INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

EVALUACIÓN DEL RIESGO INFORMÁTICO PONDERADO Y SU IMPLANTACIÓN EN EL CAMPUS ESTADO DE MÉXICO

TESIS QUE PARA OPTAR EL GRADO DE
MAESTRO EN CIENCIAS COMPUTACIONALES
PRESENTA

MANUEL DAMIÁN GUERRA FARÍAS

Asesor: Dr. ROBERTO GÓMEZ CÁRDENAS

Comité de tesis: Dr. JESÚS VÁZQUEZ GÓMEZ.
Mtro. RICARDO GONZALEZ VARGAS.
Dr. FRANCISCO CAMARGO SANTACRUZ

Jurado:		
	Dr. JESÚS VÁZQUEZ GÓMEZ.	Presidente
	Dr. FRANCISCO CAMARGO SANTACRUZ	Secretario
	Dr. ROBERTO GÓMEZ CÁRDENAS	Vocal
	Mtro. RICARDO GONZALEZ VARGAS.	Vocal

Atizapán de Zaragoza, Edo. Méx., octubre de 2006.

CONTENIDO

1. INTRODUCCIÓN 6
2. EL RIESGO 9
2.1 CONCEPTOS BÁSICOS 10
2.1.1 ACTIVOS 10
2.1.2 RIESGO 11
2.1.3 AMENAZA 12
2.1.4 VULNERABILIDAD 12
2.1.5 IMPACTO 13
2.1.6 SALVAGUARDA 13
2.1.7 INTERRELACIÓN DE LOS CONCEPTOS BÁSICOS 13
2.2 ADMINISTRACIÓN DE RIESGOS 14
2.2.1 ENFOQUES EN LA ADMINISTRACIÓN DE RIESGOS 14
2.2.1.1 Enfoque reactivo 14
2.2.1.2 Enfoque proactivo 15
2.2.2 MANEJO DEL RIESGO 16
2.2.2.1 Evitar el riesgo 16
2.2.2.2 Transferencia de responsabilidad 16
2.2.2.3 Indemnización 16
2.2.2.4 Mitigación 17
2.2.2.5 Retención 17
2.3 METODOLOGÍAS DE ADMINISTRACIÓN DE RIESGOS 17
2.3.1 AS/NZS 4360 18
2.3.2 COBIT 21
2.3.2.1 Evaluación de Riesgos del Negocio 22
2.3.2.2 Enfoque de Evaluación de Riesgos 22
2.3.2.3 Identificación de Riesgos 22
2.3.2.4 Medición de Riesgos 22
2.3.2.5 Plan de Acción contra Riesgos 23
2.3.2.6 Aceptación de Riesgos 23
2.3.2.7 Selección de Garantías o Protecciones 23
2.3.2.8 Compromiso con el Análisis de Riesgos 23
2.3.3 CRAMM 23
2.3.3.1 Identificar los activos, amenazas y vulnerabilidades 24
2.3.3.2 Identificar los posibles impactos a los activos 25
2.3.3.3 Evaluación de los activos y medición de las amenazas y vulnerabilidades. 25
2.3.3.4 Calcular el riesgo 25
2.3.3.5 Revisión de los resultados 25
2.3.4 FERMA 26
2.3.4.1 Valoración de riesgos 26
2.3.4.2 Tratamiento de riesgos 27
2.3.5 MAGERIT 28
2.3.5.1 Planeación 28
2.3.5.2 Análisis de riesgos 28
2.3.5.3 Gestión de riesgos 29
2.3.6 OCTAVE 29
2.3.6.2 Identificar las vulnerabilidades de la infraestructura 30

2.3.6.3 Desarrollar las estrategias y planes de seguridad. 30
2.3.7 MICROSOFT 31
2.3.7.1 Evaluación del riesgo 31
2.3.7.2 Apoyo a la toma de decisiones 31
2.3.7.3 Implementación de controles 32
2.3.7.4 Medición de la efectividad del programa 32
2.3.8 NIST 32
2.3.8.1 Caracterización del sistema 33
2.3.8.2 Identificación de amenazas 33
2.3.8.3 Identificación de vulnerabilidades 33
2.3.8.4 Análisis de Control 34
2.3.8.5 Determinación de la probabilidad 34
2.3.8.6 Análisis del impacto 34
2.3.8.7 Determinación del riesgo 35
2.3.8.8 Recomendaciones de controles 36
2.3.8.9 Documentación de los resultados 36
2.3.9 RISK TARGETING 36
2.3.9.1 Entender el negocio 36
2.3.9.2 Modelar los procesos críticos. 37
2.3.9.3 Determinar los componentes verdaderamente críticos. 37
2.4 FACTORES A CONSIDERAR AL ADOPTAR UNA METODOLOGÍA DE ADMINISTRACIÓN DE RIESGOS. 37
2.5 CONCLUSIONES 39
3. EVALUACIÓN DE RIESGOS 41
3.1 EVALUACIÓN CUANTITATIVA 41
3.1.1 VALORACIÓN DE ACTIVOS 42
3.1.1.1 El valor global del activo en la organización. 42
3.1.1.2 La repercusión financiera inmediata de la pérdida del activo. 43
3.1.1.3 La repercusión de negocios indirecta de la pérdida del activo. 43
3.1.2 EXPECTATIVA DE PÉRDIDA SIMPLE 43
3.1.3 LA FRECUENCIA ANUAL 44
3.1.4 EXPECTATIVA DE PÉRDIDA ANUAL 44
3.1.5 DETERMINACIÓN DEL COSTO DE LOS CONTROLES 44
3.1.6 RENDIMIENTO DE LA INVERSIÓN EN SEGURIDAD 45
3.2 EVALUACIÓN CUALITATIVA 45
3.3 COMPARACIÓN ENTRE EL ENFOQUE CUANTITATIVO Y EL ENFOQUE CUALITATIVO 46
3.4 ENFOQUE HIBRIDO 47
3.5 CONCLUSIONES 48
4. NUEVA PROPUESTA DE EVALUACIÓN DE RIESGOS 49
4.1 CRITERIOS A CONSIDERAR EN LA EVALUACIÓN DE RIESGOS. 51
4.1.1 NIVEL DE AMENAZA 52
4.1.1.1 Factores del nivel de amenaza 52
4.1.1.2 Salvaguardas del nivel de amenaza 53
4.1.2 NIVEL DE VULNERABILIDAD 53
4.1.2.1 Factores del nivel de vulnerabilidad 54
4.1.2.2 Salvaguardas del nivel de vulnerabilidad 55
4.1.3 VALOR DEL ACTIVO 56
4.1.3.1 Factores del valor del activo 56

4.1.3.2 Salvaguardas del valor del activo 59
4.2 EVALUACIÓN DE CRITERIOS Y FACTORES 59
4.2.1 EVALUACIONES DE VERDADERO/FALSO 60
4.2.2 EVALUACIONES PROPORCIONALES AL MÁXIMO DEL MISMO FACTOR. . 60
4.2.3 EVALUACIONES POR SELECCIÓN, APLICANDO UNA ESCALA. 60
4.2.4 EVALUACIONES POR PORCENTAJE DIRECTO. 61
4.3 EJEMPLO DE EVALUACIÓN DE RIESGOS. 61
4.4 VENTAJAS Y DESVENTAJAS DE LA EVALUACIÓN DE RIESGOS PONDERADA. 65
4.5 CONCLUSIONES 65
5. SISTEMA DE DEFINICIÓN Y EVALUACIÓN DE RIESGOS INFORMÁTICOS PARA EL ITESM CEM (SDERI) 67
5.1 Análisis de SDERI 67
5.1.1 DEFINICIÓN DEL NIVEL DE RIESGO. 67
5.1.2 EVALUACIÓN DEL NIVEL DE RIESGO DE EQUIPOS. 68
5.1.3 EVALUACIÓN AUTOMÁTICA DE FACTORES. 68
5.2 DISEÑO DE SDERI 69
5.2.1 DISEÑO DE LA BASE DE DATOS. 69
5.3 IMPLEMENTACIÓN DE SDERI 70
5.4 SDERI Y OTROS SISTEMAS 72
5.4.1 OSSIM 73
5.4.1.1 Arquitectura de OSSIM 73
5.4.1.2 Integración de OSSIM con SDERI 77
5.4.1.3 Integración de ISS con SDERI 80
5.5 Conclusión 84
6. RESULTADOS DEL SISTEMA SDERI EN EL ITESM-CEM 85
7. CONCLUSIONES SOBRE EVALUACIÓN DE RIESGOS INFORMÁTICOS 88
Bibliografía 89
Anexo A. Guía de instalación para OSSIM en Debian GNU/Linux 92
Anexo B. Manual de Usuario del Sistema para la Definición y Evaluación de Riesgos Informáticos (SDERI). 96
Anexo C. Código Fuente del Sistema para la Definición y Evaluación de Riesgos Informáticos (SDERI). 110

LISTA DE FIGURAS

Fig. 1 - Relación entre activos, amenazas y riesgos 13
Fig. 2 - Administración de Riesgos - AS/NZS 4360 19
Fig. 3 - Dominios de COBIT 21
Fig. 4 - El riesgo según CRAMM 24
Fig. 5 - Impactos potenciales. Combinación de amenaza, vulnerabilidad e impacto- CRAMM . 25
Fig. 6 - Administración de riesgos- FERMA 26
Fig. 7 - Administración de Riesgos según MAGERIT 28
Fig. 8 - Proceso de administración de riesgos de OCTAVE 30
Fig. 9 - Administración de riesgos según Microsoft 31
Fig. 10 - Evaluación de riesgos- NIST 33
Fig. 11 - Riesgo residual VS Inversión en salvaguardas. 38
Fig. 12 - El nivel del riesgo 52

Fig. 13 - Diagrama E-R de SDERI 70
Fig. 14 - Arquitectura de OSSIM 73
Fig. 15 - Sensores y colectores de OSSIM 74
Fig. 16 - El motor de correlación de OSSIM 75
Fig. 17 - Pizarrón del riesgo de OSSIM 77
Fig. 18 - Diagrama E-R de OSSIM 78
Fig. 19 - Diagrama E-R del factor nivel de ataque e intromisión basado en OSSIM 79
Fig. 20 - Diagrama ER del factor de cantidad de puertos abiertos 80
Fig. 21 - Diagrama E-R de ISS 81
Fig. 22 - Tabla Hosts de ISS 82
Fig. 23 - Vista (de BD) de el ultimo Job por IP de ISS 82
Fig. 24 - Vistas y tablas para crear la vista del conteo de vulnerabilidades por nivel por IP de ISS 83
Fig. 25 - Vista (de BD) del numero de vulnerabilidades por nivel por IP de ISS 83

LISTA DE TABLAS

Tabla 1 - Nivel de Riesgo- CRAMM 25
Tabla 2 - Niveles de probabilidad de amenazas. NIST 34
Tabla 3 - Niveles del impacto. NIST 34
Tabla 4 - Matriz del nivel de riesgo. NIST 35
Tabla 5 - Niveles de riesgo -NIST 35
Tabla 6 - Niveles de madurez de la administración de riesgos de seguridad 38
Tabla 7 - Ventajas y desventajas de la evaluación de los enfoques cualitativo y cuantitativo 46
Tabla 8 - Nivel de la información según su confidencialidad. 57
Tabla 9 - Nivel de seguridad de los usuarios según la política de seguridad del ITESM-CEM 57
Tabla 10 - Nivel de dependencia de los servicios 58
Tabla 11 - Nivel de criticidad según el dueño 58
Tabla 12 - Escala del factor nivel de la información 60
Tabla 13 - Ejemplo de ponderación de criterios y factores 61
Tabla 14 - Datos para la evaluación de riesgo 63
Tabla 15 - Ejemplo de evaluación de riesgo. 64
Tabla 16 - Nivel de riesgo de los servidores del ITESM-CEM 85
Tabla 17 - Evaluación del nivel de risgo de wdb-rep 86

1. INTRODUCCIÓN

El Tecnológico de Monterrey es un sistema universitario que tiene como misión formar personas comprometidas con el desarrollo de su comunidad y que sean competitivas internacionalmente en su área de conocimiento. La Misión incluye hacer investigación y extensión relevantes para el desarrollo sostenible del país.

