Universidad de las Ciencias Informáticas

Facultad 2

Título: Plugins para evaluar resultados de las auditorías a sistemas gestores de bases de datos MySQL y PostgreSQL.

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

Autores: Dilonyx Espino Díaz

Rodolfo Ernesto Sevilla Mercantety

Tutores: Ing. Annia Pimentel Rivero

Lic. Ernesto Arbois García

La Habana, Junio 2012

Año 54 de la Revolución

Si supiese que es lo que estoy haciendo, no lo llamaría investigación, ¿verdad?

Albert Einstein.

DECLARACIÓN DE AUTORÍA

Declaramos ser los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) y a la Facultad (2) a hacer uso del mismo en su beneficio.

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

_____	_____
Firma de la Autora	Firma del Autor
Dilonyx Espino Díaz	Rodolfo Ernesto Sevilla Mercantety
_____	_____
Firma de la Tutora	Firma del Tutor
Ing. Annia Pimentel Rivero	Lic. Ernesto Arbois García

DATOS DE CONTACTO

Tutora: Ing. Annia Pimentel Rivero.

Institución: Universidad de las Ciencias Informáticas

Correo electrónico: apimentel@uci.cu

Título de graduado: Ingeniero en Ciencias Informáticas.

Categoría docente: Instructor.

Dirección de la institución: Carretera a San Antonio de los Baños, Km. 2 ½, Torrens, municipio: Boyeros, La Habana, Cuba.

Tutor: Lic. Ernesto Arbois García.

Institución: Universidad de las Ciencias Informáticas

Correo electrónico: earbois@uci.cu

Título de graduado: Licenciado en Ciencias de la Computación.

Categoría docente: Ninguna.

Dirección de la institución: Carretera a San Antonio de los Baños, Km. 2 ½, Torrens, municipio: Boyeros, La Habana, Cuba.

DEDICATORIA

Dedico este trabajo de diploma:
A mi abuela materna María Salome Lanzas Tugores.
A mi hermana Dailenis Espino Díaz.
A mis padres Juan Roberto Espino Morales y Victoria de la Caridad Díaz Lanzas.
A mi novio Osmani Acosta Cantero.

Dilonyx Espino Díaz.

Dedico este trabajo de diploma:

A mis dos abuelos fallecidos recientemente Ernesto Sevilla y Juan Marcantety.

A mis padres Marlenis Mercantety Montoya y Rodolfo Sevilla Milanés.

A mi querida y estimada novia Sandra Falcón Pasamontes.

Rodolfo Ernesto Sevilla Mercantety.

AGRADECIMIENTOS

Quiero agradecerle especialmente a todos los que hoy han venido a presenciar la discusión del trabajo de diploma, por todo su apoyo muchas gracias.

A mis padres por toda la dedicación, educación, amor que me han brindado, por sus exigencias y sus regaños que han hecho de mi la persona que soy hoy, por ser los tutores de mi vida, gracias por todo.

A mi hermanita por ser mi inspiración, porque por ella es que quiero llegar lejos en esta vida.

A mi abuelichi por ser mi segunda mamá, por sus consejos, sus mimos, su cariño por quererme tanto y darme la oportunidad de disfrutar de su compañía.

A mi novio osmani por su cariño, por siempre estar a mi lado durante estos 5 años, apoyándome y queriendo que todo me salga lo mejor posible, por todos los momentos lindos y divertidos.

Quiero también agradecerle a Dios porque a pesar de que no soy de las que voy a la iglesia, tengo mucha, mucha fe en él.

A mi tía abuela Cary, mi tía Tania, María del Carmen, Pablo y Lázaro, por su ayuda, sus buenos consejos y por su apoyo.

Al profesor Renilde por ser una persona recta y por su educación.

A mi segunda familia Valdes, Marlen por todos sus buenos consejos.

A mi amiga de toda la vida Liset por ser esa persona noble y buena que me ha brindado toda su amistad desde que estábamos en la barriga de nuestras madres y quien dice que no, hasta un poquito antes también.

A la familia de mani por aceptarme como un miembro más y bridarme su cariño.

A mis amigos Wilfre, Yoli, Giselle, Yanis, Yannier por todos estos años compartiendo conmigo sus más sinceros sentimientos.

A Angel y al Chino por toda su ayuda, por su amistad, por los sustos que me daban cada vez que me decían que cambiaban algo minutos antes de las exposiciones y al final era mentira y me tenían con el corazón en la boca, gracias por todo, en ustedes muchas veces encontré un gran apoyo

A mi compañero de tesis Rodolfo Sevilla Mercantely como siempre mencione, por poder contar con su buena preocupación, porque es una persona que le gusta que las cosas salgan bien, por su organización que me hizo mucha falta, por poder compartir este tiempecito y llevarnos muy bien a pesar de que nunca habíamos coincidido ni un grupo de clases.

A todos los profesores del proyecto por el esfuerzo realizado para lograr un trabajo satisfactorio.

A mis tutores Annia y Ernesto por sus consejos, por su recomendaciones para lograr que todo saliera bien.

A Leslye porque a pesar de no ser mi tutora oficial, la considero una persona de sentimiento muy buenos, que siempre estaba al tanto de nosotros, que si el documento, que si la matriz, que si esto, que si aquello de verdad gracias por todo.

A Yasser por todo el apoyo durante este curso, por sus consejos en cuando a toda la metodología de la investigación.

A los profesores del tribunal por todas las recomendaciones para que finalmente se obtuviera un trabajo con mejor calidad.

A Oscar la pandemia, por dedicar tanto tiempo de su vida a la mía, por ser esa persona de tan buenos sentimientos y por ser mi chofer particular.

A todos los compañeros que han compartido conmigo desde los inicios en esta universidad, a Esbietta por ser aquella primera persona que compartió conmigo toda su amistad.

Dilonyx Espino Díaz.

AGRADECIMIENTOS

A mi querida madre Marlenis Mercantety Montoya agradecerle todo lo que ha hecho por mi, sin su educación, sin su ejemplo de lucha y perseverancia no hubiera llegado hasta donde estoy, gracias por todo mami, Te Quiero Mucho.

A mi peleón y caprichoso padre agradecerle todo lo que me ha dado, gracias por enseñarme como debo compórtame en la vida. Estas líneas no bastarían para expresar mi agradecimiento por todo lo que has hecho por mí, pero para ser concreto me siento HONRADO por tenerte como padre.

A mi Sandrita muchas gracias por el apoyo que siempre me has brindado, eres una persona especial en mi vida y sin tu apoyo no me sentiría satisfecho. Te amo con la vida.

A toda mi familia agradecerles por la confianza y el apoyo incondicional que siempre me brindaron, estaré eternamente agradecido por ello.

A mi compañera de tesis Dilonyx Espino Díaz primero agradecerle por el aguante y comprensión que tuvo conmigo y segundo por toda la preocupación y responsabilidad que asumió a lo largo del desarrollo de nuestro trabajo de diploma.

A mis tutores Annia Pimentel Rivero y Ernesto Arbois García gracias por el apoyo, preocupación y dedicación hacia nosotros, sin su ayuda hubiese sido imposible realizar este trabajo de diploma con la calidad que lleva.

A todo el equipo del Proyecto agradecerle por toda la colaboración, sin duda alguna en la unión esta la fuerza y si el trabajo en equipo no funciona, no existirá un resultado satisfactorio.

A mis amigos Luis Ángel Vázquez Vega, Alfredo Dunn Pellicer, Antonio Alameda Carricarte y Leandro Benítez Ortiz por su desinteresada y sincera amistad.

A todos los profesores que a lo largo de la carrera aportaron su granito de arena para hacer de mí el Ingeniero en Ciencias Informáticas que hoy soy.

A todas las amistades que he conocido en la universidad que de una forma u otra me han apoyado, especialmente a mi gente de la facultad 3, a las chicas y chicos de mi facultad y mis grandes colegas de la facultad 7.

A todos los compañeros de los grupo 2105, 2205, 2305, 2401 y 2501.

Rodolfo Ernesto Sevilla Mercantety.

RESUMEN

Con el desarrollo de las infraestructuras computacionales y el crecimiento de la información que manejan también se han incrementado los ataques informáticos enfocados al robo y destrucción de dicha información, con el objetivo de ocasionar daños severos a empresas y organizaciones. La Empresa de Telecomunicaciones de Cuba (ETECSA) posee un Departamento de Gestión de la Seguridad Informática (DGSI), que tiene el objetivo de velar por la integridad, confidencialidad y disponibilidad de toda la información que procesa la institución.

Para el cumplimiento de dicho objetivo entre otras tareas, este departamento lleva a cabo auditorías informáticas a los Sistemas Gestores de Bases de Datos (SGBD), que son realizadas prácticamente de forma manual por el personal especializado en el tema, lo que provoca que el proceso se realice en más tiempo del esperado y lo hace propenso a errores inherentes por la forma de manejar la información. Como solución a la situación planteada se propone el desarrollo de una herramienta informática para automatizar las auditorías de seguridad informática siguiendo la metodología por la que se rige ETECSA, que permita disminuir el tiempo empleado y los errores cometidos dichas auditorías. En el trabajo de diploma se expone específicamente los resultados del estudio realizado para el desarrollo de los dos complementos que formarán parte un sistema que realiza las auditorías informáticas, dichos complementos serán encargados de la evaluación de resultados obtenidos de las auditorías a SGBD MySQL y PostgreSQL.

PALABRAS CLAVE: Auditorías, Bases de Datos, Sistemas Gestores de Bases de Datos, MySQL, PostgreSQL, Plugins.

TABLA DE CONTENIDOS

INTRODUCCIÓN

En el presente siglo las nombradas Tecnologías de la Información y las Comunicaciones (TIC) forman parte del eje central en que se mueve la sociedad y la economía mundial, tomando mayor protagonismo día a día debido a las mejoras y facilidades que presenta; eliminando las barreras geográficas, facilitando el acceso a la educación y a la cultura, con el acceso a Internet las personas se convierten en receptores o en emisores de información a través de los blogs y Wikipedia, las investigaciones encuentran una rápida difusión y se enriquecen con nuevas ideas con la colaboración entre investigadores separados por miles de kilómetro, conllevando a que la humanidad esté mejor preparada y cuente con mejores comunicaciones. Cuba como país en vía de desarrollo tiene que adaptarse a este fenómeno y hacer evolucionar sus entidades económicas en el sentido de automatizar en la medida de lo posible sus procesos internos y asegurar la integridad de los mismos.

La información generada por los procesos empresariales se almacena en Bases de Datos (BD); siendo un recurso vital para las organizaciones, por lo que asegurar su correcta manipulación e integridad se convierte en primordial objetivo. En el mundo empresarial conocer qué y cómo se hace el trabajo puede ser vital para la competitividad y debido a esto hubo quienes dedicaron su esfuerzo para evitar la incorrecta manipulación de los datos o la pérdida de los mismos.

ETECSA, posee diversas BD, en las que se almacena información sensible para la compañía, registros financieros, cálculos de nómina de los empleados, etc. A nivel mundial existen estándares definidos encaminados a minimizar los riegos que provocan las vulnerabilidades de las infraestructuras computacionales. Las auditorías de seguridad informática juegan un papel fundamental en la verificación del cumplimiento de las políticas de seguridad informática dictadas por una institución.

El DGSI de ETECSA es el encargado de verificar que se cumplan principios de seguridad informática como la integridad, confidencialidad y disponibilidad de la información. En estos momentos la empresa para realizar las auditorías a los SGBD cuenta con un conjunto de scripts[1], luego de ser ejecutados estos scripts en el gestor se obtienen resultados los cuales son analizados y traspasados manualmente, lo que tiende a involucrar errores humanos.

[1] Un script no es más que un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano.

El proceso incluye una matriz elaborada a partir de las deficiencias detectadas, comparándolas con las buenas prácticas internacionales, también se insertan datos como las observaciones, la evaluación obtenida y las mejoras sugeridas, todo este proceso se efectúa de forma manual. Lo expuesto implica que el proceso se dificulte y consuma más tiempo del que realmente se necesitaría, además alguna imprecisión o equivocación por parte de quienes lo realizan, puede traer como consecuencia que la seguridad de la empresa en algún momento se vea amenazada.

Lo anterior ocurre porque ETECSA no cuenta con una herramienta que sea capaz de evaluar los resultados obtenidos al ejecutar los scripts y confeccionar la matriz de resultados con la valoración de cada parámetro medido en la realización de las auditorías informáticas.

Con la finalidad de dar solución a la **situación problemática** anteriormente expuesta se plantea como **problema a resolver:** ¿Cómo disminuir el tiempo que toma y los errores que se cometen en la evaluación de los resultados obtenidos en la realización de auditorías informáticas a los SGBD MySQL y PostgreSQL que se efectúan en ETECSA? Fijándose como **objeto de estudio** de la investigación, los procesos relacionados con las auditorías informáticas a SGBD y como **campo de acción** el proceso de evaluación de los resultados obtenidos en las auditorías informáticas a los SGBD MySQL y PostgreSQL en el DGSI de ETECSA.

Como **objetivo general** se define desarrollar plugins que permitan realizar la evaluación de los resultados obtenidos en las auditorías informáticas a los SGBD MySQL y PostgreSQL, para reducir el tiempo y los errores cometidos en las mismas.

Para su cumplimiento se han definido los siguientes **objetivos específicos:**

- Realizar el marco teórico de la investigación.
- Caracterizar el sistema informático a desarrollar.
- Integrar el plugin que realiza la evaluación de los resultados obtenidos en la realización de las auditorías informáticas a los SGBD MySQL al Sistema para la Realización de Auditorías a Sistemas Gestores de Bases de Datos (SASGBD).

- Integrar el plugin que realiza la evaluación de los resultados obtenidos en la realización de las auditorías informáticas a los SGBD PostgreSQL al SASGBD.

- Realizar pruebas al software obtenido.

Defendiendo la idea de que “si se utilizan los plugins que evaluarán los resultados obtenidos en las auditorías a los SGBD MySQL y PostgreSQL se podrán disminuir los errores cometidos y el tiempo utilizado en la obtención de los resultados de dicho proceso”.

Para cumplir con lo descrito se han propuesto las siguientes **tareas a realizar:**

- Realización de encuentros y entrevistas con el personal del DGSI de la empresa cubana ETECSA, para conocer cómo se ejecutan los procesos relacionados con las auditorías de la seguridad informática a los SGBD MySQL y PostgreSQL.

- Caracterización de los sistemas informáticos existentes en el mundo que realizan auditorías a la seguridad informática a SGBD MySQL y PostgreSQL.

- Realización de un estudio de las metodologías, herramientas y lenguajes de programación que pudieran usarse en la solución informática con el objetivo de efectuar una correcta selección.

- Selección de la metodología, herramientas y lenguajes de programación que se utilizarán para el desarrollo de los plugins.

- Análisis de las aplicaciones basadas en plugins para consolidar conocimientos sobre este tipo de desarrollo.

- Diseño de los plugins para la evaluación de los resultados obtenidos en la realización de auditorías de seguridad informática a SGBD MySQL y PostgreSQL.

- Implementación del plugin para la evaluación de los resultados obtenidos en la realización de auditorías a la seguridad informática a SGBD MySQL.

- Implementación del plugin para para la evaluación de los resultados obtenidos en la realización de auditorías a la seguridad informática a SGBD PostgreSQL.

- Realización de las pruebas de caja negra y blanca a la solución informática obtenida para detectar y corregir posibles errores o fallas.

Con el propósito de solucionar y dar cumplimiento al objetivo general y las tareas propuestas, se utilizaron varios **métodos científicos** que definen una estrategia general para enfrentar el problema que se investiga, con una dirección consciente que garantice una organización basada en un análisis teórico precedente y guarde una relación directa con la esencia misma del fenómeno, con sus leyes y regularidades y que tenga en cuenta la práctica como punto de partida y finalidad del conocimiento. (1)

- **Histórico-Lógico:** analiza la trayectoria completa del fenómeno, su condicionamiento a los diferentes periodos de la historia, además se basa en el estudio histórico del fenómeno y pone de manifiesto la lógica interna de su desarrollo. (1) Este método se utiliza para estudiar la teoría conocida hasta el momento, así como para conocer los antecedentes de las auditorías a los SGBD MySQL y PostgreSQL, específicamente de la evaluación de los resultados que arrojan las mismas.

- **Analítico-Sintético:** permite la división mental del fenómeno en sus múltiples relaciones y componentes para facilitar su estudio y establece mentalmente la unión entre las partes previamente analizadas. (1) Este método permitió analizar las teorías conocidas hasta el momento, buscar la esencia de la evaluación de los resultados de las auditorías, los rasgos y características que la distinguen, además posibilitó el estudio por partes del proceso de evaluación de los resultados de las auditorías informáticas a los SGBD MySQL y PostgreSQL. Con este método se puede estudiar los documentos referentes al objetivo de la investigación facilitando la extracción de los elementos más importantes que se relacionan con el objeto de estudio y el campo de acción.

