacena en una tabla caché real, que será actualizada de forma periódica a partir de las tablas originales.

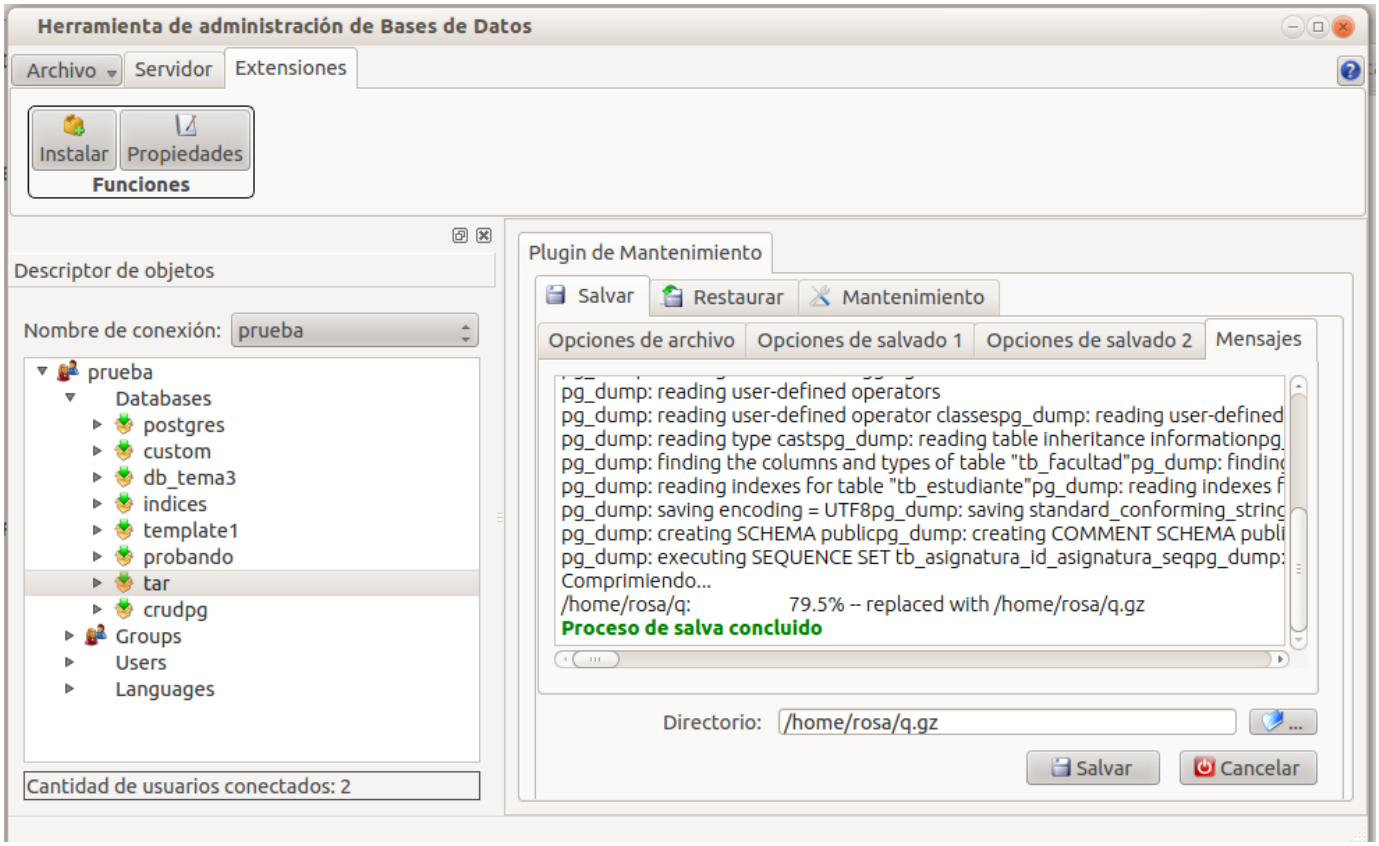
wxWidgets: Es una biblioteca de C++ que permite a los desarrolladores crear aplicaciones para Windows, OS X, Linux y UNIX.

ANEXOS

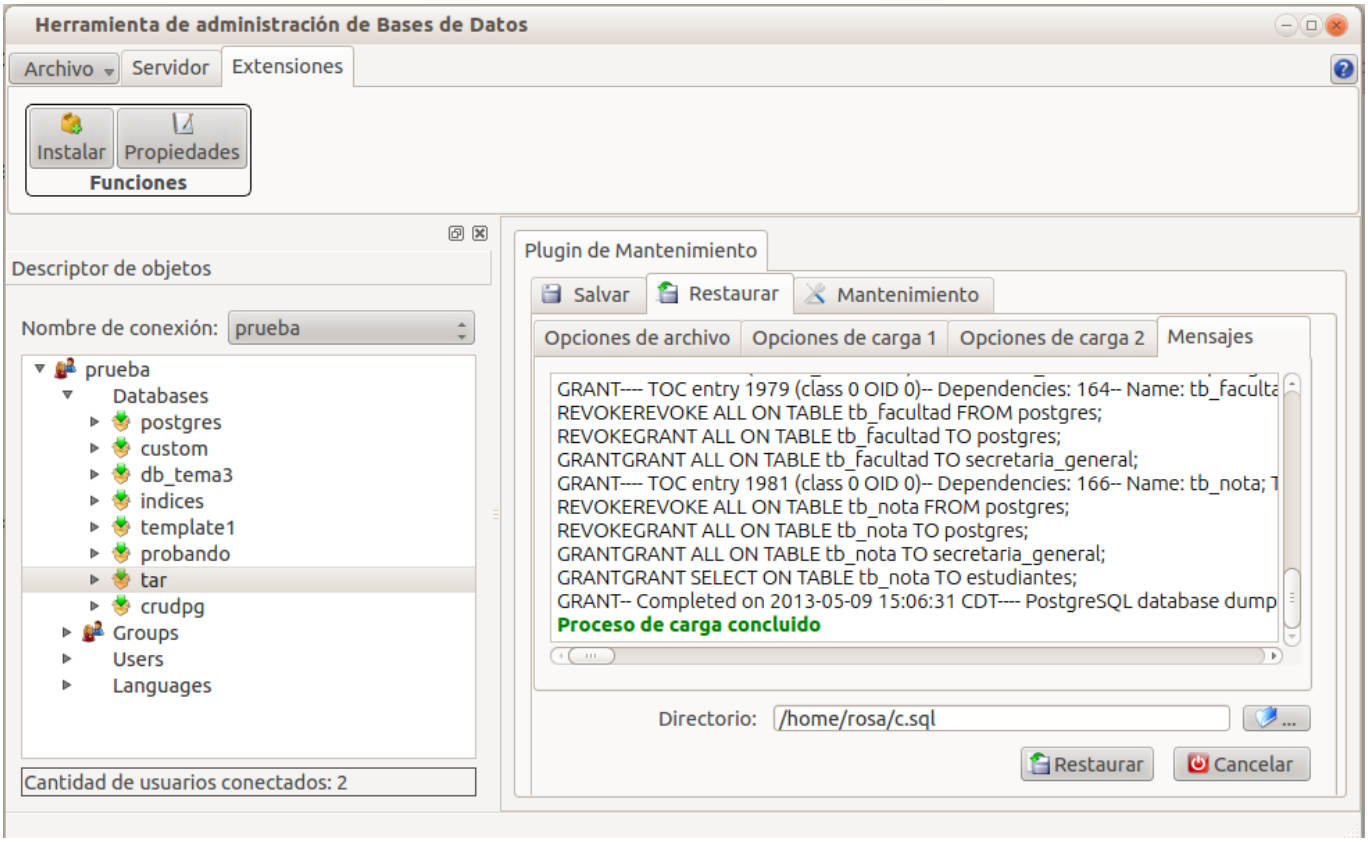
Anexo # 1: Plugin de mantenimiento integrado en la herramienta HABD.



Anexo # 2: Proceso de salva concluido.



Anexo # 3: Proceso de carga concluido.



Anexo # 4: Proceso de mantenimiento concluido.