Para lograr esta misión el Tec ha desarrollado e implantado una estrategia educativa, la cual esta sustentada en una plataforma tecnológica. Para el ITESM los sistemas informáticos juegan un papel imprescindible tanto operativamente como estratégicamente. Todos los procesos esenciales para la continuidad del ITESM están soportados por sistemas.

El departamento de Seguridad Computacional, tiene el compromiso de minimizar los riesgos informáticos a los cuales esta expuesto el instituto, es decir, debe buscar la forma de preservar la integridad, disponibilidad y confidencialidad de la información que se encuentre en los distintos equipos de cómputo de los cuales depende la organización para su operación. Para cumplir con lo anterior es necesario determinar el nivel de seguridad de los equipos de cómputo del Campus a fin de enfocarse en aquellos que presenten un mayor riesgo para la institución.

El constante cambio en el mundo de las redes y computadoras nos obliga a estar frecuentemente evaluando el nivel de seguridad de los equipos. Si en este momento, se evaluara el nivel de seguridad de un equipo muy probablemente dentro de una semana, esta evaluación solo nos serviría para darnos una simple idea del nuevo nivel de seguridad, debido a que al equipo se le pudo instalar nuevas aplicaciones, parches o servicios, por otro lado también es posible que se hayan encontrado nuevas vulnerabilidades que afecten al equipo o haya sido infectado por algún virus informático. Por ello es necesario que la evaluación de seguridad sea constante.

Sin embargo, no se cuenta con ninguna herramienta que ayude a determinar objetivamente cuales son los activos que presentan mayor riesgo para la organización y por ende deban de atenderse primero. Ante esta problemática se desarrollo una herramienta para evaluar en tiempo

real el nivel de seguridad de los servidores del ITESM-CEM en base a la actividad de la red y los objetivos del negocio.

Es evidente que se deben asegurar estos sistemas. Sin embargo, los recursos para asegurar estos sistemas son finitos y es necesario establecer un mecanismo que nos permita ser eficientes en este proceso. Es decir, hay que determinar objetivamente cuales son los recursos informáticos que presentan un mayor riesgo para el instituto y en base en ello tomar acciones que permitan minimizar las consecuencias de un incidente. En otras palabras, se necesita evaluar el nivel de seguridad de los diferentes equipos de los cuales depende la institución para su operación.

Esta valoración del nivel de seguridad, no es tan simple como podríamos pensar, pues a fin de que realmente sea de utilidad esta debe estar conformada por al menos los siguientes criterios:

- Nivel de importancia del equipo para la institución, es decir, que tanto podría afectar que el equipo sea comprometido.
- Nivel de exposición del equipo, es decir, que tan fácil es que el equipo sea comprometido.
- Nivel de ataque hacia el equipo, es decir, que tanto se intenta comprometer al equipo.

Debido al dinamismo que pueden presentar cada uno de estos criterios es necesario estar evaluándolos constantemente, a fin de que la información proporcionada sea realmente de utilidad, la evaluación a realizar deberá ser en tiempo real.

Al evaluar y conjuntar estos criterios podemos tener un excelente indicador que le permita al Departamento de Seguridad Computacional enfocarse en aquellos activos que presentan el mayor riesgo para la organización objeto de estudio.

A fin de delimitar el alcance de la presente tesis y proporcionar información relevante para el ITESM-CEM, se ha delimitado el enfoque en analizar únicamente la red de servidores del Campus Estado de México. Razón por la cuál, solo se evaluará en tiempo real el nivel de seguridad la red de servidores del ITESM-CEM en base a la actividad de la red y los objetivos del negocio.

Actualmente el ITESM-CEM cuenta con 55 servidores. Estos servidores proveen servicios de muy distinta índole e importancia, van desde aquellos que son vitales para la institución como es el caso del correo del personal directivo, hasta otros que no lo son tanto como el bazar electrónico. Aunado a ello, los servicios se proporcionan a distintas escalas que van desde un servicio para un área determinada, para el Campus, para la Rectoría o para el sistema. También es importante tomar en cuenta que se atiende a distintos públicos, entre ellos: alumnos, ex-alumnos (Ex–A–Tecs), padres de familia, profesores, personal administrativo, prospectos o público en general.

Es imprescindible contar un mecanismo que permita identificar objetivamente aquellos activos que presenten un mayor nivel de riesgo a fin de poder jerarquizar las acciones para minimizar estos riesgos de manera eficiente.

La evaluación del nivel de seguridad en tiempo real de la red de servidores del ITESM-CEM traerá muchos beneficios. Permitirá al departamento de Seguridad Computacional ser eficientes, pues dará un indicador de donde exactamente hay que enfocar los esfuerzos. Disminuirá el costo del monitoreo de equipos, ya que al automatizar la tarea, permitirá disminuir la demanda de mano de obra calificada que realice la tarea. Dará información para la toma de decisiones, debido a que permitirá conocer que tan vulnerables, atacados e importantes son los sistemas, así como la evolución de estos. Permitirá tener un ambiente más seguro, al verificar que los sistemas críticos para el ITESM-CEM mantienen un nivel de riesgo aceptable y en caso contrario poder notificar oportunamente para que los administradores puedan tomar las medidas correctivas correspondientes. Además, en el largo plazo este proyecto permitirá cambiar el paradigma de una seguridad reactiva a una proactiva.

La tesis se encuentra organizada de la siguiente forma:

En el capítulo dos se aborda el tema de riesgo, en el cuál se revisan conceptos del riesgo, las amenazas y las vulnerabilidades así como su interrelación; los distintos enfoques que se tienen actualmente en la administración de riesgos y las formas alternativas de manejar el riesgo, así como las metodologías existentes para la administración de riesgo y los factores que ayudan a determinar la adopción de una metodología.

En el capítulo tres se enfoca en la forma de evaluar el riesgo de forma cuantitativa o cualitativamente, sus ventajas y desventajas y la manera en la que ha de realizarse cada una de ellas.

En el capítulo cuatro se presenta una propuesta innovadora para evaluar el riesgo informático. En este capítulo se establecen los criterios a considerar en la evaluación de riesgos, se determinan los criterios y factores de la evaluación y se expondrá un ejemplo completo de evaluación de riesgos mediante ésta nueva metodología. Finalmente se exponen las ventajas y desventajas de una evaluación de riesgo ponderada.

El capítulo cinco muestra el desarrollo e implementación de un Sistema para la Definición y Evaluación de Riesgos Informáticos para el ITESM-CEM que ha sido denominado (SDERI). Se presenta la forma en la que serán evaluados el nivel de riesgo de los equipos, los factores a considerar y el diseño tanto del sistema como de la base de datos que utiliza y la integración de SDERI con otros sistemas complementarios.

El capitulo seis presenta los resultados de haber evaluado el nivel de riesgo de los servidores del ITESM-CEM

Por último se presentan una serie de recomendaciones y conclusiones, producto del análisis detallado y exhaustivo de la investigación que aportará al Campus Estado de México una innovación en materia de evaluación de riesgo informático.

2. EL RIESGO

De acuerdo a COBIT [1], "Un elemento crítico para el éxito y la supervivencia de las organizaciones, es la administración efectiva de la información y de la tecnología de información relacionada. En esta sociedad global (donde la información viaja a través del "ciberespacio" sin las restricciones de tiempo, distancia y velocidad) esta criticidad emerge de:

- La creciente dependencia en información y en los sistemas que proporcionan dicha información.
- La creciente vulnerabilidad y un amplio espectro de amenazas, tales como las "ciber amenazas" y la guerra de información (information warfare).
- La escala y el costo de las inversiones actuales y futuras en información y en tecnología de información.
- El potencial que tienen las tecnologías para cambiar radicalmente las organizaciones y las prácticas de negocio, crear nuevas oportunidades y reducir costos.

Para muchas organizaciones, la información y la tecnología que la soporta, representan los activos más valiosos de la empresa.

Es evidente que se deben asegurar estos sistemas. Sin embargo, los recursos para asegurar estos sistemas son finitos y es necesario establecer un mecanismo que nos permita ser eficientes en este proceso. Es decir, hay que determinar objetivamente cuales son los recursos informáticos que presentan un mayor riesgo para la organización y en base en ello tomar acciones que permitan manejar este riesgo.

En la actualidad, el Campus Estado de México cuenta con alrededor de 55 servidores que atienden diferentes servicios entre los que destacan los siguientes: de comunicación, firewalls, de nombres de dominios, de información como bases de datos, servicios de directorios, servicios de

Internet/Intranet, tales como: ftp, correo electrónico, Web, de colaboración, etc., software especializado para laboratorios y academia, aplicaciones desarrolladas internamente y por proveedores externos, servidores para el uso de sistemas en desarrollo, soluciones de respaldo, etc. Implementados en diferentes plataformas, tecnologías y sistemas operativos destacando como plataforma principal Intel, aunque aún se conservan servidores con procesadores PowerPC y como sistemas operativos, principalmente sobresalen servidores Linux y Windows.

La administración de estos equipos resulta compleja y costosa debido a la cantidad y variedad de servicios en los que se debe focalizar el resguardo, disponibilidad y seguridad de los mismos.

Como parte de la definición de administración de la TI que realiza el IT Governance Institute [2], es importante destacar dos puntos de relevancia: "El valor que la TI aporta al negocio y la reducción de riesgos informáticos. En donde el primer punto está orientado hacia la alineación estratégica de la TI con los negocios. Y el segundo se enfoca en la implantación de la responsabilidad en la empresa. En ambos casos, se exige el soporte de recursos adecuados, mismos que deben medirse para asegurar la obtención de los resultados."

Lo anterior ratifica la necesidad del Campus de contar con un modelo o sistema que identifique objetivamente, el orden en el que deben asegurarse y priorizarse los recursos informáticos dependiendo de una definición de factores y variables que coadyuven a determinar y medir el nivel de riesgo de los activos informáticos del Campus.

2.1 CONCEPTOS BÁSICOS

2.1.1 ACTIVOS

En forma genérica [3], un *activo* puede ser cualquier cosa que tenga valor a la organización. Por ejemplo: los sistemas de información, la información, el personal, la reputación, el hardware, etc.

En informática [4], "los activos son recursos del sistema de información o relacionados con éste, necesarios para que la organización funcione correctamente y alcance los objetivos propuestos por la dirección".

No se está hablando de lo que cuestan las cosas, sino de lo que valen. Si algo no vale nada, se puede prescindir de ello. Si no se puede prescindir impunemente de un activo, es que cuenta con valor. Esto es lo que hay que averiguar pues eso es lo que hay que proteger.

El valor puede ser propio, o puede ser acumulado. Se dice que los activos inferiores en un esquema de dependencias, acumulan el valor de los activos que se apoyan en ellos.

El valor nuclear suele estar en la información (o datos) que el sistema maneja, quedando los demás activos subordinados a las necesidades de explotación y protección de la información. Por otra parte, los sistemas de información explotan los datos para proporcionar servicios, internos a

la organización o destinados a terceros, apareciendo una serie de datos necesarios para prestar un servicio. Las dependencias entre activos permiten relacionar los demás activos con datos y servicios.

2.1.2 RIESGO

Existen diferentes definiciones de riesgo, las cuales varían de acuerdo al contexto en que se habla. De acuerdo a [5], "el riesgo es la posibilidad de que se produzca un contratiempo o una desgracia, de que alguien o algo sufra perjuicio o daño".

En términos de negocio [6], "el riesgo es la probabilidad de que un evento pueda reducir el valor del negocio donde ocurre. Este evento es llamado evento adverso".

En términos comerciales [7], "el riesgo es una medida de incertidumbre. Puede consistir en consecuencias positivas o negativas, aunque la mayoría de los riesgos positivos se llaman oportunidades y los riesgos negativos se les llama riesgos".

En informática contamos con las siguientes definiciones:

- De acuerdo a COBIT: [1] La probabilidad de que una amenaza explote vulnerabilidades de un activo o conjunto de activos y cause pérdida o daño a los mismos. El riesgo está medido en términos de consecuencias y posibilidad de ocurrencia."
- En la versión 2 de MARGERIT [4] se define como "La posibilidad de que se produzca un impacto determinado en un activo, en un dominio o en toda la organización".
- CRAMM [14] lo define como "El riesgo puede ser determinado como el producto de la probabilidad del impacto de un activo y el costo del impacto".
- La gente de Microsoft [8] propone "La probabilidad de que se produzca un ataque a la empresa. Esta definición requiere la inclusión de una declaración de repercusiones y una predicción de cuándo se puede producir la repercusión, es decir, la probabilidad".
- El BS 7799 [9], define al riesgo como: "La combinación de que una amenaza explote una vulnerabilidad que pueda causar un daño a un activo".

Se observa que la mayor parte de las definiciones coinciden en que el riesgo es la probabilidad de que un evento adverso ocurra. Para medirlo se toma en cuenta tanto su impacto como la probabilidad de que ocurra el evento.

2.1.3 AMENAZA

El diccionario [37] define *amenaza* como "como algo o alguien que es probable que cause daño".

En forma genérica [4], "las amenazas son eventos que pueden desencadenar un incidente en la organización, produciendo daños materiales o pérdidas inmateriales en sus activos".

En informática, se encontramos las siguientes definiciones:

- CRAMM [15] define a las amenazas como:"Algo (personas, grupos o actividades) que tratan de causar incidentes de seguridad a activos".
- En ISO 17799 [8] se definen las amenazas como "causa de repercusiones posibles en la organización".
- NIST [8] define una amenaza como un suceso o entidad con posibilidad de dañar el sistema. Las repercusiones derivadas de una amenaza normalmente se definen con conceptos como confidencialidad, integridad y disponibilidad.

Dentro de este trabajo consideraremos una amenaza como todo aquello que pueda afectar negativamente a los activos.

2.1.4 VULNERABILIDAD

Una *vulnerabilidad* [8] es un punto débil de un activo o grupo de activos que una amenaza puede atacar. De un modo simplificado, las vulnerabilidades proporcionan el mecanismo o el modo en que se pueden producir las amenazas.

En informática, la vulnerabilidad es:

- CRAMM [14] define una vulnerabilidad como: "la ausencia de protección (por ejemplo, ausencia de firewalls, passwords débiles, o la presencia de hoyos de seguridad en el software) en contra de las amenazas que a su vez están en contra de los activos".
- NIST [8] define la vulnerabilidad como una situación o punto débil en (o la ausencia de) los procedimientos de seguridad, controles técnicos, controles físicos u otros controles que puede aprovechar una amenaza.

Un error habitual al llevar a cabo una evaluación de riesgos es centrarse en vulnerabilidades técnicas. La experiencia demuestra que las vulnerabilidades más importantes se suelen producir debido a la ausencia de un proceso definido o un control no adecuado de la seguridad de información.

Para este trabajo, una vulnerabilidad es el modo en como se puede llegar a materializar una amenaza.

2.1.5 IMPACTO

El *impacto* [4] "es la consecuencia sobre la materialización de una amenaza". Es decir, el costo de que haya ocurrido un evento adverso. Este costo es tanto en perdidas económicas, como en el tiempo necesario para revisar, componer y estabilizar el efecto del evento.

2.1.6 SALVAGUARDA

La salvaguarda es la acción que reduce el riesgo. Es decir, el procedimiento o dispositivo (físico o lógico) que disminuye el riesgo.

2.1.7 INTERRELACIÓN DE LOS CONCEPTOS BÁSICOS

Esquemáticamente podemos relacionar estos conceptos en la figura 1:

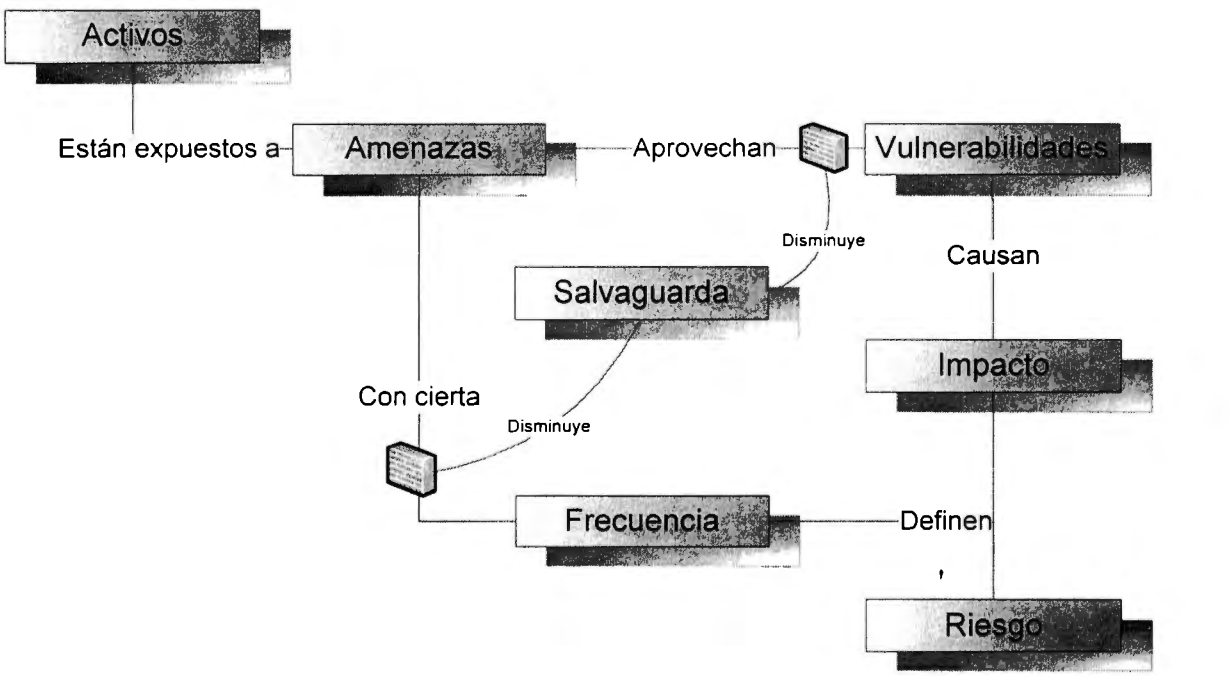

Fig. 1 - Relación entre activos, amenazas y riesgos

En resumen, los activos están expuestos a amenazas que aprovechan vulnerabilidades que al ser explotadas producen un impacto que multiplicado por la frecuencia de ocurrencia nos permite determinar el nivel de riesgo. Las salvaguardas nos permiten reducir tanto el impacto como la frecuencia de que se materialicen las amenazas.

2.2 ADMINISTRACIÓN DE RIESGOS

El impacto de la materialización de las amenazas puede llegar a ser devastadora para una organización. Es por ello, que la implantación de salvaguardas debe ser optimizada para reducir al máximo los riesgos al mismo tiempo que se reduzcan los costos de implantación de estas salvaguardas. Es decir, es necesario contar con una administración de riesgos.

La administración de riesgos [11] ha sido definida como "el proceso de identificar, reducir, controlar y minimizar el impacto de eventos indeseados. Una vez que se identifican los riesgos de un proyecto en particular, la administración de riesgos trata de minimizar sus efectos. Esto lo hace al aplicar una serie de técnicas de administración".

Desde el punto de vista de las Tecnologías de Información (TI), la administración de riesgos es un acercamiento sistemático usado para identificar, evaluar y reducir o eliminar la posibilidad de un desafortunado evento. Es un proceso permanente de evaluar los riesgos de los recursos de los sistemas de información. Entre otras cosas, la administración de riesgos de TI es usada para determinar adecuadamente los sistemas de seguridad para los sistemas al analizar las amenazas y vulnerabilidades y seleccionado los apropiados y efectivos controles (costo/beneficio) para salvaguardar y mantener un nivel de riesgo aceptable.

2.2.1 ENFOQUES EN LA ADMINISTRACIÓN DE RIESGOS

Las organizaciones requieren de la administración de riesgos al estar expuestos a incidentes de seguridad. Por ejemplo si un equipo se infecta con un virus, un experto debe averiguar cómo tiene que erradicar el virus sin destruir el equipo ni los datos que contiene. A medida que aparecen cada vez más problemas relacionados con la seguridad y comienzan a tener repercusiones en el negocio, es necesario administrar los riesgos. Se pueden tener dos enfoques:

- Un enfoque reactivo
- Un enfoque proactivo

2.2.1.1 Enfoque reactivo

En este enfoque se espera a que el incidente ocurra para tomar una acción correctiva. Una organización que dedica tiempo a responder a los incidentes de seguridad en un modo calmado y racional mientras determina los motivos subyacentes que provocan la incidencia podrá protegerse mejor de problemas similares en el futuro y responderá con más rapidez a otros problemas.

Los equipos de respuesta a incidentes se enfocan en los siguientes pasos para poder ser eficientes:

1. Proteger la vida humana y la seguridad de las personas. Ésta debe ser siempre la primera prioridad. Por ejemplo, si los equipos afectados incluyen equipos de mantenimiento de vida, apagarlos puede no ser una opción.
2. Contener el daño. Contener el daño que ha provocado el ataque ayuda a limitar daños adicionales. Si es posible aísle la fuente del daño (desconecte equipos).
3. Evaluar el daño. Si no se puede evaluar el daño de forma oportuna, se debe implementar un plan de contingencias para que las operaciones de negocios normales y la productividad puedan continuar.
4. Determinar la causa del daño. Para determinar el origen del asalto, es necesario conocer los recursos a los que iba dirigido el ataque y las vulnerabilidades que se han aprovechado para obtener acceso o interrumpir los servicios.
5. Reparar el daño. Los planes y procedimientos de continuidad de negocio de la organización deben cubrir la estrategia de restauración. Antes de devolver los sistemas reparados al servicio, se debe cuidar de que no se vuelvan a infectar inmediatamente; para ello, hay que asegurarse de que hayan mitigado las vulnerabilidades que se hayan atacado durante la incidencia.
6. Revisar las directivas de respuesta y actualización. Después de haber completado las fases de documentación y recuperación, se debe revisar a fondo el proceso. Comúnmente, es necesario modificar los procesos para poder administrar mejor las incidencias en el futuro.

2.2.1.2 Enfoque proactivo

La administración de riesgos de seguridad proactiva tiene numerosas ventajas con respecto a un enfoque reactivo. En vez de esperar a que suceda lo peor y, a continuación, llevar a cabo la respuesta, se minimiza la posibilidad de que pase lo peor antes de que se produzca. Se trazan planes para proteger los activos importantes de la organización mediante la implementación de controles que reduzcan el riesgo de que el software malintencionado, los piratas informáticos o un uso incorrecto o accidental de los sistemas aprovechen las vulnerabilidades.

Es importante señalar que en el caso de que se decida establecer un enfoque proactivo para la administración de riesgos, eso no implica la ausencia de la necesidad de establecer planes y procesos para reaccionar ante incidentes de seguridad. El enfoque proactivo permitirá evitar en gran medida los "bomberazos" en las organizaciones, sin embargo aun así existirán incidentes que habrá que manejar adecuadamente.

2.2.2 MANEJO DEL RIESGO

Una vez que se hayan identificado los riegos a los cuales están expuestos los activos de la organización es necesario determinar como se van a manejar, para ello, existen al menos 5 formas de hacerlo:

- Evitar el riesgo
- Transferencia de responsabilidad
- Indemnización
- Mitigación y
- Retención

2.2.2.1 Evitar el riesgo

Esta forma de manejar el riesgo generalmente es meramente conceptual por que implica evitar oportunidades. Por ejemplo, si buscamos que un equipo no tenga riesgos, lo más probable es que lo apaguemos y lo metamos a una caja de seguridad. Esta forma de evitar el riesgo suele ser inviable.

2.2.2.2 Transferencia de responsabilidad

La organización puede transferir la responsabilidad para un evento hacia un tercero. Esta transferencia puede ser de dos maneras por negación o por acuerdo.

La organización niega responsabilidad cuando emprende una actividad con el entendido explícito que no va a ser responsable de las consecuencias de eventos adversos, pero sin especificar quien va a ser el responsable de esas consecuencias.

La organización transfiere la responsabilidad al establecer un acuerdo, en el cual se establece que la contraparte será responsable de las consecuencias de ciertos eventos adversos.

2.2.2.3 Indemnización

Un negocio se puede indemnizar a si mismo. Hay dos tipos de indemnizaciones: reuniendo o dirigiendo.

En un esquema de reunión, varios negocios comparten el costo de ciertos eventos. Los seguros son el ejemplo más común de indemnizaciones. Actualmente existen pólizas de seguros que abarcan los ataques informáticos [38]. Los precios de las pólizas varían mucho, empezando en un par de miles de dólares para las pequeñas empresas. Las aseguradoras tradicionalmente requieren de una tercera opinión en la valoración del sistema de seguridad actual del aplicante, que cuesta

hasta 50.000 dólares, así como evidencias que la compañía ha tomado los pasos necesarios para proteger a sus redes de los "hackers".

En un esquema de dirección, una empresa hace una apuesta de que el evento pasará. Si es improbable, otras organizaciones aceptarán la apuesta. Los casinos están basados en estos esquemas. En informática, se han desarrollado retos que buscan impulsar el desarrollo y la investigación. Existen retos criptográficos como es el caso de RSA[39], retos para promover nuevas tecnologías de seguridad como es el caso del reto global de seguridad [40], etc.

2.2.2.4 Mitigación

Una empresa puede reducir su expectativa de costo de un riesgo al reducir la probabilidad de que el evento ocurra o al reducir las consecuencias si es que ocurre. Es posible reducir la probabilidad de que un evento adverso ocurra al ajustar sistemas o procesos para eliminar las causas probables y conocidas.

Las consecuencias de un evento adverso pueden ser reducidas al tomar los pasos para limitar el daño. Estos pasos previenen que el daño se difumine o acortan el tiempo que ocurre el daño al acelerar la detección y recuperación.

Este enfoque es el preponderante en la administración de riesgos informáticos.

2.2.2.5 Retención

Si un evento adverso no es muy costoso o no es probable que ocurra, o si los beneficios de tomar el riesgo son grandes, la empresa puede escoger retener el riesgo

La empresa puede escoger establecer un fondo para solventar los riesgos en caso de que ocurran.

Si la empresa retiene el riesgo sin establecer un fondo, entonces se dice que la empresa acepta el riesgo.

2.3 METODOLOGÍAS DE ADMINISTRACIÓN DE RIESGOS

Todas las metodologías de administración de riesgos de seguridad comparten algunos procedimientos de alto nivel comunes:

1. Identificar los activos de negocios.
2. Determinar el daño que un ataque a un activo podría provocar a la organización.
3. Identificar las vulnerabilidades que aprovechará el ataque.

4. Determinar el modo de minimizar el riesgo de ataque mediante la implementación de los controles adecuados.

La administración de riesgos es un tema amplio el cual ha sido estudiado por diversos grupos, entre los que destacan:

- AS/NZS 4360
- COBIT
- CRAMM
- FERMA
- MAGERIT
- Microsoft
- NIST
- OCTAVE
- Risk Targeting

A continuación daremos una breve descripción de cada uno de los anteriores.

2.3.1 AS/NZS 4360

El estándar de administración de riesgos de Australia y Nueva Zelanda 4360 [10] (The Australian and New Zealand Standard on risk management, AS/NZS 4360) fue desarrollado por Dr. Dale F Cooper para ayudar a la administración de riesgos en organismos tanto públicos como privados.

Su proceso de administración de riesgos, se puede resumir en la figura 2:

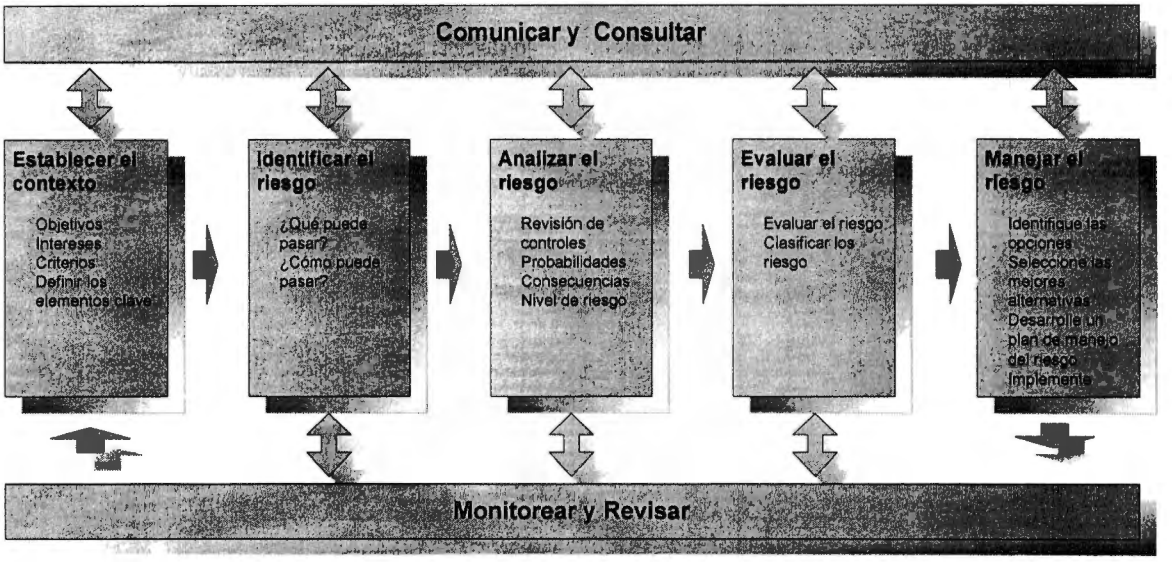

Fig. 2 - Administración de Riesgos - AS/NZS 4360

2.3.1.1 Establecer el contexto

Para reconocer el riesgo es necesario saber que esta en riesgo. El primer paso en este estándar es definir el contesto de la evaluación del riesgo, lo cual se hace tanto de forma descriptiva como creativa.

El contexto descriptivo asegura que todos los riesgos significativos son tomados en cuenta. Para ello, es necesario saber los objetivos del negocio, así como las alineaciones de los objetivos entre los niveles de la organización. También es necesario tomar en cuenta el interés de los distintos actores que intervengan en la operación de la organización. En esta etapa se deben definir los criterios de éxito, enfocándose más en los niveles de servicio y en el retorno de inversión, que en los costos y tiempos de respuesta.

El contexto creativo, trata de ver cada uno de los componentes de la empresa para identificar el riesgo. Estos componentes son definidos como elementos clave.

2.3.1.2 Identificar el riesgo

En general, hay dos formas de identificar el riesgo: por listas de chequeo o por lluvia de ideas.

Las listas de chequeo son listas estándar de riesgo, que aparecen para un contexto en particular. Estas listas son rápidas de usar, pero limitan su alcance a lo ya conocido.

La lluvia de ideas es una mejor forma de identificar los riesgos, pero demandan un mayor esfuerzo.

Lo mejor para identificar todos los posibles riesgos, es empezar por una lluvia de ideas y completarlo con listas de chequeo.

2.3.1.3 Análisis de riesgo

En esta etapa se asigna a cada riesgo una jerarquía tomando en cuenta todos los factores que permiten operar el control del riesgo. Para riesgos simples, donde el riesgo puede ser expresado como un evento improbable, puede ser usado un enfoque de impacto y probabilidad cualitativos. Cuando los riesgos son complejos, en donde se relacionan eventos e influencias, se requerirá algún tipo de modelación.

Sin importar como se definan los riesgos en detalle, esta etapa arrojará una vista inicial de los riesgos más significantes.

2.3.1.4 Evaluación del riesgo

La evaluación toma el análisis inicial y lo revisa en contra de las prioridades y requerimientos de la organización. Los riesgos son ajustados en su jerarquía.

En esta etapa es común encontrar una gran cantidad de riesgos menores, y por lo general se eliminan del proceso (después de una cierta consideración), evitando la sobresaturación del proceso.

2.3.1.5 Manejo del riesgo

El manejo del riesgo consiste en determinar que se debe hacer en respuesta a los riesgos identificados. En esta etapa es necesario definir un plan de contingencia, en caso de que un riesgo ocurra.

2.3.1.6 Monitoreo y revisión

El resultado de las cinco etapas debe mantenerse en revisión mientras transcurre el tiempo. Cambios en el entorno, pueden provocar que la valoración no este al día. En general solo es necesario modificar las partes que directamente se afecten.

La ejecución de la administración de riesgos absorbe recursos y debe ser administrada para asegurar que es conducida en forma efectiva.

2.3.1.7 Comunicar y consultar

La consulta y la comunicación son elementos claves en el proceso de la administración de riesgos. El éxito de la administración de riesgos se basa en involucrar todas las partes. Tanto en la planeación como en la ejecución de la administración de riesgos, es muy importante asegurarse que todos los que necesiten ser involucrados tengan la oportunidad de hacerlo y que se

mantengan adecuadamente informados en el entendimiento de los riesgos y salvaguardas que se tomen para manejarlos.

2.3.2 COBIT

Las siglas COBIT significan Control Objectives for Information Systems and related Technology (Objetivos de Control para la Información y Tecnologías Relacionadas) [20]. Es el resultado de una investigación con expertos de varios países, desarrollado por la "Information Systems Audit Control Association (ISACA)".

El COBIT es un modelo para evaluar y/o auditar la gestión y control de los sistemas de información y tecnología.

La estructura de COBIT propone un marco de acción donde se evalúen los requerimientos del negocio, los recursos de TI y a los procesos de TI; los cuales pueden ser enfocados desde tres puntos ventajosos como lo son los criterios de información, como por ejemplo la seguridad y calidad, se evalúan los recursos que comprenden la tecnología de información, como por ejemplo recurso humano, instalaciones, sistemas, entre otros, y finalmente se realiza una evaluación sobre los procesos involucrados en la organización.

COBIT es un modelo estructurado y lógico de mejores prácticas de Tecnología de Información, definidas por un consenso de expertos en todo el mundo en aspectos técnicos, seguridad, riesgos, calidad y control.

El modelo de COBIT [18], esta compuesto por cuatro Dominios, los cuales están formados a su vez por 34 procesos genéricos.

Fig. 3 - Dominios de COBIT

Para nuestros propósitos destaca el proceso PO9, referente a una evaluación de riesgos.

Según COBIT [18] la administración de riesgos, se realiza mediante ocho pasos:

1. Evaluación de Riesgos del Negocio
2. Enfoque de Evaluación de Riesgos
3. Identificación de Riesgos
4. Medición de Riesgos
5. Plan de Acción contra Riesgos
6. Aceptación de Riesgos
7. Selección de Garantías o Protecciones

8. Compromiso con el Análisis de Riesgos

2.3.2.1 Evaluación de Riesgos del Negocio

En esta etapa se establece un marco de referencia de evaluación sistemática de riesgos. Este marco de referencia deberá incorporar una evaluación regular de los riesgos de información relevantes para el logro de los objetivos del negocio, formando una base para determinar la manera en la que los riesgos deben ser manejados a un nivel aceptable. El proceso deberá proporcionar evaluaciones de riesgos tanto a un nivel global como a niveles específicos del sistema, para nuevos proyectos y para casos recurrentes y con participación multidisciplinaria. La administración deberá asegurar que se realicen reevaluaciones y que la información sobre evaluación de riesgos sea actualizada como resultado de auditorias, inspecciones e incidentes identificados.

2.3.2.2 Enfoque de Evaluación de Riesgos

En esta etapa se establece un enfoque general para la evaluación de riesgos que defina el alcance y los límites, la metodología a ser adoptada para las evaluaciones de riesgos, las responsabilidades y las habilidades requeridas.

Se debe adelantar la identificación de soluciones para la mitigación de riesgos e involucrarse en la identificación de vulnerabilidades. Especialistas de seguridad deben realizar identificación de amenazas y especialistas de TI deben dirigir la selección de controles. La calidad de las evaluaciones de riesgos deberá estar asegurada por un método estructurado y por asesores expertos en riesgos.

2.3.2.3 Identificación de Riesgos

La evaluación de riesgos deberá enfocarse al examen de los elementos esenciales de riesgo y las relaciones causa/efecto entre ellos. Los elementos esenciales de riesgo incluyen activos tangibles e intangibles, valor de los activos, amenazas, vulnerabilidades, protecciones, consecuencias y probabilidad de amenaza. El proceso de identificación de riesgos debe incluir una clasificación cualitativa y, donde sea apropiado, clasificación cuantitativa de riesgos y debe obtener insumos de las lluvias de ideas de la gerencia, de planeación estratégica, auditorias anteriores y otros análisis. El análisis de riesgos debe considerar el negocio, regulaciones, aspectos legales, tecnología, comercio entre socios y riesgos del recurso humano.

2.3.2.4 Medición de Riesgos

El enfoque de la evaluación de riesgos deberá asegurar que la información del análisis de la identificación de riesgos genere como resultado una medida cuantitativa y/o cualitativa del riesgo

al cual está expuesta el área examinada. Asimismo, deberá evaluarse la capacidad de aceptación de riesgos de la organización.

2.3.2.5 Plan de Acción contra Riesgos

El enfoque de evaluación de riesgos deberá proporcionar la definición de un plan de acción contra riesgos para asegurar que el costo–efectividad de los controles y las medidas de seguridad mitiguen los riesgos en forma continua. El plan de acción contra los riesgos debe identificar la estrategia de riesgos en términos de evitar, mitigar o aceptar el riesgo.

2.3.2.6 Aceptación de Riesgos

El enfoque de la evaluación de riesgos deberá asegurar la aceptación formal del riesgo residual, dependiendo de la identificación y la medición del riesgo, de la política organizacional, de la incertidumbre incorporada al enfoque de evaluación de riesgos y el costo-efectividad de la implementación de protecciones y controles. El riesgo residual deberá compensarse con una cobertura de seguro adecuada, compromisos de negociación contractual y autoaseguramiento.

2.3.2.7 Selección de Garantías o Protecciones

Mientras se logra un sistema de controles y garantías razonable, apropiado y proporcional, controles con el más alto retorno de inversión y aquellos que provean ganancia rápida deben recibir la primera prioridad. El sistema de control necesita además balancear las medidas de prevención, detección, corrección y recuperación. Adicionalmente, la gerencia necesita comunicar el propósito de las medidas de control, manejar el conflicto y monitorear continuamente la efectividad de las medidas de control.

2.3.2.8 Compromiso con el Análisis de Riesgos

La gerencia deberá motivar el análisis de riesgos como una herramienta importante para proveer información para el diseño e implementación de controles internos, en la definición del plan estratégico de tecnología de información y en los mecanismos de evaluación y monitoreo.

2.3.3 CRAMM

En 1997 el gobierno británico desarrollo CRAMM [17]. CRAMM es el acrónimo de CCTA Risk Analysis and Management Methodology. A su vez la CCTA es el acrónimo de la Central Computer and Telecommunications Agency. Traduciendo al español CRAMM es la Metodología para el Análisis y Administración del Riesgo de la CCTA (Agencia Central de Computación y Telecomunicación). CRAMM es consistente con los estándares y políticas de seguridad del

gobierno británico y del BS7799 (el código de prácticas para la administración de seguridad de la información).

El método para el análisis de riesgo usado por CRAMM [3] consiste en evaluar los siguientes tres factores:

- Las amenazas que pueden afectar a los activos,
- Las vulnerabilidades que pueden ser aprovechadas por esas amenazas y
- El costo en caso del impacto hacia el activo

Y a partir de esto se determina el nivel de riesgo o se establece una medida del riesgo. Este concepto se ilustra en la siguiente figura:

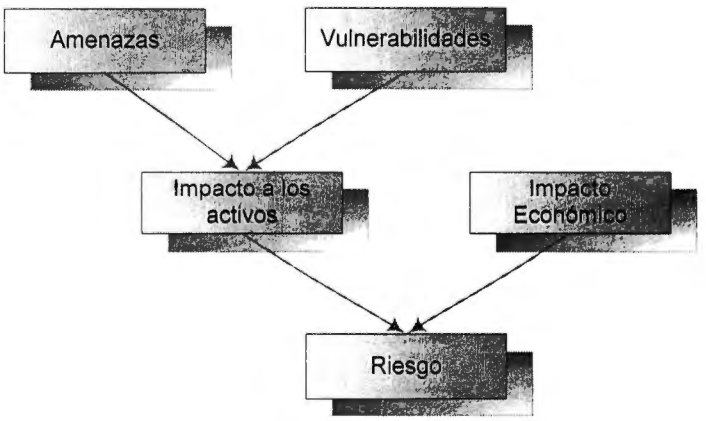

Fig. 4 - El riesgo según CRAMM

El análisis de riesgo en si mismo consiste en cinco pasos:

1. Identificar los activos, amenazas y vulnerabilidades.
2. Identificar los posibles impactos a los activos.
3. Evaluación de los activos y medición de las amenazas y vulnerabilidades.
4. Calcular el riesgo
5. Revisión de los resultados

2.3.3.1 Identificar los activos, amenazas y vulnerabilidades.

Cada activo potencial de ser impactado tiene que ser identificado. Listas de todas las amenazas imaginables, de todas las vulnerabilidades relevantes y todos los activos potencialmente afectados son establecidos.

2.3.3.2 Identificar los posibles impactos a los activos.

Una lista de todas las combinaciones de amenazas y vulnerabilidades que potencialmente puedan causar un impacto a un activo son identificadas.

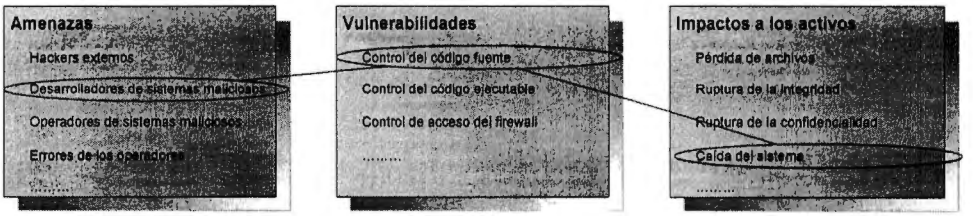

Fig. 5 - Impactos potenciales. Combinación de amenaza, vulnerabilidad e impacto- CRAMM

2.3.3.3 Evaluación de los activos y medición de las amenazas y vulnerabilidades.

Cada activo potencialmente afectado tiene que ser evaluado de acuerdo a los costos de pérdida o daño del activo. Todos los valores son transcritos a una escala de 1 a 10. La fuerza de las amenazas y el nivel de vulnerabilidades debe ser cuantificado. Los posibles valores para las amenazas y las vulnerabilidades son: bajo, medio y alto.

2.3.3.4 Calcular el riesgo

Para este cálculo se usa una tabla tridimensional donde la fuerza de la amenaza, el nivel de vulnerabilidad y el valor del activo son los parámetros de entrada, al final da el requerimiento de seguridad (dígase nivel de riesgo) en un rango de uno a cinco.

Tabla 1 - Nivel de Riesgo- CRAMM

Nivel de amenaza	Bajo			Medio			Alto		
Vulnerabilidad	B	M	A	B	M	A	B	M	A
Valor del activo									
1	1	1	1	1	1	1	1	1	2
2	1	1	1	1	1	1	1	1	2
3	1	1	2	1	2	2	2	2	3
4	1	2	2	2	2	3	3	3	4
5	2	2	3	2	3	3	3	3	4
6	2	3	3	3	3	4	3	4	4
7	3	3	4	3	4	4	4	4	5
8	3	4	4	4	4	5	4	5	5
9	4	4	5	4	5	5	5	5	5
10	4	5	5	5	5	5	5	5	5

2.3.3.5 Revisión de los resultados

En este punto se revisan los datos. El común tiene que ser usado para ver si los resultados parecen razonables. Usualmente, un ajuste de los datos de entrada es necesario.

Después del análisis de riesgos, las mejores salvaguardas pueden ser seleccionadas y el análisis de riesgo puede ser hecho nuevamente contando con las nuevas salvaguardas, para así determinar si se ha podido reducir el riesgo a un nivel aceptable.

2.3.4 FERMA

La FERMA [27] es la federación europea de asociaciones de administración de riesgo (Federation European Risk Management Associations).

La propuesta de administración de riesgos de FERMA tiene un enfoque de negocios más que de TI, pero se puede aplicar a esta sección de la empresa.

Según FERMA la administración de riesgos se realiza mediante siete pasos que se ilustran a continuación:

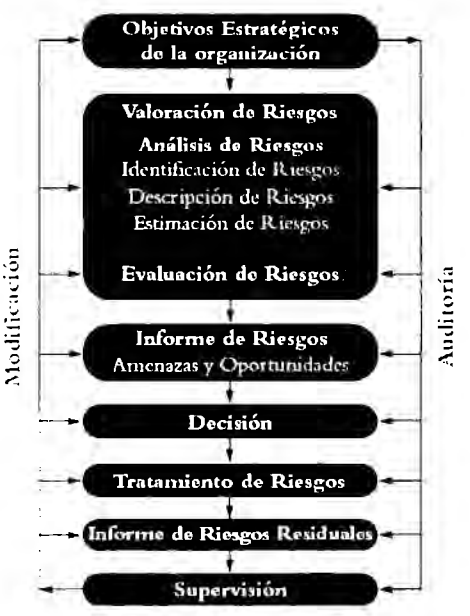

Fig. 6 - Administración de riesgos- FERMA

2.3.4.1 Valoración de riesgos

La valoración de riesgos está definida como el proceso general de análisis y de evaluación de riesgos

2.3.4.1.1 Análisis de riesgos

Comprende la identificación, descripción y estimación de riesgos.

2.3.4.1.1.1 Identificación de riesgos

La identificación de riesgos se propone identificar la exposición de una empresa a la incertidumbre. Ello requiere un conocimiento detallado de dicha empresa, del mercado en el que opera, del entorno legal, social, político y cultural que le rodea, así como el desarrollo de una visión común coherente de su estrategia y de sus objetivos operacionales, incluyendo los factores críticos para su éxito y las amenazas y oportunidades relacionadas con la consecución de estos objetivos

2.3.4.1.1.2 Descripción de riesgos

El objetivo de la descripción de riesgos es mostrar los riesgos identificados de una forma estructurada, con todos los datos relevantes, entre ellos:

1. Nombre del riesgo
2. Alcance del riesgo
3. Naturaleza del riesgo
4. Interesados
5. Cuantificación del riesgo
6. Tolerancia del riesgo
7. Tratamiento del riesgo y mecanismos de control
8. Acción potencial de mejora
9. Política y estrategia a desarrollar

2.3.4.1.1.3 Estimación de riesgos

La estimación de riesgos puede ser cuantitativa, semi-cuantitativa o cualitativa en términos de probabilidad de ocurrencia y de sus posibles consecuencias.

2.3.4.1.2 Evaluación de riesgos

Cuando el proceso de análisis de riesgos se ha llevado a cabo, es necesario comparar los riesgos estimados con los criterios de riesgo establecidos por la empresa. Los criterios de riesgo pueden incluir costos y beneficios asociados, requisitos legales, factores socioeconómicos y medioambientales, preocupaciones de los interesados, etc. Por tanto, se usa la evaluación de riesgos para tomar decisiones acerca de la importancia de los riesgos para la empresa y sobre si se debe aceptar o tratar un riesgo específico.

2.3.4.2 Tratamiento de riesgos

El tratamiento de riesgos es el proceso que consiste en seleccionar y aplicar medidas para modificar el riesgo. El tratamiento de riesgos incluye, como principal elemento, el control o mitigación del riesgo, pero también se extiende más allá, por ejemplo, a la elusión de riesgos, a la transferencia de riesgos, a la financiación de riesgos, etc.

2.3.5 MAGERIT

El Consejo Superior de Administración Electrónica de España ha desarrollado la Metodología de Análisis y Gestión de Riesgos de los sistemas de información (MAGERIT) [4]. En junio del 2005 publicaron la segunda versión de esta metodología.

Esta metodología se puede resumir en la siguiente figura:

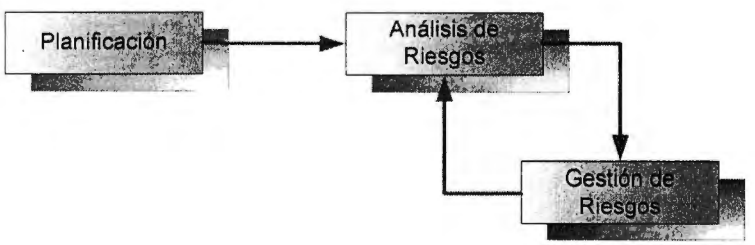

Fig. 7 - Administración de Riesgos según MAGERIT

2.3.5.1 Planeación

En esta fase, se establece el objetivo del proyecto, el dominio (ámbito) de estudio y las restricciones generales (recursos materiales/humanos para realizarlo). Deben también definirse las métricas con las que se valorarán los diferentes elementos de seguridad, de manera que los resultados finales de medición del riesgo sean definidos en función de los parámetros adecuados para cuantificar el riesgo por la organización (por ejemplo, definir la escala de frecuencias para medir la vulnerabilidad, definir las cantidades monetarias por las que cuantificar el impacto, etc.).

2.3.5.2 Análisis de riesgos

Una vez definido el dominio, los analistas de riesgos procederán a realizar las entrevistas al personal de la organización para la obtención de información. En esta fase se identificarán los activos de la organización, identificando las relaciones que se establecen entre activos. De esta forma se obtiene el "árbol de activos" que representan las distintas dependencias y relaciones entre activos, es decir, todos aquellos elementos que están "encadenados entre sí" en términos de seguridad.

También se identifican el conjunto de amenazas, estableciendo para cada activo, cual es la vulnerabilidad que presenta frente a dicha amenaza. Además, se cuantifica el impacto, para el caso en el que la amenaza se materializase. Dado que los activos se encuentran jerarquizados y se encuentran establecidas las relaciones de dependencia entre los activos de las diferentes categorías, se puede documentar la "cadena de fallo" en caso de un incidente de seguridad. La experiencia y la sucesiva revisión de la información generada en estudios de riesgos anteriores permitirán ajustar de forma más exacta las diferentes dependencias entre activos.

Con toda esta información, se estima el costo que podría producir la materialización de una amenaza sobre un activo. Teniendo en cuenta las relaciones funcionales y de dependencias entre activos, se hallan los valores de riesgo.

2.3.5.3 Gestión de riesgos

En esta fase, se procede a la interpretación del riesgo. Una vez identificado los puntos débiles, deben seleccionarse el conjunto de funciones de salvaguarda que podrían ser usados para disminuir los niveles de riesgo a los valores deseados. Para ello, deberán especificarse los mecanismos de salvaguarda que se encuentran implantados hasta ese momento y cual es su grado de cumplimiento.

Este proceso se ayuda de la simulación. Se van probando selecciones de diferentes mecanismos de salvaguarda y se estudia en que medida reducen los niveles de riesgo a los márgenes deseados. Es muy importante realizar las correctas estimaciones de la efectividad de los diferentes mecanismos de salvaguarda para ajustar de forma precisa los valores de riesgo.

2.3.6 OCTAVE

OCTAVE [25] es el acrónimo de Operationally Critical Threat, Asset, and Vulnerability Evaluation (en español: Evaluación de Amenazas, Activos y Vulnerabilidades de Operaciones Criticas). Es una evaluación de riesgos estratégica y una técnica de planeación para la seguridad. OCTAVE se enfoca en el riesgo organizacional y en su estrategia. Busca balancear el riesgo operacional, las mejores prácticas de seguridad y la tecnología.

Esta metodología consta de tres pasos los cuales permiten construir una imagen de las necesidades de seguridad. Estos pasos son:

1. Construir los perfiles de amenazas basados en los activos.
2. Identificar las vulnerabilidades de la infraestructura
3. Desarrollar las estrategias y planes de seguridad.

Estos pasos se ilustran en la figura 8:

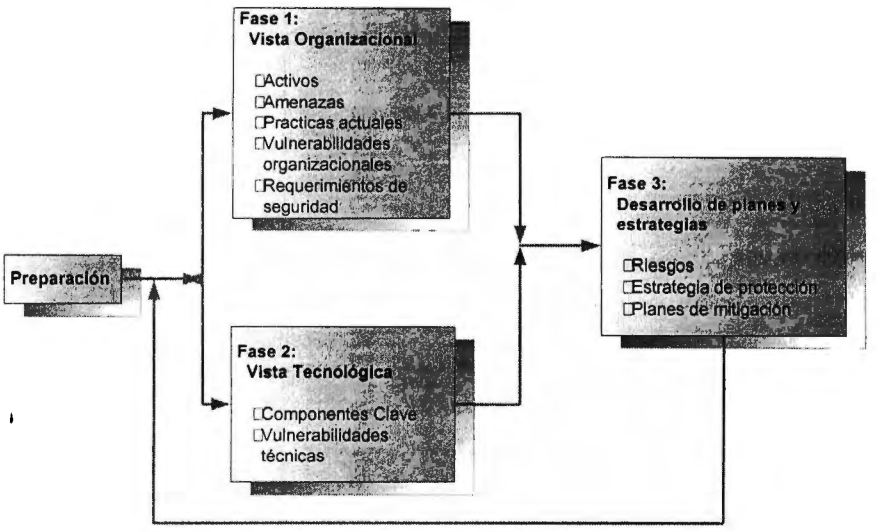

Fig. 8 - Proceso de administración de riesgos de OCTAVE

2.3.6.1 Construir los perfiles de amenazas basados en los activos.

Esta etapa es una evaluación organizacional. Se determina que es lo importante para la organización (que información con cuales activos relacionados) y que es lo que actualmente se esta haciendo para proteger estos activos. Posteriormente se seleccionan aquellos activos que sean más importantes para la organización y se describen los requerimientos de seguridad para cada activo crítico. Finalmente, se identifican las amenazas para cada activo crítico, creando un perfil de amenaza para ese activo.

2.3.6.2 Identificar las vulnerabilidades de la infraestructura

Este paso es una evaluación de la infraestructura informática. Se examinan los caminos de acceso de la red, identificando los tipos de tecnología de información relacionados con cada activo. Finalmente se determina el grado de resistencia de cada tipo hacia los ataques de red.

2.3.6.3 Desarrollar las estrategias y planes de seguridad.

Durante esta etapa de evaluación, se identifican los riesgos a los que se están expuestos los activos críticos y se decide que hacer con ellos. Se desarrolla una estrategia de protección para la organización y un plan de mitigación hacia los riesgos de los activos críticos basado en la información generada.

2.3.7 MICROSOFT

Microsoft ha definido la administración de riesgos [8] como un proceso continuo con cuatro fases principales, como se puede ver en la figura 9:

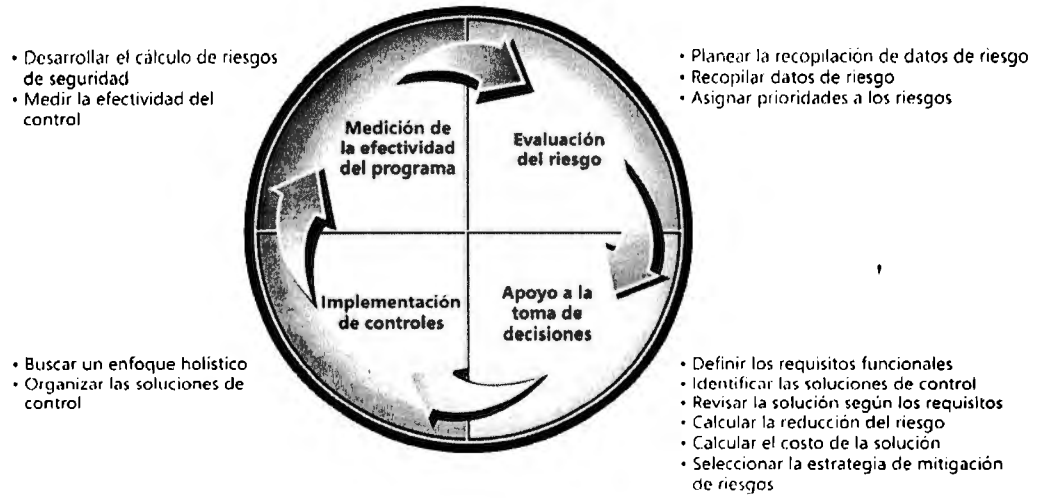

Fig. 9 - Administración de riesgos según Microsoft

2.3.7.1 Evaluación del riesgo

La fase de evaluación de riesgos representa un proceso formal para identificar y asignar prioridades a los riesgos en la organización. El proceso de administración de riesgos de seguridad de Microsoft proporciona indicaciones detalladas acerca de cómo realizar las evaluaciones de riesgos y divide el proceso de la fase de evaluación de riesgos en los tres pasos siguientes:

1. Planeamiento: establecer las bases para una evaluación de riesgos correcta.
2. Recopilación de datos facilitados: recopilar información de riesgos mediante los debates sobre riesgos facilitados.
3. Asignación de prioridades a riesgos: clasificar los riesgos identificados en un proceso coherente y repetible.

2.3.7.2 Apoyo a la toma de decisiones

El proceso de apoyo a la toma de decisiones incluye un análisis de costo-beneficio formal con funciones y responsabilidades definidas en los límites organizativos. El análisis de costo-beneficio proporciona una estructura coherente y exhaustiva para identificar, determinar el alcance y seleccionar la solución más eficaz y asequible para reducir el riesgo a un nivel aceptable.

Durante la fase de apoyo a la toma de decisiones, se debe determinar cómo afrontar los riesgos clave del modo más eficaz y asequible. El resultado final serán planes claros para controlar,

aceptar, transferir o evitar cada uno de los riesgos principales identificados en el proceso de evaluación de riesgos. Los seis pasos de la fase de apoyo a la toma de decisiones son:

1. Definir los requisitos funcionales.
2. Seleccionar las soluciones de control.
3. Revisar las soluciones según los requisitos.
4. Estimar la reducción del nivel de riesgo que cada control proporciona.
5. Estimar los costos de cada solución.
6. Seleccionar la estrategia de mitigación de riesgos.

2.3.7.3 Implementación de controles

Durante esta fase, los responsables de mitigación emplean los controles especificados durante la fase anterior. Un factor de éxito fundamental en esta fase del proceso de administración de riesgos de seguridad de Microsoft consiste en que los responsables de mitigación busquen un enfoque holístico al implementar las soluciones de control. Deben tener en cuenta todo el sistema de tecnología de la información (TI), toda la unidad de negocios o, incluso, toda la empresa al crear los planes para adquirir e implementar soluciones de mitigación.

2.3.7.4 Medición de la efectividad del programa

La fase de medición de la efectividad del programa permite que el equipo de administración de riesgos de seguridad documente formalmente el estado actual de los riesgos de la organización. A medida que la empresa continúa por el ciclo de administración de riesgos, esta fase también contribuye a demostrar el progreso de administrar el riesgo con un nivel aceptable a lo largo del tiempo.

2.3.8 NIST

El Instituto Nacional de Estándares y Tecnología (NIST: Nacional Institute of Standanrds and Technology) de Estados Unidos desarrollo en 2002 una Guía de administración de riesgos para las tecnologías de sistemas de información. [26].

Esta guía consta de nueve pasos:

1. Caracterización del sistema
2. Identificación de amenazas
3. Identificación de vulnerabilidades
4. Análisis de Control

5. Determinación de la probabilidad
6. Análisis del impacto
7. Determinación del riesgo
8. Recomendaciones de controles
9. Documentación de los resultados

Estos pasos se esquematizan en la figura 10:

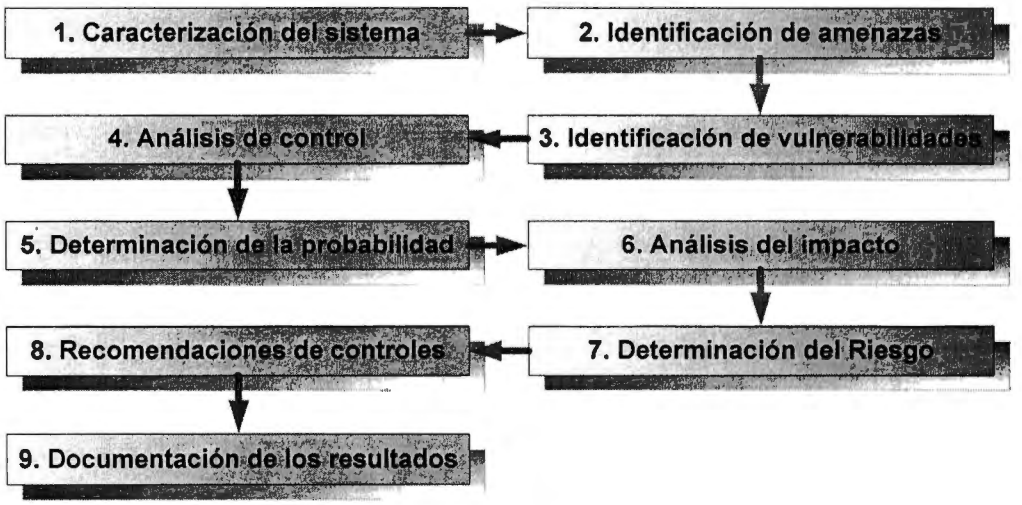

Fig. 10 - Evaluación de riesgos- NIST

2.3.8.1 Caracterización del sistema

El primer paso es determinar el alcance. Se identifican los límites del sistema, así como los recursos y la información que constituyen el sistema.

2.3.8.2 Identificación de amenazas

Se identifican tanto las amenazas potenciales como las causas de las mismas, pudiendo ser estas últimas naturales (terremotos, tornados, etc.), humanas (hackers, terroristas, etc.) o ambientales (fallas eléctricas, contaminación, etc.). En esta etapa también se identifica la motivación de los ataques humanos.

2.3.8.3 Identificación de vulnerabilidades

En esta etapa se crea una lista de posibles vulnerabilidades que puedan ser explotados por las fuentes de amenazas.

2.3.8.4 Análisis de Control

En esta etapa se analizan los controles que hayan sido implementados o que estén planeados a implementarse, para que se minimice la probabilidad de que las amenazas se materialicen.

2.3.8.5 Determinación de la probabilidad

Para determinar la probabilidad de una vulnerabilidad, se debe tomar en cuenta:

1. La motivación de la fuente de la amenaza y su capacidad,
2. La naturaleza de la vulnerabilidad
3. La existencia y efectividad de los controles existentes.

Esta probabilidad se expresa como alta, media o baja, según la tabla 2:

Tabla 2 - Niveles de probabilidad de amenazas. NIST

Nivel de probabilidad	Definición
Alta	La fuente de amenaza esta altamente motivada y suficientemente capaz, los controles para prevenir la vulnerabilidad son inefectivos
Media	La fuente de amenaza es motivada y capaz, pero los controles pueden impedir la explotación de la vulnerabilidad.
Baja	Existe poca motivación o capacidad de la fuente de la amenaza, o los controles existentes impiden significativamente que la vulnerabilidad se pueda explotar.

2.3.8.6 Análisis del impacto

En esta etapa se determina el impacto resultante de que puedan explotar una vulnerabilidad. Este impacto se describe en términos de pérdida o degradación de la integridad, disponibilidad o confidencialidad de los sistemas. Este impacto se cuantifica según la tabla 3:

Tabla 3 - Niveles del impacto. NIST

Nivel del impacto	Definición
Alto	La materialización de la amenaza produce una costosa pérdida de los activos más sensibles. Puede significar la violación, daño o impedimento de la misión, reputación o interés de la organización. Puede causar muertes humanas o lesiones serias.

Medio	La materialización de la amenaza puede causar una costosa perdida de los activos. Puede violar, dañar o impedir la misión, reputación o el interés de la organización. Puede causar heridas a humanos.
Bajo	La materialización de la amenaza puede causar algún daño a los activos. Puede afectar la misión, reputación o interés de la organización

2.3.8.7 Determinación del riesgo

En esta etapa, se determina el nivel de riesgo del sistema. El riesgo se determina en función de:

- La probabilidad de que una fuente de amenaza intente explotar una vulnerabilidad
- La magnitud del impacto si la fuente de amenaza explota la vulnerabilidad
- La adecuación de controles existentes o planeados para reducir o eliminar el riesgo.

El riesgo se calcula con la tabla 4:

Tabla 4 - Matriz del nivel de riesgo. NIST

Probabilidad de la amenaza	Impacto		
	Bajo(10)	Medio(50)	Alto(100)
Alto (1.0)	Bajo (10x1.0=10)	Medio(50x1.0=50)	Alto (100x1.0=100)
Medio(0.5)	Bajo (10x0.5=5)	Medio(50x0.5=25)	Medio (100x0.5=50)
Bajo(0.1)	Bajo (10x0.1=1)	Bajo(50x0.1=5)	Bajo (100x0.1=10)

El riesgo se interpreta según la tabla 5:

Tabla 5 - Niveles de riesgo -NIST

Nivel de riesgo	Descripción y acciones necesarias
Alto	Existe una fuerte necesidad para implementar controles correctivos. El sistema puede seguir operando, pero una acción correctiva debe ponerse en práctica cuanto antes.
Medio	Las acciones correctivas son necesarias y se debe desarrollar un plan para incorporar estas acciones en un tiempo razonable
Bajo	Es necesario determinar si realmente se requieren acciones correctivas o decidir aceptar el riesgo.

2.3.8.8 Recomendaciones de controles

Durante esta etapa los controles que puedan mitigar o eliminar los riesgos identificados son probados. El objetivo es reducir el riesgo a un nivel aceptable.

2.3.8.9 Documentación de los resultados

Una vez que la evaluación del riesgo ha sido completada (al identificar amenazas y vulnerabilidades, y determinar los riesgos y controles correspondientes), los resultados deben ser documentados en un reporte final

Un reporte de evaluación de riesgo, es un reporte administrativo que ayuda a la gerencia a tomar decisiones en políticas, procedimientos, presupuestos y cambios operacionales y administrativos. A diferencia de un reporte de investigación, que busca las áreas de oportunidad, un reporte de evaluación de riesgo no debe ser presentado de manera acusatoria. Este reporte debe tener un enfoque sistemático y analítico que le permita a la administración entender los riesgos y reasignar los recursos necesarios para reducir o corregir las pérdidas potenciales.

2.3.9 RISK TARGETING

Risk Targeting [24] fue desarrollado en 1998, con el propósito de ayudar a reducir el riesgo del problema del año 2000. Sin embargo vale la pena resaltar su metodología de administración de riesgos.

Risk Targeting es el proceso de establecer una jerarquía de componentes por los efectos en las fallas en la operación de la compañía y entonces dirigir los esfuerzos que provean de una reducción mayor de riesgo global hacia los componentes identificados como los más críticos. Ello requiere de un entendimiento de los procesos clave de negocio y los componentes que integran estos procesos. Por lo tanto, una perspectiva de negocio, y no una perspectiva de TI, tiene que encaminar el esfuerzo. Son tres pasos básicos:

1. Entender el negocio
2. Modelar los procesos críticos.
3. Determinar los componentes verdaderamente críticos.

2.3.9.1 Entender el negocio

Cada negocio opera usando una variedad de procesos que manejan las actividades desde la creación y entrega del producto hasta la compensación de los empleados. Estos procesos varían grandemente en prioridad. El fallo de un proceso de misión crítica puede ocasionar que el negocio fracase, mientras que la falla en un proceso menos crítico puede que no cree problemas insuperables por días o incluso semanas. Si la compañía ha recientemente experimentado un

proceso de reingeniería o un plan de recuperación de desastres, estos proceso ya se han identificado y priorizado. Si no, cada área funcional de la compañía tiene que identificar sus propios procesos básicos. Para determinar la lista de las funciones verdaderamente criticas, un ejecutivo debe jerarquizar los procesos en importancia.

2.3.9.2 Modelar los procesos críticos.

Se documenta cada proceso de negocio de inicio a fin ignorando las fronteras corporativas u organizacionales. Este mapa no necesita ser muy detallado, pero tiene que ser completo e incluir todas las entidades internas y externas. Por ejemplo, el mapa de procesos de la creación de un producto empieza con el material en crudo, pasa a través de proveedores intermedios hacia su manufactura, pasa a través de los distribuidores y finalmente a las manos del consumidor.

Después, es necesario identificar las identidades internas y externas, los sistemas computacionales y los equipos de negocio o manufactura que soporta cada proceso. Por último es necesario asegurarse de incluir entidades como las agencias de gobierno, los agentes de transferencia y los distribuidores.

2.3.9.3 Determinar los componentes verdaderamente críticos.

Un componente critico es aquel de que este en riesgo de falla (especialmente si no cuenta con la adecuada protección) o aquel cuya falla pueda poner en peligro la operación de todo el negocio. Para cada componente se determina el impacto de la falla en la operación. Se debe saber donde se existe una continencia alternativa. Aplicar este filtro hace mas claro que una fuente de suministros es especialmente crítica. Finalmente se determina la probabilidad de la falla.

Usando este método para identificar los componentes críticos se obtendrán unos descubrimientos sorprendentes. Algunos de los componentes más críticos no están dentro de la compañía. Otros componentes no son tan críticos como estaban catalogados (debido a que atienden un proceso de negocio menos importante). También, encontrara que otros componentes catalogados como no importantes, realmente lo son (pues soportan procesos críticos de negocio).

2.4 FACTORES A CONSIDERAR AL ADOPTAR UNA METODOLOGÍA DE ADMINISTRACIÓN DE RIESGOS.

La administración de riesgos busca contar con un equilibrio entre el costo de las inversiones en salvaguardas y el valor propio de los activos. Es decir, no debe gastar más allá de lo que valgan los activos. En la figura 11, se contrasta el nivel de riesgo contra la inversión requerida para alcanzar este nivel.

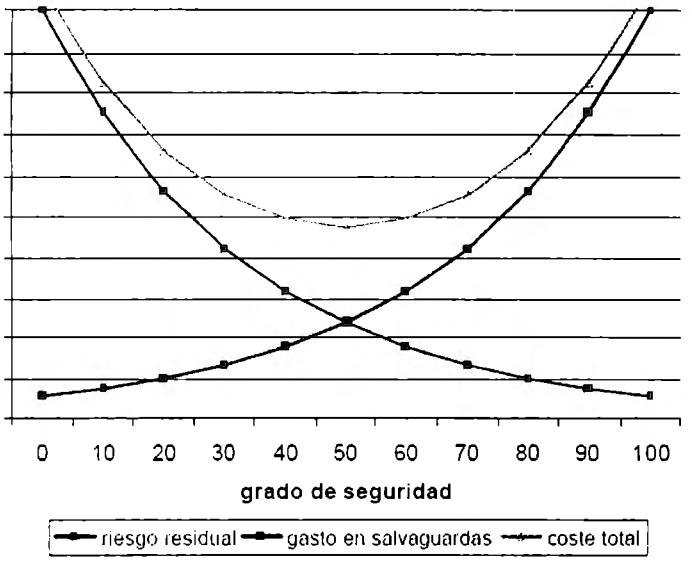

Fig. 11 - Riesgo residual VS Inversión en salvaguardas.

Como se observa claramente en la figura, disminuir el nivel de riesgo, aumenta el costo de la inversión realizada en salvaguardas. Es común, que el nivel de riesgo disminuya significativamente con pequeñas inversiones y el costo de estas inversiones se dispare al tratar de reducir a niveles cercanos al 0% de inseguridad.

Se recomienda que el proceso de administración de riesgos vaya madurando poco a poco y que primero trate de abarcar las secciones más críticas de la empresa y poco a poco trate de abarcar más áreas, al mismo tiempo de que trate de implantar procesos más rigorosos en cuanto a su documentación. Según el ISO 17799, los niveles de administración de riesgos se describen en la tabla 6:

Tabla 6 - Niveles de madurez de la administración de riesgos de seguridad

Nivel	Estado	Definición
0	No existe	La directiva (o el proceso) no está documentada y la organización, anteriormente, no ha tomado conciencia del riesgo de negocios asociado a esta administración de riesgos. Por lo tanto, no ha habido comunicados al respecto.
1	Ad hoc	Es evidente que algunos miembros de la organización han llegado a la conclusión de que la administración de riesgos tiene valor. No obstante, los esfuerzos de administración de riesgos se han llevado a cabo de un modo ad hoc. No hay directivas o procesos documentados y el proceso no se puede repetir por completo. En general, los proyectos de administración de riesgos parecen caóticos y sin coordinación; los resultados no se han medido ni auditado.
2	Repetible	Hay una toma de conciencia de la administración de riesgos en la organización. El proceso de administración de riesgos es repetible aunque inmaduro. El proceso no está totalmente documentado; no obstante, las actividades se realizan periódicamente y la organización está trabajando en establecer un proceso de administración de riesgos exhaustivo con la participación de los directivos. No hay cursos formales

		ni comunicados acerca de la administración de riesgos; la responsabilidad de la implementación está en manos de empleados individuales.
3	Proceso definido	La organización ha tomado una decisión formal de adoptar la administración de riesgos incondicionalmente con el fin de llevar a cabo su programa de seguridad de información. Se ha desarrollado un proceso de línea de base en el que se han definido los objetivos de forma clara con procesos documentados para lograr y medir el éxito. Además, todo el personal dispone de algunos cursos de administración de riesgos rudimentaria. Finalmente, la organización está implementando de forma activa sus procesos de administración de riesgos documentados.
4	Administrado	Hay un conocimiento extendido de la administración de riesgos en todos los niveles de la organización. Los procedimientos de administración de riesgos existen, el proceso está bien definido, la comunicación de la toma de conciencia es muy amplia, hay disponibles cursos rigurosos y se han implementado algunas formas iniciales de medición para determinar la efectividad. Se han dedicado recursos suficientes al programa de administración de riesgos, muchas partes de la organización disfrutan de sus ventajas y el equipo de administración de riesgos de seguridad puede mejorar continuamente sus procesos y herramientas. Se utilizan herramientas de tecnología como ayuda para la administración de riesgos, pero la mayoría de los procedimientos, si no todos, de evaluación de riesgos, identificación de controles y análisis de costo-beneficios son manuales.
5	Optimizado	La organización ha dedicado recursos importantes a la administración de riesgos de seguridad y los miembros del personal miran al futuro intentando determinar los problemas y soluciones que habrá en los meses y años venideros. El proceso de administración de riesgos se ha comprendido bien y se ha automatizado considerablemente mediante el uso de herramientas (desarrolladas internamente o adquiridas a proveedores de software independientes). La causa principal de todos los problemas de seguridad se ha identificado y se han adoptado medidas adecuadas para minimizar el riesgo de repetición. El personal dispone de cursos en distintos niveles de experiencia.

2.5 CONCLUSIONES

En este capítulo se definieron los conceptos de: activo, riesgo, amenaza, vulnerabilidad, impacto y salvaguarda. Posteriormente, se definió la administración de riesgos que es el proceso de identificar, evaluar y reducir los riesgos, seleccionando los controles apropiados (costo/beneficio) para garantizar la operación de la empresa. Para cada riesgo identificado es necesario determinar el tipo de manejo que se le vaya a dar, es decir si se transfiere la responsabilidad, se indemniza, mitiga, retiene o se evita el riesgo.

También se explicaron las principales metodologías para administrar el riesgo, desde las muy sencillas como Risk Targeting y OCTAVE hasta las mas estrictas como NIST y COBIT. En forma genérica, las metodologías de administración de riesgos se resumen en identificar los activos, identificar las amenazas y finalmente determinar la forma en que se manejen los riesgos.

Independientemente de que metodología de administración de riesgos se trate de implementar, el punto critico dentro de este proceso es la evaluación objetiva del riesgo, por ello en el resto de este trabajo estudiaremos a fondo la evaluación de riesgos.

Es necesario resaltar que un proceso de administración de riesgos debe tener un enfoque olístico, pero con una tendencia marcada y predominante hacia el negocio y no dejarse llevar por los problemas o amenazas técnicas que en algún momento se pudieran presentar. Ante algún incidente siempre hay que preguntarse que tanto puede afectar al negocio, y en base a ello, asignarle una prioridad y recursos para resolverlo.

Según Meta Group [41]," mas del 70 % de las organizaciones globales de TI del 2000 (Global 2000 IT Orgarnizations (ITO)) no tienen un proceso de administración de riesgos. Esto es debido a que las ITOs han hecho un trabajo pobre en el entendimiento de la tolerancia del riesgo para los negocios, y generalmente no están disponibles para categorizar los riesgos, evaluar las amenazas, identificar las vulnerabilidades y comunicar el riesgo residual hacia el negocio".

A mi parecer, dependiendo del tamaño de la organización y el nivel de madurez que estas tengan, estas pueden tratar de usar:

- Las pequeñas organizaciones con poca o ninguna regulación, puede emplear un modelo simple e informal, similar a los modelos de Risk Targeting u OCTAVE.
- Organizaciones medianas o más maduras pueden usar: CRAMM, Microsoft o FERMA.
- Organizaciones grandes o con requerimientos más estrictos pueden probar con AS/NZS 4360, COBIT, MAGERIT o NIST

El ITESM-CEM debe tomar un enfoque inicial como Risk Targeting, que permita entender de forma clara y sencilla los procesos de negocios y los recursos informáticos que permiten la operación de estas actividades, para que de esta manera se puedan tomar decisiones que tengan un mayor impacto al reducir el riesgo del ITESM-CEM. Una vez que se logre establecer esta administración de riesgos basado en Risk Targeting, sería posible intentar el uso de un marco de referencia mas estructurado y rígido con el propósito de hacer mejor este trabajo, sin embargo, lo más relevante es no perder el objetivo principal, que es una operación eficiente y eficaz de la Institución y no solamente el logro de alguna certificación que acredite alguna mejor práctica en especifico.

3. EVALUACIÓN DE RIESGOS

La evaluación de riesgos se define como el proceso de identificar y asignar prioridades a los riesgos para la empresa. Para determinar el valor del riesgo en una organización, se realiza una evaluación de riesgos cualitativa o cuantitativa. A continuación describiremos cada uno de estos enfoques.

3.1 EVALUACIÓN CUANTITATIVA

En las evaluaciones de riesgos cuantitativas, el objeto es intentar calcular valores numéricos objetivos para cada uno de los componentes recopilados durante la evaluación de riesgos y el análisis de costo-beneficio. Por ejemplo, se estima el valor verdadero de cada activo de negocios en función de lo que costaría reemplazarlo, lo que costaría en pérdida de productividad, lo que costaría en reputación de marca y en otros valores de negocios directos e indirectos.

Normalmente el análisis de riesgos cuantitativo implica:

- Determinar las amenazas a los que es vulnerable la organización.
- Para cada una de las amenazas, determinar la posibilidad de que se materialicen, es decir, de que realmente ocurra la amenaza.
- Para cada una de las amenazas, determinar el impacto económico de las pérdidas que se producirán en caso de concretarse la amenaza.

Existen algunos puntos débiles importantes que son inherentes a este enfoque y que no se pueden solventar fácilmente. En primer lugar, no existe un modo formal y riguroso de calcular de forma eficaz los valores de los activos y de los controles. Es decir, aunque pueda parecer que ofrece más

detalle, en realidad los valores financieros oscurecen el hecho de que las cifras se basan en estimaciones.

En segundo lugar, las organizaciones que han intentado aplicar meticulosamente todos los aspectos de la administración de riesgos cuantitativa han comprobado que el proceso es excesivamente costoso. Dichos proyectos suelen tardar mucho tiempo en completar su primer ciclo completo y normalmente implican a muchos miembros del personal con discusiones acerca de cómo se han calculado los valores fiscales específicos. En tercer lugar, en organizaciones con valores de alto valor, el costo de exposición puede ser tan alto que se gastaría una ingente cantidad de dinero en mitigar los riesgos a los que estuvieran expuestas. Pero esto no es realista, una organización no gastaría todo su presupuesto en proteger un solo activo, ni siquiera los cinco principales.

Al hacer una evaluación de riesgos cuantitativa, se calculan los siguientes factores y valores:

- Valoración de activos,
- El costo de los controles,
- La determinación del rendimiento de la inversión en seguridad (ROSI)
- El cálculo de valores para la expectativa de pérdida simple (SLE)
- La frecuencia anual (ARO)
- La expectativa de pérdida anual (ALE).

3.1.1 VALORACIÓN DE ACTIVOS

Las cifras calculadas en realidad son estimaciones subjetivas: no existe ninguna herramienta o método para determinar el valor de un activo. Para asignar un valor a un activo, se deben calcular los siguientes tres factores:

- El valor global del activo en la organización.
- La repercusión financiera inmediata de la pérdida del activo.
- La repercusión de negocios indirecta de la pérdida del activo.

3.1.1.1 El valor global del activo en la organización.

Se calcula o estima el valor del activo en términos financieros directos. Consideremos el ejemplo simplificado de las repercusiones de la interrupción temporal de un sitio Web de comercio electrónico que normalmente funciona siete días a la semana, 24 horas al día, y que genera un promedio de $2,000.00 dólares por hora en ingresos procedentes de los pedidos de los clientes.

Se puede establecer con seguridad que el valor anual del sitio Web en términos de ingresos por ventas es de $17,520,000.00 dólares.

3.1.1.2 La repercusión financiera inmediata de la pérdida del activo.

Si simplificamos deliberadamente el ejemplo anterior y suponemos que el sitio Web genera una tasa constante por hora y el mismo sitio Web deja de estar disponible durante seis horas, la exposición calculada es de un 0.000685% por año. Al multiplicar este porcentaje de exposición por el valor anual del activo, se podrá predecir que las pérdidas directamente atribuibles en este caso serían de $12,000.00 dólares. En realidad, la mayoría de los sitios Web de comercio electrónico generan ingresos con unas tasas muy distintas según la hora del día, el día de la semana, la estación, las campañas de publicidad y otros factores. Además, algunos clientes pueden encontrar un sitio Web alternativo que prefieran al original, por lo que dicho sitio Web puede tener una pérdida de usuarios permanente. En realidad, calcular la pérdida de ingresos resulta bastante complejo si se quiere ser preciso y tener en cuenta todos los tipos posibles de pérdida.

3.1.1.3 La repercusión de negocios indirecta de la pérdida del activo.

En este ejemplo, la empresa estima que gastará $10,000.00 dólares en publicidad para contrarrestar la propaganda negativa de una incidencia. Asimismo, la empresa también estima una pérdida de un 0.01% a un 1% de ventas anuales, o $17,520.00 dólares. