UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 2

SUBSISTEMA DE EVALUACIÓN DEL RIESGO DE SEGURIDAD DE LA INFORMACIÓN DEL SASGBD

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

Autores:

Sailyn María Parra López

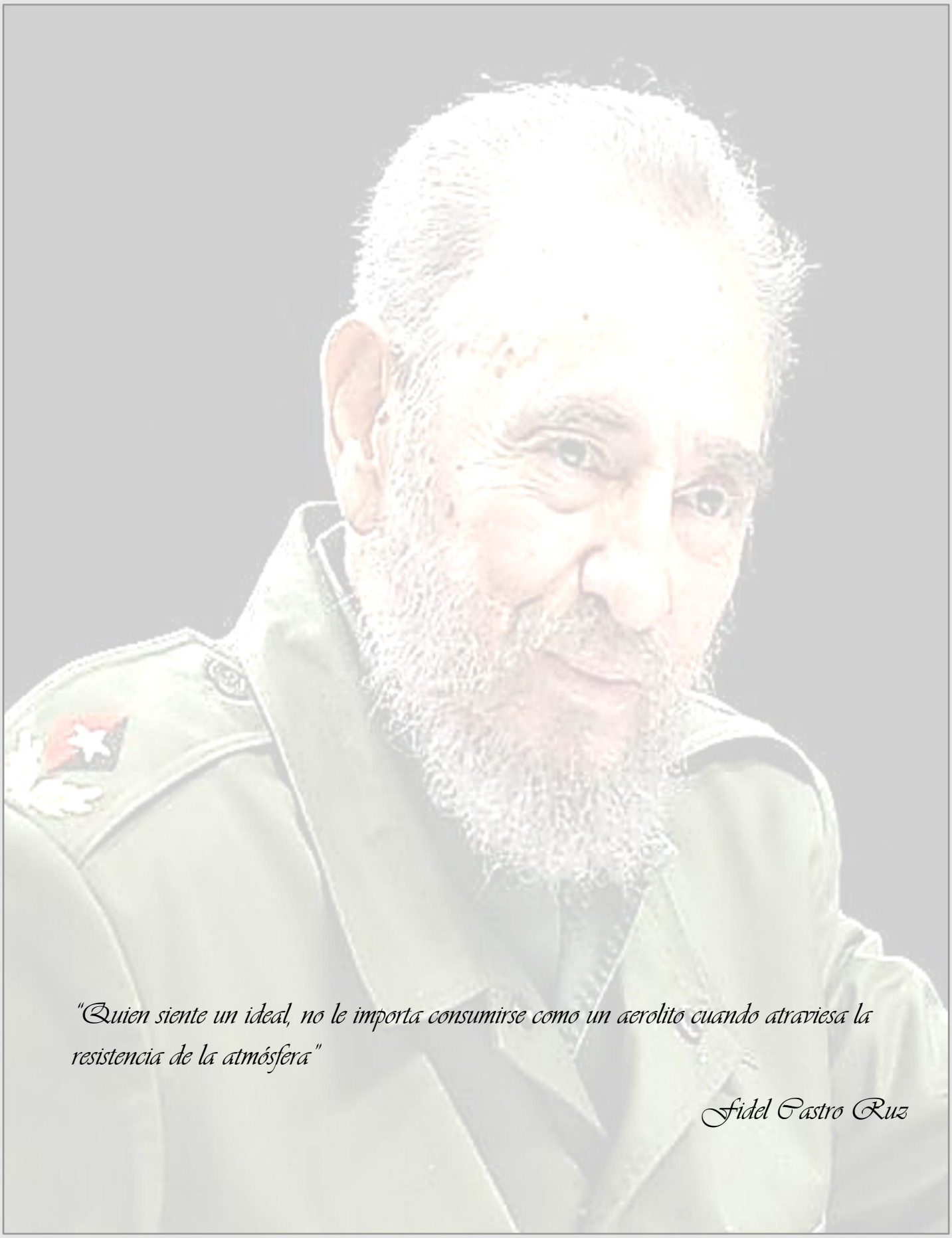
Ubel Angel Fonseca Cedeño

Tutor:

MSc. Yasser Azán Basallo

La Habana, Junio 2015

"Año 57 de la Revolución"



"Quien siente un ideal, no le importa consumirse como un aerolito cuando atraviesa la resistencia de la atmósfera"

Fidel Castro Ruz

DECLARACIÓN DE AUTORÍA

Nosotros, Sailyn María Parra Lopez y Ubel Ángel Fonseca Cedeño, declaramos ser los únicos autores de este trabajo y autorizamos a la Facultad 2 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Autor

Sailyn María Parra López

Autor

Ubel Ángel Fonseca Cedeño

Firma del Autor

Firma del Autor

Tutor

MSc. Yasser Azán Basallo

Firma del Tutor

AGRADECIMIENTOS

Agradezco en especial a mis padres por su ejemplo, dedicación, comprensión y sacrificio durante todos estos años de estudio. Gracias por confiar en mí y trasmitirme fuerza y energía para seguir adelante.

A mi novio Walter por su apoyo incondicional, comprensión, y amor todos estos años. Por hacerme muy feliz y demostrarme amor en los buenos y malos momentos de la vida.

Muchas gracias, Sailyn

Agradezco en especial a mi madre y a mi hermana por todo el sacrificio, amor y comprensión que me han brindado todos estos años de la carrera.

Muchas gracias, Ubel

DEDICATORIA

A mis padres por el amor, confianza y educación que me han dado desde que nací.

A mi hermanita por su fe en mí y verme como un modelo a seguir.

En especial a mi madre por ser mi motor impulsor, por apoyarme en todas las decisiones que he tomado en la vida.

Sailyn

En especial a mi madre por toda la confianza depositada en mí y por su amor incondicional.

Ubel

RESUMEN

El Departamento de Seguridad Informática (DSI) de la Empresa de Telecomunicaciones de Cuba S.A. (ETECSA) tiene entre sus principales responsabilidades garantizar y mantener la integridad de los datos en los Sistemas Gestores de Bases de Datos (SGBD). Por tal motivo, las auditorías de seguridad informática son parte esencial de las actividades llevadas a cabo por el departamento, siendo las evaluaciones del riesgo de seguridad de la información uno de los principales aspectos a tener en cuenta. Para realizar todo el proceso de auditoría se cuenta con el Sistema para la realización de Auditorías a los Sistemas Gestores de Bases de Datos (SASGBD), pero en estos momentos la aplicación no es capaz de ofrecer una evaluación de las auditorías tan acertada como la puede ofrecer un especialista del DSI.

Con el propósito de contribuir con el proceso de toma de decisiones a la hora de evaluar las auditorías, se desarrolla un subsistema para la evaluación del riesgo de seguridad de la información; utilizando el razonamiento basado en casos y la lógica difusa. De esta manera se aprovecha la experiencia acumulada en las auditorías anteriores de este tipo y se evalúa el nivel de riesgo al cual está sometido el sistema auditado.

Para la implementación de la propuesta de solución, se seleccionaron como principales tecnologías: los marcos de trabajo Spring e Hibernate para la programación en Java, el SGBD PostgreSQL 9.1 y Visual Paradigm 8.0 como herramienta para el modelado. El proceso de desarrollo fue guiado por la metodología RUP.

Palabras claves: auditoría de seguridad informática**,** evaluación de riesgo, sistemas gestores de bases de datos**,** razonamiento basado en casos, lógica difusa

ÍNDICE DE CONTENIDOS

ÍNDICE DE CONTENIDOS

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El surgimiento de los ordenadores trajo consigo un creciente desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC). A raíz de este auge, los volúmenes de información que se generaban fueron aumentando por lo que fue necesario almacenarlos y protegerlos, surgiendo así los Sistemas de Información (SI).

Los avances de los SI y las tecnologías han producido grandes resultados para las organizaciones, negocios y otras agencias en términos de productividad del trabajo, almacenamiento de la información, administración y oportunidad de ventajas competitivas. Sin embargo, a pesar de los disímiles beneficios brindados por los SI, estos también representan un indicador de riesgo bastante elevado para las organizaciones (Quigley 2008).

Uno de los SI que con mucha frecuencia es objetivo de ataque, son los Sistemas Gestores de Bases de Datos (SGBD) como exponen (Ramakanth y Vinod 2011): El 17 de agosto de 2009, el Departamento de Justicia de los Estados Unidos acusó a un ciudadano por el robo de 130 millones en tarjetas de crédito usando ataques de inyección de SQL. Aproximadamente 500.000 páginas web que usaban como servidor el Microsoft Internet Information Services (IIS) y el servidor de SQL, fueron atacadas entre abril y agosto del 2008 usando la inyección de SQL.

La cantidad de vulnerabilidades reportadas en este tipo de ataque han ido aumentando en los últimos años según el Instituto Nacional de Vulnerabilidades de Estados Unidos de América. En la tabla 1 se muestran datos estadísticos que reflejan el aumento de las vulnerabilidades.

Año	Numero de Vulnerabilidades	% de Total
2007	116	2.59
2008	237	3.87
2009	252	4.48
2010	289	6.96
2011	515	11.10
2012	948	16.54
2013	1.092	19.39

Tabla 1: Datos estadísticos de la vulnerabilidad: Inyección SQL (NIST 2013).

INTRODUCCIÓN

Los SGBD poseen mecanismos para garantizar la integridad, la confidencialidad y disponibilidad de la información. Sin embargo, como se pudo observar en la tabla 1, estos mecanismos, no los eximen de los efectos de la ciberguerra llevada a cabo por piratas informáticos. Motivo por el cual, se ha incrementado la búsqueda por parte de las empresas, de soluciones eficientes, que brinden la posibilidad de auditar la seguridad de sus servidores de datos, con el objetivo de conocer sus puntos débiles y tratar de erradicarlos.

Uno de los pasos para garantizar la integridad de los datos, es a través de la realización de auditorías informáticas periódicas a las tecnologías de cómputo. Para las auditorías de seguridad informática[1] se pueden utilizar las listas de chequeo[2].

El Departamento de Seguridad Informática (DSI) de la Empresa de Telecomunicaciones de Cuba S.A. (ETECSA) tiene entre sus principales responsabilidades garantizar y mantener la integridad de los datos en los SGBD. Para ello realiza las auditorías de seguridad informática, siendo las evaluaciones de Riesgo de Seguridad de la Información (RSI) uno de los principales aspectos a tener en cuenta en dichas auditorías.

El departamento mencionado tiene estandarizado con listas de chequeo todo el proceso de revisión de los sistemas SGBD, las cuales no ofrecen la posibilidad de realizar la evaluación del riesgo; pero permiten detectar las vulnerabilidades y las amenazas en las que están expuestos los sevidores de datos, premisa importante para determinar el nivel de RSI.

Los problemas en las auditorías de seguridad informática utilizando las listas de chequeo vienen dados muchas veces en el nivel de experiencia de un auditor para evaluar el nivel de riesgo de cada parámetro existente en la lista de chequeo. Por lo que el resultado de la evaluación de una auditoría de seguridad informática puede estar sustentado en buena medida de la experticia del auditor presente, por lo que existe una alta dependencia del mismo. Otro fenómeno presente en la evaluación del nivel del riesgo es que debido a las diferencias existentes en cuanto a la experticia de los expertos, los resultados pueden no tener la misma consistencia en todos los casos. Además el RSI se da en los términos: Alto, Medio o Bajo, por lo que se pueden generar ambigüedades en el resultado de la evaluación.

[1] Es el estudio que comprende el análisis y gestión de sistemas para identificar y posteriormente corregir las diversas vulnerabilidades que pudieran presentarse en una revisión exhaustiva de las estaciones de trabajo, los servidores y las redes de comunicaciones (Bahamontes 2013).

[2] Las Listas de Control, Check Lists u Hojas de Verificación, son formatos creados para realizar actividades repetitivas, controlar el cumplimiento de una lista de requisitos o recolectar datos ordenadamente y de forma sistemática.

INTRODUCCIÓN

Aunque se cuenta actualmente con el Sistema para la realización de Auditorías a Sistemas Gestores de Bases de Datos (SASGBD), el mismo no es capaz de ofrecer una evaluación de las auditorías tan acertada como la puede ofrecer un especialista del DSI; además no le brinda a los especialistas elementos o criterios para tomar decisiones en el momento de realizar el diagnóstico del RSI en los SGBD.

Este problema repercute en mayor tiempo y esfuerzo invertido por los especialistas para realizar las auditorías a los SGBD. Teniendo en cuenta los señalamientos anteriores se plantea como problema de la investigación: **¿Cómo contribuir con el proceso de toma de decisiones en las evaluaciones del riesgo de seguridad de la información del SASGBD?**

Para la realización de la investigación se define como **objeto de estudio**: El proceso de auditoría de seguridad informática.

Para dar solución al problema planteado se define como **objetivo general:** Desarrollar un subsistema de evaluación del riesgo de seguridad de la información, que contribuya con el proceso de toma de decisiones del SASGBD**.**

Enfocándose en el **campo de acción**: El proceso de auditoría de seguridad informática efectuado por los especialistas del DSI de ETECSA.

Con el propósito de cumplimentar gradualmente el objetivo general antes mencionado, el mismo se ha descompuesto en los siguientes **objetivos específicos**:

- Caracterizar los fundamentos teóricos relacionados con el proceso de auditoría de seguridad informática.
- Definir las tecnologías, herramientas y la metodología para la implementación del subsistema de evaluación del riesgo de seguridad de la información.
- Diseñar el subsistema de evaluación del riesgo de seguridad de la información.
- Implementar el subsistema de evaluación del riesgo de seguridad de la información.
- Validar la solución propuesta.

Luego de haber tratado los elementos fundamentales del área de la ciencia a incidir y los objetivos primordiales, se formula la siguiente **idea a defender:** El desarrollo de un subsistema que permita evaluar el riesgo de seguridad de la información del SASGBD, utilizando la experiencia de resultados anteriores y la lógica difusa, facilitará el proceso de toma de decisiones.

A partir de lo anterior, se establecen un conjunto de **tareas de la investigación** que guiarán la realización del trabajo**:**

- Caracterización de soluciones informáticas existentes en el mundo que permitan la evaluación de riesgo de seguridad de la información a los SGBD.
- Definición de las tecnologías, herramientas, y metodologías adecuadas para la construcción de la solución informática.
- Elaboración de los artefactos requeridos por la metodología de desarrollo seleccionada.
- Implementación de la propuesta de solución.
- Documentación de las pruebas realizadas para validar la solución implementada.

Para cumplimentar las tareas de investigación, se utilizaron los siguientes métodos de investigación:

Métodos teóricos:

1. Análisis-síntesis: utilizado en el estudio de las fuentes bibliográficas existentes referente al proceso de auditoría de seguridad informática, identificando los elementos más importantes y necesarios para dar solución al problema planteado.
2. Histórico-lógico: permitió el estudio del comportamiento y evolución de las diferentes herramientas de evaluación del riesgo de seguridad de la información para los sistemas de cómputo.
3. Modelación: utilizado en la representación, mediante el uso de diagramas, de las características del sistema, relaciones entre objetos; y las actividades que intervienen en los procesos implementados por el subsistema de evaluación del riesgo de seguridad de la información.

Métodos empíricos:

1. Entrevista: se utiliza el presente método particular para llevar a cabo el levantamiento de requisitos con el cliente.

Estructura del documento

El presente trabajo de diploma está estructurado en 4 capítulos cómo se describe a continuación:

Capítulo 1. **Evaluación del riesgo de seguridad de la información. Marco teórico de la investigación**: Tiene como objetivo analizar los diferentes aspectos teóricos relacionados con la evaluación de riesgos de seguridad de la información. Además se analizan y adoptan los principales conceptos que facilitan el estudio y compresión de la temática.

Capítulo 2. Diseño de la propuesta de solución: Tiene como objetivo el diseño de la propuesta de solución haciendo uso del razonamiento basado en casos para la implementación de un algoritmo que realice la evaluación del RSI en los SGBD.

Capítulo 3. Análisis y diseño del sistema para la evaluación del riesgo de seguridad de la información en los SGBD: Se exponen las características del sistema, incluyendo los requisitos funcionales y no funcionales, patrones de diseño y arquitectura utilizados; además de algunos de los correspondientes artefactos que requiere la metodología de desarrollo utilizada.

Capítulo 4. Implementación y prueba del sistema para la evaluación del riesgo de seguridad de la información en los SGBD: Se exponen algunos aspectos asociados con la implementación de la solución informática, así como los componentes que la integran. Además, se presentan los diseños de casos de prueba a utilizar en la validación del sistema y se analizan los resultados de las pruebas realizadas que permiten evaluar la calidad de la propuesta de solución.

FUNDAMENTACIÓN TEÓRICA

CAPÍTULO 1. EVALUACIÓN DEL RIESGO DE SEGURIDAD DE LA INFORMACIÓN. MARCO TEÓRICO DE LA INVESTIGACIÓN

1.1 Introducción

El objetivo fundamental del presente capítulo es abordar los aspectos más relevantes que conforman el fundamento teórico de la investigación. Se describen los conceptos principales. Se realiza el estudio de los principales sistemas inteligentes para el análisis de riesgo de seguridad de la información. Se realiza además una valoración de las principales herramientas existentes a nivel nacional e internacional que están relacionados con la evaluación de la seguridad de la información y que pueden servir como referencia.

1.2 Fundamentos teóricos asociados al dominio del problema

Entre los factores más importantes para alcanzar el éxito en una investigación se encuentran la calidad y claridad de las bases conceptuales relacionadas con el dominio del problema. Con este objetivo se definen a continuación los principales conceptos tratados en el desarrollo de la presente investigación.

1.2.1 **Auditoría de seguridad informática**:

Una auditoría de seguridad informática es el estudio que comprende el análisis y gestión de sistemas para identificar y posteriormente corregir las diversas vulnerabilidades que pudieran presentarse en una revisión exhaustiva de las estaciones de trabajo, los servidores y las redes de comunicaciones. Una vez obtenidos los resultados, se detallan, archivan y reportan a los responsables quienes deberán establecer medidas preventivas de refuerzo y/o corrección siguiendo siempre un proceso secuencial que permita a los administradores mejorar la seguridad de sus sistemas aprendiendo de los errores cometidos con anterioridad. (Ruiz 2011).

1.2.2 **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(Loza y Cargnelutti 2012):

- Quién y cuándo se accedió a los datos.
- Desde qué tipo de dispositivo/aplicación y que ubicación en la red.
- Cuál fue la sentencia SQL ejecutada y el efecto del acceso a la base de datos.

A través de las auditorías se demuestra la integridad de la información y las organizaciones puedan mitigar los riesgos que tienen relación con la pérdida de datos y la fuga de información. Los datos son la parte fundamental del negocio son analizados para que no tengan ninguna alteración, por lo que debe existir un control con el acceso a estos para velar su seguridad.

1.2.3 **Análisis de riesgos**:

El análisis de riesgos informáticos es un proceso que comprende la identificación de activos informáticos, sus vulnerabilidades y amenazas a los que se encuentran expuestos así como su probabilidad de ocurrencia y el impacto de las mismas, a fin de determinar los controles adecuados para aceptar, disminuir, transferir o evitar la ocurrencia del riesgo.

Teniendo en cuenta que la explotación de un riesgo causaría daños o pérdidas financieras o administrativas a una empresa u organización, se tiene la necesidad de poder estimar la magnitud del impacto del riesgo a que se encuentra expuesta mediante la aplicación de controles. Dichos controles, para que sean efectivos, deben ser implementados en conjunto formando una arquitectura de seguridad con la finalidad de preservar las propiedades de confidencialidad, integridad y disponibilidad de los recursos objetos de riesgo((MS-ISAC) 2010).