- **Modelación:** La modelación es el método mediante el cual se crean abstracciones con el objetivo de explicar la realidad (1). Se pone en práctica en el momento de realizar los diagramas y modelos que permitirán la posterior implementación de las funcionalidades, además posibilitará descubrir y estudiar nuevas relaciones y cualidades del objeto de estudio, siendo el objeto de estudio el eslabón principal de la relación entre el modelo y el objeto que se desea modelar.

Dentro de los métodos de investigación científica también se hizo uso de los **métodos empíricos**, específicamente la entrevista y la experimentación, que representan un nivel de investigación cuyo contenido procede de la experiencia y es sometido a cierta elaboración racional. (1)

La entrevista se emplea con la perspectiva de obtener un mayor conocimiento a la hora de entender el negocio, viene a ser una conversación planificada entre el investigador y el personal con el conocimiento necesario en el tema de las auditorías informáticas, específicamente de las formas de evaluación de los distintos parámetros a medir para obtener información acerca de este proceso. (1)

La experimentación es el método empírico para el estudio de un objeto en el cual el investigador crea condiciones o adapta las existentes para el esclarecimiento de las propiedades, leyes y relaciones del objeto, para verificar una hipótesis, una teoría o un modelo. El experimento puede ser transformador o comprobador. En el primero se revela la realidad y se actúa sobre ella para transformarla, es un experimento creador, en el segundo caso se verifica el estado del fenómeno. (1) Este método se utilizó como comprobador, en la realización de las pruebas de calidad con el objetivo de validar la solución informática.

Este trabajo de diploma está estructurado como se describe a continuación:

Capítulo 1. "Fundamentación Teórica"

Permite encontrar los principales conceptos que se manejan a lo largo del trabajo; estado del arte de las herramientas existentes en el mundo, así como las tendencias, técnicas, tecnologías, metodologías y software usados para la solución a la problemática actual.

Capítulo 2: "Características del sistema"

Se realiza el análisis del problema existente para entender y llegar a una solución, se construye el modelo de negocio adecuado, la especificación de los requisitos que constituirán las bases de la propuesta del sistema, así como la elaboración de los diagramas y la descripción de los casos de uso del sistema.

Capítulo 3: "Análisis y Diseño del sistema"

Representa la base de la futura implementación del sistema, se definen los patrones de diseño y arquitectura, también se elaboran de diagramas de clases del diseño y los diagramas de interacción, específicamente los de secuencias.

Capítulo 4: "Implementación y Prueba"

Muestra cómo será implementado el sistema en términos de componentes, realizando posteriormente las pruebas de calidad al software.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción.

En el presente capítulo se expone el estudio y análisis del estado del arte de las soluciones existentes que realizan auditorías de seguridad informática a SGBD. También se muestran los principales conceptos y los resultados del estudio realizado referente a la selección de la metodología, lenguaje de programación y el marco de trabajo o framework, así como la descripción de las herramientas que se utilizarán para el desarrollo, las cuales facilitarán obtener la solución a la problemática planteada.

1.2. Conceptos.

1.2.1 Bases de Datos.

Una base de datos es la representación integrada de los conjuntos de entidades, instancias correspondientes a las diferentes entidades tipo del sistema informático y de sus interrelaciones. Esta representación informática (o conjunto estructurado de datos) debe poder ser utilizada de forma compartida por muchos usuarios de distintos tipos. (2)

1.2.2 Gestores de Bases de Datos.

Un gestor de base de datos o sistema de gestión de base de datos (SGBD o DBMS) es un software que permite introducir, organizar y recuperar la información de las bases de datos; en definitiva, administrarlas. Existen distintos tipos de sistemas gestores de bases de datos, relacionales, jerárquicos y de red. (3)

1.2.3 Auditoría Informática.

La auditoría informática es el proceso de recoger, agrupar y evaluar evidencias para determinar si un sistema de información salvaguarda el activo empresarial, mantiene la integridad de los datos, lleva a cabo eficazmente los fines de la organización y utiliza eficientemente los recursos. (4)

1.2.4 Auditoría a Bases de Datos.

Auditar las bases de datos no es más que el proceso que permite medir, asegurar, demostrar, monitorear y registrar los accesos a la información almacenada en las bases de datos incluyendo la capacidad de determinar:

- Quien accede a los datos.

- Cuando se accedió a los datos.
- Desde qué tipo de dispositivo/aplicación.
- Desde que ubicación en la red.
- Cual fue la sentencia SQL ejecutada.
- Cual fue el efecto del acceso a la base de datos. (5)

Las auditorías a bases de datos tienen gran importancia debido a que toda la información financiera, de recursos humanos y además datos que manejan las empresas están almacenados en bases de datos, por lo que debe existir un control sobre el acceso a estas para velar su seguridad. A través de estas auditorías también se demuestra la integridad de la información y las organizaciones pueden mitigar los riesgos que tienen relación con la pérdida y fuga de datos.

1.3. Sistemas de Auditorías de Seguridad Informática a SGBD en el mundo.

En esta sección del presente capítulo se aborda sobre las soluciones informáticas que existen en el mundo para realizar las auditorías a los SGBD. También se analizará su competitividad, sus licencias, si son multiplataforma, entre otros aspectos con el objetivo de estudiar sus funcionalidades y características.

1.3.1. DB AuditExpert.

DB AuditExpert es una poderosa solución de auditorías a bases de datos con capacidades nativas en Oracle, Sybase, SQL y Ds2 con una consola de administración central intuitiva, un depósito central de datos auditados, compilación de reportes predefinidos, reportes, análisis forense y servicios de análisis y desempeño. (6)

Esta herramienta extrae de cada base de datos las configuraciones de parámetros para su auditoría en una interfaz gráfica única, teniendo como ventaja que se disminuya el tiempo en aprender de los errores que se cometen, permite a los administradores de las bases de datos y sistemas, administradores de seguridad, auditores y los operadores poder analizar cualquier actividad que se realice como por ejemplo el acceso a la información, la creación, modificación y eliminación de los datos, todo esto en una sola herramienta.

Beneficios de DB AuditExpert:

- Mejora la seguridad del sistema y asegura responsablemente el sistema.
- Captura los accesos traseros o “back-door” a las bases de datos no permitidos.

- Presenta características de seguridad centralizadas y control de auditoría de múltiples sistemas de Base de Datos desde una sola ubicación otorgando una fácil administración.
- Presenta características unificadas en una interfaz gráfica acortando la curva de aprendizaje y es fácil de usar.
- Provee reportes analíticos, reduciendo así grandes cantidades de datos en una auditoría presentando resúmenes comprensivos que permiten identificar fácilmente varias violaciones a la seguridad de base de datos.
- Provee reportes analíticos que ayudan a identificar qué usuarios y procesos ahogan los recursos del sistema.
- Proporciona la capacidad de generar alarmas en tiempo real al correo electrónico del personal clave cuando se presentan cambios en datos sensibles.
- Proporciona una total transparencia a nivel de sistema y los datos de cambio de las auditorías de todas las aplicaciones existentes sin necesidad de realizar cambios en esas aplicaciones.
- Compatibilidad total con todos los Sistemas Operativos (SO) nativos con el soporte para ejecutar bases de datos, incluidos pero no limitados; Windows NT, UNIX, Linux, VMS, OS/390, z/OS.

Otras informaciones

- Última versión: 4.2.26.1
- Fecha de Liberación: Mayo 20, 2010

Especificaciones técnicas

- Espacio en Disco - 42-66 MB, Memoria - 512 MB.
- CPU - Pentium o compatible.
- GUI Console OS: Windows 2000/XP/2003/2008/Vista/7 (DB Audit Management Console).
- WebConsole: Cualquier servidor web capaz de correr Java Services Page (JSP), DB Audit Web-based Management Console, Apache Tomcat recomendado.

SGBD soportados:

- Oracle 7.3, 8.0, 8i, 9i, 10g, 11g
- Microsoft SQL Server 6.5, 7, 2000, 2005, 2008
- Sybase SQL Server y Sybase Adaptive Server Enterprise 10.x, 11.x, 12.x, 15.x
- MySQL 4.2, 5.0, 5.1, 5.4

- Conexiones vía drivers nativos de BD: ODBC, JDBC o OLEDB/ADO.

Esta es una herramienta con numerosos beneficios que hacen de ella una solución poderosa para la realización de auditorías a SGBD, pero tiene como desventaja que la consola de administración solo trabaja en los sistemas operativos desde Windows 2000 hasta Windows 7 cuando ETECSA también utiliza sistemas operativos GNU/Linux, además no realiza auditorías para el SGBD PostgreSQL, por lo que no es una solución factible.

1.3.2. MysqlPasswordAuditor

MysqlPasswordAuditor es un programa gratuito de recuperación de contraseña de un servidor MySQL y puede ser usada para auditar SGBD MySQL y descubrir contraseñas débiles.

MysqlPasswordAuditor es muy fácil de usar, por fuerza bruta empieza a probar todas las contraseñas que estén en nuestro diccionario. Por defecto se incluye una lista de contraseñas de archivos pequeños. MysqlPasswordAuditor trabaja solo en Windows, funciona desde el XP hasta Windows 7. (7)

Características dominantes de "MysqlPasswordAuditor":

- Software libre y simple para recuperar y auditar contraseñas de MySQL
- Muy útil para administradores y probadores de penetración.
- El diccionario se basa en el método de recuperación de contraseña.
- Presenta estadísticas detalladas tales como prueba de contraseñas, tiempo transcurrido, la barra de progreso que se exhibe durante la operación de la intervención.
- La Interfaz Gráfica es simple y fácil de usar.

A pesar de ser uno de los programas de bases de datos más potentes y utilizados por la mayoría de las aplicaciones para detectar y descubrir contraseñas débiles, solo contempla ese aspecto, dejando a la deriva otros aspectos importantes, además que, solo funciona sobre Windows siendo una gran desventaja ya que los servidores de ETECSA operan bajo diferentes sistemas operativos.

1.3.3. DbProtect.

DbProtect es una suite de seguridad para bases de datos, esta herramienta es capaz de combinar detección, escaneo de vulnerabilidades, monitoreo de actividades en tiempo real, auditoría, y cifrado opcional, respondiendo a su principio el cual es ayudar a las organizaciones a reducir el riesgo y mejorar el cumplimiento de sus objetivos. Esta herramienta protege contra mal uso y abuso a más de cien mil bases de datos en todo el mundo.

DbProtect monitorea toda la actividad de usuario y cambios del sistema, vigilar ataques y amenazas complejas así como conductas maliciosas y dar alertas en tiempo real con detalles de contexto.
El asistente de filtrado de Application Security Inc. permite a las organizaciones afinar sus parámetros de detección y personalizar qué eventos de auditoría y seguridad se deben monitorear, enfocando así los esfuerzos de seguridad en la información que es relevante al tiempo que se pasan por alto los falsos positivos y eventos irrelevantes. El mecanismo ASAP Update de DbProtect asegura que la protección permanezca actualizada a medida que se identifican nuevas vulnerabilidades y se publican parches. (8)

El monitoreo de actividades privilegiadas con evidencia de alteraciones:
Defiende contra el mal uso, fraude y abuso por parte de usuarios internos y externos.
La completa evaluación de las vulnerabilidades:
Identifica y reduce el riesgo que presenta las bases de datos.
El monitoreo y detección de intrusiones en tiempo real:
Identifica inmediatamente los ataques a la base de datos o su mal uso.
La administración de brechas de parches:
Ayuda a las organizaciones a priorizar los parches de seguridad de la base de datos y a defenderse contra ataques. Una mejor integración permite informar sobre el progreso de los parches de seguridad, el impacto de la mitigación de riesgos, y el estado del cumplimiento general de las tareas.
El conocimiento de la aplicación:
Proporciona una perspectiva esencial de la infraestructura de las tecnologías de la información y las comunicaciones, permitiendo a las organizaciones comprender mejor su inventario de bases de datos y mitigar con ello los factores de riesgo del cumplimiento de los objetivos, así como enfrentar las necesidades de seguridad de las bases de datos.

DbProtect soporta las siguientes categorías y reglas de evento/ataque:

- Acceso a recursos del SO.
- Eventos de auditoría.
- Ataques de desbordamiento de búfer.
- Ataques de contraseña.
- Intentos de escalamiento de privilegios.
- Herramienta de seguridad en uso.
- Eventos del sistema.

- Ataques de aplicaciones Web.
- Firmas y reglas creadas por el usuario.

Principales Características:

- Detección de intrusiones en tiempo real y auditoría de actividades integradas en la base de datos.
- La arquitectura híbrida que presenta maximiza la profundidad y amplitud del monitoreo.
- Mínimo impacto sobre el desempeño.
- El análisis de actividad en tiempo real maximiza la protección y reduce los gastos generales.
- Integración de informes y alertas automatizados mediante SNMP[2]y SMTP[3]. (8)

Esta herramienta no resulta una solución a ser utilizada por el Departamento de la Gestión de la Seguridad Informática de ETECSA debido a que este cuenta con script para auditar cada SGBD con las consultas que les son necesarias para auditar dichos servidores y esta herramienta no contempla entre sus funcionalidades este aspecto.

1.3.4. AppDetectivePro.

AppDetectivePro es una herramienta de análisis de vulnerabilidades en bases de datos. Es un escáner de red y evaluador de vulnerabilidades, detecta las aplicaciones de bases de datos existentes en la infraestructura y evalúa sus características de seguridad.

Con el respaldo de una probada metodología y un amplio conocimiento de las vulnerabilidades a nivel de aplicación, AppDetectivePro localiza, examina, reporta y repara brechas de seguridad y configuraciones erróneas.

Como consecuencia, las empresas pueden mejorar proactivamente sus aplicaciones de bases de datos, fortalecer sus operaciones de seguridad y demostrar el cumplimiento de las normas regulatorias.

AppDetectivePro soporta bases de datos:

MySQL, Oracle, Sybase, IBM DB2, IBM DB2 en Mainframe, Microsoft SQL Server, Oracle Application Server y Lotus Notes/Domino.

AppDetectivePro tiene como principales características:

[2] SMNP: (**S**imple **N**etwork **M**anagement **P**rotocol o Protocolo Simple de Administración de Red) está diseñado para facilitar el intercambio de información entre dispositivos de red y es ampliamente utilizado en la administración de redes para supervisar el desempeño y el bienestar de una red, equipo de cómputo y otros dispositivos.

[3] SMTP: El protocolo SMTP (**S**imple **M**ail **T**ransfer **P**rotocol o Protocolo Simple de Transferencia de Correo) es el protocolo estándar que permite la transferencia de correo de un servidor a otro mediante una conexión punto a punto.

- Evaluación de vulnerabilidades específicas de base de datos.
- La más completa y actualizada base de conocimiento.
- Búsqueda y administración escalables.
- No requiere la instalación de agentes en los servidores de base de datos.
- Reportes avanzados y configurables por el usuario.
- Instalación sencilla y fácil utilización. (8)

Esta es una suite integrada para evaluación de vulnerabilidades en bases de datos y monitoreo de actividades de base de datos en tiempo real, que protegen contra mal uso y abuso a más de cien mil bases de datos en todo el mundo, esta no realiza auditorías para el SGBD PostgreSQL, por lo que no es una solución viable para ser utilizada por el DGSI de ETECSA.

En este punto resulta evidente que ninguna de las herramientas estudiadas, que son relevantes en el tema de realización de auditorías a SGBD, se ajustan completamente a las necesidades de ETECSA, por lo que resulta necesario construir una aplicación que permita suplirla, siendo parte de dicha solución los plugins que se presentan en este documento.

1.4. Metodología de Desarrollo.

La ingeniería del software se ha vuelto un elemento indispensable en el desarrollo de sistemas computacionales, debido a la magnitud que poseen algunos, resultando su construcción extremadamente compleja. Esta disciplina agrupa técnicas, procedimientos y estrategias que guían el desarrollo de un producto partiendo de las necesidades del cliente.

El objetivo principal de una metodología es presentar un conjunto de técnicas tradicionales y modernas de modelado de sistemas para crear un software de calidad. Para materializar estas técnicas se usan herramientas de análisis y diseño auxiliándose de diagramas, especificaciones y criterios de aplicación de las mismas.

Teniendo en cuenta la filosofía de desarrollo de las metodologías, aquellas con mayor énfasis en la planificación y control del proyecto, en especificación precisa de requisitos y modelado, reciben el apelativo de Metodologías Tradicionales o Pesadas. (9)

1.4.1. Proceso Unificado de Modelado (RUP).

Entre las metodologías tradicionales más usadas se encuentra RUP, un proceso de ingeniería del software que facilita un acercamiento disciplinado a la asignación de tareas y responsabilidades en un

equipo de desarrollo. Su propósito es asegurar que la producción de software de alta calidad se ajuste a las necesidades de sus usuarios finales con costos y calendarios predecibles. (10)

Principales características de RUP:

- **Guiado por casos de uso:** La razón de ser de un sistema software es servir a usuarios ya sean humanos u otros sistemas; un caso de uso es una facilidad que el software debe proveer a sus usuarios. Los casos de uso reemplazan la antigua especificación funcional tradicional y constituyen la guía fundamental establecida para las actividades a realizar durante todo el proceso de desarrollo incluyendo el diseño, la implementación y las pruebas del sistema.
- **Centrado en arquitectura:** La arquitectura involucra los elementos más significativos del sistema y está influenciada entre otros por plataformas software, sistemas operativos, manejadores de bases de datos, protocolos, consideraciones de desarrollo como sistemas heredados y requerimientos no funcionales. Es como una radiografía del sistema que estamos desarrollando, lo suficientemente completa como para que todos los implicados en el desarrollo tengan una idea clara de qué es lo que están construyendo.
- **Iterativo e Incremental:** Para hacer más manejable un proyecto se recomienda dividirlo en ciclos. Para cada ciclo se establecen fases de referencia, cada una de las cuales debe ser considerada como un mini-proyecto cuyo núcleo fundamental está constituido por una o más iteraciones de las actividades principales básicas de cualquier proceso de desarrollo. En concreto RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en las distintas actividades. (10)

RUP es un proceso de desarrollo de software que constituye una de las metodologías más utilizadas en el desarrollo de sistemas e involucra el uso de Unified Modeling Lenguage (UML) para la modelación. Como RUP es un proceso, en su modelación define como sus principales elementos:

- Trabajadores (“quién”): Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto.
- Actividades (“cómo”): Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.
- Artefactos (“qué”): Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.

- Flujo de actividades (“cuándo”): Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.(11)

Se selecciona como metodología de desarrollo a RUP analizando que esta genera abundante documentación, resultando un argumento de vital importancia para el desarrollo nuevas funcionalidades en los complementos, así como nuevos plugins para la evaluación de los resultados de las auditorías. Otro aspecto que se tuvo en cuenta para la selección es la inestabilidad del equipo de desarrollo, este está conformado en su mayoría por estudiantes de 4to y 5to año, lo que permitiría que todo lo realizado por los estudiantes que en el futuro no se encuentren quedará plasmado en los artefactos generados por la metodología, valiendo como referencia para los nuevos integrantes del equipo.

1.4.2. Notación de Modelado de los Procesos del Negocio.

Hace varias décadas los lenguajes y notaciones de procesos de negocio se usan en numerosos campos de la industria del software. Su principal objetivo es la optimización de costes y tiempo, sin embargo en la ingeniería de software esta técnica lleva poco tiempo en uso. No solo se aplica al intentar optimizar el proceso de desarrollo de software en sí mismo sino también en la comunicación durante la obtención de los requisitos.

Business Process Modeling Notation (BPMN) es una notación gráfica que describe la lógica de los pasos de un proceso de Negocio. Esta notación ha sido especialmente diseñada para coordinar la secuencia de los procesos y los mensajes que fluyen entre los participantes de las diferentes actividades.

- BPMN es un estándar internacional de modelado de procesos aceptado por la comunidad e independiente de cualquier metodología de modelado de procesos.
- BPMN crea un puente estandarizado para disminuir la brecha entre los procesos de negocio y la implementación de estos.
- BPMN permite modelar los procesos de una manera unificada y estandarizada permitiendo un entendimiento a todas las personas de una organización. (11)

Un Diagrama de Procesos de Negocio (DPN o BPD) está formado por un conjunto de elementos gráficos, que habilitan el fácil desarrollo de diagramas simples que serán familiares para la mayoría de analistas de negocio (diagrama de flujo). Los elementos se eligen para ser distinguibles los unos de los otros y para usar formas familiares para la mayoría de modeladores. Por ejemplo, las actividades son rectángulos y las decisiones son rombos. Debe notarse que uno de los objetivos del desarrollo de BPMN

es crear un mecanismo simple para crear modelos de procesos de negocio, y al mismo tiempo que sea posible gestionar la complejidad inherente en dichos procesos. (11)

Otros objetivos importantes que plantea esta notación son:

- Crear enlaces entre la implementación del proceso de negocio y su diseño.
- Que los lenguajes basados en XML para describir procesos tengan una notación gráfica.

Un factor importante de BPMN es que esta notación abarca solamente a los procesos de negocio, lo que significa que otro tipo de modelos relacionados como por ejemplo estructura de la organización, recursos, modelos de datos, estrategias, reglas de negocio quedan fuera de la especificación de los procesos.

1.5. Herramientas y lenguajes para el desarrollo.

1.5.1. Lenguaje Unificado de Modelado (UML).

En décadas pasadas los diseños se representaban de diversas maneras. Por no existir una forma común de representar dichos gráficos no se podían compartir fácilmente entre los diseñadores los esquemas confeccionados. Con el objetivo de solucionar este inconveniente y para la mejor comprensión y análisis de los problemas se creó UML.

UML permite modelar sistemas de información y su objetivo es lograr modelos que, además de describir con cierto grado de formalismo los sistemas, puedan ser entendidos por los clientes o usuarios de aquello que se modela. También se usa para entender, diseñar, hojear, configurar, mantener y controlar la información sobre los sistemas. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. (12)

Resumen de los objetivos de UML:

Visualizar: UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.

Especificar: UML da la posibilidad de especificar cuáles son las características de un sistema antes de su construcción.

Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.

Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

Para poder representar un sistema desde varios puntos de vistas, UML ofrece varios diagramas como el Diagrama de Casos de Usos del Sistema (DCUS), diagrama de clases, diagrama de objetos, diagrama de actividades, y otros.

UML se seleccionó como lenguaje de modelado, pues brinda soporte a la metodología escogida (RUP), es independiente del lenguaje de programación y proporciona rigor en la especificación.

1.5.2. Herramienta de modelado Visual Paradigm.

Visual Paradigm es una herramienta CASE (Ingeniería de Software Asistida por Computación). Esta herramienta se utiliza para modelar con UML y está presente en todo el desarrollo del software en cuestión, empezando por la planificación que se lleva a cabo, pasando por el modelamiento de negocio, el análisis y diseño del software, la elaboración del código y toda la documentación necesaria.

Características que presenta Visual Paradigm 8.0: (13)

- Es un software libre y disponibilidad en múltiples plataformas (Windows, Linux).
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanecen sincronizado en todo el ciclo de desarrollo
- Licencia: gratuita y comercial.
- Las imágenes y reportes generados, no son de muy buena calidad.
- Generación de código para Java y exportación como HTML.
- Ingeniería inversa, Java, C++, Esquemas XML, XML, NET exe/dll, CORBA IDL.

En el desarrollo de este trabajo se utiliza esta herramienta por las características antes expuestas y además por ser fácil de usar, permite modelar los diagramas de entidad relación así como convertirlos a tablas de la bases de datos a utilizar.

1.5.3. Lenguaje de programación Java.

Java es un lenguaje de programación creado por la compañía informática SunMicrosystem en el año 1995. Este fue la primera plataforma creada por esta compañía y se usa en más de 850 millones de computadores a través de herramientas, juegos y aplicaciones de negocios, entre otros. También se ejecuta en otros dispositivos como son los móviles y los televisores, es un lenguaje orientado a objetos e independiente de la plataforma.

Algunas características notables: (14)

- Robusto.
- Gestiona la memoria automáticamente.
- No permite el uso de técnicas de programación inadecuadas.
- Multihilo.
- Cliente-Servidor.
- Mecanismos de seguridad incorporados.
- Herramientas de documentación incorporadas.

1.5.4. Entorno de Desarrollo Integrado (IDE).

SpringSource Tool Suite ™ (STS) 2.8.1ofrece la mejor potencia de Eclipse entorno de desarrollo para la construcción de Primavera-potencia las aplicaciones empresariales. STS proporciona herramientas para todos los de la última versión de Java de la empresa. SpringSource Tool Suite (STS) es un IDE basado en la versión Java EE de Eclipse, pero altamente customizado para trabajar con Spring Framework. Entre las características más destacadas que STS proporciona se encuentran:

- Asistentes para la creación de proyectos Spring.
- Herramientas para la gestión de beans.
- Editores gráficos de archivos de configuración de Spring.
- Herramientas de desarrollo para Spring Web Flow y Spring Batch. (15)

1.6. Marco de Trabajo o Framework Spring.

Spring 3.0.6 es un framework contenedor liviano basado en la técnica Inversión de Control (IoC)[4] e implementación de desarrollo según el Paradigma de Orientación a Aspectos (POA)[5] (16)

Algunas características del framework Spring: (16)

- La motivación inicial es facilitar el desarrollo de aplicaciones J2EE, promoviendo buenas prácticas de diseño y programación, tratando de utilizar patrones de diseño como Factory, Abstract Factory, Builder, Decorator, ServiceLocator, entre otros bastante conocidos en la ingeniería del software.

[4] Inversión de Control (IoC): Consiste en ceder el control a una entidad externa a la aplicación, llamada "Contenedor", que se encargará de gestionar las instancias así como sus creaciones y destrucciones.

[5] Paradigma de Orientación a Aspecto (POA): Paradigma de programación cuya intención es permitir una adecuada modularización de las aplicaciones y posibilitar una mejor separación de conceptos, gracias al POA se pueden encapsular los diferentes conceptos que componen una aplicación en entidades bien definidas, eliminando las dependencias entre cada uno de los módulos.

- Es de código abierto.
- Está enfocado al manejo de objetos de negocio, dentro de una arquitectura en capas.
- Una ventaja de Spring es su modularidad, pudiendo usar algunos de los módulos sin comprometerse con el uso del resto:
  - ❖ Framework con programación orientada a aspectos, que trabaja con soluciones que son utilizadas en numerosos lugares de una aplicación, lo que se conoce como asuntos transversales.
  - ❖ Acceso a datos que facilita el trabajo de usar un API con JDBC, (Hibernate).
  - ❖ Acceso Remoto. Facilita la existencia de objetos en el servidor que son exportados para ser usados como servicios remotos.
  - ❖ Spring Web MVC. Maneja la asignación de peticiones a controladores y desde estos a las vistas. Implica el manejo y validación de formularios.

1.7. Conclusiones.

En el presente capítulo se realizó un estudio de las principales herramientas en el mundo para realizar auditorías a los SGBD MySQL y PostgreSQL, llegando a la conclusión de que existen diversas herramientas que realizan auditorías a SGBD con numerosas ventajas y que efectúan un análisis exhaustivo de las amenazas que existen en estos sistemas pero con el inconveniente de que no suplen completamente las necesidades de ETECSA. Para el desarrollo, implementación y diseño de los plugins se escogieron herramientas que por sus ventajas y características son idóneas para ser utilizadas. Para guiar el proceso de desarrollo de software se utilizó RUP. Se escoge BPMN para la notación gráfica que describe la lógica de los procesos del negocio, Visual Paradigm como herramienta CASE para modelar con UML. Java como lenguaje de programación y como IDE escogido SpringSource Tool Suite ™ el cual soporta Spring, marco de trabajo seleccionado.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción.

En el presente capítulo se describen las características de los plugins que realizan la evaluación de los resultados obtenidos de las consultas que serán ejecutadas en los SGBD MySQL y PostgreSQL como parte de la auditoría informática, permitiendo determinar cuán vulnerable es en un momento determinado la información contenida en los SGBD, y a partir de ahí tomar las medidas pertinentes para evitar la mala manipulación y pérdida de los datos que podría conllevar a un mal funcionamiento del sistema de comunicaciones en Cuba.

Específicamente en este capítulo se identifican los procesos de negocio y se describen los mismos con el objetivo de garantizar una mejor comprensión de cómo se realizan las auditorías en el DGSI de ETECSA. También se detallan los requisitos funcionales y no funcionales de los plugins y se modelan los diagramas de los casos de usos con sus respectivas descripciones, al igual que las especificaciones de los actores que intervienen.

2.1.1 Problema y situación problemática.

El DGSI de ETECSA es responsable de velar por la integridad de toda la información manipulada por los SGBD para ello realiza auditorías de seguridad informática a los SGBD, que actualmente se efectúan de forma semi-automatizada. El proceso de revisión se realiza según guías que contienen todos los aspectos a medir en dependencia del SGBD a auditar, los aspectos a tratar en una auditoría determinada se vuelcan en un script que será ejecutado en el servidor a revisar. Los resultados de la ejecución del script se traspasan manualmente hacia una matriz de resultado, y acto seguido se conforma un informe que es elaborado a partir de las deficiencias detectadas en la matriz al comparar los resultados obtenidos con las buenas prácticas internacionales. Este proceso manual puede traer consigo errores humanos a la hora de copiar los datos y ocasionar demoras, trayendo como consecuencia que la seguridad de la empresa en algún momento se pueda ver amenazada. Por lo planteado surge la necesidad de desarrollar una aplicación que automatice los procesos que se realizan en el DGSI de ETECSA para efectuar las auditorías informáticas.

Como parte de esta aplicación se desarrollarán dos plugins capaces de realizar la evaluación de los resultados obtenidos en las auditorías de seguridad informática a los SGBD MySQL y PostgreSQL para dar un dictamen de cómo se encuentra la seguridad de un servidor en un momento determinado,

dándole una visión a los auditores de la parte que necesita refuerzo y así se puede evitar un posible ataque.

2.1.2 Propuesta del sistema.

La aplicación al integrar los plugins brindará varias funcionalidades al Supervisor, las cuales permitirán cargar el fichero de resultado de extensión XML, esto implícitamente comprueba que el SGBD se corresponda con el plugin indicado y que la versión se corresponda con el gestor en análisis. Permite realizar el análisis de los diferentes ficheros XML con el objetivo de evaluar los resultados por indicador y pasar posteriormente a elaboración de la matriz de resultados, de donde se obtendrán los datos para realizar el informe final, también permitirá que el usuario pueda modificar esta matriz de resultados en caso de que exista una evaluación que depende de factores que no están contemplado en el proceso. Para el desarrollo de los plugins que evalúan los resultados adquiridos en las auditorías a los SGBD MySQL y PostgreSQL se decidió implementar funcionalidades, tales como la validación de las consultas SQL que se pueden insertar o modificar en el sistema con el objetivo de que las mismas luego sean utilizadas para auditar los SGBD. También se realiza la funcionalidad de confeccionar la matriz de resultados, dicha matriz es la que permite mostrar las vulnerabilidades del gestor analizado, mostrando el resultado y el nivel de riesgo en el que se encuentra el SGBD en ese momento. Esta matriz también contiene las buenas prácticas y las observaciones asociadas a cada parámetro analizado en la auditoría, al igual que le permitirá agregar algunas recomendaciones sobre la evaluación. Los plugins están involucrados en múltiples procesos como la lectura de los resultados que llegan en un archivo XML, y la validación de las consultas SQL a través de la librería General SQL Parser (GSP). Estos procesos utilizan los resultados obtenidos de las consultas y las sentencias SQL correspondientes de cada script, como puede suponerse es de suma importancia controlar dicha información y la forma en que se maneja por el papel que juegan en la realización de la auditoría como tal.

Para la evaluación de los resultados de las consultas ejecutadas se realizó un estudio profundo de los scripts definidos por ETECSA para auditar los SGBD MySQL y PostgreSQL y de las guías que contienen las buenas prácticas internacionales precisadas para cada gestor, esto permitió el desarrollo de formas evaluativas a través de las cuales se evalúan los resultados obtenidos una vez realizadas las consultas, dentro de estas se utilizan los términos; *valores sugeridos*, estos no son más que los resultados correctos que deberían devolverse cuando se ejecutan las consultas, los *valores encontrados* son los resultados reales que se obtienen al ejecutar el script.

Buscar en lista: Busca si alguno de los valores encontrados se encuentra en la lista de los valores sugeridos, si coincide se le asigna una evaluación de coincidencia y si no coincide se le asigna una de no coincidencia, dicha evaluación será definida por el auditor.

Buscar en variantes: Busca el valor encontrado dentro de alguna de las listas definidas y asigna una evaluación en correspondencia con la lista donde se encuentre el valor.

Las listas de las variantes son:

- Lista de valores para una evaluación de Bien.
- Lista de valores para una evaluación de Mal.
- Lista de valores para una evaluación de Regular.

Buscar en intervalos: Busca el valor encontrado dentro de alguno de los intervalos definidos y se le otorga una evaluación en correspondencia con el intervalo donde se encuentre el valor.

Intervalos:

- Intervalos de valore para una evaluación de Bien.
- Intervalos de valore para una evaluación de Mal.
- Intervalos de valore para una evaluación de Regular.

2.2 Modelo de Negocio.

El análisis del flujo de trabajo se realiza para llegar a un mejor entendimiento de la organización donde se va a implantar el producto. Las principales motivaciones son las siguientes: asegurarse de que el producto será útil, no un obstáculo; conseguir que encaje de la mejor forma posible en la organización y tener un marco común para los desarrolladores, los clientes y los usuarios finales. (10)

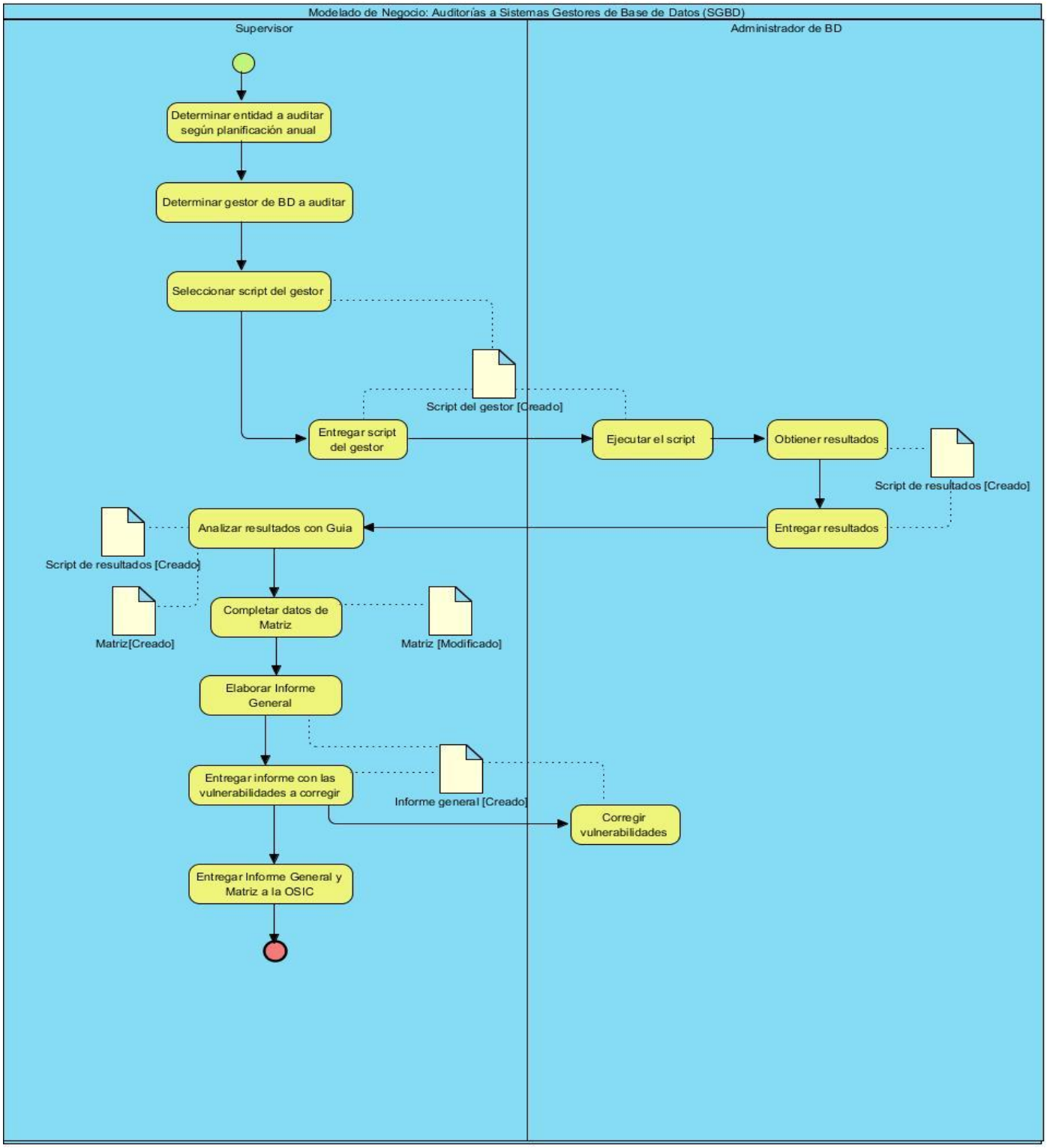


Figura No. 1. Modelo de Negocio.

2.2.1 Descripción de las actividades del proceso de negocio Auditorías a SGBD.

Las actividades identificadas en el DGSI de ETECSA a la hora de realizar las auditorías a los SGBD son:

2.2.1.1 Determinar gestor de BD a auditar.

Entradas: Auditorías planificadas con anterioridad.

Salidas: Sistema Gestor de BD seleccionado.

Responsable del proceso: Supervisor de ETECSA.

Descripción:

Es la actividad que da inicio al proceso de negocio y necesita que las auditorías estén planificadas con anterioridad, así como que los SGBD estén instalados. Este proceso comienza cuando se acuerda realizar alguna auditoría a uno o varios SGBD, para esto debe seleccionar cuál es el gestor o los gestores que se van a auditar y las versiones correspondientes al gestor.

2.2.1.2 Seleccionar script del gestor.

Entradas: Sistema Gestor de BD seleccionado.

Salidas: Script con los parámetros a evaluar para el SGBD.

Responsable del Proceso: Supervisor de ETECSA.

Descripción:

Luego de seleccionar el SGBD que se quiere auditar y la versión correspondiente se procede a la selección del script adecuado.

2.2.1.3 Entregar script del gestor.

Entradas: Script con los parámetros a evaluar en el SGBD.

Salidas: Script con los parámetros a evaluar en el SGBD.

Responsable: Supervisor de ETECSA.

Descripción:

Esta actividad la inicia el Supervisor y consiste en la entrega del script con los parámetros a evaluar en el SGBD al Administrador de BD.

2.2.1.4 Ejecutar el script.

Entradas: Script con los parámetros a evaluar en el SGBD.

Salidas: No aplica.

Responsable: Administrador de BD.

Descripción:

El Administrador de BD ejecuta el script que le entregó el Supervisor de ETECSA.

2.2.1.5 Obtener resultados.

Entradas: No aplica.

Salidas: Fichero de resultados.

Responsable: Administrador de BD.

Descripción:

Una vez ejecutado el script por el Administrador de BD, este adquiere los resultados de la supervisión realizada en un fichero.

2.2.1.6 Entregar resultados.

Entradas: Fichero de resultados.

Salidas: No aplica.

Responsable: Administrador de BD.

Descripción:

Una vez obtenido el fichero de resultados el Administrador de BD entrega al Supervisor dicho fichero para su posterior análisis.

2.2.1.7 Analizar resultados con Guía.

Entradas: Fichero de resultados y la Guía.

Salidas: No aplica.

Responsable: Supervisor de ETECSA.

Descripción:

Actividad que se realiza cuando el Supervisor compara el fichero de los resultados con la Guía, donde se encuentran los medidores para evaluar los resultados obtenidos.

2.2.1.8 Completar datos de Matriz.

Entradas: Matriz.

Salidas: Matriz.

Responsable: Supervisor de ETECSA.

Descripción:

Actividad en donde se analizan los resultados almacenados en el fichero de resultado y se completa la matriz, dicha matriz es la que almacena el resultado final de la auditoría al SGBD en cuestión.

2.2.1.9 Elaborar Informe general.

Entradas: Matriz.

Salidas: Informe general.

Responsable: Supervisor de ETECSA.

Descripción:

Actividad en el cual queda elaborado el informe general usando los datos almacenados en la matriz, este reporte final sintetiza los resultados de la Auditoría.

2.2.1.10 Entregar Informe general

Entradas: Informe general y Matriz.

Salidas: No aplica.

Responsable: Supervisor de ETECSA.

Descripción:

Se entrega el Informe general y la Matriz a la OSIC (Oficina de Seguridad Informática de Cuba) para su revisión, registro y distribución a las personas indicadas.

2.3 Especificación de los requisitos de software.

Posiblemente la etapa más compleja a lo largo de la producción de un software sea la de captura de requerimientos. En esta etapa el cliente determina que características y funcionalidades desea que contenga el software a desarrollar. Para lograr esto se hace necesario el establecimiento de una visión común entre el cliente y el equipo de desarrollo sobre los procesos que se llevan a cabo en el negocio y cómo estos se verán reflejados y mejorados en el futuro software. Esta tarea es generalmente complicada ya que la mayoría de los clientes no dominan las tecnologías informáticas. Esta es una etapa de vital importancia puesto que de ella depende que los resultados sean los esperados y que el cliente quede satisfecho con el producto.

2.3.1 Requisitos funcionales.

Los requisitos funcionales reflejan las capacidades con las que debe contar el sistema. Estos se mantienen invariables sin importar con qué propiedades o cualidades se relacionen. A continuación serán tratados los requerimientos funcionales asociados a los plugins desarrollados:

RF1. Leer configuración inicial del plugin.

Permite obtener la información inicial que tenga el plugin a la hora de ser instalado en el SASGBD, tales como consultas, forma de evaluación de cada consulta, versiones de los gestores a los que se asocia cada consulta y los indicadores que agrupan las consultas.

RF2. Evaluar parámetros.

Permite analizar los resultados de las consultas pertenecientes a un indicador específico usando la forma de evaluación asociada, al tiempo que se conforma la matriz de resultados que contienen las evaluaciones obtenidas, las buenas prácticas y las recomendaciones asociadas a cada parámetro analizado en la auditoría.

Indicadores del SGBD MySQL.

RF2.1 Realizar análisis del indicador Configuración del Sistema Operativo.

RF2.2 Realizar análisis del indicador Permisos sobre el File System.

RF2.3 Realizar análisis del indicador Logs.

RF2.4 Realizar análisis del indicador General.

RF2.5 Realizar análisis del indicador Permisos en MySQL.

RF2.6 Realizar análisis del indicador Configuración de arranque de MySQL.

Indicadores del SGBD PostgreSQL.

RF2.7 Realizar análisis del indicador Accesos de usuarios y permisos sobre objetos de la base de datos.

RF2.8 Realizar análisis del indicador Configuración de cuentas de usuarios.

RF2.9 Realizar análisis del indicador Configuración de parámetros.

RF2.10 Realizar análisis del indicador Configuración del servidor de bases de datos.

RF3. Validar consulta.

Permite a través de la librería GSP realizar el análisis sintáctico de las sentencias SQL en el momento que se vayan a insertar o modificar en la BD de la aplicación.

2.3.2 Requisitos no funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Representan características que hacen al producto atractivo, rápido, confiable, entre otras. En muchos casos los requerimientos no funcionales son fundamentales en el éxito del producto. Normalmente están vinculados a requerimientos funcionales, es decir una vez que se conozca lo que el sistema debe hacer, se puede determinar cómo ha de comportarse y las cualidades que debe tener.

2.3.2.1 Soporte.

Se requiere que los plugins cuenten con la documentación apropiada que permitan agilizar la gestión y administración de los procesos que automatizarán los plugins en las auditorías informáticas. Se brindarán cursos de capacitación al personal que utilizará el software en caso de ser necesario, además se le dará soporte y mantenimiento a los plugins por 6 meses en el sitio de despliegue.

2.3.2.2 Portabilidad, reusabilidad y escalabilidad.

La aplicación a la cual se integrarán los plugins debe ser multiplataforma para que pueda instalarse y trabajar sobre diferentes sistemas operativos como Windows y GNU/Linux. Habiendo analizado cómo cambia la economía del país y como las empresas u organizaciones cambian sus procesos, los plugins deberán ser capaces de adaptarse a estos cambios y dar la posibilidad de agregar nuevas funcionalidades.

2.3.2.3 Seguridad:

Como parte de la ética profesional que le corresponde a cada desarrollador de software se mantendrá en la implementación de los plugins el estricto cumplimiento de los siguientes principios de seguridad informática:

- **Confiabilidad:** A la información que es manejada por los plugins solo deben acceder las personas que tengan los privilegios concedidos para realizar tal acción.
- **Integridad:** La información que es manejada por el sistema puede ser modificada solo por el personal con los permisos correspondientes y será transformada solo si es para bien y mejora del proceso, manteniendo así una correcta protección contra la corrupción y estados de inconsistencia.
- **Disponibilidad:** La información deberá estar disponible en todo momento para las personas que la requieran en pos de realizar acciones que estimen convenientes.

2.3.2.4 Legales:

Los plugins se desarrollarán utilizando herramientas de software libre, la propiedad intelectual del producto será de la Universidad de las Ciencias Informáticas (UCI), desarrollado para ETECSA, específicamente para el DGSI.

2.3.2.5 Software:

Los plugins deberán ser capaces de integrarse al SASGBD por lo que esta aplicación debe estar ejecutándose en el momento de la integración. En los ordenadores donde se vaya a ejecutar la

aplicación se necesita tener instalado los requisitos mínimos necesarios para que funcione correctamente.

- Máquina virtual de Java (JVM).
- Servidor de Base de Datos PostgreSQL.

2.4 Modelo de Casos de Usos del Sistema.

2.4.1 Definición de los actores del sistema a automatizar.

Actores del Sistema	Justificación.
SASGBD.	Es el encargado de activar o instalar el plugin correspondiente al gestor (MySQL o PostgreSQL), de cargar el fichero de resultados obtenido de la ejecución de un script de consultas y da inicio al proceso de evaluación de estos resultados. A través de este sistema se insertan o modifican las consultas, dando este inicio a la funcionalidad validar consulta.

2.4.2 Diagrama de Casos de Usos del Sistema.

Para modelar los aspectos estáticos de un sistema se utilizan los diagramas de caso de uso. Estos muestran un conjunto de casos de usos, actores y sus relaciones. También son importantes para visualizar y especificar el comportamiento de un elemento.

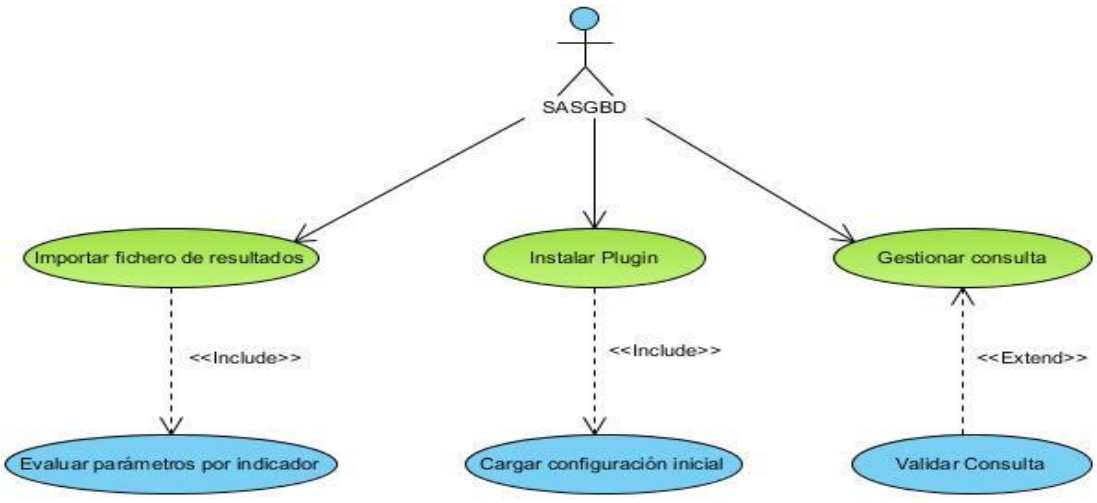


Figura No. 2. Diagrama de Casos de Usos del Sistema.

2.4.3 Descripción de casos de uso del Plugin para MySQL.

2.4.3.1 Descripción del CU Cargar configuración inicial.

Objetivo	Cargar configuración inicial.
Actores	Sistema para la realización de auditorías a Sistemas Gestores de Bases de Datos (SASGBD).
Resumen	El caso de uso inicia cuando en el SASGBD se va a instalar el plugin MySQL, específicamente cuando se va a importar la configuración del plugin MySQL, se inserta en la base de datos toda la información contenida dentro del fichero de configuración XML (fichero de configuración XML para MySQL), el caso de uso termina cuando se inserta satisfactoriamente toda la información a la base de datos.
Complejidad	Alta
Prioridad	Alta
Precondiciones	Debe haberse activado el plugin correspondiente al gestor MySQL.
Poscondiciones	Se debe haber insertado satisfactoriamente a la base de datos toda la información contenida dentro del fichero de configuración XML.
Requisitos no funcionales	RF 1
Flujo de eventos	
Flujo básico: Leer Fichero de Configuración XML.	
Actor	**Sistema**
1. Llama la funcionalidad leer configuración inicial.	2. Obtiene el elemento raíz, dada la dirección donde se encuentra el fichero de configuración XML.
	3. Obtiene la lista de los elementos “indicador” del fichero de configuración. Los indicadores son: • Configuración del Sistema Operativo. • Permisos sobre el File System. • Logs.

	• General. • Permisos en MySQL. • Configuración de arranque de MySQL.
	4. Obtiene cada elemento “indicador” de la lista de indicadores.
	5. Obtiene de cada elemento “indicador” los siguientes atributos: • “nombre” • “descripción”. • “estado” (si está activo o no)
	6. Asigna los diferentes atributos a una instancia de “tbdIndicador” del SASGBD.
	7. Obtiene del elemento “indicador” la lista de los elementos “consultas” pertenecientes al “indicador”.
	8. Obtiene cada elemento “consulta” de la lista de consultas.
	9. Obtiene de cada elemento “consulta” los siguientes atributos: • “nombre” • “nivel de riesgo”. • “estado”(si está activo o no) • “descripción” • “impacto” • “sql”(sentencia SQL) • “mejores prácticas” • “peso” • “versiones soportadas

	• "id"
	10. Asigna a una instancia de la clase "tbdConsulta" del SASGBD, los diferentes atributos: • "nombre" • "nivel de riesgo". • "estado"(si está activo o no) • "descripción" • "impacto" • "sql"(sentencia SQL) • "mejores prácticas" • "peso" • "versiones soportadas • "id"
	11. Obtiene del elemento "consulta "el atributo: • "forma de evaluar".
	12. Obtiene del elemento "consulta" las diferentes evaluaciones de la "forma evaluativa" "variante" que pueden ser: • "bien" • "regular" • "mal" • "no evaluado"
	13. Obtiene de cada evaluación los diferentes "valores asociados" del elemento "consulta".
	14. Asigna a una instancia de la clase "tbdVariante" del SASGBD, los diferentes atributos:

	• "nivel de evaluación". • "valores asociados" a cada evaluación.
	15. Inserta en una "lista de variantes" una instancia de la clase "tbdVariante".
	16. Inserta la "lista de variantes" al atributo correspondiente de una instancia de la clase "tbDfeVariante".
	17. Devuelve la "forma evaluativa tbDfeVariante" que contiene una "lista de variantes" y cada elemento de la "lista" tiene por cada "tipo de evaluación" los "valores predeterminados".
	18. Asigna "tbDfeVariante" al atributo correspondiente de una instancia de "tbdConsulta" del SASGBD.
	19. Inserta una instancia de la clase "tbdConsulta" en una "lista de consultas".
	20. Inserta en una "lista de consultas" una instancia de la clase "tbdIndicador" SASGBD.
	21. Inserta el "tbdIndicador" en una "lista de indicadores".
	22. Devuelve la "lista de indicadores" SASGBD y termina el caso de uso.
Flujos alternos	
Nº 11a. Forma Evaluativa Lista.	

Autor	Sistema
	11a.1. Obtiene del elemento consulta el nivel de evaluación si hay coincidencia de la forma evaluativa Lista que pueden ser: • "bien" • "mal" • "regular" • "no evaluado"
	11a.2. Obtiene del elemento "consulta" el "nivel de evaluación si no hay coincidencia" de la forma evaluativa "lista" que pueden ser: • "bien" • "mal" • "regular" • "no evaluado"
	11a.3. Obtiene del elemento "consulta" la "lista de valores predeterminados" de la forma evaluativa" lista".
	11a.4. Asigna a una instancia de la clase "tbdLista" del SASGBD, los diferentes atributos: • "nivel de coincidencia" • "nivel de no coincidencia" • "valores predeterminados"
	11a.5. Devuelve la forma evaluativa "tbDfeLista" que contiene el "nivel de evaluación si hay coincidencia", el "nivel de evaluación si no hay coincidencia" y los "valores predeterminados".

	11a.6. Asigna el atributo “tbDfeLista” a una instancia de la clase “tbdConsulta” del SASGBD y regresa a la acción 19 del flujo normal de eventos.
Flujos alternos	
Nº 11b. Forma Evaluativa Intervalos.	
Autor	**Sistema**
	11b.1. Obtiene del elemento “consulta” el “nivel de evaluación” de la forma evaluativa Intervalo que puede ser: • “bien” • “mal” • “regular” • “no evaluado”
	11b.2. Obtiene del “nivel de evaluación” una lista de elementos tipo “intervalo”.
	11b.2. Obtiene cada elemento “intervalo” de la lista de elementos de tipo intervalo.
	11b.3. Obtiene los diferentes atributos del elemento intervalo: • “límite superior” • “límite Inferior” • “inferior cerrado” (si es inferior cerrado) • “superior cerrado” (si es superior cerrado)
	11b.4. Asigna los diferentes atributos a una instancia de la clase “tbdIntervalos” del SASGBD: • “límite superior” • “límite inferior” • “inferior cerrado” (si es inferior cerrado)

		• “superior cerrado” (si es superior cerrado)
		11b.5. Inserta en una “lista de intervalos” los diferentes objetos “tbdIntervalos”.
		11b.6. Asigna la “lista de intervalos” al atributo correspondiente de una instancia de la clase “tbDfeIntervalo” del SASGBD.
		11b.7. Asigna el atributo “tbDfeVariante” al atributo correspondiente de una instancia de la clase “tbdConsulta” del SASGBD y regresa a la acción 19 del flujo normal de eventos.
	CU Incluido	Incluido del caso del caso de uso “Instalar Plugin” del SASGBD.
	CU Extendidos	No Procede

2.4.4 Descripción de casos de uso del Plugin para PostgreSQL.

2.4.4.1 Descripción del CU Validar Consulta.

Objetivo	Validar Consulta.
Actores	SASGBD.
Resumen	El caso de uso inicia cuando en el SASGBD el supervisor inserta o modifica una consulta SQL, el SASGBD envía la consulta al plugins para que este la evalúe si la consulta SQL que se va a insertar o modificar esta correctamente estructurada, además de comprobar que la consulta en su sintaxis no contenga ninguna operación de Insert, Create, Drop, Delete o Update y el caso de uso termina cuando el sistema le da una respuesta positiva o negativa al actor.
Complejidad	Media
Prioridad	Alta
Precondiciones	Se debe haber instalado el Plugin correspondiente al SGBD PostgreSQL.

Postcondiciones	Se debe haber validado si la estructura de la consulta SQL es correcta o no y si posee o no operaciones indebidas.
Requisitos funcionales	RF 3.
Flujo de eventos	
Flujo básico Validar consulta.	
Actor	**Sistema**
1. El SASGBD realiza la funcionalidad de Insertar o Modificar una consulta.	2. Obtiene la "consulta" SQL para analizarla.
	3. Analiza sintácticamente la "consulta" SQL obteniéndose una respuesta de la clase que implementa la librería GSP.
	4. Verifica que la respuesta de la evaluación sintética es positiva (Valor 1).
	5. Analiza si la "consulta" contiene en su sintaxis alguna operación de Insert, Create, Drop, Delete o Update, operación que no se deben realizar en las BD.
	6. Verifica que la respuesta de la operación indebida es positiva (Valor 1).
	7. Envía un mensaje indicando: "La consulta SQL está estructurada correctamente y no contiene operaciones incorrectas" y termina el caso de uso.
Flujos alternos	
Nº Evento 4a Respuesta de evaluación sintáctica con valor 0.	
Actor	**Sistema**
	4a.1. Envía un mensaje indicando: La "consulta" esta sintácticamente mal estructurada en la línea correspondiente. Regresa a la acción 9 del caso de uso *"Gestionar*

		consulta", sección *"Insertar consulta".* (Ver caso de uso *"Gestionar consulta")* Regresa a la acción 12 del caso de uso *"Gestionar consulta",* sección *"Modificar consulta"*. (Ver caso de uso *"Gestionar consulta*").
Nº Evento 6a.Respuesta de evaluación de las operaciones indebidas con valor 0.		
		6a.1. Envía un mensaje indicando: "La consulta no puede contener operaciones de Update Insert, Create, Drop, Delete o Update". Regresa a la acción 9 del caso de uso *"Gestionar consulta",* sección *"Insertar consulta".* (Ver caso de uso *"Gestionar consulta")* Regresa a la acción 12 del caso de uso *"Gestionar consulta",* sección *"Modificar consulta"*. (Ver caso de uso *"Gestionar consulta*").
	CU Incluido	No procede.
	CU Extendidos	CU Validar consulta extiende del CU Gestionar consulta sección "Insertar consulta" y sección "Modificar consulta."
Requisitos no funcionales	No procede.	
Asuntos pendientes	No procede.	

2.5 Conclusiones.

En el presente capítulo se expuso el análisis de la situación problemática y de la información que manejan los plugins, y se presentó una propuesta del sistema a desarrollar, el modelo de negocio que se lleva a cabo en ETECSA y se realizó una descripción detallada de cada uno de los procesos que se ejecutan en él, sirviendo de guía para identificar los requisitos funcionales del sistema, detallando cada uno de estos al igual que los requisitos no funcionales y se realizó el DCUS junto a una descripción detallada de los CUS críticos.

CAPÍTULO 3: DISEÑO DEL SISTEMA

3.1. Introducción.

En el presente capítulo luego de los resultados obtenidos en el capítulo anterior, se profundizará en los casos de usos detallándolos para lograr una vista de la estructura interna del sistema, donde se reflejan las clases necesarias para dar cumplimiento a las funcionalidades del sistema descritos en los casos de uso. También se define el estilo arquitectónico, patrones de diseños, los diagramas de clases del diseño y los diagramas de interacción, específicamente los diagramas de secuencia.

3.2. Arquitectura de software.

3.2.1. Estilo arquitectónico.

Los plugins se basan en una arquitectura n capas, los mismos tendrán una capa **Clase Principal** que es la encargada de heredar e implementar las funciones de la interfaz de comportamiento definida en la capa Comunicador definida en la arquitectura del SASGBD, con la finalidad de que los plugins puedan comunicarse y ser cargado por el sistema al cual se integrarán. Además, esta capa es quien sabe cómo acceder a las funciones de los propio plugins.

La **Capa Negocio** tendrá toda la lógica de negocio de los plugins. Habrá una carpeta *Interfaces* donde estarán las clases interfaces de los plugins, también presentará una segunda carpeta *Implementación* que es donde se encontrarán las clases que implementarán las mencionadas interfaces.

La **Capa Recursos** es donde estarán las librerías utilizadas por los plugins y los ficheros XML con toda la información inicial de los plugins, como son el nombre del plugin, las versiones, los indicadores y las consultas asociadas a cada indicador; así como el fichero con los resultados a evaluar.

Lo que se conoce como arquitectura en capas es en realidad un estilo de programación donde el objetivo principal es separar los diferentes aspectos del desarrollo, tales como las cuestiones de presentación, lógica de negocio y mecanismos de almacenamiento.

La necesidad de contar con fracciones en una aplicación es que se puedan “intercambiar” o “realizar cambios” sin tener que modificar el resto de la aplicación, este principio es lo que impulsa el desarrollo en capas.

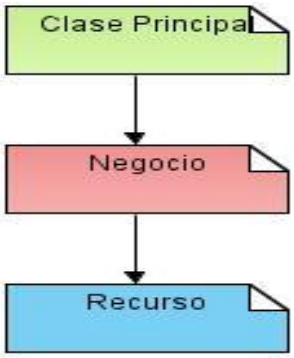


Figura No. 3. Arquitectura de los plugins

Ventajas del modelo.

- Desarrollos paralelos (en cada capa).
- Aplicaciones más robustas debido al encapsulamiento.
- Mantenimiento y soporte más sencillo (es más sencillo cambiar un componente que modificar una aplicación monolítica).
- Mayor flexibilidad (se pueden añadir nuevos módulos para dotar al sistema de nueva funcionalidad).
- Alta escalabilidad. La principal ventaja de una aplicación distribuida bien diseñada es su buen escalado, es decir, que puede manejar muchas peticiones con el mismo rendimiento simplemente añadiendo más hardware. El crecimiento es casi lineal y no es necesario añadir más código para conseguir esta escalabilidad.

Los plugins se integrarán al SASGBD responsable de las auditorías a los SGBD de ETECSA. Posee la capacidad de importar-exportar plugins, gestionar las consultas e indicadores, agregar nuevas versiones de los plugins, entre otras funcionalidades.

SASGBD cuenta con una arquitectura en capas:

La **Capa GUI** es la que contendrá las interfaces gráficas mediante las cuales los usuarios harán uso de la aplicación. La misma tendrá una carpeta *UI* que almacenará las interfaces gráficas, una segunda carpeta *Interfaces* que contendrá las clases interfaces que definirán el comportamiento de las interfaces de usuario. Una tercera carpeta llamada *Implementación* donde estarán las implementaciones de las clases interfaces.

La **Capa Entorno**, es una capa de abstracción entre la capa GUI y la capa de Negocio, es la encargada de establecer la comunicación entre ambas capas, además de ejecutar los entornos correspondientes a

los módulos en caso de que se quiera utilizar alguno de estos. Es quien sabe que funcionalidad del negocio corresponde con determinada interfaz gráfica, así como orientar al negocio sobre qué acción realizó el usuario. En esta capa se pone en práctica el patrón Observer, ya que estará al tanto de cuál fue la acción que seleccionó el usuario. Esta capa tendrá una carpeta *Interfaces* que es donde estarán las clases interfaces de los entornos, y tendrá una segunda carpeta *Implementación* donde estarán las clases que implementarán las interfaces de los entornos.

La **Capa de Negocio** contendrá toda la lógica de negocio, es quien implementa todas las clases y funciones de la aplicación. La misma tiene una carpeta *Interfaces* que es donde estarán las clases interfaces que definirán el comportamiento de las clases del negocio. Habrá además una carpeta *Implementación* donde estarán las clases que implementarán las interfaces del negocio. Esta capa se comunica con la capa AD.

La **Capa AD** manejará todo lo relacionado con el acceso a los datos, es una capa que abstrae al negocio del modo en que la aplicación se conectará a la base de datos, esto permitirá que la base de datos pueda cambiar en un futuro y no haya que hacer muchos cambios en el negocio. Esta capa también contendrá una carpeta *Interfaces* con sus clases de interfaces correspondientes y una carpeta *Implementación* que tendrá las clases que implementarán las interfaces.

La **Capa Comunicador**, es una capa vertical que tendrá la interface de comportamiento que deberá ser implementada por cada uno de los plugins para que los mismos se puedan comunicar con la aplicación. Es quien posee el conocimiento para cargar y como comunicarse con los plugins.

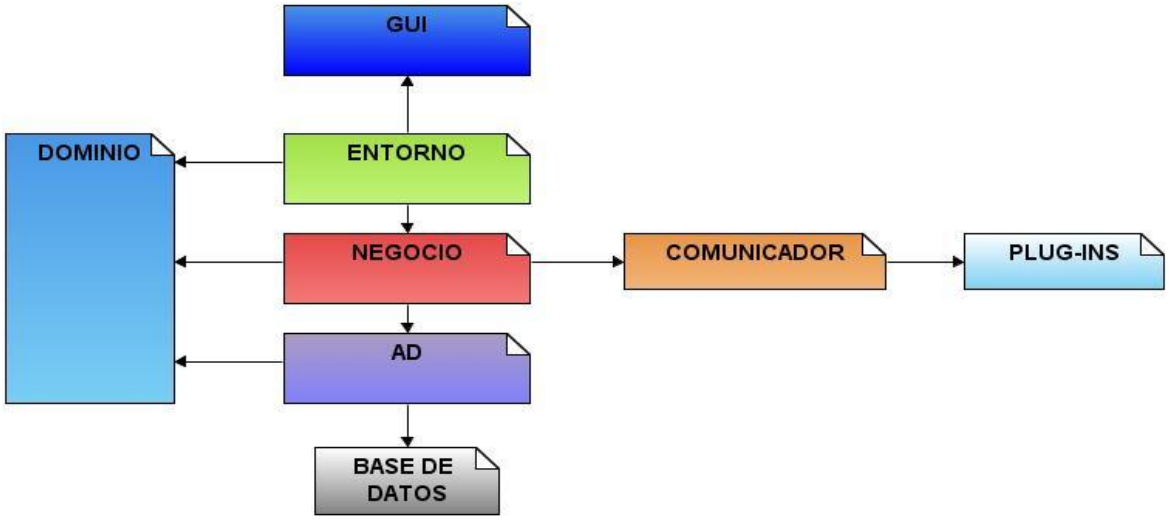


Figura No. 4. Arquitectura del Módulo de BD.

3.3. Patrones de Diseño.

El diseño de software es tanto un proceso como un modelo. El proceso de diseño es una secuencia de pasos que hacen posible que el diseñador describa todos los aspectos del software que se va construir. Sin embargo, es importante destacar que el proceso de diseño simplemente no es un recetario. Un conocimiento creativo, experiencia en el tema, un sentido de lo que hace que un software sea bueno y un compromiso general con la calidad son factores críticos de éxito para un diseño competente. El modelo de diseño es el equivalente a los planes de un arquitecto para una casa. Comienza representando la totalidad de todo lo que se va a construir. (17)

Para definir bien estos planes se han creados los patrones de diseños, concepto introducido en el año 1997, cuando se publica el libro: "A Pattern Language: Towns/Building/Construction", de Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King y Shlomo Angel, Oxford University Press. Uno de los autores de este libro comento sobre el concepto de patrón de diseño: “Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo siquiera dos veces de la misma forma”. (18)

3.3.1. Patrones Gang of Four (GOF).

Singleton.

El patrón Singleton asegura que exista una única instancia de una clase. A primera vista, uno puede pensar que pueden utilizarse clases con miembros estáticos para el mismo fin. Sin embargo, los resultados no son los mismos, ya que en este caso la responsabilidad de tener una única instancia recae en el cliente de la clase. El patrón Singleton hace que la clase sea responsable de su única instancia, quitando así este problema a los clientes. Adicionalmente, si todos los métodos de esta clase son estáticos, éstos no pueden ser extendidos, desaprovechando así las capacidades polimórficas que nos proveen los entornos orientados a objetos.

El funcionamiento de este patrón es muy sencillo y podría reducirse a los siguientes conceptos:

- Ocultar el constructor de la clase Singleton, para que los clientes no puedan crear instancias.
- Declarar en la clase Singleton una variable miembro privada que contenga la referencia a la instancia única que queremos gestionar.

- Proveer en la clase Singleton una función o propiedad que brinde acceso a la única instancia gestionada por el Singleton. Los clientes acceden a la instancia a través de esta función o propiedad.

Estas reglas se cumplen en todas las implementaciones del Singleton, independientemente de los recaudos que deban tomarse para soportar la correcta ejecución en entornos multihilo. (19)
Este patrón es usado cuando debe haber exactamente una instancia de una clase y ésta deba ser accesible a los clientes desde un punto de acceso conocido. La única instancia debería ser extensible mediante herencia y los clientes deberían ser capaces de utilizar una instancia extendida sin modificar su código.

3.3.2. Patrones de Asignación de Responsabilidades (GRASP).

Los patrones GRASP son parejas de problema-solución con un nombre, que codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. GRASP es un acrónimo que significa General Responsibility Asignment Software Patterns (Patrones Generales de Software para Asignar Responsabilidades). El nombre se eligió para indicar la importancia de captar (grasping) estos principios, si se quiere diseñar eficazmente el software orientado a objetos. (20)

Experto.

Problema:

- ¿Cuál es el principio fundamental en virtud del cual se asignan las responsabilidades en el diseño orientado a objetos?
- Un modelo de clase puede definir docenas y hasta cientos de clases de software, y una aplicación tal vez requiera el cumplimiento de cientos o miles de responsabilidades.
- Si estas se asignan en forma adecuada, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, y se nos presenta la oportunidad de reutilizar los componentes en futuras aplicaciones.

Solución:

- Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Beneficios:

- Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento.
- El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clase “sencillas” y más cohesivas, siendo más fáciles de comprender y de mantener. Así se brinda soporte a una alta cohesión.

Controlador.

Problema:

- ¿Quién debería encargarse de atender un evento del sistema?
- Un evento del sistema es un evento de alto nivel generado por un actor externo; es un evento de entrada externa. Se asocia a operaciones del sistema: las que emite en respuesta a los eventos del sistema.
    - Por ejemplo, cuando un cajero que usa un sistema de terminal en el punto de venta oprime el botón "Terminar Venta”, está generando un evento sistémico que indica que “la venta ha terminado”.
- Un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación.

Solución:

- Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones:
    - el “sistema” global (controlador de fachada).
    - la empresa u organización global (controlador de fachada).
    - algo en el mundo real que es activo (por ejemplo, el papel de una persona) y que pueda participar en la tarea (controlador de tareas).

- un manejador artificial de todos los eventos del sistema de un caso de uso, generalmente denominados “Manejador<NombreCasodeUso>” (controlador de casos de uso).

Beneficios:

- Mayor potencial de los componentes reutilizables. Garantiza que la empresa o los procesos de dominio sean manejados por la capa de los objetos del dominio y no por la de la interfaz.
- Reflexionar sobre el estado del caso de uso. A veces es necesario asegurarse de que las operaciones del sistema sigan una secuencia legal o poder razonar sobre el estado actual de la actividad y las operaciones en el caso de uso subyacente.
- Un corolario importante del patrón Controlador es que los objetos de la interfaz (por ejemplo, objetos de ventanas, applets) y la capa de presentación no deberían encargarse de manejar los eventos del sistema.

Alta cohesión.

Problema:

- ¿Cómo mantener la complejidad dentro de límites menejables?
- La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase.
- Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.
- Una baja cohesión hace muchas cosas no afines o realiza trabajo excesivo. Esto presenta los siguientes problemas:
    - Son difíciles de comprender.
    - Difíciles de reutilizar.
    - Difíciles de conservar.
    - Las afectan constantemente los cambios.

Solución:

- Asignar una responsabilidad de modo que la cohesión siga siendo alta.
- En la práctica, el nivel de cohesión no puede ser considerado independiente de los otros patrones y principios (e.j. Patrones “Experto” y “Bajo Acoplamiento”).

Beneficios:

- Mejoran la claridad y facilidad con que se entiende el diseño.
- Se simplifica el mantenimiento y las mejoras de funcionalidad.
- A menudo se genera un bajo acoplamiento.
- Soporta mayor capacidad de reutilización.

Bajo acoplamiento.

Problema:

- ¿Cómo dar soporte a una dependencia escasa y a un aumento de la reutilización?
- El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas.
    - Acoplamiento bajo significa que una clase no depende de muchas clases.
    - Acoplamiento alto significa que una clase recurre a muchas otras clases. Esto presenta los siguientes problemas:
        - Los cambios de las clases afines ocasionan cambios locales.
        - Difíciles de entender cuando están aisladas.
        - Difíciles de reutilizar puesto que dependen de otras clases.

Solución:

- Asignar una responsabilidad para mantener bajo acoplamiento.
- El grado de acoplamiento no puede considerarse aisladamente de otros principios como Experto y Alta Cohesión. Sin embargo, es un factor a considerar cuando se intente mejorar el diseño.

- En los lenguajes OO como C++ y JAVA las formas comunes de acoplamiento de TipoX a TipoY son las siguientes:
    - TipoX posee un atributo (miembro de datos o variable de instancia) que se refiere a una instancia TipoY o al propio TipoY.
    - TipoX tiene un método que a toda costa referencia una instancia de TipoY o incluso el propio TipoY. Suele incluirse un parámetro o una variable local de TipoY o bien el objeto devuelto de un mensaje es una instancia de TipoY.
    - TipoX es una subclase directa o indirecta del TipoY.
    - TipoY es una interfaz y TipoX la implementa.

Beneficios:

- No se afectan por cambios de otros componentes.
- Fáciles de entender por separado.
- Fáciles de reutilizar.

Algunos escenarios:

- Muy baja cohesión: Una clase es la única responsable de muchas cosas en áreas funcionales heterogéneas.
- Baja cohesión: Una clase tiene la responsabilidad exclusiva de una tarea compleja dentro de un área funcional.
- Alta cohesión: Una clase tiene responsabilidades moderadas en un área funcional y colabora con las otras para llevar a cabo las tareas.
- Cohesión moderada: Una clase tiene peso ligero y responsabilidades exclusivas en unas cuántas áreas que están relacionadas lógicamente con el concepto de clase pero no entre ellas.

Creador.

Problema:

- ¿Quién debería ser responsable de crear una nueva instancia de alguna clase?

- La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos.
- En consecuencia, conviene contar con un principio general para asignar las responsabilidades concernientes a ella.
- El diseño, bien asignado, puede soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reutilizabilidad.

- El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos.
- El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento.
- Al escogerlo como creador, se da soporte al bajo acoplamiento.

Solución:

- Asignarle a la clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos:
  - B agrega los objetos A.
  - B contiene los objetos A.
  - B registra las instancias de los objetos A.
  - B utiliza especialmente los objetos A.
  - B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado (así que B es un Experto respecto a la creación de A). B es un creador de los objetos A.
- Si existe más de una opción, prefiera la clase B que agregue o contenga la clase A.

3.3.3. Patrón Inversión de Control.

El marco de trabajo Spring brindo una potente gestión de configuración basada en JavaBeans, aplicando los principios de IoC haciendo que la configuración de aplicaciones sea rápida y sencilla. Estas definiciones de beans se realizan en lo que se llama el contexto de aplicación.

La Inversión de Control es un patrón de diseño pensado para permitir un menor acoplamiento entre componentes de una aplicación y fomentar así el reúso de los mismos. (21)

Problema:

Muchas veces, un componente tiene dependencias de servicios o componentes cuyos tipos concretos son especificados en tiempo de diseño.

Ejemplo: Clase A depende de ServicioA y de ServicioB.

Los problemas que esto plantea son:

- Al reemplazar o actualizar las dependencias, se necesita cambiar el código fuente de la clase A.
- Las implementaciones concretas de las dependencias tienen que estar disponibles en tiempo de compilación.
- Las clases son difíciles de testear aisladamente porque tienen directas definiciones a sus dependencias.
- Las clases tienen código repetido para crear, localizar y gestionar sus dependencias.

Solución:

Delegar la función de seleccionar una implementación concreta de las dependencias a un componente externo.

El control de cómo un objeto A obtiene la referencia de un objeto B es invertido. El objeto A no es responsable de obtener una referencia al objeto B sino que es el Componente Externo el responsable de esto, esta es la base del patrón IoC.

Usos:

El patrón IoC se puede utilizar cuando:

- Se desee desacoplar las clases de sus dependencias de manera de que las mismas puedan ser reemplazadas o actualizadas con muy pocos o casi ningún cambio en el código fuete de sus clases.
- Desea escribir clases que dependan de clases cuyas implementaciones no son conocidas en tiempo de compilación.
- Desea testar las clases aisladamente sin sus dependencias.

- Desea desacoplar sus clases de ser responsables de localizar y gestionar el tiempo de vida de sus dependencias.

Técnicas de implementación.

Según diversos enfoques, este patrón puede implementarse de diversas maneras. Entre las más conocidas tenemos:

Inyección de dependencias.

Una dependencia entre un componente y otro, puede establecerse estáticamente o en tiempo de compilación, o bien, dinámicamente o en tiempo de ejecución. Es en éste último escenario donde cabe el concepto de inyección, y para que esto fuese posible, debemos referenciar interfaces y no implementaciones directas. En general, las dependencias son expresadas en términos de interfaces en lugar de clases concretas. Esto permite un rápido reemplazo de las implementaciones dependientes sin modificar el código fuente de la clase.

La esencia de la inyección de las dependencias es contar con un componente capaz de obtener instancias validas de las dependencias del objeto y pasárselas durante la creación o inicialización del objeto.

Inyección basada en Constructor

Las dependencias se inyectan utilizando un constructor con parámetros del objeto dependiente. Éste constructor recibe las dependencias como parámetros y las establece en los atributos del objeto.

Inyección basada en métodos setters

En este tipo de inyección, se utiliza un método setters para inyectar una dependencia en el objeto que la requiere. Se invoca así al setter de cada dependencia y se le pasa como parámetro una referencia a la misma.

3.4. Diagrama de Paquetes del plugin que evalúa los resultados de las auditorías a SGBD MySQL.

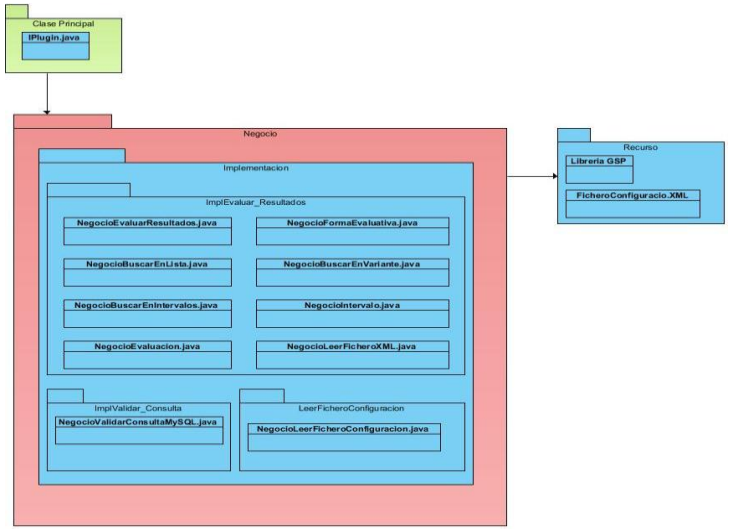


Figura No. 5. Estructura del plugin para evaluar los resultados de las auditorías a SGBD MySQL.

3.5. Diagrama de Paquetes del plugin que evalúa los resultados de las auditorías a SGBD PostgreSQL.

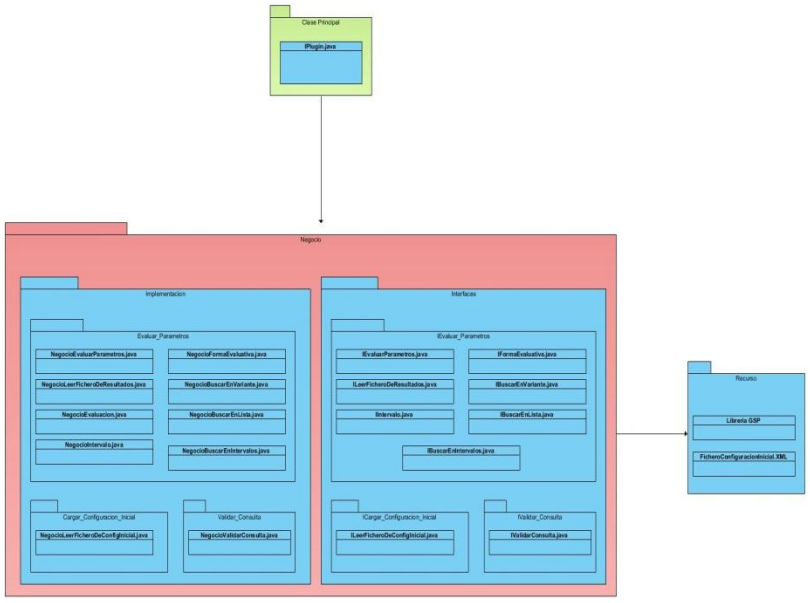


Figura No. 6. Estructura del plugin para evaluar los resultados de las auditorías a SGBD PostgreSQL.

3.6. Diagramas de clases del diseño del plugin que evalúa los resultados de las auditorías a SGBD MySQL.

3.6.1. Diagrama de Clases del diseño. CU Validar consulta.

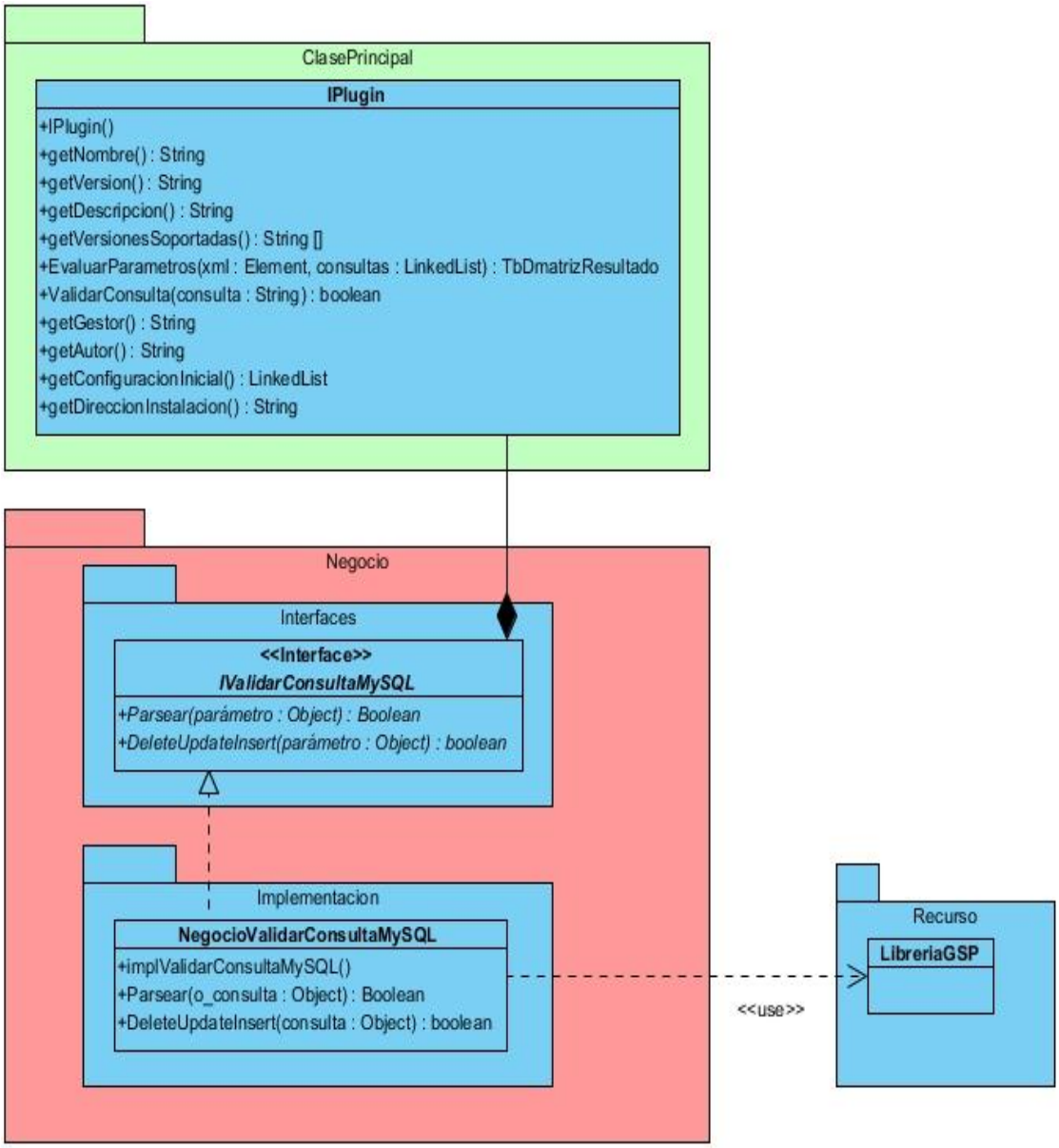


Figura No. 7. Diagrama de Clases del diseño. CU Validar consulta.

3.7. Diagramas de clases del diseño del plugin que evalúa los resultados de las auditorías a SGBD PostgreSQL.

3.7.1. Diagrama de Clases del diseño. CU Validar consulta.

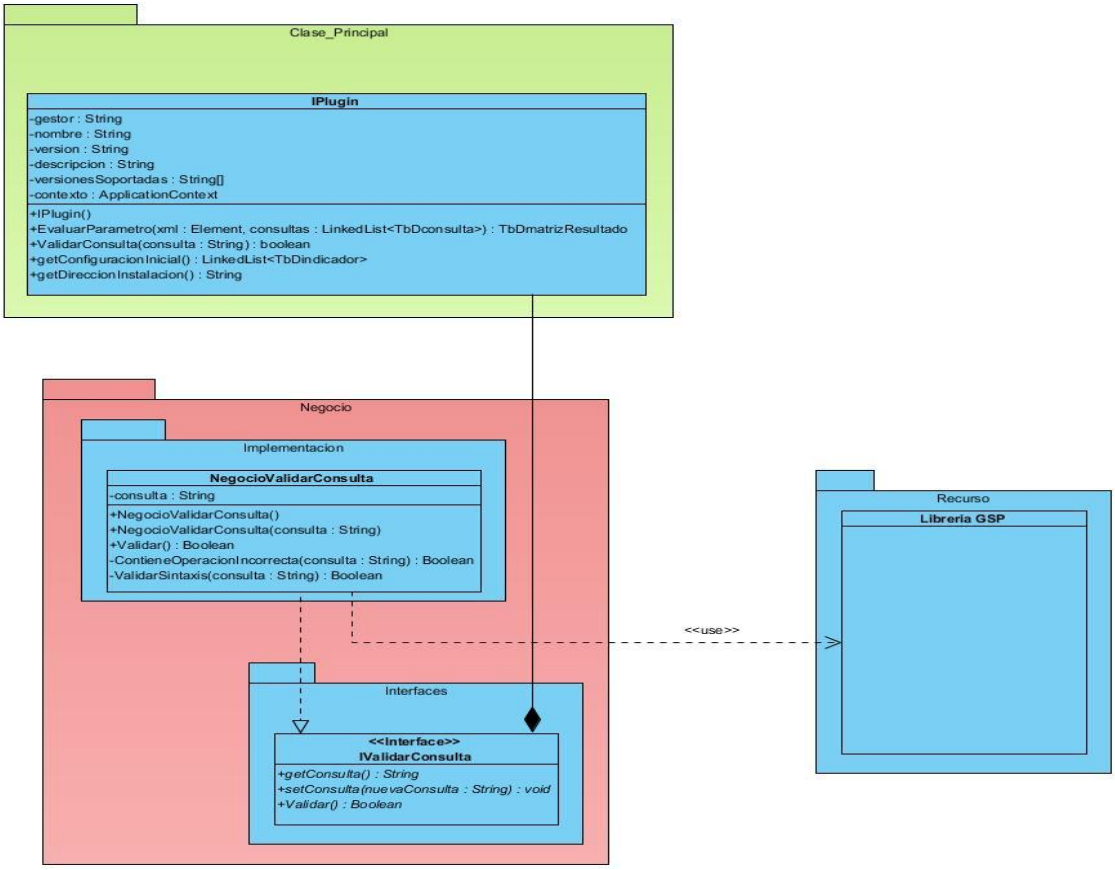


Figura No. 9. Diagrama de Clases del diseño. CU Validar consulta.

3.8. Conclusiones.

En el presente capítulo se obtuvieron los artefactos fundamentales correspondientes al diseño del sistema, como el modelo de diseño, el cual constituye la entrada principal para la futura implementación del sistema. Dentro de este modelo se construyeron los diagramas de clases del diseño así como los diagramas de secuencia correspondientes a cada uno de estos diagramas para representar la interacción entre las clases. También se realizó una explicación detallada de la arquitectura utilizada en la construcción del sistema para su mejor comprensión y mejor construcción del plugin.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

4.1 Introducción.

En el presente capítulo se documenta todo lo referente a la fase de implementación en la cual se da inicio al desarrollo del sistema en términos de componentes, es decir, ficheros de código fuente y ejecutables a partir de los artefactos generados durante la fase de diseño fundamentada en el capítulo anterior. Además se realizan las pruebas de los plugins con el objetivo de comprobar si las funcionalidades que se ejecutan devuelven los resultados esperados y verificar la existencia de errores.

4.2 Diagramas de Componentes.

Luego de la Fase de Elaboración y de establecer la arquitectura del sistema, la Fase de Construcción tiene como propósito completar el desarrollo del sistema. En esta fase se comienza el desarrollo de los diagramas de componentes para mostrar las dependencias lógicas entre componentes del software.

Un componente es una parte física y reemplazable de un sistema, debe tener un nombre simple, ejemplo: *cliente.java* o *nombre de camino*, cuando está incluido en un paquete.

Un componente puede contener adornos, valores etiquetados e información adicional, ejemplo: *referencia a las interfaces que realiza*. Un componente posee características similares a una clase, tiene nombre, realiza interfaces, puede participar de relaciones, puede tener instancias y puede participar en interacciones. (22)

Se muestran los diferentes diagramas de componentes de los plugins que evaluarán los resultados obtenidos en las auditorías a los SGBD MySQL y PostgreSQL.

4.2.1 Diagrama de Componentes del Plugin para evaluar los resultados de las auditorías de los SGBD MySQL.

Diagrama de Componentes. CU Cargar configuración inicial MySQL.

DIAGRAMA DE COMPONENTES
<CU Cargar configuración inicial MySQL.>

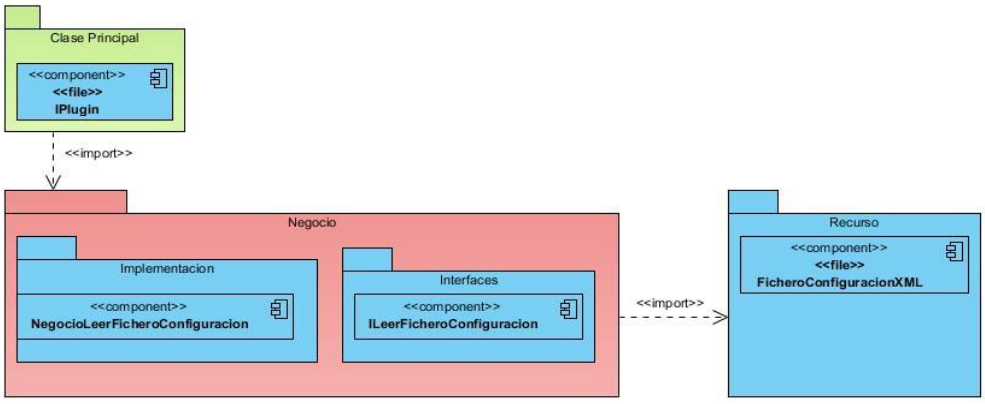


Figura No. 17. Diagrama de Componente del CU Cargar configuración inicial MySQL.

4.2.2 Diagrama de Componentes del Plugin para evaluar los resultados de las auditorías de los SGBD PostgreSQL.

Diagrama de Componentes. CU Validar Consulta PostgreSQL.

DIAGRAMA DE COMPONENTES
<CU Validar Consulta PostgreSQL >

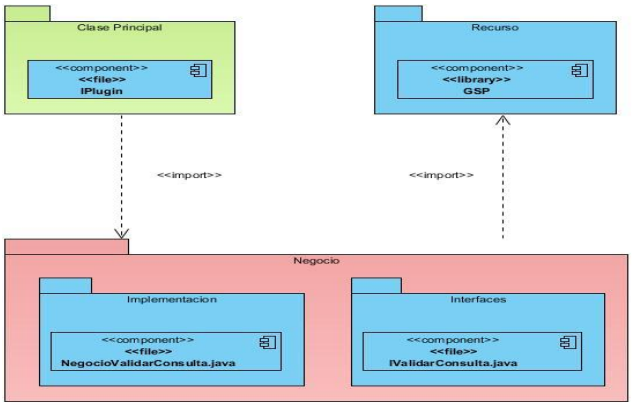


Figura No. 19. Diagrama de Componente del CU Validar consulta PostgreSQL.

4.3 Pruebas.

Las pruebas son las encargadas de evaluar la calidad del producto que se está desarrollando, no para aceptar o rechazar el producto al final del proceso de desarrollo, sino que debe ir integrado en todo el ciclo de vida. “El papel del testeo no es asegurar la calidad, pero sí evaluarla y proporcionar una realimentación a tiempo, de forma que las cuestiones de calidad puedan resolverse de manera efectiva en tiempo y coste.” (10)

Luego de la implementación del sistema y para la validación del mismo las pruebas toman protagonismo, en busca de errores en la implementación y para perfeccionar el funcionamiento de la aplicación, definiéndose inicialmente una estrategia para la ejecución de estas. Uno de los objetivos de las pruebas es validar que el software trabaje como fue diseñado, que cumpla con los requisitos pactados y que estos fueron implementados correctamente. Las Pruebas verifican que el producto satisface los requerimientos y que se comporta tal y como se desea.

Estrategias de pruebas:

- **Niveles:** es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo.
- **Tipos de pruebas:** cada tipo de prueba tiene un objetivo específico y una técnica que lo soporte.
- **Método de Prueba:**
    - Caja Blanca (código)
    - Caja Negra (interfaz)

Para los Plugins se definió la siguiente estrategia de prueba:

Niveles
•Integración
Tipo de pruebas
•Descripción: funcionalidad. •Tipo de prueba: función
Métodos de pruebas
•Pruebas de caja blanca •Pruebas de caja negra

Como se especifica entre los método de pruebas a utilizar se encuentran las pruebas de caja negra. Los Plugins no presentan interfaz de usuario, por lo que se utilizaron las interfaces del SASGBD para probar las funcionalidades, verificando que sean operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto.

Igualmente se realizarán pruebas de caja blanca, estas permiten que se comprueben los caminos lógicos del software. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado. Se comprobarán los métodos fundamentales como Evaluar resultados por indicador, devolver Matriz de resultados que hace uso entre otros métodos del Evaluar resultados por indicador, también se le realizarán pruebas al código del método Validar consulta.

4.3.1 Prueba de Caja Negra al caso de uso Validar Consulta.

4.3.1.1 Descripción General

El caso de uso inicia cuando en el SASGBD, el supervisor desea insertar o modificar una consulta SQL, luego el Plugin MySQL es capaz de evaluar si la consulta SQL que se va a insertar o modificar esta correctamente estructurada, además de comprobar que la consulta en su sintaxis no contenga ninguna operación de Insert, Update, Delete, Drop o Create y el caso de uso termina cuando el sistema envía un mensaje indicando si la consulta está apta para estar contenida dentro de la base de datos (que puede ser insertada o que pueda modificar alguna consulta específica) al SASGBD.

4.3.1.2 Condiciones de ejecución

El supervisor desea insertar o modificar una consulta MySQL a través del SASGBD.

4.3.1.3 Secciones y Escenarios del Caso de Uso Validar consulta.

Secciones	Escenarios
1. Validar Consulta	1.1 Validar satisfactoriamente la consulta sintácticamente.
	1.2 Consulta a insertar con operaciones prohibidas.
	1.3 Campos vacíos.

Descripción de las variables.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Consulta MySQL	Campo de texto	No	Es una consulta MySQL que cuando el supervisor intente insertar o modificar se debe validar sintácticamente y a la vez que no contenga operaciones prohibidas.

Escenarios.

Escenario	Descripción	Consulta MySQL	Respuesta del sistema	Flujo central
EC 1.1 Validar satisfactoriamente la consulta sintácticamente.	Se valida utilizando la librería GSP una consulta de tipo SQL cuando se intenta modificar o insertar una consulta a la base de datos.	V: select host, user from mysql.user;	Envía una respuesta booleana al SASGBD con valor true lo que indica que la consulta está sintácticamente correcta.	1. El supervisor inserta los datos correspondientes para insertar una consulta a través del SASGBD. 2. Presiona el botón “aceptar”, enviando “la consulta” al Plugin para evaluar resultados de auditorías a sistemas gestores de bases de datos MySQL. 3. El sistema comprueba que el campo “consulta” no este vacío. 4. El sistema comprueba que la consulta no contenga operaciones prohibidas (insert, update, drop, create, delete).
	Se valida utilizando la librería una consulta de tipo SQL cuando se intenta modificar o insertar una consulta a la base de datos.	I: select host; user from mysql.user;	Envía una excepción al SASGBD indicando que la consulta no está sintácticamente correcta.	
EC 1.2 Consulta a insertar con operaciones	Cuando se intenta insertar una consulta con operaciones indebidas a la	V: Insert into mysql.user (col1,col2)	Envía un mensaje, indicando que la consulta contiene	

prohibidas.	base de datos.	values (col2*2,15);	operaciones prohibidas.	5. El sistema valida sintácticamente la consulta a través de la librería GSP. 6. Se envía posteriormente una respuesta al SASGBD indicando si la consulta esta apta para ser insertada en la base de datos.
		I: Insert into mysql.user (col1,col2) values (col2*2,15);	Envía un mensaje, indicando que la consulta contiene operaciones prohibidas.	
		V: Update table mysql.user user= “Armando” WHERE a=root;	Envía un mensaje, indicando que la consulta contiene operaciones prohibidas.	
		I: Update table mysql.user user= “Armando” WHERE a=root;	Envía un mensaje, indicando que la consulta contiene operaciones prohibidas.	
		V: Delete FROM mysql.user WHERE user= “Armando”;	Envía un mensaje, indicando que la consulta contiene operaciones prohibidas.	
		I:	Envía un	

		Delete From mysql.user WHERE user= “Armando”;	mensaje, indicando que la consulta contiene operaciones prohibidas.	
1.3 Campos vacíos.	Se comprueba que la consulta de tipo SQL no sea enviada al Plug-in cuando se intenta modificar o insertar una consulta a la base de datos.	I:	Envía un mensaje, indicando que la consulta no contiene ningún valor.	

Registro de defectos y dificultades detectados

Elemento	No	Ubicación de la No Conformidad	Etapa de Detección	Significativa	Estado NC	Respuesta del Equipo de Desarrollo
En el EC 1.1 Validar satisfactoriamente la consulta de forma sintáctica, el sistema no efectúa la validación correctamente.	1	Validar Consulta.	Pruebas	Si	-2/5/2012 pendiente -5/5/2012 resuelta	-
En el EC 1.1 Validar satisfactoriamente la consulta de forma sintáctica, el sistema no efectúa la validación correctamente.	2	Validar Consulta.	Pruebas	Si	-2/5/2012 pendiente -5/5/2011 resuelta	-
EC 1.2 Consulta a insertar con operaciones prohibidas el sistema no efectúa la validación correctamente.	3	Validar Consulta.	Pruebas	Si	-2/5/2012 pendiente -5/5/2011 resuelta	-
En el EC1.3 Campos vacíos el sistema no efectúa la validación correctamente.	4	Validar Consulta.	Pruebas	Si	-2/5/2012 pendiente -5/5/2011 resuelta	-
En el EC 1.1 Evaluar correctamente la consulta por la forma de evaluar "buscar en lista", el	5	Evaluar Resultados por indicador.	Pruebas	Si	-3/5/2012 pendiente -6/5/2011 resuelta	-

sistema no otorga la evaluación correcta a la consulta.						
En el EC 1.1 Evaluar correctamente la consulta por la forma de evaluar "buscar en lista", el sistema no otorga la evaluación correcta a la consulta.	6	Evaluar Resultados por indicador.	Pruebas	Si	-3/5/2012 pendiente -6/5/2011 resuelta	-
En el EC 1.2 Evaluar correctamente al indicador por la forma de evaluar “buscar en variante”, el sistema no otorga la evaluación correcta a la consulta.	7	Evaluar Resultados por indicador.	Pruebas	Si	-3/5/2012 pendiente -6/5/2011 resuelta	-
En el EC 1.2 Evaluar correctamente al indicador por la forma de evaluar “buscar en variante”, el sistema no otorga la evaluación correcta a la consulta.	8	Evaluar Resultados por indicador.	Pruebas	Si	-3/5/2012 pendiente -6/5/2011 resuelta	-
En el EC 1.2 Evaluar correctamente al indicador por la forma de evaluar “buscar en intervalo”, el sistema no otorga la evaluación correcta a la consulta.	9	Evaluar Resultados por indicador.	Pruebas	Si	-3/5/2012 pendiente -6/5/2011 resuelta	-

4.3.2 Pruebas de Caja Blanca.

Requieren del conocimiento de la estructura interna del programa y son derivadas a partir de las especificaciones internas de diseño o el código.

Mediante los métodos de prueba de la caja blanca, el ingeniero de software puede obtener casos de prueba que garanticen que:

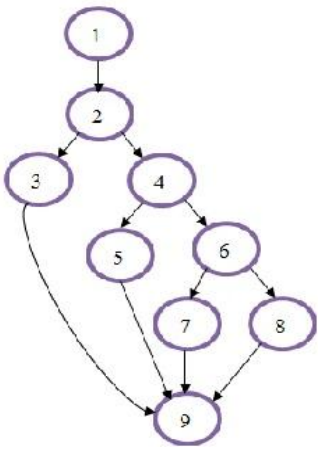
- Se ejerciten por lo menos una vez todos los caminos independientes para cada módulo.
- Se ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsa.
- Ejecuten todos los bucles en sus límites y con sus límites operacionales.
- Se ejerciten las estructuras internas de datos para asegurar su validez.

4.3.2.1 Pruebas al Caso de Uso Validar consulta del Plugin que evalúa los resultados del SGBD MySQL.

4.3.2.1.1 Método validar consulta MySQL

```
public boolean ValidarConsulta(String consulta)throws Exception {1
                NegocioValidarConsultaMySQL obj_implValidarConsultaMySQL = new NegocioValidarConsultaMySQL();
                boolean siDelectUpdateInsert = obj_implValidarConsultaMySQL.DeleteUpdateInsert(consulta);
                String consulta_1 = (String) consulta;
                if(consulta_1.isEmpty())2
                {
                throw new Exception("La consulta no contiene ningun contenido");3
                }
                else if (siDelectUpdateInsert)4
                {
                        throw new Exception("La consulta no debe contener oprecaiones Update, Insert o Delete");5
                }
                try {6
                        obj_implValidarConsultaMySQL.Parsear(consulta);6
                } catch (Exception e) {7
                        throw e;7
                }

                return true;8
}9
```



Complejidad ciclomática V (G)

- V (G) = A - N + 2.
- V (G) = 11- 9 + 2 = 4

- V (G) = r
- V (G) = 4 Regiones Cerradas.

- V (G) = c + 1
- V (G) = 3+1= 4 Condiciones

Dónde:

- a: # de arcos o aristas del grafo.
- n: # de nodos.
- r: # de regiones cerradas del grafo.
- c: # de nodos de condición.

Caminos: 1-2-3-9, 1-2-4-5-9, 1-2-4-6-7-9, 1-2-4-6-8-9.

- Cuando V (G) <10 la probabilidad de defectos del método es mínima.

4.4 Conclusiones

En el presente capítulo se describieron los elementos correspondientes a los flujos de trabajo de implementación y prueba, donde en la implementación se desarrolló el sistema en términos de componentes, se realizaron los diagramas de componentes para cada caso de uso. Además se realizaron las pruebas tanto de caja blanca como de caja negra, para así, poder detectar posibles defectos en los plugins.

CONCLUSIONES GENERALES

La elaboración del presente trabajo surge por la necesidad de reducir los errores cometidos y el tiempo empleado por el Departamento de la Gestión de la Seguridad Informática de ETECSA en la evaluación de los resultados de las auditorías a SGBD. Inicialmente se realizó una investigación sobre las tecnologías existentes que realizan las auditorías a los SGBD MySQL y PostgreSQL, verificándose que ninguna se adecúa completamente al negocio. Acto seguido se llevó a cabo una selección de las metodologías, lenguajes y herramientas que servirían de guía en el desarrollo del proyecto y se utilizarían en el diseño, construcción y validación de la solución. El resultado concreto de este trabajo son dos plugins capaces de evaluar los resultados obtenidos de la auditorías a SGBD MySQL y PostgreSQL que se integran al SASGBD, permitiendo: la validación de las consultas de los respectivos leguajes SQL usados por estos gestores, la importación del conocimiento inicial concerniente a sus auditorías y la conformación de la matriz de resultados con los datos de los parámetros medidos, lo que permite agilizar la realización de dichas auditorías y disminuir los errores inherentes a las operaciones manuales, cumpliendo satisfactoriamente con el objetivo general propuesto.

RECOMENDACIONES

Se recomienda:

- En la validación de la consultas, agregar las funcionalidades adecuadas para impedir que se enmascaren inyecciones SQL a través de las consultas que se van a insertar o modificar en la base de datos.

REFERENCIAS BIBLIOGRÁFICAS

1. **Hernández León, Rolando Alfredo y Coello González, Sayda .** *El paradigma cuantitativo de la investigación científica.* Cuidad de la Habana : Editorial Universitaria, 2002. ISBN 978-959-16-0343-2.

2. **Paré, Rafael Camps, y otros, y otros.** [En línea] [Citado el: 16 de Enero de 2012.] http://www.sw-computacion.f2s.com/Linux/007-Bases_de_datos.pdf. 71Z799014MO.

3. [En línea] [Citado el: 16 de Enero de 2012.] http://wwwdi.ujaen.es/~barranco/publico/ofimatica/tema7.pdf.

4. Sitio - Hoy Digital. [En línea] [Citado el: 17 de Enero de 2012.] http://hoy.com.do/investigacion/2009/1/5/261742/TecnologiaQue-es-auditoria-informatica.

5. **A. Loza , Patricia, R. Cargnelutti, Pablo y Sosa Agüero, Paula.** Auditoria a Bases de Datos. [En línea] [Citado el: 17 de Enero de 2012.]

6. SoftTree Technologies Inc. [En línea] [Citado el: 18 de Enero de 2012.] http://www.softtree.jkmst.com/images/DBAuditExpert.pdf.

7. **Auditor, MySQL Password.** *Auditoría & Recuperación de Contraseñas MySQL.* 2011.

8. Data Security Solutions. [En línea] [Citado el: 18 de Enero de 2012.] http://www.datasecuritys.com/dbProtect.html.

9. EcuRed_Metodologias. *EcuRed_Metodologias.* [En línea] [Citado el: 19 de Enero de 2012.] http://www.ecured.cu/index.php/Metodolog%C3%ADas_Tradicionales.

10. **Martínez, Alejandro ; Martínez, Raúl.** Guía a Rational Unified Process. [En línea] [Citado el: 19 de Enero de 2012.] http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-Guia RUP.pdf.

11. **Juan Diego Pérez.** Notaciones y lenguajes de procesos. Una visión global. [En línea] [Citado el: 16 de Enero de 2012.] www.lsi.us.es/docs/doctorado/memorias/Perez,%20Juan%20D.pdf.

12. **Scribd, Denis Y.** [En línea] [Citado el: 20 de Enero de 2012.] http://es.scribd.com/doc/57061047/Grapple..

13. EcuRed_Visual Paradigm. [En línea] [Citado el: 20 de Enero de 2012.] http://www.ecured.cu/index.php/Visual_Paradigm.

14. Java desde Cero. [En línea] [Citado el: 20 de Enero de 2012.] http://mmc2.geofisica.unam.mx/cursos/mcst-2007-II/Java/Java%20desde%20Cero.pdf.

15. **Comunidad de spring source.** Spring Framework. *Spring Framework.* [En línea] 25 de enero de 2012. [Citado el: 22 de febrero de 2012.] http://www.springsource.org/spring-framework.

16. Sitio de la Universidad de Palermo. *Sitio de la Universidad de Palermo.* [En línea] [Citado el: 20 de Enero de 2012.] http://www.palermo.edu/ingenieria/downloads/introduccion_spring_framework_v1.0.pdf.

17. **Pressman, Roger S.** *Software Engineering: A Practitioner's, Approach, Seventh Edition.* New York : McGraw-Hil Companies, 2010. 978-0-07-337 597 -7.

18. **Lago, Ramiro.** Patrones de diseño software. [En línea] Abril de 2007. [Citado el: 9 de Marzo de 2012.] http://www.proactiva-calidad.com/java/patrones/index.html.

19. Blog sobre el patrón Singleton. [En línea] [Citado el: 26 de Abril de 2012.] http://msdn.microsoft.com/es-es/library/bb972272.aspx.

20. **Visconti, Marcello y Astudillo, Hernán.** Fundamentos de Ingeniería de Software. [En línea] [Citado el: 29 de Abril de 2012.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.

21. **Guillerón, Gastón.** Inversion de Control « Gln Blog. [En línea] [Citado el: 10 de Abril de 2012.] http://glnconsultora.com/blog/?tag=inversion-de-control.

22. **Daniele, Ing. Marcela.** Teoría 11: El Arte de Modelar. [En línea] 2007 . [Citado el: 15 de Mayo de 2012.] http://fineans.usac.edu.gt:8001/rid=1HV0BP15X-15DBYBZ-FH/UML-diagramaComponentes.pdf.

BIBLIOGRAFÍA

1. **Hernández León, Rolando Alfredo y Coello González, Sayda .** *El paradigma cuantitativo de la investigación científica.* Cuidad de la Habana : Editorial Universitaria, 2002. ISBN 978-959-16-0343-2.

2. **Paré, Rafael Camps, y otros, y otros.** [En línea] [Citado el: 16 de Enero de 2012.] http://www.sw-computacion.f2s.com/Linux/007-Bases_de_datos.pdf. 71Z799014MO.

3. [En línea] [Citado el: 16 de Enero de 2012.] http://wwwdi.ujaen.es/~barranco/publico/ofimatica/tema7.pdf.

4. Sitio - Hoy Digital. [En línea] [Citado el: 17 de Enero de 2012.] http://hoy.com.do/investigacion/2009/1/5/261742/TecnologiaQue-es-auditoria-informatica.

5. **A. Loza , Patricia, R. Cargnelutti, Pablo y Sosa Agüero, Paula.** Auditoria a Bases de Datos. [En línea] [Citado el: 17 de Enero de 2012.]

6. SoftTree Technologies Inc. [En línea] [Citado el: 18 de Enero de 2012.] http://www.softtree.jkmst.com/images/DBAuditExpert.pdf.

7. **Auditor, MySQL Password.** *Auditoría & Recuperación de Contraseñas MySQL.* 2011.

8. Data Security Solutions. [En línea] [Citado el: 18 de Enero de 2012.] http://www.datasecuritys.com/dbProtect.html.

9. EcuRed_Metodologias. *EcuRed_Metodologias.* [En línea] [Citado el: 19 de Enero de 2012.] http://www.ecured.cu/index.php/Metodolog%C3%ADas_Tradicionales.

10. **Martínez, Alejandro ; Martínez, Raúl.** Guía a Rational Unified Process. [En línea] [Citado el: 19 de Enero de 2012.] http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-Guia RUP.pdf.

11. **Juan Diego Pérez.** Notaciones y lenguajes de procesos. Una visión global. [En línea] [Citado el: 16 de Enero de 2012.] www.lsi.us.es/docs/doctorado/memorias/Perez,%20Juan%20D.pdf.

12. **Scribd, Denis Y.** [En línea] [Citado el: 20 de Enero de 2012.] http://es.scribd.com/doc/57061047/Grapple..

13. EcuRed_Visual Paradigm. [En línea] [Citado el: 20 de Enero de 2012.] http://www.ecured.cu/index.php/Visual_Paradigm.

14. Java desde Cero. [En línea] [Citado el: 20 de Enero de 2012.] http://mmc2.geofisica.unam.mx/cursos/mcst-2007-II/Java/Java%20desde%20Cero.pdf.

15. **Comunidad de spring source.** Spring Framework. *Spring Framework.* [En línea] 25 de enero de 2012. [Citado el: 22 de febrero de 2012.] http://www.springsource.org/spring-framework.

16. Sitio de la Universidad de Palermo. *Sitio de la Universidad de Palermo.* [En línea] [Citado el: 20 de Enero de 2012.] http://www.palermo.edu/ingenieria/downloads/introduccion_spring_framework_v1.0.pdf.

17. **Pressman, Roger S.** *Software Engineering: A Practitioner's, Approach, Seventh Edition.* New York : McGraw-Hil Companies, 2010. 978-0-07-337 597 -7.

18. **Lago, Ramiro.** Patrones de diseño software. [En línea] Abril de 2007. [Citado el: 9 de Marzo de 2012.] http://www.proactiva-calidad.com/java/patrones/index.html.

19. Blog sobre el patrón Singleton. [En línea] [Citado el: 26 de Abril de 2012.] http://msdn.microsoft.com/es-es/library/bb972272.aspx.

20. **Visconti, Marcello y Astudillo, Hernán.** Fundamentos de Ingeniería de Software. [En línea] [Citado el: 29 de Abril de 2012.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.

21. **Guillerón, Gastón.** Inversion de Control « Gln Blog. [En línea] [Citado el: 10 de Abril de 2012.] http://glnconsultora.com/blog/?tag=inversion-de-control.

22. **Daniele, Ing. Marcela.** Teoría 11: El Arte de Modelar. [En línea] 2007 . [Citado el: 15 de Mayo de 2012.] http://fineans.usac.edu.gt:8001/rid=1HV0BP15X-15DBYBZ-FH/UML-diagramaComponentes.pdf.

23. Sitio Oficial de la Universidad Catolica de Pereira. [En línea] [Citado el: 1 de Diciembre de 2011.] http://biblioteca.ucp.edu.co:8080/jspui/bitstream/10785/145/1/completo.pdf.

24. Informática-Auditoría. *Informática-Auditoría.* [En línea] [Citado el: 1 de Diciembre de 2011.] http://auditoria3.obolog.com/auditoria-base-datos-876651.

25. [En línea] [Citado el: 1 de Diciembre de 2011.] http://www.etecsa.cu.

26. Portal de Disca(Universidad Politécnica de Valencia). [En línea] [Citado el: 16 de Enero de 2012.] www.disca.upv.es/enheror/pdf/ActaUML.PDF.

27. Sitio Oficial de Java. [En línea] [Citado el: 17 de Enero de 2012.] http://www.java.com/es/download/faq/whatis_java.xml.

28. SoftTree Technologies Inc. [En línea] [Citado el: 18 de Enero de 2012.] http://www.jkmst.com/softtree/dbaudit.htm#.

29. GratisPrograma. *GratisPrograma.* [En línea] [Citado el: 20 de Enero de 2012.] http://www.gratisprogramas.org/descargar/netbeans-ide-v7-0-1-full-jdk-v7-00-incluido-espanol-fls-ul-ups/.

30. SoftTree Technologies Inc. [En línea] [Citado el: 20 de Enero de 2012.] http://www.softtree.jkmst.com/images/DBAuditExpert.pdf.

31. **A. Loza , Patricia, R. Cargnelutti, Pablo y Sosa Agüero, Paula.** Auditoria a Bases de Datos. [En línea] [Citado el: 17 de Enero de 2012.] http://subversion.assembla.com/svn/.../Auditoria_de_Bases_de_Datos.pdf.

32. **Jacobson, Ivar, Rumbauch, James y G.B.** *El proceso unificado de desarrollo de software.*

33. Business Process Modeling Notation. *Business Process Modeling Notation.* [En línea] http://www.bizagi.com/docs/BPMNbyExampleSPA.pdf..

34. **Jerónimo Calvo Sánchez.** Lycka Bonita. [En línea] [Citado el: 16 de Febrero de 2012.] http://www.hachisvertas.net/blog/01/2008/03/27/inversion-de-control-ioc-inyeccion-de-dependencias-di.

35. Sitio español de PostgreSQL. [En línea] [Citado el: 20 de Enero de 2012.] http://www.postgresql.org.es/sobre_postgresql#intro.

36. **Lago, Ramiro.** Framework Spring. Inversión de Control. [En línea] Enero de 2008. [Citado el: 28 de Abril de 2012.] http://www.proactiva-calidad.com/java/spring/introduccionSpring.html.

37. **Pressman, Roger S.** *Ingeniería de software: Un enfoque práctico.* s.l. : Editorial McGraw-Hil, 2002.

38. visual-paradigm.com. [En línea] [Citado el: 07 de Enero de 2011.] http://www.visual-paradigm.com/product/vpuml/.

39. www.uml.org. [En línea] [Citado el: 08 de Enero de 2011.] http://www.uml.org/.

40. www.buenastareas.com. [En línea] [Citado el: 10 de Diciembre de 2010.] http://www.buenastareas.com/ensayos/Metodo-Empirico-Analitico/278643.html.

41. Sitio Oficial del Netbeans. [En línea] [Citado el: 20 de Enero de 2012.] http://netbeans.org/community/releases/61/index_es.html.

42. easyeclipse.org. [En línea] [Citado el: 08 de Enero de 2011.] http://www.easyeclipse.org/site/home/.

GLOSARIO DE TÉRMINOS

Bases de Datos: es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

CASE: Computer Aided Software Engineering en español Ingeniería de Software Asistida por Computación, son aplicaciones informáticas para aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

C++: lenguaje de programación.

Clase: descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semánticas. Una clase puede utilizar un conjunto de interfaces para especificar recopilaciones de operaciones que proporciona a su entorno.

Consulta SQL: es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en estas.

Diagrama: dibujo en el que se muestran las relaciones entre las diferentes partes de un conjunto o sistema.

Indicador: nombre con que se denomina a un grupo o conjunto de consultas que se relacionen entre sí, teniendo en cuenta el tipo de información que analizan.

MySQL: es un sistema de gestión de bases de datos relacional, multihilo y multiusuario.

Plugin: es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica.

PostgreSQL: es un sistema de gestión de base de datos relacional orientada a objetos y libre.

Sistemas Gestores de Bases de Datos:

TIC: agrupan los elementos y las técnicas usados en el tratamiento y la transmisión de la información, principalmente la Informática, Internet y las Telecomunicaciones.

XML: lenguaje de marcas extensible, es un metalenguaje extensible de etiquetas.