1.3 Sistemas basados en el conocimiento

Los Sistemas Basados en el Conocimiento (SBC) representan un paso delante de los sistemas de información convencionales al pretender representar funciones cognitivas del ser humano como el aprendizaje y el razonamiento. Esta clase de aplicaciones descansan en las contribuciones de la Inteligencia Artificial en lo general y en la Ingeniería del Conocimiento en lo particular. Su orientación es la automatización del análisis de problemas, la búsqueda de soluciones, la toma de decisiones y el empleo de conocimiento especializado en un campo específico de aplicación(Dr. Peña Ayala 2006).

Un SBC "*puede definirse como un sistema computarizado que utiliza conocimiento específico de un dominio y se emplea para dar solución a problemas en dicho dominio*"(Univ. Nacional Colombia 2003). Entre los productos más significativos de los SBC se encuentran los Sistemas Expertos, los cuales están encargados de representar el conocimiento de los especialistas de una rama en la procura de su aprovechamiento para tareas de diagnóstico, enseñanza y control (Dr. Peña Ayala 2006).

La composición de los SBC consta de: Un mecanismo de aprendizaje, una base de conocimientos (BC), un motor de razonamiento o inferencia (MI), y medios de comunicación hombre-máquina, es decir interfaz de usuario. En la BC, es donde se almacena el conocimiento necesario y las experiencias de los expertos para resolver los problemas del dominio; el MI, se encarga de realizar los procesos de inferencia entre la información contenida en la base de datos o memoria de trabajo y la base de conocimiento, con el fin de obtener las conclusiones que sean necesarias; y la interfaz de usuario, que es la parte del sistema experto que permite la comunicación entre el usuario y el motor de inferencias, además de que permite introducir la información que necesita el sistema y comunica las respuestas del sistema experto al usuario(Univ. Nacional Colombia 2003). Los Sistemas Expertos emplean una amplia variedad de arquitecturas específicas a las aplicaciones, sin embargo se puede generalizar un módulo de componentes que normalmente se deben integrar en cualquier ámbito, cuyos elementos se ilustran en la siguiente Figura 1:

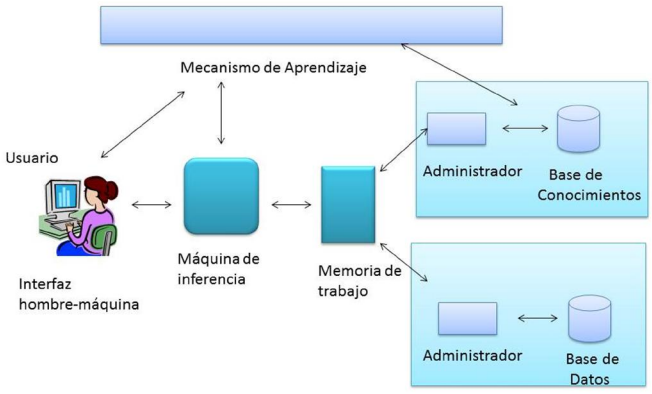


Figura 1: Arquitectura general de los componentes de un Sistema Experto.

Ventajas de los SBC

Algunas ventajas de los SBC son cuando existen las siguientes situaciones:

- El experto no está disponible.
- La experiencia se va a guardar para uso futuro o cuando la experiencia va a ser clonado o multiplicado.
- Se requiere la asistencia o entrenamiento inteligente para la toma de decisiones la solución del problema.
- El conocimiento de más de un expertos tienen que ser agrupados en una sola plataforma.

Tipos de SBC

El conocimiento almacenado en la BC puede ser de diferentes tipos: simbólicos, pesos de una red neuronal, ejemplos o casos de problemas del dominio, entre otros. Esto da lugar a la existencia de diferentes SBC, entre los que se encuentran:

- **Sistemas basados en reglas**

Los Sistemas Basados en Reglas (SBR) se caracterizan porque la forma de representación del conocimiento son las reglas de producción, y como método de inferencia utilizan la regla de modus ponens (Lío y Gálvez 1998). Este tipo de regla expresa siempre una condicional, con antecedentes y un consecuente. La interpretación de una regla surge del hecho que si los antecedentes se satisfacen entonces se logra el consecuente.

Como ventaja fundamental muestran su capacidad de interpretación y explicación de la inferencia. El proceso de solución de problemas en un SBR es crear una cadena de inferencias que constituye un camino entre la definición del problema y su solución, mediante la utilización del encadenamiento hacia atrás y hacia adelante. Una de las facilidades que brinda es la manipulación de incertidumbre.

Entre sus desventajas se encuentra el encadenamiento infinito, este se produce debido a que la mayoría de los SBR realizan una búsqueda primero a profundidad; este método padece dicho problema, si no es implementado correctamente. La adición de nuevo conocimiento puede resultar contradictorio, ya que puede darse el caso que al incorporar nuevas reglas sus conclusiones entren en contradicción con las de las reglas ya existentes en la BC. La modificación de reglas existentes, para considerar casos específicos, puede llevar a un incremento sustancial de la BC si las modificaciones no se realizan convenientemente. El reconocimiento de qué reglas son aplicables en cada ciclo es altamente ineficiente; ya que durante cada ciclo es necesario examinar cada regla para encontrar la aplicable. Hay dominios en que las entradas varían mucho y requerirán miles de reglas para considerar todas las situaciones. Por lo planteado anteriormente se descarta esta técnica para dar solución a la problemática.

- **Sistemas basados en probabilidades**

En los Sistemas Basados en Probabilidades la adquisición del conocimiento consiste en coleccionar muestras y realizar un procesamiento estadístico que produzca las probabilidades o frecuencias que forman la base de conocimiento. Éstos utilizan generalmente el Teorema de

Bayes[3] como método de solución de problemas (Lío y Gálvez 1998). A pesar de que los sistemas probabilísticos se consideran uno de los tipos más importantes de sistemas basados en el conocimiento no son factibles para todo tipo de dominio, pues se dificulta construir las redes con ayuda de expertos humanos cuando existen carencias de conocimiento. No son viables para explicar el razonamiento, ya que los métodos y modelos que utiliza están aún lejos de ofrecer explicaciones comprensibles. Este método se utiliza principalmente en sistemas donde la naturaleza de las inferencias sea probabilística. También se utiliza en sistemas donde el conocimiento provenga de múltiples fuentes, algunas fiables y otras no. Por lo planteado anteriormente no resulta conveniente utilizar esta técnica, ya que no se ajusta a los objetivos que se pretenden alcanzar con la investigación.

- **Redes expertas**

En las Redes Expertas la adquisición del conocimiento incluye la selección de los ejemplos, el diseño de su topología y el entrenamiento de la red para hallar el conjunto de pesos. Facilitan el trabajo con información incompleta y brindan algoritmos poderosos de aprendizaje para crear la base de conocimiento. Estas generalmente utilizan pesos como forma de representación del conocimiento y el cálculo de niveles de activación de las neuronas como método de solución de problemas (Lío y Gálvez 1998). Estos sistemas requieren de muchos ejemplos y son cajas negras que no explican cómo se alcanza la solución, lo que se considera una gran desventaja para la investigación, ya que se requiere de un mecanismo que muestre como se llegó a la solución propuesta.

- **Sistemas de razonamiento basados en casos**

En los Sistemas de Razonamiento Basados en Casos (SRBC) la adquisición del conocimiento se reduce a la selección de un conjunto de ejemplos o casos resueltos y su organización en la base de casos. Argumenta una solución mediante los casos que son relevantes al nuevo problema. Cada caso es la experiencia anterior almacenada. Su dificultad radica en la definición adecuada de la función de semejanza, al no existir una función de semejanza general apropiada para cualquier problema. En este paradigma la base del comportamiento inteligente de un sistema radica en recordar situaciones similares existentes en el pasado. Los dominios abordados mediante técnicas RBC implican generalmente tareas de análisis, clasificación, interpretación, diagnóstico, diseño, planificación o asesoría (Lío y Gálvez 1998).

[3] En la teoría de la probabilidad el teorema de Bayes es un resultado enunciado por Thomas Bayes en 1763 que expresa la probabilidad condicional de un evento aleatorio A dado B en términos de la distribución de probabilidad condicional del evento B dado A y la distribución de probabilidad marginal solo de A.

Después de haber analizado las diferentes técnicas de la IA para la construcción de sistemas inteligentes, se opta por la utilización del Razonamiento Basado en Casos (RBC); ya que se cuenta con una base de datos referencial de cientos de auditorías de seguridad informáticas. Estos datos no se han utilizado a no ser para mantenerlos archivados por seguridad, como vestigio comprobatorio de las auditorías realizadas. En la siguiente figura se muestra un esquema de la relación del RBC y el Subsistema de evaluación del RSI, que justifica el uso de dicha técnica.

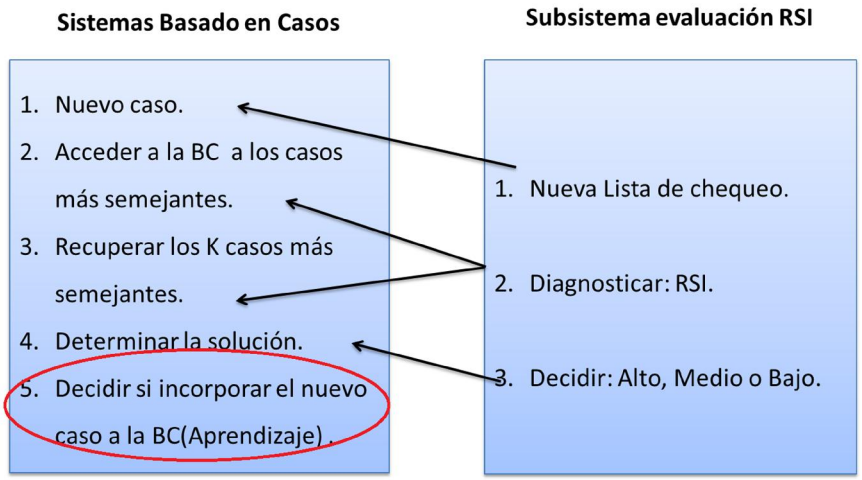


Figura 2: Relación entre el SBC y el subsistema de evaluación del RSI.

Existen varias razones que justifican el uso del RBC en la implementación del subsistema de evaluación del RSI entre los que se pueden citar:

- Es difícil conseguir pensamiento sin memoria.
- Las limitaciones que presentan los otros SBC para alcanzar existo en cualquier dominio amplio o captar efectivamente la noción del sentido común, mucho de lo cual, se cree está basado esencialmente en la memorización de la experiencia pasada.
- No siempre el pensamiento humano está regido por la reglas de la lógica, en ocasiones es básicamente un proceso de información recuperada con el tiempo.
- Para los expertos explicar la evaluación del RSI mediante cadenas de reglas es un trabajo muy engorroso, siéndole más factible describirlo a través de rasgos seleccionados previamente a partir de ejemplos existentes.
- La hipótesis de que “problemas similares tienen soluciones similares”, es común al RBC y al subsistema de evaluación de RSI, donde “auditorías similares tienen evaluaciones similares”.

1.3.1 Razonamiento basado en casos

Los SRBC son una de las tecnologías actuales para construir sistemas basados en el conocimiento para la toma de decisiones. Estos sistemas utilizan el RBC como método de solución de problemas para resolver nuevas situaciones. Los SRBC, apoyan sus predicciones en ejemplos (casos) que se almacenan en la fase de aprendizaje. Una función de distancia o semejanza determina los casos más semejantes al nuevo problema y las soluciones de los casos recuperados se adaptan para obtener una nueva solución (Martínez Sánchez, Ferreira Lorenzo y García Lorenzo 2008).

El RBC representa un método para resolver problemas no estructurados, en el cual el razonamiento se realiza a partir de una memoria asociativa que usa un algoritmo para determinar una medida de semejanza entre dos objetos (Lío y Gálvez 1998). Debe destacarse que es una técnica, en la cual la memoria se sitúa como fundamento de la Inteligencia Artificial y más concretamente de los sistemas basados en el conocimiento. Denota un método en el cual la solución de un nuevo problema se realiza a partir de las soluciones conocidas para un conjunto de problemas previamente resueltos (o no resueltos) del dominio de aplicación.

La idea básica del RBC es recuperar, adaptar y validar las soluciones encontradas en experiencias previas en un intento de relacionarlas con un problema actual. Las experiencias previas están representadas como una biblioteca de casos que reside en memoria (Lío y Gálvez 1998). Una de las formas más sencillas de dar solución a un nuevo problema es recuperando el caso más similar, y la solución del mismo se adapta al nuevo problema en un intento para resolverlo.

Como se puede apreciar el RBC es básicamente el procesamiento de la información apropiada, recuperada en el momento oportuno. De modo que el problema central es la identificación de la información pertinente cuando se necesite. El RBC tiene diferentes formas de manifestarse, como por ejemplo puede razonar considerando precedentes, usar los casos viejos para explicar nuevas situaciones o usar los casos viejos para criticar nuevas soluciones. El funcionamiento del RBC parte de estos principios y para ello realiza cuatro actividades fundamentales (Figura 3):

1. Recuperación.

2. Reutilización o adaptación.

3. Revisión.

4. Almacenamiento o retención.

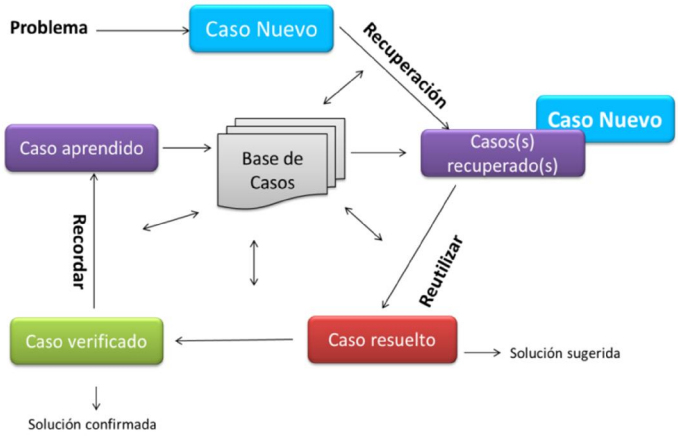


Figura 3: Ciclo de vida de un SBC**.**

Como se muestra en la figura anterior, el RBC consiste en la recuperación de los casos anteriores ya resueltos que sean similares al nuevo caso que se desea resolver; la adaptación de estos al caso a resolver; la revisión de la solución propuesta y finalmente el almacenamiento de la información relevante obtenida.

Ventajas y desventajas del razonamiento basado en casos

El RBC como método de solución de problemas para el desarrollo de sistemas basados en el conocimiento presenta diversas ventajas y desventajas entre las que se encuentran:

Ventajas:

- Facilita la adquisición de conocimiento porque a los expertos les resulta mucho más fácil solucionar el caso que proponer reglas de aplicación general.
- Facilita el aprendizaje porque es capaz de aprender por simple acumulación de casos, ya que cada vez que se soluciona un nuevo problema este puede ser incorporado como un nuevo caso a la base de casos.
- Propone soluciones en dominios poco formalizados o que no se comprenden del todo, no es necesario que toda la información del dominio esté completa.
- Permite construir razonadores más eficientes, puesto que suele ser menos costoso modificar una solución previa que construir una nueva solución desde cero.
- Se facilita el mantenimiento de la base de conocimiento, ya que los usuarios pueden añadir nuevos casos sin ayuda de los expertos.

- Es más fácil conseguir que los usuarios acepten las sugerencias del sistema ya que estas están avaladas por una situación previa.
- Los casos pueden proporcionar también “información negativa”, alertando sobre posibles fallos.

Desventajas:

- Requiere de una base de datos considerablemente grande y bien seleccionada.
- La consistencia entre varios casos es difícil de mantener.
- El RBC depende de una adecuada función de semejanza la cual no es fácil de encontrar para cada aplicación.

1.3.2 Base de Casos

El componente principal de un SRBC, es la Base de Casos (BC), la cual constituye la memoria del sistema. La BC contiene las experiencias, ejemplos o casos a partir de los cuales el sistema hace sus inferencias. Esta base puede ser generada a partir de los casos o ejemplos resultantes del trabajo de expertos humanos o por un procedimiento automático o semiautomático que construye los casos desde datos existentes registrados, por ejemplo en una base de datos.

Formas de organización de la BC

Dado que los casos constituyen el elemento principal de todo SRBC, la manera de almacenarlos repercutirá directamente en el rendimiento del mismo. Tradicionalmente se han propuesto dos estructuras principales para el almacenamiento de casos:

Plana: En esta estructura se presentan los casos completos de forma secuencial en una lista simple, un arreglo o un fichero. Esta estructura tiene el principal inconveniente de que la búsqueda de casos es menos eficiente, por lo que no es recomendable utilizarla en sistemas que trabajan con una base de casos de gran tamaño y necesitan respuesta en tiempo real. Por otra parte, la inserción de nuevos casos en este tipo de estructuras es muy sencilla ya que basta con incluir un nuevo registro con el nuevo (Mora 2008).

Jerárquica: En una estructura con memoria jerárquica se utilizan representaciones en forma de árbol, en los que cada nodo interior representa un atributo del caso y en las hojas se almacenan las soluciones de los mismos. Cada recorrido desde la raíz hasta una de las hojas del árbol representa un caso completo. La gran ventaja de este tipo de almacenamiento es la eficiencia en la búsqueda, pero a cambio se sacrifica la sencillez de inserción de nuevos casos (Mora 2008).

Memoria plana **Memoria jerárquica**

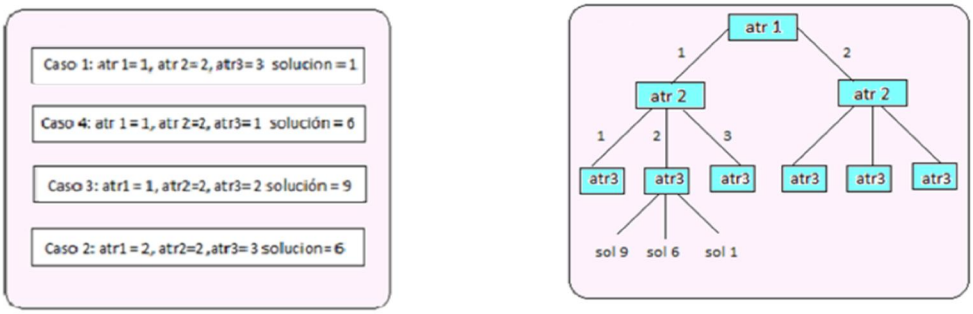


Figura 4: Estructura plana y jerárquica.

Descripción de los módulos de un SRBC

Los componentes del sistema no se pueden analizar separadamente pues la arquitectura de cada uno puede influir en la eficiencia del trabajo de la otra; por ejemplo, el modelo de la base de casos influye en el modelo del recuperador de casos. A continuación se hace una breve descripción de cada uno de los componentes de un SRBC.

Módulo de recuperación:

La peculiaridad del RBC implica que la solución de un problema se realice recuperando, desde la memoria, los problemas ya resueltos almacenados en la BC que tienen relación con este. Este proceso está compuesto por dos etapas: el acceso a un conjunto de elementos de memoria o casos que resulten promisorios para el problema a resolver y la recuperación del o los elementos de memoria de este conjunto que sean más semejantes al problema.

La recuperación o selección de los ítems de la memoria permanente, semejantes al problema actual se realiza utilizando una función de semejanza, la cual determina una medida numérica del grado de similitud de cada ítem de la memoria con respecto al problema a resolver. Esta función considera las diferencias y semejanzas entre la descripción de ambos problemas. El mecanismo de recuperación en un SBC asegura que los casos más relevantes sean recuperados para el problema dado, siendo este el resultado más importante del RBC. Los casos relevantes son aquellos que coinciden con la mayoría de los rasgos del problema presente.

En el proceso de recuperación influyen otros elementos que cuentan con gran importancia. Uno de ellos es el umbral de semejanza que no es más que el valor mínimo que debe tener un caso para ser considerado a la hora de dar solución a nuevos problemas.

Módulo de adaptación:

Después de la determinación de los casos más semejantes, las soluciones contenidas en dichos casos pueden usarse directamente como solución al nuevo problema, pero comúnmente necesitan ser modificadas. Este nuevo caso obtenido como resultado final será incorporado a la BC como uno más, lo que permitirá que el sistema se desarrolle "aprendiendo" de sí mismo. En la implementación de los métodos de adaptación se hace imprescindible el conocimiento de los expertos en este dominio.

Módulo de almacenamiento:

En la fase de almacenamiento se procede a almacenar el caso recién resuelto en la base de casos para uso futuro. Este proceso comprende tres fases: determinar qué se necesita almacenar, definir cómo será indexado el nuevo caso y finalmente integrar el caso a la memoria de casos. A este módulo se le conoce también como módulo de aprendizaje.

1.4 Estado del arte

En este epígrafe se presentan algunas de las soluciones que existen en el mundo que poseen relación con la presente investigación.

- **Sistemas inteligentes para el análisis de riesgo de seguridad de la información**

ALRAM y BDSS

El Sistema de Soporte de Decisión Bayesiana (con las siglas en inglés: BDSS) y el Automatizada Metodología (con las siglas en inglés: ALRAM) aplican algoritmos de evaluación del riesgo de la industria nuclear para la seguridad de la información, remplazando el algoritmo FIPSPUB-65. BDSS es un sistema experto que proporciona al usuario una base de conocimiento amplia que dirige las vulnerabilidades y los resguardos, así como los factores de exposición de las amenazas y la frecuencia de los datos. Todos son totalmente mapeados y cruzados con algoritmos de modelación del riesgo e interfaces y presentación de lenguaje natural. ALRAM, sin embargo, requiere un experto para construir y mapear la base de conocimiento y entonces conducir a una evaluación del riesgo personalizada(Peltier 2001).

Estos sistemas aunque están enfocados al riesgo de seguridad en el área de la industria nuclear y no a los sistemas informáticos, proponen a través de técnicas de Inteligencia Artificial una forma de evaluar, por lo que abren el camino para que puedan ser utilizadas las mismas en el área de la seguridad informática. Conociendo que el valor del riesgo es un número probabilístico, una solución a través de un modelo bayesiano se nos puede presentar idóneo. El problema en este enfoque es la de prevenir los suficientes datos a priori que permitan realizar la estimación

probabilística del fenómeno. Si se encuentra una auditoría donde las características la convierten en única o nueva, se podría entrar en dificultades para utilizar el modelo bayesiano.
En cambio, una base de conocimiento a disposición, que no dependa de la probabilidad de ocurrencia del fenómeno de un ataque a la seguridad de la información, puede servir para adaptar una de sus soluciones al caso que puede ser nuevo o no y emitir una evaluación del riesgo.

MIDS

Es un Sistema de Detección de Intruso (con las siglas en inglés: MIDS). Tiene como principal ventaja una proporción baja de falsos positivos y las acciones preventivas y correctivas pueden ser realizadas fácilmente. El sistema puede ser dividido en varias categorías, una de ellas es sistema experto. En esta categoría el sistema tiene un conjunto de reglas que describen los ataques empleados. Los eventos de auditorías son traducidos como hechos, llevando su significación semántica en el sistema experto y el motor de inferencia traza conclusiones usando ambas, reglas y hechos (Chou 2011).

La baja proporción de falsos positivos demuestra la efectividad de utilizar las reglas y hechos en el área de la seguridad informática, en la detección de intrusos, por lo que a pesar de que este sistema no está pensado para la evaluación del riesgo de seguridad de la información, sí demuestra la efectividad de estas técnicas en este escenario. Proporcionando propuestas de soluciones a los problemas planteados en este trabajo.

@RISK

Este sistema realiza análisis de riesgo utilizando la simulación para mostrar múltiples resultados posibles en un modelo de hoja de cálculo y le indica qué probabilidad hay de que se produzcan. Computa y controla matemática y objetivamente gran número de escenarios futuros y luego le indica las probabilidades y riesgos asociados con cada uno. Esto quiere decir que permite decidir qué riesgos tomar y cuáles evitar, tomando la mejor decisión en situaciones de incertidumbre(Palisade 2014).

@RISK planifica estrategias de administración de riesgo mediante la integración de RISKOptimizer, que utiliza la simulación Monte Carlo[4] para el cálculo del riesgo. Usando algoritmos genéticos u OptQuest, junto con las funciones de @RISK, RISKOptimizer puede

[4] El **método de Monte Carlo** es un método no determinista o estadístico numérico, usado para aproximar expresiones matemáticas complejas y costosas de evaluar con exactitud(Peña Sánchez 2011).

determinar la mejor asignación de recursos, la distribución óptima de activos, el calendario más eficiente y otras más(Palisade 2014).

El método de Monte Carlos está basado en números generados al azar. Los escenarios que construye están basados en estos números. La probabilidad de ocurrencia de un ataque a las vulnerabilidades a un SGBD no es obra del azar, sino de un estudio del atacante. Por lo que, si las vulnerabilidades conocidas están fuertemente protegidas o eliminadas, puede repercutir en que la frecuencia de ocurrencia de un ataque puede ser cero o es muy baja, por la protección existente. Por lo que podría considerarse que el riesgo es casi nulo, mientras que los números aleatorios pueden estar proponiendo otros valores.

- **Sistemas que automatizan el análisis de las auditorías de seguridad informáticas**

Desde el punto de vista práctico existen varios sistemas informáticos que permiten la automatización de un grupo de actividades relacionadas a las auditorías informáticas y dentro de estas, la evaluación del RSI. En este sentido cabe mencionar las siguientes:

- Sistemas de análisis automático de las TI, en busca de posibles vulnerabilidades y fallos de seguridad. Pueden encontrar errores de administración y fallos de seguridad comunes, tales como la falta de actualizaciones del sistema, contraseñas caducadas, existencia de más de una cuenta de administrador o de una cuenta de invitado, qué directorios se comparten y con quién, posibles fallos en Internet Explorer, Microsoft Office, SQL Server, Internet Information Server, etc. Además detecta los errores más comunes de configuración de seguridad y actualizaciones de seguridad que falten en los sistemas informáticos. Existen aplicaciones enfocadas a varios tipos de tecnologías como es el caso de Microsoft Baseline Security Analyze (Softonic 2014). Otras están enfocadas solamente a los SGBD como es el caso de DB Audit, Secure Oracle Auditor y Auditor de Seguridad del SQL, Microsoft Forefont, MSAT (Technologies 2013), entre otras aplicaciones. Estos sistemas a pesar de ser capaces de detectar las vulnerabilidades de las TI, no llegan a evaluar el riesgo de seguridad de la información.
- Aplicaciones que automatizan las listas de chequeo. Tienen como finalidad evaluar el cumplimiento de las listas de chequeo. Esencialmente comparan las configuraciones de seguridad de los sistemas de TI según las listas de chequeo. Una de estas herramientas es el CIS-CAT, el cual trabaja según las listas de chequeo de la CIS (de inglés: Center for Internet Security) (CIS, 2013) y brinda una puntuación de los parámetros cumplidos de la lista de chequeo pero no evalúa el riesgo de seguridad de la información.

- **Las listas de chequeo de seguridad para los SGBD**

De acuerdo a los autores(Broder y Tucker 2011), las listas de chequeo son usadas para facilitar la recolección de información pertinente. Estas pueden tener muchas formas. Pueden ser simples listas de preguntas de si o no, o de preguntas de final abierto requiriendo respuestas en formas de redacción. Ellas pueden ser breves y enfoques limitados en la específica operación o actividad en cuestión, o pueden ser extensas en alcance y cubrir lo común en lo concerniente a la seguridad de todo las operaciones de la compañía. El propósito de las listas de chequeo es proporcional una grabación lógica de la información y asegurar que preguntas no importantes vayan sin ser solicitadas.

Listas de chequeo de las CIS (Center for Internet Security)

Las listas de chequeo de la CIS son una norma extensamente aceptada según el sitio web de la CIS. Son basadas entre acuerdos de expertos y se aceptan ampliamente por las agencias del gobierno americanas para la complacencia de FISMA y por interventores para la complacencia con las normas de ISO(CIS 2014).

Las listas de chequeo de la CIS están disponibles para los SGBD: MySQLDatabase, FreeBSD, IBM DB2, Microsoft MS SQL Server, Oracle Database Server, Sybase ASE y otros(CIS 2014). Además para otro conjunto de aplicaciones utilizadas para servidores como Apache y Microsoft Exchange. Con estas listas de chequeo los especialistas del DSI de la empresa ETECSA, se apoyan para realizar el proceso de auditoría de seguridad informática a los SGBD. En la siguiente figura se muestra un fragmento de la lista de chequeo de parámetros a evaluar para el SGBD MySQL 4.1(CIS 2014).

Parámetro	Abreviatura
Configuración del Sistema Operativo	CSO
Máquina dedicada	MD
Ejecutar MySql en modo chroot.	ECR
Cuentas de servicio	CS
Permisos sobre el File System	PFS
Logs de errores	LE
Permisos sobre los directorios de datos	PDD
Permisos sobre los ficheros binarios	PFB
Ficheros de configuración	FC
Logs	L
Logs de errores	LE
Directorio de Logs	DL
Log Update	LU
General	G
Version del SGBD	VS
Cuenta del usuario root	CUR

Tabla 2: Fragmento de la lista de chequeo de parámetros a evaluar para el SGBD MySQL(CIS 2014).

Análisis del estudio realizado

Actualmente existen varios sistemas inteligentes relacionados con la evaluación de riesgo de la seguridad de la información. Sin embargo con la revisión bibliográfica realizada se concluye que los mismos están enfocados de forma genérica para cualquier tipo de entidad a donde se quieran aplicar. Esta característica puede ser una fortaleza, pero trae como consecuencia que se necesita la intervención del experto en bases de datos con su experiencia en el proceso de evaluación del RSI para arrojar un resultado más ajustado a la realidad, según las características propias de cada entidad auditada.

Las listas de chequeo facilitan la recolección de la información para la detección de las vulnerabilidades necesarias para la evaluación del RSI, pero no orientan el proceso necesario para evaluar el riesgo de la TI. Debido a su reconocimiento internacional y su empleo por los auditores de seguridad informática, es importante tenerlas en cuenta.

A los sistemas informáticos inteligentes estudiados, no se les encontró un enfoque en el área de la seguridad de la información para SGBD, ni utilizan las listas de chequeo. Sin embargo, manifiestan la utilidad de las técnicas de la IA para tareas de análisis, clasificación, interpretación, diagnóstico, diseño y planificación. Por lo planteado anteriormente, se concluye que es necesaria una solución que permita realizar una evaluación del RSI a los SGBD utilizando técnicas de IA para la toma de decisiones.

1.5 Tecnologías y Herramientas

1.5.1 Metodología de desarrollo de software

Para asegurar el éxito durante el desarrollo de software no es suficiente contar con notaciones de modelado y herramientas, hace falta un elemento importante: la metodología de desarrollo, la cual provee una dirección a seguir para la correcta aplicación de los demás elementos. Actualmente las metodologías se clasifican en Metodologías pesadas y Metodologías ágiles.

Las metodologías ágiles, por lo general se centran en desarrollar productos funcionales más que en conseguir una buena documentación (Calderón y Dámaris 2007). Son metodologías centradas en la implementación, que evita cualquier tipo de documentación fuera del código.

Por otra parte, las Metodologías Pesadas están orientadas al control de los procesos, estableciendo rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán. Teniendo en cuenta lo antes expuesto y la necesidad de contar con la documentación necesaria para la futura implementación del sistema, se propone el uso de la metodología pesada Proceso Unificado de Rational (RUP).

Proceso unificado de Rational / Rational Unified Process (RUP)

El proceso unificado conocido como RUP, es un modelo de software que permite el desarrollo de software a gran escala, mediante un proceso continuo de pruebas y retroalimentación, garantizando el cumplimiento de ciertos estándares de calidad. Proporciona un enfoque disciplinado para la asignación tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de alta calidad software que satisfaga las necesidades de sus usuarios finales, dentro de un horario predecible y presupuesto. Aunque con el inconveniente de generar mayor complejidad en los controles de administración del mismo. Sin embargo, los beneficios obtenidos recompensan el esfuerzo invertido en este aspecto(Flores y Lourdes 2012).

Para el desarrollo del presente trabajo se seleccionó la Metodología de Desarrollo de Software RUP, pues genera con su utilización gran cantidad de documentación, cuestión importante en la investigación, ya que facilitará el trabajo futuro de los desarrolladores o administradores. Además, con RUP se hace una adecuada captura de requisitos, lo cual sin dudas contribuirá a que el trabajo sea más rápido y que a la vez se cumpla con las expectativas y funcionalidades que se quieren lograr. Brinda también la facilidad de que una vez detectado un posible error se pueda retroceder y corregirlo, sin que atente en un futuro con el correcto funcionamiento del sistema.

RUP es un proceso de desarrollo de software que constituye una metodología, utilizando UML (Lenguaje Unificado de Modelado) para la modelación es una de las más utilizadas para desarrollo de sistemas.
Propone 9 flujos de trabajo: 6 de ingeniería y 3 de apoyo:

Flujos de trabajo de ingeniaría: Modelamiento del Negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba, y Despliegue.
Flujos de trabajo de apoyo: Administración del proyecto, Configuración y control de cambios, Entorno.

En la **Figura 5** se muestran las 4 fases de desarrollo y los 9 flujos de trabajo que conforman esta metodología de desarrollo de software.

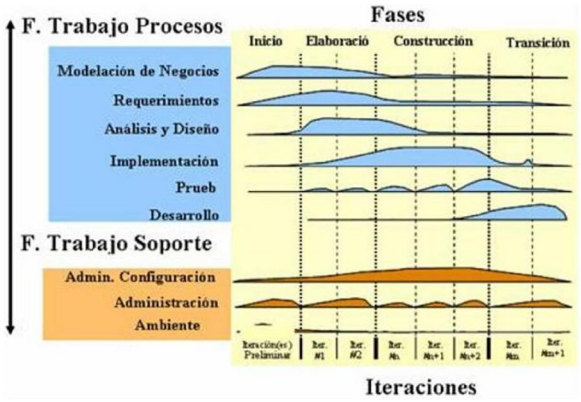


Figura 5: Fases y flujos de trabajo de RUP(Montero 2015).

1.5.2 Lenguaje unificado de modelado (UML)

El Lenguaje Unificado de Modelado (UML, Unified Modeling Language) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios(Jacobson, Rumbaugh y Booch 2007).
El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo

iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos (Jacobson, Rumbaugh y Booch 2007)

1.5.3 Herramienta case. Visual Paradigm

Como herramienta Case se propone la utilización del **Visual Paradigm**. La misma es una herramienta que es soportada por cualquier sistema operativo. Es una herramienta visual de ingeniería de software para el modelado, que tiene un entorno de trabajo que muestra colección de menús, barra de herramientas y ventanas que permite realizar diagramas. Puede ser utilizada por una gran variedad de usuarios como analistas de sistemas, analistas de negocios e ingenieros de software. Provee soporte para la generación de código y la ingeniería inversa. Se integra con algunas herramientas como: Eclipse, Netbeans, Jbuilder, Oracle, entre otras. Se utilizará Visual Paradigm Suite versión 8.0.

1.5.4 Lenguajes de programación. Java

Java es un lenguaje de programación orientado a objetos ideado por Sun Microsystem. Es un lenguaje de propósito general por lo que con él se podría crear cualquier tipo de aplicación. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria (Belmonte Fernández 2005) .

De acuerdo con el libro blanco de Java, este lenguaje de programación se encamina a ser "*un lenguaje sencillo, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutral, portátil, de gran rendimiento, multitarea y dinámico*"(Zukowski 2003).

Como parte de las tendencias de lenguajes de programación actuales, Java presenta los siguientes beneficios:

- Es independiente de la plataforma de desarrollo.
- Existen dentro de su librería, clases gráficas que permiten crear objetos visuales comunes altamente configurables y con una arquitectura independiente de la plataforma.
- Java permite a los desarrolladores aprovechar la flexibilidad de la Programación
- Orientada a Objetos en el diseño de sus aplicaciones.
- El manejo de las bases de datos es uniforme, es decir, transparente y simple.
- El sistema de Java es seguro ya que tiene muchas funciones de seguridad integradas, que garantizan la seguridad del código que se está ejecutando.

1.5.5 Marco de trabajo

En la actualidad existen varias herramientas imprescindibles en el proceso de desarrollo de un software, entre las que se encuentran los marcos de trabajo, o frameworks por su terminología en inglés. "*Un Framework es un diseño reutilizable de todo o parte de un sistema de software descrito por varias jerarquías de herencia de clases (generalmente algunas abstractas), y por las colaboraciones que se establecen entre las instancias de estas clases"* (Johnson y Foote 1988).

Marco de trabajo Hibernate

En la actualidad existen varios frameworks ORM capaces de solucionar el problema de la diferencia entre dos modelos utilizados para organizar y manipular datos: el orientado a objetos en las aplicaciones y el relacional en las bases de datos. Sin embargo se decide utilizar el marco de trabajo Hibernate, debido a que es gratuito y de código abierto, cuenta con amplia documentación y aceptación dentro de la comunidad Java. Además entre sus características principales se encuentran:

- Soporta el paradigma de orientación a objetos de una manera natural: herencia, polimorfismo, composición y la librería de colecciones de java.
- Provee un sistema de cache de dos niveles y puede ser usado en un clúster. Permite inicialización perezosa (lazy) de objetos y colecciones.
- Presenta un potente mecanismo de transacciones.
- Proporciona el lenguaje HQL el cual provee una independencia del SQL de la base de datos, tanto para el almacenamiento de objetos como para su recuperación.

Marco de trabajo Spring

Spring es un marco de trabajo de código abierto para el desarrollo de aplicaciones en la plataforma Java, cuenta con un conjunto de librerías en las que podemos escoger aquellas que faciliten el desarrollo de nuestra aplicación. Entre sus posibilidades más potentes está su contenedor de Inversión de Control (Inversión de Control, también llamado Inyección de Dependencias, es una técnica alternativa a las clásicas búsquedas de recursos vía JNDI. Permite configurar las clases en un archivo XML y definir en él las dependencias. De esta forma la aplicación se vuelve muy modular y a la vez no adquiere dependencias con Spring), la introducción de aspectos, plantillas de utilidades para Hibernate, iBatis y JDBC así como la integración con JSF.

Fue seleccionado debido a que puede emplearse en cualquier aplicación desarrollada en Java, brinda muchas libertades a los desarrolladores y ofrece soluciones bien documentadas. Además de permitir escribir un código más limpio, manejable y fácil de probar.

1.5.6 Gestor de base de datos

Los Sistemas Gestores de Bases de Datos (SGBD), son software que permiten la administración y mantenimiento de los ficheros y datos de una Base de Datos o de varias. Proporcionan funcionalidad añadida al sistema de ficheros para facilitar la gestión de datos. Además, proporcionan valor añadido, como seguridad en cuanto a acceso, copias de respaldo, entre otras. En la actualidad existen disímiles SGBD. Para la futura implementación del sistema se propone el uso de PostgreSQL, esencialmente porque es de software libre, posibilitando su uso sin restricciones.

PostgreSQL: Dentro de los gestores de bases de datos existentes, se nombra como uno de los más distintivos. Está diseñado para soportar volúmenes masivos de datos, sin que ello afecte en lo absoluto su rendimiento. Ofrece una fortaleza adicional sustancial al incorporar cuatro conceptos adicionales básicos: Clases, Herencia, Tipos, Funciones. Cuenta además con características que aportan potencia y flexibilidad adicional: Restricciones, Disparadores, Reglas, Integridad transaccional (Postgresql 2000). Se propone el uso de PostgreSQL 8.4.

1.6 Conclusiones parciales

En este capítulo se trataron los elementos teóricos que dan sustento a la propuesta de solución del problema planteado, arribando a las siguientes conclusiones:

- El estudio de los conceptos asociados a la auditoría de seguridad informática y evaluación de riesgo de seguridad de la información permitió lograr un mejor entendimiento de la investigación que se realiza.
- El estudio de los diferentes sistemas inteligentes para el análisis del riesgo de seguridad de la información permitió identificar las facilidades que brindan las técnicas de la IA para apoyar el proceso de toma de decisiones.
- La selección de la metodología, herramientas y tecnologías con soporte multiplataforma y basadas en software libre, posibilitó obtener una base tecnológica enfocada en las necesidades de la investigación.
- La falta de una herramienta informática que realice evaluaciones del riesgo en los SGBD, hacen necesario el desarrollo de un subsistema de evaluación del riesgo de seguridad de la información para el SASGBD.

CAPÍTULO 2. DISEÑO DE LA PROPUESTA DE SOLUCIÓN

2.1 Introducción

El objetivo fundamental del presente capítulo es el diseño propuesta de solución. Se identifican los rasgos predictores y objetivos que conformaran los casos. Se proponen además los algoritmos a utilizar en las fases de recuperación, adaptación y aprendizaje. Posteriormente se hace uso de la lógica difusa para manipular las ambigüedades de valores e imprecisiones en las evaluaciones del RSI.

2.2 Propuesta de solución

La solución propuesta está concebida en la implementación de un subsistema de evaluación del RSI para SASGBD, utilizando el RBC y la lógica difusa y de esta forma apoyar el proceso de toma de decisiones. El subsistema tiene como entrada, las listas de chequeo de seguridad, luego de las mismas haber recogido los datos en el monitoreo de los SGBD para realizar la evaluación del riesgo. La salida del subsistema es la evaluación del RSI en los términos en los que son establecidos por los especialistas. Además de esta salida, con el sistema se obtiene la evaluación numérica del RSI, para cada parámetro que existente en las listas de chequeo, así como del resultado final de la evaluación. La siguiente figura muestra la propuesta de solución antes descrita.

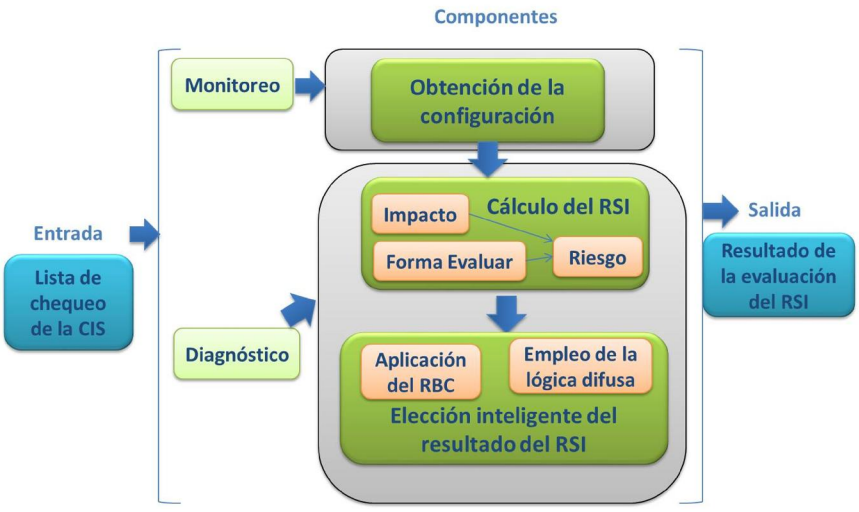


Figura 6: Propuesta de solución.

Descripción de los componentes de la propuesta de solución

a) **Componente: Obtención de la configuración**

Este componente contiene la Herramienta Colaborativa para la Realización de Auditorías (HCRA), la cual está destinada a apoyar las auditorías que se realizan a través del SASGBD. Esta herramienta se enfoca en obtener las configuraciones de seguridad del servidor auditado, organizado a través de los parámetros extraídos del SASGBD.

La solución informática HCRA es capaz de monitorear los SGBD siguientes: PostgreSQL, MySQL, SQL Server y Oracle, que son los principales gestores hospedados en los servidores de ETECSA.

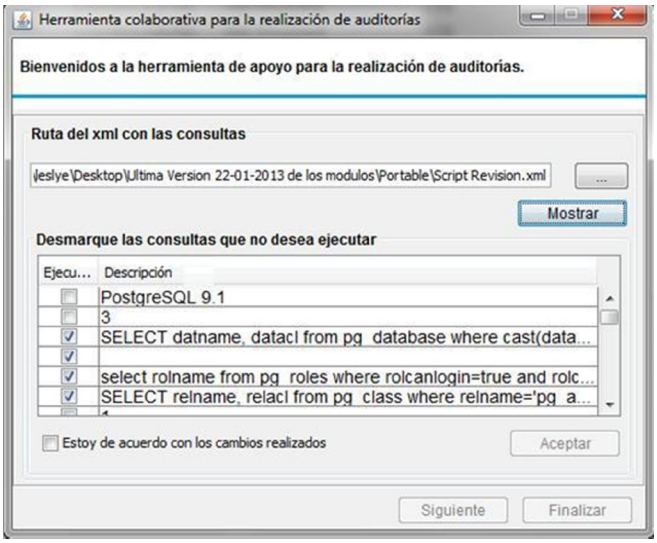


Figura 7: Interfaz de la HCRA.

Con esta herramienta se mejora el tiempo utilizado para el monitoreo, además el archivo exportado por HCRA, se convierte en la entrada del siguiente componente de la propuesta de solución en dicha investigación.

b) **Componente: Cálculo del RSI**

En este componente se define la manera de determinar el impacto: el peso del costo de la pérdida de un recurso y la probabilidad de ataque a un parámetro n de la lista de chequeo. Cada parámetro n de la lista de chequeo tiene un impacto cuyo valor es un término lingüístico de los vistos anteriormente para los niveles de riesgos. Se tomó en acuerdo con los expertos los siguientes valores para el impacto. Ver **Anexo # 1**:

$$\text{Impacto} = \begin{Bmatrix} Si\ Alto & Entonces\ 1 \\ Si\ Medio & Entonces\ 0{,}66 \\ Si\ Bajo & Entonces\ 0{,}33 \end{Bmatrix}$$

Se plantea utilizar la siguiente fórmula, en acuerdo con los especialistas para realizar una evaluación del riesgo:

R_i = Impacto$_i$ * Evaluación del parámetro$_i$

La variable R_n representa el valor del riesgo para un parámetro i de la lista de chequeo que se esté utilizando para llevar a cabo la auditoría de seguridad informática para el SGBD en específico.

c) **Componente: Elección inteligente del resultado del riesgo**

Para la selección del resultado, este componente se apoya en el uso de las técnicas de la inteligencia artificial: razonamiento basado en casos (RBC) y la lógica difusa.

El acceso de los casos en el componente se realiza utilizando la técnica Acceso Secuencial.

El desarrollo de dicho componente de la propuesta de solución se detalla a continuación:

2.3 Diseño de la base de casos

2.3.1 Selección de los rasgos predictores

A continuación se muestra un fragmento de los parámetros de la lista de chequeo para PostreSQL

Rasgos predictores	Impacto
Cuentas con seguridad integrada	Alto
Logines del motor de la base de datos	Medio
Pertenencia de usuarios a grupos	Alto
Usuarios con claves nulas	Alto
Contraseñas que no caducan	Alto
Roles del motor de base de datos	Alto
Modificación del esquema de bases de datos	Alto
Actualización del catálogo del sistema	Alto
Utilización del Primary Domain Controller	Medio
Permisos sobre directorios de la base de datos	Alto
Ejecución de otras aplicaciones	Medio
Utilitarios para el manejo y administración de la base de datos (R2.4)	Bajo

Tabla 3: Representación de los parámetros de la lista de chequeo.

.

Como se pudo observar en la tabla anterior los riesgos de los parámetros de la lista de chequeo son los rasgos predictores que se calculan como se describió en el componente anterior. Cada SGBD tiene dedicada una lista de chequeo. En el **anexo # 2** se encuentra todos los parámetros de la lista de chequeo para PostreSQL. Para consultar las listas de chequeo para los SGBD, puede consultar en(CIS 2013). En el caso del SGBD PostgreSQL, la lista fue confeccionada por los propios expertos del DSI de ETECSA debido a su inexistencia en el sitio oficial del Centro de Seguridad de la Internet (por sus siglas en inglés: CIS). La elaboración de esta lista es guiada utilizando las mismas pautas que en el resto de las listas de chequeo publicadas.

La BC se encuentra alojada en un SGBD, donde se encuentran las auditorías de seguridad informáticas anteriores. La figura 8, ilustra el modo el con el cual está estructurada la BC. Como se aprecia, la BC está particionada u organizada en forma de árbol, la cual permite que los casos que corresponden a un SGBD, no estén relacionados con los de otro SGBD. A su vez existen otros niveles o ramas, antes de llegar a los casos como es la versión y la evaluación. La forma de organizar los casos, favorece el acceso y recuperación de los casos. La variable n en la figura se utiliza para transmitir el significado de un número indeterminado.

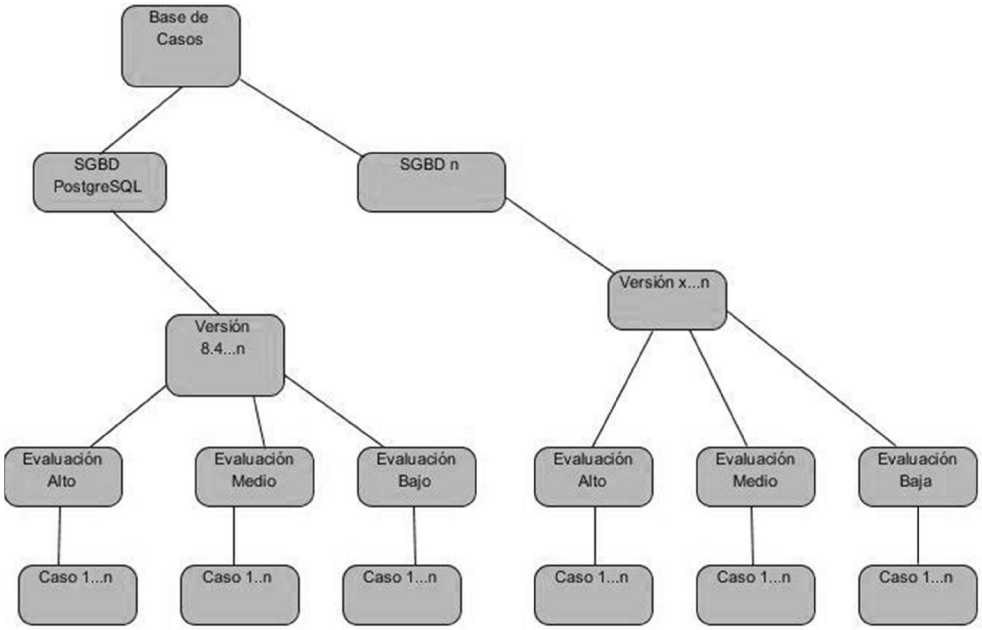


Figura 8: Representación de la estructura de la BC.

2.3.2 Selección del rasgo objetivo

El rasgo objetivo no es más que la evaluación del riesgo de seguridad de la información para un determinado gestor, que puede contener los siguientes valores: ALTO, MEDIO Y BAJO.

2.4 Funciones de comparación entre rasgos y función de semejanza entre casos

La función de comparación utilizada es la misma para todos los rasgos predictores que componen los casos, debido a que se opera sobre un mismo dato, el riesgo. El dominio de los rasgos predictores a utilizar es numérico, debido que son los parámetros los utilizados para el cálculo del riesgo.

Función de comparación para atributos numéricos.

La distancia absoluta (llamada de Manhattan o de City Block) se representa como la diferencia absoluta sobre todas las dimensiones. Se selecciona la función distancia de Manhattan, apoyado en los resultados obtenidos en el modelo de los autores (Zhang y otros, 2011), donde se propone para la semejanza entre atributos, el uso de la misma.

$$\mathrm{d}(\mathrm{X_{nr}}(\mathrm{O_r}), \mathrm{X_{ni}}(\mathrm{O_t})) = |\mathrm{X_{nr}} - \mathrm{X_{ni}}|$$

Donde de un conjunto n de parámetros existente en la lista de chequeo, la variable $\mathrm{X_{nr}}(\mathrm{O_r})$, corresponde a la evaluación dada a un rasgo $\mathrm{X_{nr}}$ de en un caso $\mathrm{O_r}$ que represente la auditoría de seguridad informática que se esté llevando a cabo. La variable $\mathrm{X_{ni}}(\mathrm{O_t})$ corresponde a la evaluación dada a un rasgo $\mathrm{X_{ni}}$ en un caso t ($\mathrm{O_t}$), almacenado en la base de casos.

Se utiliza la misma función de Manhattan para todos los rasgos predictores como se hace en el modelo referenciado anteriormente y debido a que se mide una misma característica de los rasgos y el dominio de los valores también es el mismo.

En esta función no se utilizan las variables para la normalización de la función (máx ($\mathrm{X_{ni}}$) y mín ($\mathrm{X_{ni}}$)), porque los valores normalizados de la evaluación de los parámetros están en el rango entre 0 y 1, como se definió en el componente anterior a través de los expertos, se simplifica la cantidad de operaciones matemáticas.

Función de semejanza entre casos

La propuesta general de distancia entre los casos es:

$$\mathrm{f}(\mathrm{O_r}, \mathrm{O_t}) = \sum_{\mathrm{i=1}}^{\mathrm{n}} \left(\left| \frac{w_i * d(\mathrm{X_{nr}}(\mathrm{O_r}), \mathrm{X_{ni}}(\mathrm{O_t}))}{\sum_{\mathrm{i}}^{\mathrm{n}} w_i} \right| \right)$$

Donde $\mathrm{w_i}$ es el peso de importancia de los rasgos, en este caso, el valor a utilizar es el del impacto especificado por los expertos para cada parámetro de la lista de chequeo.

Para aplicar la función de distancia, no se prevé que existan rasgos descriptores sin valores. La aplicación SASGBD, obliga al experto a corregir o adicionar los valores de los rasgos descriptores.

Esto puede tener lugar debido a que existen parámetros en las listas de chequeo que obligatoriamente necesitan ser evaluadas por el auditor para llegar a contener un valor.

2.5 Umbral de semejanza

El valor del umbral está relacionado con el comportamiento de los casos en su semejanza, un umbral muy bajo puede conllevar a que se tengan muchos casos en la recuperación y un umbral muy elevado puede conllevar a que no se pueda dar una respuesta en la búsqueda. De ahí que el umbral sea dado por un experto. En caso de que el experto tenga problemas con la definición del umbral, el propio módulo de recuperación debe ser capaz de brindar la posibilidad de calcular el valor umbral de los rasgos.

Para ello construye una matriz cuadrada donde la fila y las columnas están representadas por los casos que se encuentran almacenados en la BC y en la intersección está el valor de semejanza (β).

	Caso 1	Caso 2	Caso 3	Caso n
Caso1	$\beta(C1,C1)$	$\beta(C1,C2)$	$\beta(C1,C3)$	$\beta(C1,Cn)$
Caso 2	$\beta(C2,C1)$	$\beta(C2,C2)$	$\beta(C2,C3)$	$\beta(C2,Cn)$
Caso 3	$\beta(C3,C1)$	$\beta(C3,C2)$	$\beta(C3,C3$	$\beta(C3,Cn)$
Caso n	$\beta(Cn,C1)$	$\beta(Cn,C2)$	$\beta(Cn,C3)$	$\beta(Cn,Cn)$

Tabla 4: Matriz de semejanza entre casos.

Para permitir que se tengan en cuenta los casos de la BC, la Dra. Natalia Martínez propone (Martínez Sánchez, Ferreira Lorenzo y García Lorenzo 2008) un cálculo del umbral de semejanza, que viene dada por una media aritmética con los valores de semejanza entre los casos, mediante la siguiente expresión:

$$\beta = \frac{2}{m(m-1)} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \beta(O0, Ot)$$

Dónde:

m: Número de casos

i: Casos en filas

j : Casos en columnas

$\beta(O0, Ot)$: Función de semejanza entre los rasgos Oo y Ot.

2.6 Propuesta de algoritmos a utilizar en las fases de un SRBC

2.6.1 Método de indexación de los casos

Para la indexación hay algoritmos que se ocupan una parte en la clasificación para una rápida recuperación. El nombre del SGBD y la versión se pueden utilizar para aplicar como primer paso en la indexación de los casos. Además los casos almacenados tienen en su rasgo objetivo un valor de tres posibles, visto anteriormente, que puede permitir una clasificación ya dada por el mismo experto. Por lo que utilizando estas informaciones, se puede crear un mecanismo más directo para la indexación de los casos, tomando la misma intención de la política de particionamiento y del algoritmo de indexado en (Wong, Mauricio y Papa 2014).

2.6.2 Método de acceso y recuperación de los casos

El algoritmo de acceso y recuperación tiene una secuencia lógica de pasos como se describe en(Martínez y Pérez 2010). Basado en esta misma descripción se propone un método apoyado por el método de indexación descrito anteriormente.

Los casos se almacenan en una base de datos referencial, posibilitando la recuperación de los casos a través de consultas SQL. Como variables de entrada, son importantes el nombre (GestorBD) y la versión del SGBD (VersionBD), premisas indispensables del algoritmo.

2.6.3 Método de adaptación de los casos para la respuesta del sistema

El método de adaptación utiliza la lista de casos semejantes que devuelve el algoritmo anterior, para de este sacar el resultado más preciso. El método de adaptación utilizado es el nulo desde el punto de vista del tipo de transformación al valor del rasgo objetivo, debido que no hay necesidad de hacerle algún tipo de transformación. Para que este método sea aplicado, se necesita como entrada el caso actual que se está analizando. Para evaluar cada rasgo, se toma el caso más similar de la lista de casos determinada.

El primer paso en la adaptación es comparar la distancia entre los riesgos de cada parámetro X_n del caso a resolver con los k casos similares. Utilizar como solución a cada parámetro X_n del caso a resolver entre los k casos más similares el riesgo objetivo que más aparezca. Para la solución del caso, tomar el valor de riesgo objetivo del caso más similar (Rich 1998).

2.6.4 Modelo de aprendizaje del sistema basado en casos

El autoaprendizaje se logra con la incorporación de los nuevos problemas que se van solucionando a través de la decisión del experto. Para el autoaprendizaje se toma el valor obtenido

de la fórmula de semejanza. Si la semejanza sobrepasa el umbral, se toman como un caso totalmente nuevo y se introduce en la BC.

Para que el aprendizaje sea efectivo, que sea de modo ordenado. Por eso es importante en esta etapa, asegurar una correcta ubicación, la cual define el tiempo de respuesta. Por eso, se utiliza dentro del algoritmo de aprendizaje, el algoritmo de indexación especificado.

La entrada de un nuevo caso a la BC afecta en variables predeterminadas por otros algoritmos que no han tenido en cuenta el nuevo caso incorporado como son: el umbral y los pesos de los rasgos. Por lo que precisa en el mismo método de aprendizaje, reajustar las variables mencionadas y mejorar la respuesta de solución, con cada nuevo conocimiento incorporado.

2.7 La lógica difusa

La lógica difusa posibilita manipular las ambigüedades de valores e imprecisiones existentes en las evaluaciones del RSI.

Además se selecciona por los resultados obtenidos en el modelo publicado en (Zhang, Lu y Zhang 2011), donde se resuelve el cálculo del riesgo para la detección temprana de la influencia aviar e interviene el empleo de términos ambiguos. Teniendo como diferencia que en esta investigación solo se utiliza en ocasiones donde no se puede emplear el RBC.

La función miembro $\mu(R_n)$ definida para los números difusos es la triangular, donde a es el límite inferior y b el límite superior de un valor del RSI en un parámetro X_n. La variable m denota el valor modal del número difuso.

$$\mu(R_n) = \begin{cases} 0, \text{si } R_n \leq a \\ \dfrac{R_n - a}{m - a} \text{ si } R_n \in (a, m) \\ \dfrac{b - R_n}{b - m} \text{ si } R_n \in (m, b) \\ 0, \text{si } R_n > b \end{cases}$$

Los criterios son evaluados en un conjunto lingüístico de términos lingüísticos de 3 términos, simétricamente y uniformemente distribuidos con la sintaxis representada en la Figura 9 según los criterios de los expertos.

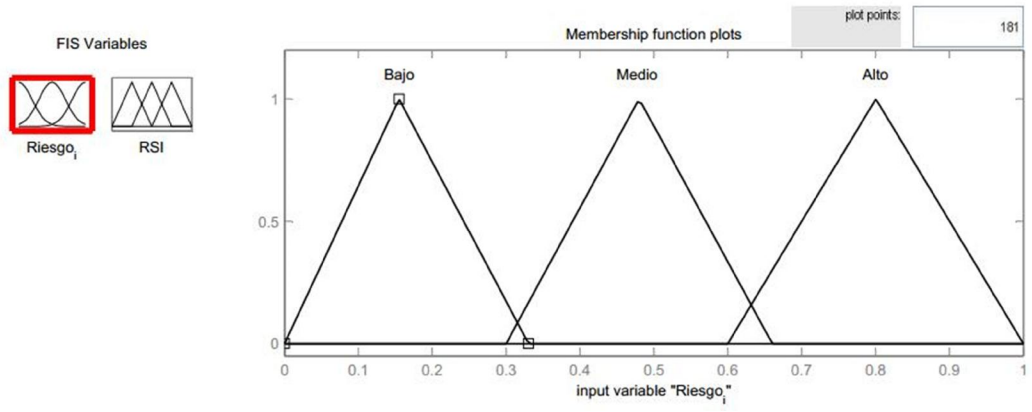


Figura 9: Conjunto de términos lingüísticos.

El uso de la lógica difusa permite una mejor exactitud en la evaluación del impacto y del riesgo de seguridad de la información en los SGBD ya que posibilita manejar los términos imprecisos del resultado de la evaluación del RSI.

Para la inferencia se utilizarán las siguientes reglas difusas debido a que los expertos encuestados, como se refleja en el resultado plasmado en el **anexo # 3**, no utilizan una metodología en específico de las reconocidas internacionalmente:

- Si al menos un Ri es “Alto” entonces el RSI es “Alto”.
- Si al menos un Ri es “Medio” y nunca Ri es “Alto” entonces el RSI es “Medio”.
- Si al menos un Ri es “Bajo” y nunca (Ri es “Alto” o Ri es “Medio”) entonces el RSI es “Bajo”

Dónde:

Ri: Es el riesgo calculado para el parámetro i, calculado en el componente anterior.

RSI: Es el riesgo de seguridad de la información del servidor auditado.

Por las características de las reglas, donde el antecedente y el consecuente de las reglas son conjuntos borrosos; se utiliza el sistema de inferencia de Mandani.

El método para la defusificación es el más grande de máximos, determinado por el resultado de la encuesta en el **anexo # 3.**

d) **Componente: Resultado de la evaluación del RSI**

Por último, el componente debe terminar con la evaluación de la auditoría de seguridad informática, donde debe reflejar el resultado final de la evaluación de RSI. Para esto, debe tener en cuenta, cuantos servidores hospedan el SGBD que se quiere auditar y efectuar el mismo procedimiento con todos y obtener la evaluación del RSI. Este tipo de situación sucede, cuando la

aplicación informática que se audita contiene más de un servidor donde tiene hospedado su base de datos.

Una vez obtenida la variable RSI por cada servidor, se tienen en cuanta un conjunto de reglas para determinar la evaluación final de RSI de la auditoría de seguridad informática para los SGBD donde va a ser plasmada en el informe redactado por el auditor o el equipo al frente de la auditoría.

Las reglas son:

- Si al menos un servidor es evaluado de Alto entonces la evaluación en el informe es Alto.
- Si al menos un servidor es evaluado de Medio y ningún otro evaluado de Alto entonces la evaluación en el informe es Medio.
- Si al menos un servidor es evaluado de Bajo y ningún otro evaluado de Alto y Medio entonces la evaluación en el informe es Bajo.

Las reglas plasmadas son extraídas en consenso con los expertos en auditorías de seguridad informática para los SGBD. Para determinar las reglas, se aplicó la encuesta ubicada en el **Anexo # 2**.

2.8 Conclusiones parciales

En este capítulo se abordaron una serie de aspectos correspondientes a la propuesta de solución llegándose a la siguiente conclusión:

- La identificación de los rasgos predictores y objetivos que componen los casos permitió el diseño de la Base de Casos.
- La forma de organización de la Base de Casos de manera jerárquica permitió eficiencia en la búsqueda.
- El uso de las técnicas de indexación permitió mejorar la eficiencia en la recuperación de los casos más semejantes.
- El uso de la lógica difusa permitió manipular las ambigüedades de los valores e imprecisiones en las evaluaciones del RSI.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA PARA LA EVALUACIÓN DEL RIESGO DE SEGURIDAD DE LA INFORMACIÓN EN LOS SGBD

3.1 Introducción

En este capítulo se abordarán los aspectos fundamentales relacionados con el diseño del sistema a desarrollar. Entre los elementos a destacar se encuentran el diagrama del modelo del dominio, mediante el cual se representan las clases conceptuales significativas del problema a resolver. Como vía para definir las futuras funcionalidades de la aplicación y qué usuarios podrán tener acceso a las mismas, se generaron los artefactos relacionados a la especificación de los requerimientos funcionales y no funcionales que deberá poseer el software; así como la especificación de los casos de uso del sistema. Como parte del diseño de la aplicación se definieron los estilos y patrones de arquitectura. Se presenta el diseño de la base de datos que se empleará para la persistencia de la información a manipular. A lo largo del capítulo se mostrarán los principales artefactos de ingeniería de software correspondientes a las funcionalidades o casos de uso más críticos.

3.2 Modelo del dominio

Una de las primeras actividades centrales de un ciclo de desarrollo consiste en crear un modelo conceptual para los casos de uso, en el cual se explican a sus creadores los conceptos significativos en un dominio del problema (Jacobson, Rumbaugh y Booch 2007).

El Modelo de Dominio también conocido como Modelo Conceptual es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Este representa clases conceptuales del dominio del problema, conceptos del mundo real y no de los componentes de software. Una clase conceptual también conocida como entidad puede ser una idea o un objeto físico (símbolo, definición y extensión).

3.2.1 Descripción de clases del modelo del dominio

La modelación del dominio constituye la herramienta fundamental para garantizar la comprensión y descripción de las clases o conceptos y sus relaciones más importantes dentro del contexto del problema. A continuación se presenta la descripción de los conceptos identificados.

CONCEPTO	DESCRIPCIÓN
Usuario	Define a los actores del sistema: supervisor y el

	auditor.
Monitoreo BD	Define la revisión o el seguimiento que se realiza a los SGBD
Servidores de BD	Se le aplica las listas de chequeo para dar un diagnóstico de la evaluación del riesgo del mismo
Lista de chequeo de seguridad	Está orientada a la revisión de la configuración de seguridad informática
Parámetros	Son los rasgos predictores por cada SGBD
Calculo del RSI	Cálculo del riesgo de cada parámetro
Evaluación del RSI	Alto, Medio o Bajo
Informe de diagnóstico	Informe generado sobre el resultado de la auditoría
Diagnóstico del servidor de BD	Resultado de la auditoría

Tabla 5: Descripción de las clases del modelo del domino.

3.2.2 Diagrama de clases del modelo del dominio

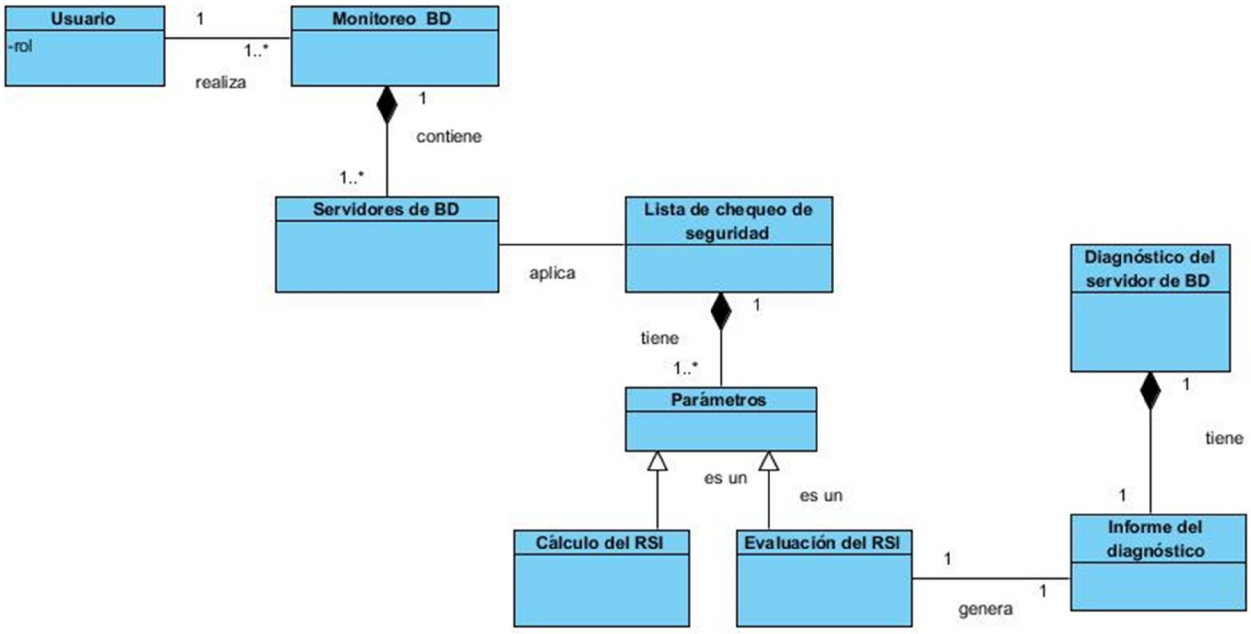


Figura 10: Modelo del dominio.

3.3 Especificación de los requisitos del software

Los requisitos del software constituyen las descripciones de los servicios y funcionalidades que brinda una aplicación informática, teniendo en cuenta además las restricciones operativas a las cuales están sujetas. De manera general, estos requisitos expresan las necesidades objetivas que

presentan los usuarios, ante un sistema que resuelve un problema en particular de un determinado dominio(SOMMERVILLE 2005).

Una vez definidos los conceptos principales relacionados con el dominio, se muestran a continuación los requerimientos funcionales y no funcionales, de acuerdo con el objetivo planteado al inicio de este trabajo.

3.3.1 Requisitos funcionales

Los requisitos funcionales (RF) son capacidades o condiciones que el sistema debe cumplir, o sea, acciones que debe realizar, deben ser comprensibles por los clientes, usuarios y desarrolladores, deben tener una sola interpretación y estar definidos en una forma medible y verificable. A continuación se definen los siguientes RF:

RF1. Mostrar casos: permite listar los casos existentes en la Base de Casos (BC):

- Evaluación del usuario
- Fecha
- Valor del riesgo
- Observación

RF 2. Ver contenido de un caso: permite ver el contenido de un caso de la BC:

- Evaluación del usuario
- Fecha
- Finalizada
- Riesgo
- Parámetro
- Evaluación
- Valor
- Mejoras sugeridas

RF3. Adicionar casos: permite al experto añadir nuevo conocimiento a la BC.

RF4. Deshabilitar caso: permite al experto deshabilitar un caso de la BC, es decir, que el mismo deje de ser caso aunque se encuentre físicamente en la BD.

RF5. Recuperar casos: permite recuperar los casos más semejantes al caso nuevo.

RF6. **Adaptar casos:** permite adaptar el nuevo caso al caso recuperado más semejante.

RF7. Aprendizaje de casos: permite la incorporación de nuevos casos que se van solucionando a la BC.

RF8. Mostrar listado de los n casos más similares: permite listar los n casos más similares para la recuperación del caso más semejante.

RF 9. Entorno de configuración: permite al experto configurar el entorno del Sistema Inteligente:

- Impacto del riesgo
- Valor de la evaluación
- Rango de valores para la evaluación del riesgo
- Umbral dado por el sistema

RF 9.1. Proponer umbral de similitud: permite al experto proponer un umbral en caso de no estar de acuerdo con el brindado por el sistema.

RF 10. Concluir matriz: permite proponer una evaluación inteligente a la auditoría actual:

- Datos de la matriz
- Elaborar observación
- Valores encontrados por matriz

RF 10.1. Modificar evaluación de un caso: permite al experto modificar la evaluación de un caso.

RF 11.Generar informe: permite generar el informe del resultado del diagnóstico realizado al servidor.

RF 11.1. Exportar informe: permite exportar el informe de diagnóstico para posterior análisis**.**

RF12. Indexar casos: permite mayor eficiencia en la recuperación del caso más semejante.

3.3.2 Requisitos no funcionales

Los requerimientos no funcionales (RNF) se presentan, en la mayoría de los casos, como las propiedades o cualidades que el sistema debe poseer. Sin embargo, desde otras aristas, pueden concebirse como las restricciones de las funcionalidades del sistema (PRESSMAN 2010). Independientemente del punto de vista, estos requisitos deben tratarse como las características que hacen al producto usable, rápido y/o confiable. Los requerimientos no funcionales aunque no aportan funcionalidades propiamente dichas dentro de la aplicación, son de vital importancia para

una puesta en marcha exitosa del software, y para lograr que este responda a las expectativas del usuario. A continuación se defines los siguientes RNF:

Apariencia o interfaz de usuario

RNF 1. El sistema brindará una interfaz amigable para sus usuarios, con predominio de color gris en diversas tonalidades y azul. Las letras serán legibles con formato Arial de 12 píxeles, de color negro.

RNF 2. Los errores que sean visibles al usuario, se marcarán de color rojo y deben señalar y mostrar un mensaje con las posibles causas.

Portabilidad

RNF 3. La aplicación podrá ser usada bajo cualquier distribución del sistema operativo Linux así como Windows.

Usabilidad

RNF 4. Se requiere que la aplicación pueda ser utilizada por usuarios con conocimientos básicos de seguridad informática.

Requerimientos de diseño e implementación

RNF 7. Se requiere el uso del paradigma de Programación Orientada a Objetos (POO).

RNF 8. Se requiere el uso de Spring e Hibernarte como marco de trabajo.

RNF 9. Se requiere PostgreSQL 9.1 como Sistema Gestor de base de Datos.

Eficiencia

RNF 10. Los tiempos de respuestas deben ser mejores significativamente a los realizados por los especialistas. Los cuales tardan varias horas para determinar la evaluación de riesgo.

3.4 Modelo de casos de uso del sistema

El modelo de casos de uso del sistema es un diagrama que explica de forma simbólica la interacción entre los actores y los casos de uso (CU) que representan las funcionalidades de la aplicación. Además, también son representadas las relaciones existentes entre casos de uso. El actor es una entidad externa del sistema que participa en la ejecución del caso de uso. Generalmente el actor estimula el sistema con eventos de entrada o recibe algo de él.

3.4.1 Definición de los actores del sistema

ACTORES	JUSTIFICACIÓN
Auditor	Este autor debe autenticarse en el sistema. Es el encargado de realizar las auditorías a los SGBD.
Supervisor	Es una especialización del autor Auditor, es el encargado supervisar el sistema inteligente ya que es el experto del mismo.

3.4.2 Diagrama de casos de uso del sistema

A continuación se presenta el diagrama de casos de uso del sistema donde se representa la relación existente entre los actores que deberán interactuar con el sistema y los casos de usos con los que interactúan cada uno de ellos, teniendo en cuenta los permisos y roles correspondiente.

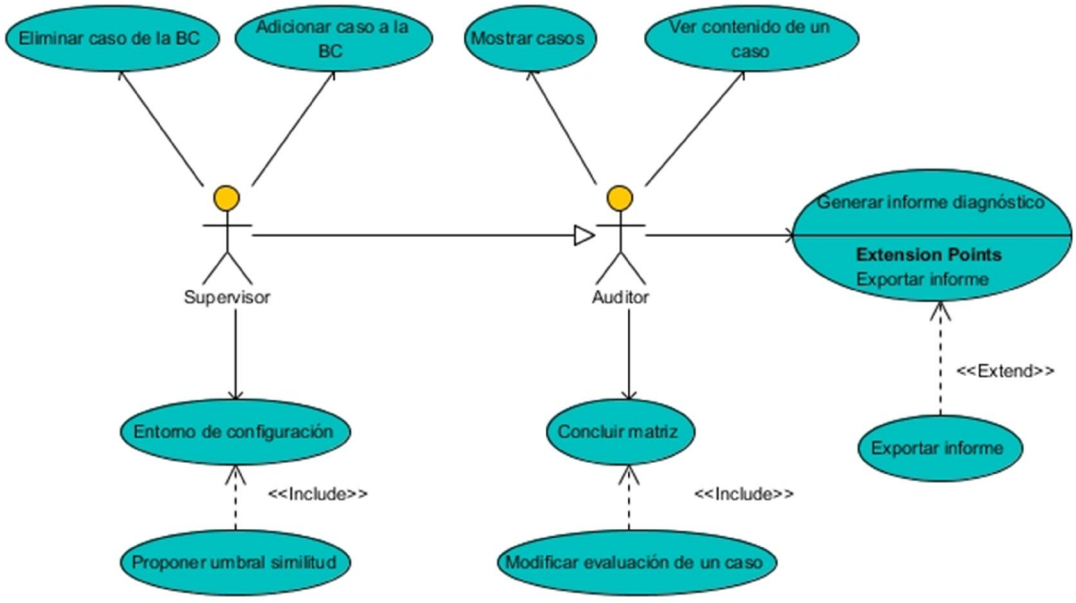


Figura 11: Diagrama de casos de uso del sistema.

En el diagrama anterior se encuentran representados en total 10 casos de uso y la jerarquía de actores que interactúan con cada uno de ellos.

3.4.3 Patrones de casos de uso utilizados

Los patrones de casos de uso permiten una mejor comprensión del comportamiento del sistema y generalmente son utilizados como plantillas que describen cómo deberían ser estructurados y organizados los casos de uso. A continuación se describen aquellos empleados en la representación de los diagramas de casos de uso descritos en la sección anterior.

Concordancia - Reutilización: Extrae una sub-secuencia de acciones que aparecen en diferentes lugares como parte del flujo de varios casos de uso. Este patrón incluye al menos tres casos de uso, uno de ellos llamado “sub-secuencia común”, que modela una secuencia de acciones que aparecen en múltiples casos de uso en el modelo; mientras que el resto describe el comportamiento del sistema que comparte la sub-secuencia común de acciones.

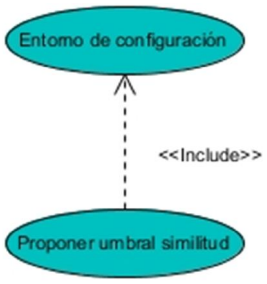


Figura 12: Ejemplo de uso del patrón de CU Concordancia – Reutilización.

Servicio Opcional – Adición: Consiste de dos casos de uso vinculados mediante una relación de extensión, donde uno de ellos modela una secuencia obligatoria de acciones y el segundo una secuencia adicional de acciones que permite extender el comportamiento del primero. Es importante señalar que el caso de uso extendido debe ser un caso de uso abstracto, lo que significa que no podrá ser instanciado de forma independiente.

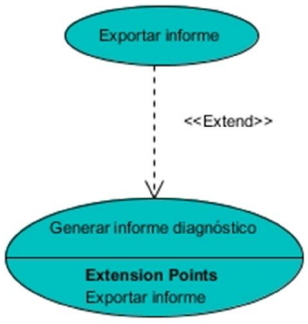


Figura 13: Ejemplo de uso del patrón de CU Servicio Opcional – Adición.

Múltiples Actores - Rol Común: Este patrón es utilizado cuando existe una sola entidad externa que interactúa con cada instancia de un caso de uso; por lo general esta situación ocurre cuando dos o más personas con roles diferentes dentro del negocio interactúan de la misma forma con un caso de uso.

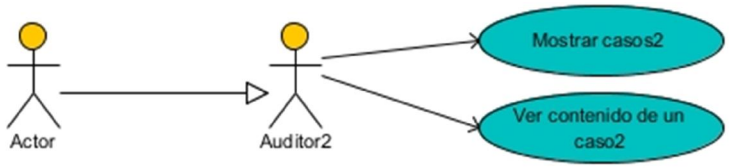


Figura 14: Ejemplo de uso del patrón de CU Múltiples Actores - Rol Común.

CRUD Parcial: El empleo de este patrón permite simplificar el tamaño y facilita el análisis del modelo mediante el agrupamiento de varias operaciones dentro de un caso de uso, cuando todas estas contribuyen al mismo valor del negocio, y existe alguna que por su complejidad, extensión o significación, debe modelarse como un caso de uso independiente.

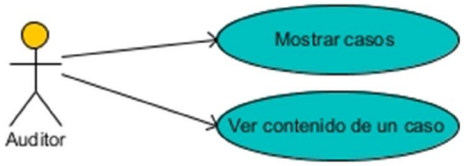


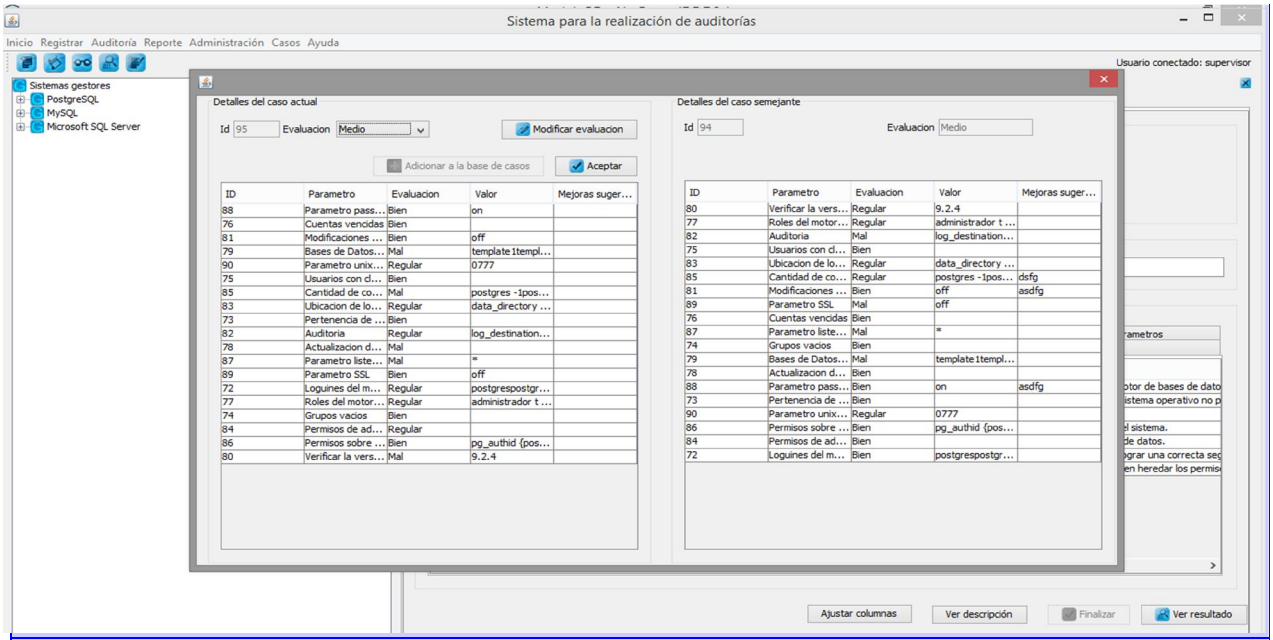
Figura 15: Ejemplo de uso del patrón de CU CRUD Parcial.

3.4.4 Especificación de casos de uso

A continuación se realiza la descripción los CU más crítico del sistema, el resto de las descripciones se encuentran en el **Anexo # 4.**

Descripción del CU "Concluir Matriz"

Objetivo	Permite proponer una evaluación inteligente a la auditoria actual.
Actores	Supervisor
Resumen	Este caso de uso es iniciado por el (Supervisor) cuando necesita realizar una acción sobre una matriz ya sea concluir matriz o modificar evaluación de la misma.
Complejidad	Alta
Prioridad	Alta
Precondiciones	El usuario ha accedido correctamente a la vista. ***Ver CU Concluir matriz***
Postcondiciones	Se realizó exitosamente cualquier acción desde la vista "Concluir matriz"
Prototipo	



Flujo de eventos		
Flujo básico		
	Acciones del actor	**Respuesta del sistema**
1	Selecciona de la interfaz la opción “**Concluir matriz**”.	
2		El sistema muestra la vista de matrices inconclusas
3	Selecciona la matriz que desea concluir Presiona el botón “**continuar**”.	
4		Muestra los detalles para editar la matriz abierta ✓ Datos de la matriz ✓ Elaborar observación ✓ Valores encontrados por matriz
5	Presiona el botón “**Evaluar**”.	
6		Muestra la vista con los detalles de la evaluación del caso actual y el caso semejante por el cual fue evaluado.
7	Presiona el botón “**Aceptar**”.	
8		Termina CU.
Sección 1: “Modificar evaluación”.		
Flujo básico “Modificar evaluación”.		
	Acciones de actor	**Respuesta del sistema**
1	Presiona el botón **“Evaluar”.**	

2		Muestra la vista con los detalles de la evaluación del caso actual y el caso semejante por el cual fue evaluado.
3	Selecciona la opción por la que desea evaluar y presiona el botón **"Modificar evaluación".**	
4		Muestra un mensaje indicando que la operación fue realizada correctamente.
5		Termina Sección 1.
Flujos alternos		
1. Error de selección		
Acciones del actor		**Respuesta del sistema**
1	Presiona el botón **"continuar"** sin haber seleccionado la matriz a editar de la lista de matrices inconclusas.	
2		Muestra un mensaje indicando que debe seleccionar al menos una matriz para el informe.
2. Cancelar		
1	Presiona el botón **"modificar evaluación".**	
2		Muestra un mensaje indicando si está seguro de realizar la operación.
3	Presiona el botón **"cancelar".**	
4		Regresa al flujo normal de eventos.

Tabla 6: Descripción del CU "Concluir matriz".

3.5 Arquitectura del sistema

3.5.1 Estilo arquitectónico

Los estilos arquitectónicos son el nivel de abstracción mayor para estructurar el sistema, la elección del mismo está dada por el tipo de aplicación que se vaya a desarrollar (S.Pressman 2006). En el presente trabajo se pretende desarrollar una aplicación de escritorio y para ello se seleccionó como estilo arquitectónico Envío- Retorno debido a que presenta una estructura fácil de modificar.

La arquitectura seleccionada para la aplicación es multicapas la cual tiene como ventajas la organización del módulo de diseño en capas, facilidad en la corrección y detección de errores. Permite un mejor entendimiento del sistema y reduce las dependencias, esto facilita que los componentes de una capa sólo puedan hacer referencia a componentes de capas inferiores o superiores (S.Pressman 2006).

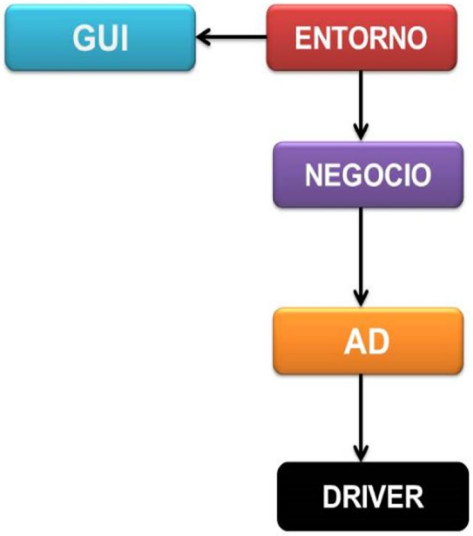


Figura 16: Arquitectura de la propuesta de solución.

La capa **GUI** es la capa que contiene las interfaces gráficas mediante las cuales los usuarios harán uso de la aplicación. La misma tiene un subpaquete UI (Interfaz Usuario) que almacena las interfaces gráficas, un segundo subpaquete Interface que contiene las clases interfaces que definen el comportamiento de las interfaces de usuario. Un tercer subpaquete llamada Implementación donde están 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. Es quien sabe que funcionalidad del negocio corresponde con determinada interfaz gráfica, así como orientar al negocio sobre qué acción realizó el usuario.

La capa de **Negocio** contiene toda la lógica de negocio, es quien implementa todas las clases y funciones de la aplicación. La misma tiene un subpaquete Interfaces que es donde están las clases interfaces que definen el comportamiento de las clases del negocio. Tiene además otro subpaquete Implementación donde están las clases que implementan las interfaces del negocio. Esta capa se comunica con la capa Acceso a Dato.

La capa **Acceso a Dato** maneja todo lo relacionado con el acceso a los datos, es una capa que abstrae al negocio del modo en que la aplicación se conecta a la base de datos, esto permite que la base de datos pueda cambiar en un futuro y no haya que hacer muchos cambios en el negocio. Esta capa también contiene un subpaquete Interfaces con sus clases de interfaces correspondientes y un subpaquete Implementación que tiene las clases que implementan las interfaces.

La capa **Drivers** contiene los driver de cada gestor de BD a auditar.

3.5.2 Patrones de diseño

Los patrones de diseño representan la descripción de un problema particular y recurrente, que aparece en contextos específicos, y presenta un esquema genérico demostrado con éxito para su solución; este último se especifica mediante la descripción de los componentes que la constituyen, sus responsabilidades y desarrollos, así como también la forma como estos colaboran entre sí (Larman 2004).

En el diseño de la propuesta de solución se tuvieron en cuenta los siguientes patrones GRASP (Patrones Generales de Software para Asignación de Responsabilidades), que describen los principios fundamentales de la asignación de responsabilidades a objetos:

Experto: este patrón plantea que se debe asignar una responsabilidad al experto en información, en otras palabras, a la clase que cuenta con los datos necesarios para cumplir la responsabilidad. De esta forma se conserva el encapsulamiento de la información, puesto que los objetos ejecutan las tareas que le corresponden de acuerdo a la información que poseen, lo que da lugar a sistemas más robustos y fáciles de mantener (Larman 2004). En el marco de la presente investigación siguiendo el patrón Experto en Información se le asignaron responsabilidades determinadas solamente a las clases que cuentan con la información necesaria para dar cumplimiento a las mismas, esto se evidencia en la Tabla 7. De esta forma se mantiene el encapsulamiento de la información, puesto que los objetos utilizan su propia información para llevar a cabo las tareas.

CLASE OBJETO	RESPONSABILIDAD
Persistencia	Permite obtener, adicionar, modificar y eliminar objetos de la BD.
IA	Conocer el estado en que se encuentra el proceso de actualización de la matriz de semejanza.

Tabla 7: Uso del patrón experto en información.

Creador: la instanciación de una clase es una de las actividades fundamentales en un sistema orientado a objetos. Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos, con lo que se logra menos dependencia y mayores oportunidades de reutilización de código (Larman 2004).

La clase "*ImpCasos*" es la encargada de implementar la lógica del negocio de los casos, pasada en el parámetro "id matriz", para esto crea instancias de la clase "TbDBaseCasosMatriz", esta crea los casos y los almacena en la BC. La 17 muestra el flujo de la creación de los objetos.

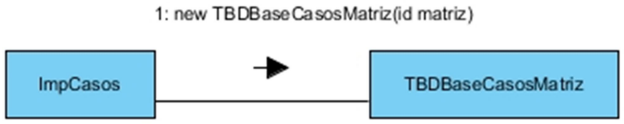


Figura 17: Creación de objetos en la clase "ImpCasos".

Controlador: este patrón tiene como objetivo asignar la responsabilidad a una clase de recibir o manejar un mensaje de evento del sistema generado por un actor externo, por lo general a través de una interfaz gráfica de usuario a la que accede un usuario para realizar ciertas operaciones en el sistema (Larman 2004).La utilización de este patrón se evidencia en la clase *"ImpAdicionarCasos"*, la misma se encarga de atender y ofrecer respuesta a cada una de las peticiones realizadas por el usuario mediante la interfaz Adicionar Casos.

Bajo acoplamiento: El acoplamiento es una medida de la fuerza con que un elemento está conectado, tiene conocimiento o confía en otros elementos. El objetivo de este patrón consiste en mantener un bajo nivel de dependencia de otros elementos, por lo que constituye un principio que debe estar presente en todas las decisiones de diseño con lo que se reduce el impacto de los cambios (Larman 2004).

Spring plantea el uso de interfaces para evitar un alto acoplamiento, por lo que en el sistema este patrón se ve claramente reflejado, en las capas de negocio y acceso a datos no existe dependencia a las implementaciones de sus atributos si no a las interfaces que definen el comportamiento de estos.

Alta cohesión: En el diseño orientado a objetos, la cohesión es una medida de la fuerza con la que se relacionan y del grado de focalización de las responsabilidades de un elemento (clase o subsistema). Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas, que colaboran entre sí y con otros objetos para simplificar su trabajo. Una clase con alta cohesión es relativamente fácil de mantener, entender y reutilizar (Larman 2004).

Anteriormente se representó el uso del patrón experto, el cual distribuye el comportamiento entre las clases que cuentan con la información, alentando con ello definiciones de clases sencillas y más cohesivas que son más fáciles de comprender y mantener. Así se brinda soporte a la alta cohesión.

3.6 Modelo del diseño

El modelo de diseño es aquel que se encarga de describir la realización de los casos de uso del sistema, y se utiliza como medio de abstracción del modelo de implementación y el código fuente del software. Su objetivo fundamental es transmitir, a través de la representación mediante diagramas, una comprensión en profundidad de los aspectos relacionados con los requerimientos no funcionales y restricciones concernientes a los lenguajes de programación.

A continuación se relaciona los diagramas de clases del diseño correspondiente a los CU descritos anteriormente. El resto de los diagramas de clases del diseño se encuentran en el **Anexo # 5**.

3.6.1 Diagrama de clase del diseño: CU Concluir matriz

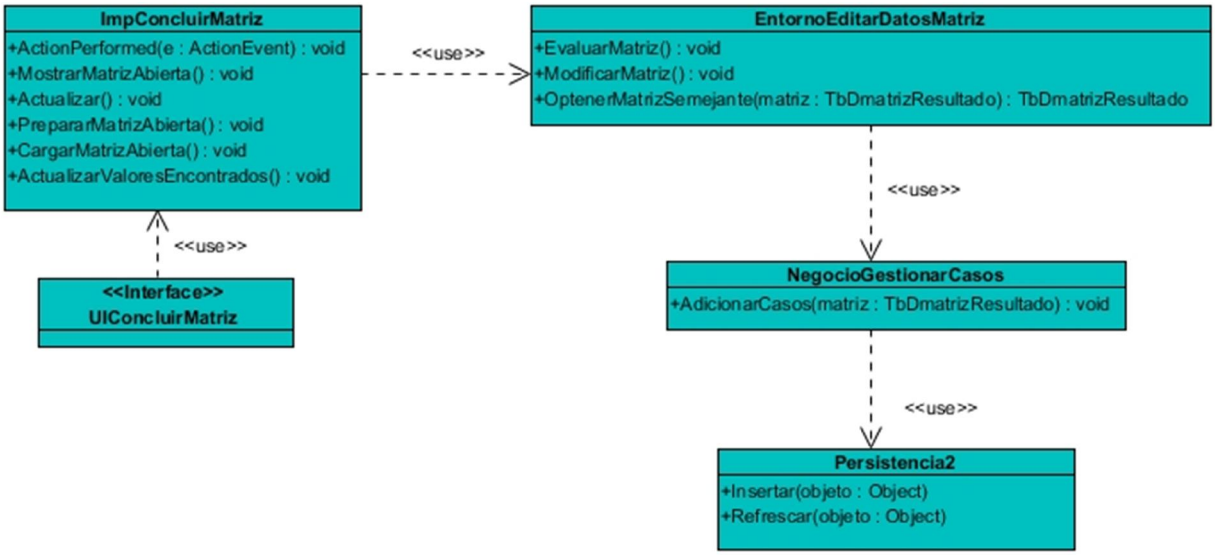


Figura 18: Diagrama de clase del diseño. CU Concluir matriz.

3.7 Diseño de la base de datos

La persistencia de la información es uno de los requerimientos claves a tener en cuenta a la hora de diseñar la arquitectura de una aplicación informática. En este sentido, es de vital importancia la realización de un diseño de base de datos coherente con las necesidades puntuales del sistema a desarrollar. El objetivo del diseño de una base de datos, es la representación de las clases

entidades y sus relaciones, que permitan almacenar la información con un mínimo de redundancia, pero que a la vez faciliten la recuperación de la información.

La aplicación que se propone hace uso de una base de datos, puesto que se hace imprescindible el almacenamiento de una gran cantidad de información relacionada

3.7.1 Modelo físico de datos

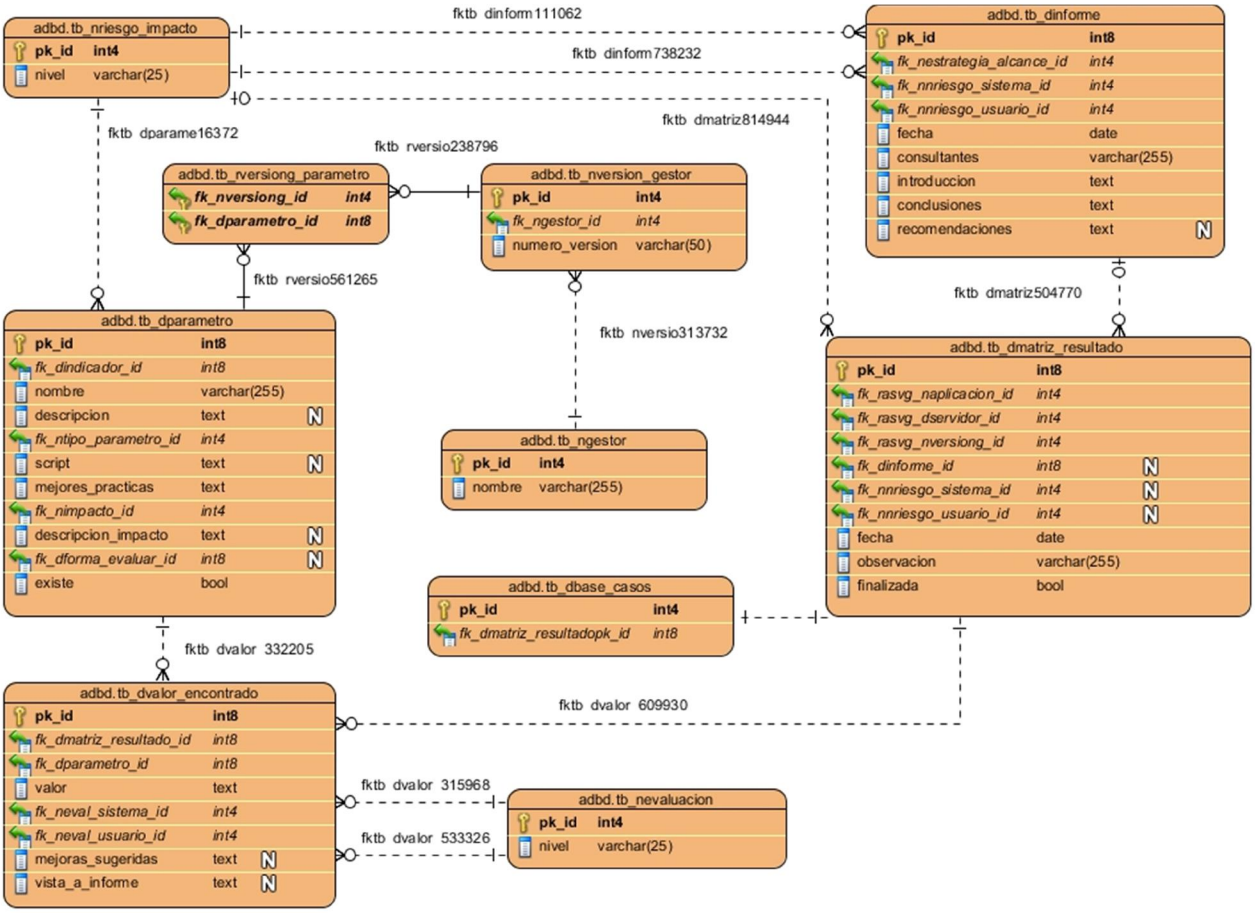


Figura 19: Modelo físico de los datos.

3.8 Modelo despliegue

Un modelo de despliegue representa la relación física que se establece entre los distintos componentes o nodos que describen la topología de un sistema. Detalla las capacidades de red, las especificaciones del servidor, los requisitos de hardware y la información relacionada con la forma en la que los componentes se comunicarán a lo largo de la infraestructura del sistema.

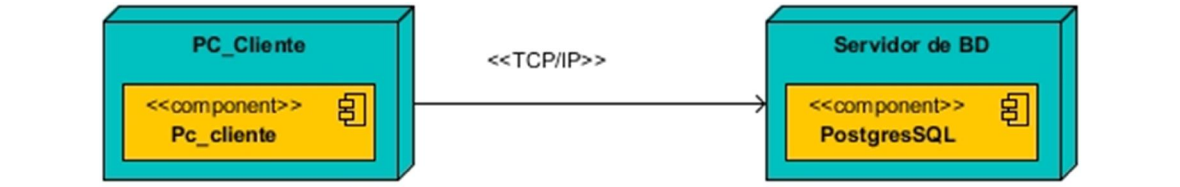


Figura: Diagrama de Despliegue de la propuesta.

PC del Cliente: En esta computadora se ejecuta el subsistema de evaluación de riesgo de seguridad de la información.

PC Servidor de BD: Estará el SGBD Oracle, MySQL, Postgres y SQServer que será objeto de una evaluación de riesgo de seguridad de la información cada vez que se realice una auditoría.

Conector: La comunicación se realiza mediante el protocolo TCP/IP porque (Pérez 2012):

- El conjunto TCP/IP está diseñado para enrutar.
- Tiene un grado muy elevado de fiabilidad.
- Es adecuado para redes grandes y medianas, así como en redes empresariales.
- Se utiliza a nivel mundial para conectarse a Internet y a los servidores web. Es compatible con las herramientas estándar para analizar el funcionamiento de la red.

3.9 Conclusiones parciales

En este capítulo se abordaron una serie de aspectos correspondientes al análisis y diseño del subsistema de evaluación del RSI a los SGBD llegándose a la siguiente conclusión:

- La representación y descripción de los artefactos generados garantizaron un mejor entendimiento de los flujos de trabajos presentes en el proceso de evaluación del riesgo.
- La especificación de los requisitos funcionales y no funcionales del sistema, dieron paso a una mejor comprensión, por parte de los autores, de los resultados que se pretenden obtener de una manera precisa y sirvieron de guía para la codificación del sistema.
- La definición de la arquitectura y los patrones de diseño utilizados, permitieron establecer las bases para fomentar la reutilización y las buenas prácticas de programación entre los desarrolladores durante la fase de implementación, así como disminuir el impacto de los cambios futuros en el código fuente.
- La elaboración del diagrama de despliegue permitió identificar la disposición física de los artefactos del subsistema a desarrollarse.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA DEL SISTEMA PARA LA EVALUACIÓN DEL RIESGO DE SEGURIDAD DE LA INFORMACIÓN EN LOS SGBD

4.1 Introducción

La implementación del sistema es una de las fases imprescindibles dentro del proceso de desarrollo de software. Esta fase comprende la materialización, en forma de código, de todos los artefactos, descripciones y arquitectura propuestos en la etapa de análisis y diseño; con el objetivo de conformar el producto final requerido por el cliente.

Aparejado al proceso de implementación, el software que se construye debe ser sometido a determinadas pruebas que corroboren la correspondencia entre el producto y los requisitos definidos en las etapas anteriores. A esta etapa se le conoce como validación del sistema y en ella, pueden realizarse diferentes tipos de pruebas en función de los objetivos de las mismas.

4.2 Modelo de componentes que integran la solución informática

El modelo de componentes representa la forma en que es estructurado un sistema informático atendiendo a las diferentes partes que lo componen. Partiendo de este punto, (SOMMERVILLE 2005) puntualiza que cada componente debe ser tratado como una unidad de composición independiente e indispensable dentro de un sistema, y que puede contraer relaciones de dependencia con otros componentes. Algunos ejemplos de componentes físicos lo constituyen los archivos, módulos, librerías, ejecutables, binarios, entre otros.

4.2.1 Diagrama de componentes

Diagrama que se utiliza para modelar la vista estática del sistema, puede mostrar un conjunto de elementos tales como componentes, subsistemas de implementación e interfaces. A continuación se muestra el diagrama de componentes correspondiente a la propuesta a desarrollar.

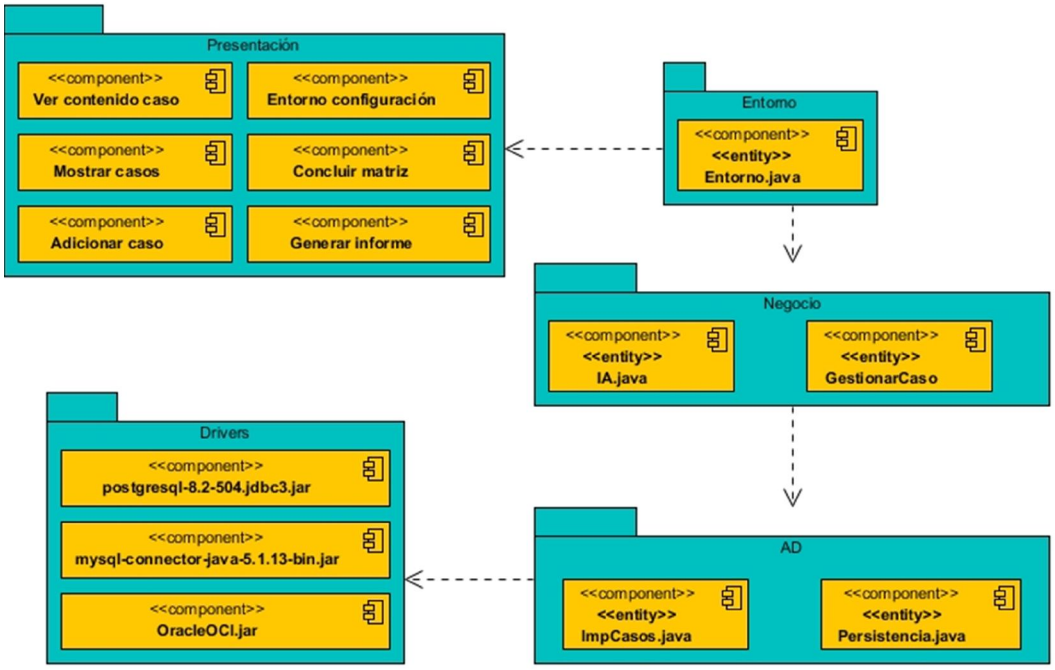


Figura 20: Diagrama de componentes de la propuesta de solución.

Paquete Presentación: Muestra varios componentes asociados a cada una de las funcionalidades. Estos componentes representan por cada funcionalidad las clases de implementación, los formularios.

Paquete Entorno: Están los componentes entidades los cuales son encargados de establecer la comunicación entre ambos paquetes, además sabe que funcionalidad del negocio corresponde con determinada interfaz gráfica, existe un entorno para cada interfaz.

Paquete Negocio: Están los componentes entidades que son los encargados de implementar todas las funciones de la aplicación.

Paquete AD: Están los componentes entidades que manejarán todo lo relacionado con el acceso a los datos.

Paquete Drivers: Este contiene los driver de los gestores Postgres, Oracle, MySQL, SQLServer.

4.3 Validación del subsistema de evaluación del RSI en los SGBD.

A continuación se detallan los tipos de pruebas de software aplicados al sistema implementado. Las mismas persiguen como objetivo fundamental, la detección de las no conformidades respecto a las funcionalidades de la aplicación, las vulnerabilidades que atentan contra la seguridad de la

información que se manipula con el software, la medición del grado de usabilidad de las funcionalidades implementadas, así como también la correcta integración entre los diferentes componentes de la arquitectura del sistema.

4.3.1 Pruebas funcionales

Las pruebas funcionales son aquellas que se aplican a un software determinado, con el objetivo de validar que las funcionalidades implementadas funcionen de acuerdo a las especificaciones de los requisitos definidos con anterioridad. Para la ejecución de este tipo de pruebas, suelen emplearse dos métodos fundamentales: el método de Caja Blanca y el método de Caja Negra. El primero se centra en las pruebas al código de las aplicaciones; mientras que el segundo permite a los probadores enfocar su atención en el funcionamiento de la interfaz, a través del análisis de los datos de entrada y los de salida.

En este epígrafe se exponen los aspectos concernientes a las pruebas funcionales realizadas utilizando el método de Caja Negra, a partir de los casos de prueba diseñados, empleando la técnica Partición Equivalente y el método de Caja Blanca, empleando la técnica Camino Básico.

Técnica de prueba de Caja Negra: Diseño de los casos de prueba.

A continuación se muestra un fragmento del diseño del caso de prueba basado en caso de uso del CU “Concluir matriz”. El resto de los diseños de casos de prueba se pueden encontrar en el **Anexo # 6**.

Caso de uso: Concluir matriz.
Condiciones de ejecución:

- El usuario ha sido autenticado en el sistema.
- El usuario autenticado es un Supervisor o Auditor.
- El usuario ha accedido a la vista. ***Ver CU Concluir matriz.***

Escenario	Descripción	V1	V2	V3	Respuesta del sistema	Flujo central
EC 1.1 Concluir matriz de forma exitosa.	El sistema propone una evaluación a la auditoría actual.	NA	NA	NA	El sistema evalúa el nuevo caso forma correcta.	1-El usuario se autentica en el sistema con rol de Auditor o Supervisor. 2-El usuario selecciona de la barra de herramientas el botón "**Auditoría**". 3- El usuario selecciona la opción "**Concluir matriz**" 4-El sistema muestra la lista de matrices inconclusas. 5-El usuario selecciona la matriz a evaluar y presiona el botón "**Continuar**". 6- El sistema muestra los detalles de la matriz. 7-El usuario presiona el botón "**Evaluar**". 8-El sistema muestra los detalles de la evaluación del caso. 9-El usuario presiona el botón "**Aceptar**". 10-El usuario presiona el botón "**Finalizar**".
EC 1.2 Modificar evaluación	El sistema modifica la evaluación de la auditoría.	NA	NA	NA	El sistema permite modificar la evaluación de la auditoria y muestra el mensaje "**Operación realizada correctamente**".	
EC 1.3 Cancelar	El sistema cancela la opción modificar evaluación.	NA	NA	NA	El sistema pregunta si está seguro de realizar dicha operación y cancela.	

Técnica de prueba de Caja Blanca: Camino Básico.

La prueba de la ruta básica o camino básico, es una técnica de caja blanca. Permite que se obtenga una medida de complejidad lógica de un diseño procedimental y que use esta medida como guía para definir un conjunto básico de rutas de ejecución(Pressman 2005).

La complejidad se denota como V (G) y se calcula de las siguientes formas:

Tabla 8: Fragmento del caso de prueba basado en el CU "Concluir Matriz**"**.

- V (G)= A-N+2, siendo A la cantidad de aristas y N la cantidad de nodos del grafo.
- V (G)= P+1, siendo P la cantidad de nodos predicados.
- V (G)= R, siendo R la cantidad de regiones que contiene el grafo.

La complejidad calculada es correcta, si al calcularse por las tres vías anteriormente expuestas, el valor resultante es el mismo.

A continuación se muestran dos ejemplos de pruebas realizadas con la técnica Camino Básico:

Ejemplo # 1:

V (G)= 15-13+2 = 5

V (G)= 4+1=5

VG= 5

```
public float Distancia(TbDmatrizResultado m1, TbDmatrizResultado m2) throws Exception {
    if (!m1.equals(m2)) {
        float Wj = m1.getTbNriesgoImpactoByFkNnriesgoUsuarioId().getValor();
        List<TbDparametro> parametros1 = new LinkedList<>();
        List<TbDparametro> parametros2 = new LinkedList<>();
        obtenerValoresEncontrados2(parametros1, parametros2, m1, m2);
        float distancia = 0;
        float WjAux = 0;
        for (int i = 0; i < parametros1.size(); i++) {
            float Xnr = 0;
            float Xni = 0;
            if (parametros1.get(i) != null) {
                Xnr = parametros1.get(i).getTbNriesgoImpacto().getValor();
            }
            if (parametros2.get(i) != null) {
                Xni = parametros2.get(i).getTbNriesgoImpacto().getValor();
            }
            WjAux += Wj;
            distancia += Wj * (Xnr - Xni);
        }
        distancia = distancia / WjAux;
        distancia = (float) (Math.rint(distancia * 1000) / 1000);
        return distancia >= 0 ? distancia : distancia * -1;
    }
    return 0;
}
```

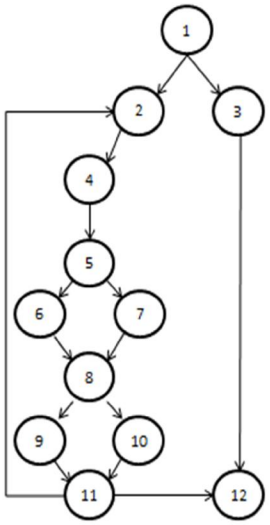


Figura 21: Distancia.

Ejemplo # 2

V (G)= 12-10+2 = 4

V (G)= 3+1=4

V (G)= 4

```
public void Umbral() throws Exception {
    setAccion("Umbral");
    setUmbralesFinalizados(false);
    LinkedList<Object> listCasos = persistencia.ObtenerTodos(TbDbaseCasosMatriz.class, TipoBusqueda.TODOS);
    int m = listCasos.size();
    float sumatoria = 0;
    for (int i = 0; i < listCasos.size() - 1; i++) {
        setPorciento(i * 100 / listCasos.size());
        for (int j = i + 1; j < listCasos.size(); j++) {
            sumatoria += Distancia(matrixDadoCaso((TbDbaseCasosMatriz) listCasos.get(i
                    matrixDadoCaso((TbDbaseCasosMatriz) listCasos.get(j)));
        }
    }
    float denominador = (m * (m - 1));
    float umbral = (2 / denominador) * sumatoria;
    umbral = (float) Math.rint(umbral * 1000) / 1000;
    LinkedList<Object> listUmbral = persistencia.ObtenerTodos(TbDumbral.class, TipoBusqueda.TODOS);
    TbDumbral u = (TbDumbral) listUmbral.get(0);
    if (umbral < 0) {
        umbral = umbral * -1;
    }
    u.setUmbral_system(umbral);
    persistencia.Modificar(u);
    persistencia.Refrescar(u);
    setUmbralesFinalizados(true);
}
```

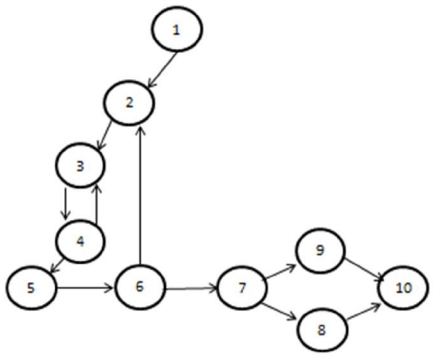


Figura 22: Umbral de similitud.

Las pruebas de caja blanca se le realizaron a las funcionalidades del subsistema mediante el método del camino básico para verificar el correcto funcionamiento del flujo de los datos.

Resultados de las pruebas funcionales

Para llevar a cabo la detección de las no conformidades presentes en la aplicación desarrollada, se realizaron 3 iteraciones de pruebas funcionales. En la siguiente tabla se muestran los resultados obtenidos en cada iteración de prueba al subsistema para evaluar el riesgo de seguridad de la información en los SGBD, así como la corrección de cada uno de los errores.

No conformidades	1ra Iteración	2da Iteración	3ra Iteración
Detectadas	10	8	5
Resueltas	6	5	5
Pendientes	4	3	0

Tabla 9: Cantidad de no conformidades por cada iteración de las pruebas funcionales.

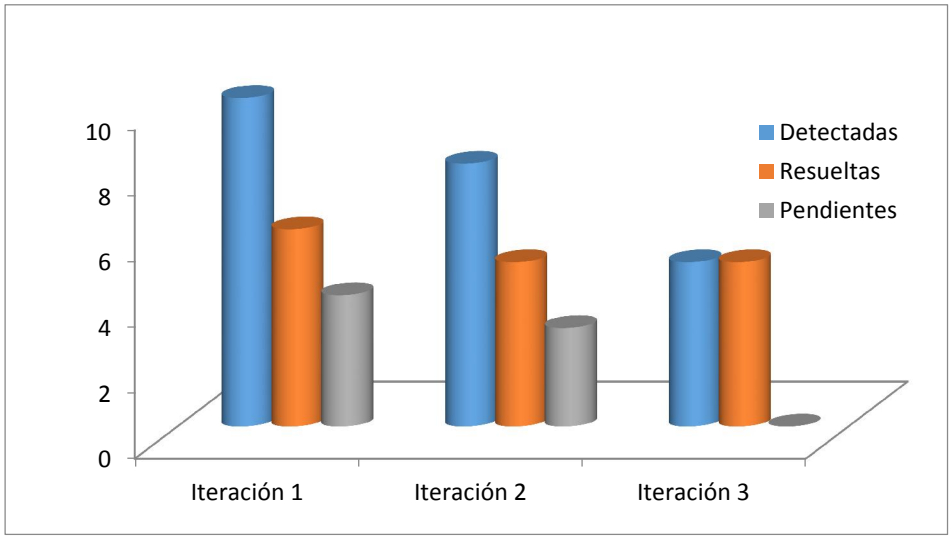


Figura 23: Comportamiento de las no conformidades por cada iteración de las pruebas funcionales.

4.3.2 Pruebas de integración

Las pruebas de integración son definidas para verificar el correcto ensamblaje entre los distintos módulos que conforman un sistema informático. Las mismas validan que estos componentes realmente funcionan juntos, son llamados correctamente y además, transfieren los datos correctos en el tiempo preciso y por las vías de comunicación establecidas(Sommerville 2005).

Una vez realizadas las pruebas funcionales a cada componente interno de manera independiente, y verificado que las funcionalidades implementadas se corresponden de acuerdo a los requisitos funcionales y no funcionales establecidos; se pudo comprobar el correcto funcionamiento de los componentes mediante el estudio del flujo de datos entre ellos. Posterior a estas pruebas, se hace necesaria la realización de pruebas de integración, con la finalidad de validar la compatibilidad y el funcionamiento de las interfaces que comunican las diferentes partes que componen la solución informática.

Para la realización de las pruebas de integración se llevaron a cabo diferentes acciones, a continuación se mencionan las más fundamentales:

- Comprobación del funcionamiento del enlace entre el SASGBD y el algoritmo basado en casos para evaluar el riesgo de seguridad de la información a los SGBD implementado.
- Verificación de la conexión entre el sistema y la base de datos en PostgreSQL.
- Correspondencia entre los datos manejados por los usuarios en el sistema y los existentes en las los sevidores de ETECSA.

La ejecución de las pruebas de integración permitió verificar el trabajo conjunto de los componentes del algoritmo en cuestión, junto a los componentes que conforman el SASGBD.

4.3.3 Tiempo de respuesta del subsistema

Las técnicas de indexación garantizan una recuperación eficiente de los casos, entendiéndose por eficiencia tanto los aspectos relacionados con el tiempo de recuperación como la garantía de que ningún caso relevante quede fuera de la búsqueda. Los índices son una estructura de datos que se mantiene en memoria y que puede ser recorrida de forma rápida para evitar la búsqueda directa en el contenido de la BC. Normalmente, los índices estarán constituidos por combinaciones de los descriptores más importantes de los casos para permitir diferenciar unos casos de otros de manera poco costosa (Althoff 2001).

Con el propósito de evaluar el tiempo de respuesta del sistema, se diseñó un experimento para probar las mejoras del comportamiento del RBC en el proceso de recuperación de los casos más semejantes con respecto al uso o no de técnicas de indexación para mejorar la eficiencia.

Se tomó una muestra de 4 casos para realizar el proceso de recuperación con un total de 100 casos existentes en la BC.

Nota: En la Tabla 10 el tiempo de recuperación está expresado en milisegundos (ms).

Nota: El tiempo de respuesta se calculó hallando el promedio de recuperación de los casos.

Nro	Caso nuevo	Caso más cercano	Distancia más cercana	Evaluación del caso más cercano	Caso encontrado	Tiempo de recuperación	Tiempo de recuperación usando indexación	Evaluación final
1	178	160	0.126	Bajo	160	534700	4329	Bajo
2	175	172	0.053	Medio	172	492621	4016	Medio
3	**171**	**172**	**0.052**	**Medio**	**172**	**442311**	**3950**	**Medio**
4	162	161	0.063	Medio	161	450002	4063	Medio

Tabla 10: Resultados experimentales del tiempo de recuperación con el uso o no de indexación.

El tiempo de respuesta del subsistema depende del tiempo de recuperación del SBC, como se pudo observar en la tabla anterior el uso de la indexación mejora en eficiencia el tiempo de respuesta con un promedio de 4758 ms.

Tiempo de respuesta del subsistema: 4089,5 ms.

Con el experimento realizado se puede llegar a la conclusión que la solución desarrollada da cumplimiento al objetivo planteado en la investigación; apoyando la toma de decisiones en cuanto a agilidad del proceso y ayuda a los auditores en el análisis de los riegos. Según la entrevista que se encuentra en el **anexo # 7** realizada a uno de los auditores de ETECSA, no se puede dar un

tiempo exacto de cuanto demore un proceso de evaluación actualmente, ya que depende de varios factores. Sin embargo, se puede afirmar que el tiempo de respuesta de la solución desarrollada agiliza considerablemente la evaluación del riego, evitando el tiempo y esfuerzo de los auditores a la hora de evaluar las auditorías.

4.4 Conclusiones parciales

En este capítulo se abordaron una serie de aspectos correspondientes a la implementación y validación del subsistema de evaluación de RSI llegándose a la siguiente conclusión:

- La representación y descripción del diagrama de componentes permitió visualizar con más facilidad la estructura general del subsistema.

- La ejecución de pruebas al subsistema permitió detectar las deficiencias presentes, subsanarlas en el menor tiempo posible y ofrecer una aplicación con mayor calidad y eficiencia.

- La aplicación del método experimental aportó elementos sustanciales que demostraron como el subsistema de evaluación del RSI apoya el proceso de toma de decisiones, en cuanto al tiempo de respuesta y la ayuda en el análisis de los riesgos a los auditores.

CONCLUSIONES

Una vez completada la presente investigación, se puede concluir que:

- Existen una serie de Sistemas Inteligentes para el análisis del RSI, los cuales no están enfocados en el área de la seguridad de la información para los SGBD; definiéndose una propuesta de solución de acuerdo a las necesidades existentes.
- La metodología RUP y el uso de las tecnologías y herramientas seleccionadas, permitieron analizar y describir los subprocesos que se debían ejecutar, concretando así las características que debía tener el subsistema a desarrollar.
- Se modelaron los artefactos que contribuyeron al diseño de la propuesta de solución posibilitando un mayor soporte a la implementación de los requisitos y la disminución del impacto ante futuras modificaciones en la aplicación.
- La implementación subsistema de evaluación del RSI en los SGBD utilizando las herramientas y tecnologías estudiadas, orientada por las fases de la metodología RUP permitió solucionar los problemas existentes planteados en la problemática de la presente investigación.
- La evaluación de las pruebas de software realizadas permitió erradicar las insuficiencias detectadas en el subsistema desarrollado, logrando así el cumplimiento del objetivo trazado en la investigación.

RECOMENDACIONES

Una vez concluida la investigación y el desarrollo de la propuesta de solución, los autores del presente trabajo recomiendan:

- Implementar algoritmos para el mantenimiento automático de la BC.
- Utilizar técnicas de programación paralela al momento de actualizar la matriz de semejanza para mejorar el rendimiento del sistema.

REFERENCIAS BIBLIOGRÁFICAS

1 ALTHOFF, K.-D., 2001. *Case-Based Reasoning. Handbook of Software Engineering and Knowledge Engineering. Institute for Experimental Software Engineering (IESE).* [en línea]. 2001. S.l.: s.n. Disponible en: http://www.iis.uni-hildesheim.de/files/staff/althoff/Publications/althoff-CBR-update.pdf.

2 BAHAMONTES, A. 2013. Auditoría de Seguridad Informática. [en línea]. [Consulta: 4 noviembre 2014]. Disponible en: .http://www.antpji.com/antpji2013/index.php/articulos2/111-auditoria-de-seguridad-informatica.

3 BELMONTE FERNÁNDEZ, O., 2005. *Introducción al lenguaje de programación Java. [En línea] 2005.* 2005. S.l.: s.n.

4 BRODER, J.F. y TUCKER, G. 2011. Risk analysis and the security survey. [en línea]. [Consulta: 16 noviembre 2014]. Disponible en: http://www.google.com.cu/books?hl=es&lr=&id=fLmgIGT18jIC&oi=fnd&pg=PP1&dq=risk+assessment%2Bformula%2Bsecurity&ots=q1K-pmlGAY&sig=lwkAVW1G2jr7wkBjlR4dQftaQ2k&redir_esc=y#v=onepage&q=risk%20assessment%2Bformula%2Bsecurity&f=false.

5 CALDERÓN, A. y DÁMARIS, S., 2007. *Metodologías Ágiles. Trujillo, Perú : s.n.,*. 2007. S.l.: s.n.

6 CHOU, T. 2011. *Information Assurance and Security Technologies for Risk Assessment and Threat Management: Advances* [en línea]. S.l.: s.n. Disponible en: http://books.google.com.cu/books?id=2zlES3JYNpoC&pg=PA243&dq=cramm+risk+assessment+tool&hl=es&sa=X&ei=S3_oUrCuI8abrAGe9IG4Cw&ved=0CGwQ6AEwCA#v=onepage&q=cramm%20risk%20assessment%20tool&f=fals.

7 CIS, C. f. l.2013. CIS Benchmarks. [en línea]. [Consulta: 16 noviembre 2014]. Disponible en: http://benchmarks.cisecurity.org/downloads/multiform/.

8 CIS, C. f. l. 2014. CIS Security Benchmarks Membership. Security Expertise Made Affordable. [en línea]. [Consulta: 16 noviembre 2014]. Disponible en: http://benchmarks.cisecurity.org/membership/.

9 DR. PEÑA AYALA, A., 2006. *Sistemas basados en Conocimiento: Una Base para su Concepción y Desarrollo. INSTITUTO POLITÉCNICO NACIONAL - México* -. 2006. S.l.: s.n.

10 FLORES, M. y LOURDES, S. 2012. Desarrollando aplicaciones informáticas con el Proceso de Desarrollo Unificado (RUP). [en línea]. [Consulta: 4 noviembre 2014]. Disponible en: . http://www.utvm.edu.mx/OrganoInformativo/orgJul07/RUP.htm.

11 JACOBSON, RUMBAUGH y BOOCH, 2007. *El Lenguaje Unificado de desarrollo (UML), Manual de Referencia.* [en línea]. 2007. S.l.: s.n. [Consulta: 4 noviembre 2014]. Disponible en: http://ingenieriasoftware2011.files.wordpress.com/2011/07/el-lenguaje-unificado-de-modelado-manual-de-referencia.pdf.

REFERENCIAS BIBLIOGRÁFICAS

12 JOHNSON, R.E. y FOOTE, B. 1988. *Desingning reausable classes. Journal of Object-Oriented Programming, Capítulo 1(Tema 2): Pag 22-35.* S.l.: s.n.

13 LÍO y GÁLVEZ, Daniel 1998. . *Curso de sistemas basados en el conocimiento. Universidad de las Villas, Cuba: s.n.* S.l.: s.n.

14 LOZA, A. y CARGNELUTTI, P., 2012. *Auditoría a Bases de Datos*. 2012. S.l.: s.n.

15 MARTÍNEZ, I.G. y PÉREZ, R.E.B. 2010. *Un Modelo para la Toma de Decisiones usando Razonamiento Basado en Casos en condiciones de Incertidumbre. Tesis Doctoral, Universidad Central Marta ,Abreu Santa Clara.* S.l.: s.n.

16 MARTÍNEZ SÁNCHEZ, Ms.N., FERREIRA LORENZO, D.G. y GARCÍA LORENZO, D.M.M. 2008. El Razonamiento Basado en Casos en el ámbito de la Enseñanza/Aprendizaje. 25-32, s.l. : Revista de Informática Educativa y Medios Audiovisuales. Vol. V. 1667-8338.

17 MONTERO, P.C. 2015. Proceso RUP - Análisis. Fases y Flujos del RUP. - Documents. [en línea]. [Consulta: 11 mayo 2015]. Disponible en: http://myslide.es/documents/proceso-rup-analisis-prof-cesar-luza-montero-fases-y-flujos-del-rup.html.

18 MORA, A., 2008. *Modelos de Organización de Memoria.* 2008. S.l.: s.n.

19 (MS-ISAC), M.-S.I.S. 2010. Cyber Security: Risk Management A Non-Technical Guide Essential for Business Managers Office Managers Operations Managers 2013. [en línea]. [Consulta: 10 noviembre 2014]. Disponible en: http://msisac.cisecurity.org/resources/guides/documents/Risk-Management-Guide.pdf#page=3&zoom=auto,179,0.

20 PALISADE 2014. El futuro en una hoja de trabajo. [en línea]. [Consulta: 20 octubre 2014]. Disponible en: http://www.palisade-lta.com/risk/.

21 PELTIER, T.R. 2001. *Information Security Risk Analysis (pp. 296). Retrieved from* [en línea]. S.l.: s.n. Disponible en: http://books.google.com.cu/books?id=O0_fO2Xvp98C&pg=PA231&dq=risk+assessment+security+%2B+expert+system&hl=es&sa=X&ei=Tr_qUq3tF6H7yAHc_ICQAg&ved=0CEQQ6AEwAg#v=onepage&q=risk%20assessment%20security%20%2B%20expert%20system&f=false.

22 PEÑA SÁNCHEZ, D., 2011. *Deducción de distribuciones: el método de Monte Carlo», en Fundamentos de Estadística. Madrid: Alianza Editorial. ISBN 84-206-8696-4.* 2011. S.l.: s.n.

23 PEREZ, L. 2012. Modelo de protocolo: Modelo TCP/IP. [en línea]. [Consulta: 27 marzo 2015]. Disponible en: http://liliperez160.blogspot.com/2012/10/modelo-tcpip.html.

24 POSTGRESQL, 2000. *El equipo de desarrollo de. Postgresql. [En línea] .* [en línea]. 2000. S.l.: s.n. [Consulta: 12 noviembre 2014]. Disponible en: http://www.postgresql.org.

25 PRESSMAN, R., 2005. *Estrategias de pruebas del software pag.305.* 2005. S.l.: s.n.

REFERENCIAS BIBLIOGRÁFICAS

26 PRESSMAN, R.S., 2010. *Ingeniería de software. 7 ed. McGraw-Hill Interamericana de España S.L, 959 p. ISBN 9786071503145.* 2010. S.l.: s.n.

27 QUIGLEY, M. 2008. Encyclopedia of information ethics and security Version details - Trove. [En línea]. [en línea]. [Consulta: 3 noviembre 2014]. Disponible en: http://trove.nla.gov.au/work/26382737?Selectedversion=NBD42065776.

28 RAMAKANTH, D. y VINOD, K., 2011. *SQL Injection - Database Attack Revolution And Prevention. Journal of International Commercial Law and Technology, 6 (4), 2 24 - 231.* [en línea]. 2011. S.l.: s.n. Disponible en: http://www.jiclt.com/index.php/jiclt/article/view/141/139.

29 RICH, E., 1998. *Inteligencia Artificial. Edit. Gustavo Gili, S.A., Barcelona.* 1998. S.l.: s.n.

30 RUIZ, A.J., 2011. *myEchelon: Un sistema de Auditoría de Seguridad Informática Avanzado bajo GNU/Linux. Titulación de Ingeniero en Informática, Universidad de Almería, Almería, España.* [en línea]. 2011. S.l.: s.n. [Consulta: 13 noviembre 2014]. Disponible en: http://www.adminso.es/images/9/9c/Alberto_PFC.pdf.

31 SOMMERVILLE, I., 2005. *Ingeniería del software. 7 ed. Pearson Educación, 712 p. ISBN 84-7829-074-5.* 2005. S.l.: s.n.

32 SOMMERVILLE, I., 2005. *Ingeniería del Software. , Madrid: Pearson Educación.* 2005. S.l.: s.n.

33 S.PRESSMAN, R. 2006. *Diseño arquitectónico. Ediciones del Pressman. Pressman_6ta_edicion Capitulo 10, Pag 281, Parte 1.* S.l.: s.n.

34 UNIV NACIONAL COLOMBIA, 2003. *Universidad Nacional de Colombia. Sistemas Basados en el Conocimiento.* [en línea]. 2003. S.l.: s.n. [Consulta: 10 noviembre 2014]. Disponible en: http://dis.unal.edu.co/profesores/lucas/iartificial/IA0011l.pdf.

35 WONG, L.R., MAURICIO, D. y PAPA, E.A. 2014. Un modelo de Razonamiento Basado en Casos para la captación de requisitos en el desarrollo de proyectos de software. Revista de investigación de Sistemas e Informática. , pp. 27- 36.

36 ZHANG, J., LU, J. y ZHANG, G., 2011. *A Hybrid Knowledge-based Risk Prediction Method Using Fuzzy Logic and CBR for Avian Influenza Early Warning. Journal of Multiple-Valued Logic & Soft Computing, 17(4), 363-386.* [en línea]. 2011. S.l.: s.n. Disponible en: http://web.ebscohost.com/ehost/pdfviewer/pdfviewer?sid=83c1714d-fa11-46ea-a0f3-c955d04587e0%40sessionmgr14&vid=1&hid=19.

37 ZUKOWSKI, J. 2003. *Programación JAVA 2 JSE 1.4. [ed.] Ignacio Luca de Tena. Madrid: ANAYA Multimedia, 2003*. S.l.: s.n.

BIBLIOGRAFÍA

1 **BELMONTE FERNÁNDEZ**, 2005. *Introducción al lenguaje de programación Java. [En línea] 2005*. S.l.: s.n.

2 **DR. PEÑA AYALA**, 2006. *Sistemas basados en Conocimiento: Una Base para su Concepción y Desarrollo. INSTITUTO POLITÉCNICO NACIONAL - México* -. 2006. S.l.: s.n.

3 **JACOBSON, RUMBAUGH y BOOCH**, 2007. *El Lenguaje Unificado de desarrollo (UML), Manual de Referencia.* [En línea]. 2007. S.l.: s.n. [Consulta: 4 noviembre 2014]. Disponible en: http://ingenieriasoftware2011.files.wordpress.com/2011/07/el-lenguaje-unificado-de-modelado-manual-de-referencia.pdf.

4 **JOHNSON, R.E. y FOOTE, B.** 1988. *Desingning reausable classes. Journal of Object-Oriented Programming, Capítulo 1(Tema 2): Pag 22-35.* S.l.: s.n.

5 **LÍO y GÁLVEZ, Daniel** 1998. . *Curso de sistemas basados en el conocimiento. Universidad de las Villas, Cuba: s.n.* S.l.: s.n.

6 **LOZA, A. y CARGNELUTTI, P**., 2012. *Auditoría a Bases de Datos*. 2012. S.l.: s.n.

7 **MARTÍNEZ, I.G. y PÉREZ, R.E.B**. 2010. *Un Modelo para la Toma de Decisiones usando Razonamiento Basado en Casos en condiciones de Incertidumbre. Tesis Doctoral, Universidad Central Marta , Abreu Santa Clara.* S.l.: s.n.

8 **MARTÍNEZ SÁNCHEZ, Ms.N., FERREIRA LORENZO, D.G. y GARCÍA LORENZO, D.M.M**. 2008. El Razonamiento Basado en Casos en el ámbito de la Enseñanza/Aprendizaje. 25-32, s.l. : Revista de Informática Educativa y Medios Audiovisuales. Vol. V. 1667-8338.

9 **PEÑA SÁNCHEZ, D.**, 2011. *Deducción de distribuciones: el método de Monte Carlo», en Fundamentos de Estadística. Madrid: Alianza Editorial. ISBN 84-206-8696-4.* 2011. S.l.: s.n.

10 **PRESSMAN, R.S**., 2010. *Ingeniería de software. 7 ed. McGraw-Hill Interamericana de España S.L, 959 p. ISBN 9786071503145.* 2010. S.l.: s.n.

11 **RICH, E**., 1998. *Inteligencia Artificial. Edit. Gustavo Gili, S.A., Barcelona.* 1998. S.l.: s.n.

12 **RUIZ, A.J.,** 2011. *MyEchelon: Un sistema de Auditoría de Seguridad Informática Avanzado bajo GNU/Linux. Titulación de Ingeniero en Informática, Universidad de Almería, Almería, España.* [En línea]. 2011. S.l.: s.n. [Consulta: 13 noviembre 2014]. Disponible en: http://www.adminso.es/images/9/9c/Alberto_PFC.pdf.

13 **SOMMERVILLE,** I., 2005. *Ingeniería del software. 7 ed. Pearson Educación, 712 p. ISBN 84-7829-074-5.* 2005. S.l.: s.n.

14 **UNIV NACIONAL COLOMBIA,** 2003. *Universidad Nacional de Colombia. Sistemas Basados en el Conocimiento.* [En línea]. 2003. S.l.: s.n. [Consulta: 10 noviembre 2014]. Disponible en: http://dis.unal.edu.co/profesores/lucas/iartificial/IA0011l.pdf.

BIBLIOGRAFÍA

15 **WONG, L.R., MAURICIO, D. y PAPA, E.A**. 2014. Un modelo de Razonamiento Basado en Casos para la captación de requisitos en el desarrollo de proyectos de software. Revista de investigación de Sistemas e Informática. , pp. 27- 36.

16 **ZUKOWSKI, J**. 2003. *Programación JAVA 2 JSE 1.4. [ed.] Ignacio Luca de Tena. Madrid: ANAYA Multimedia, 2003*. S.l.: s.n.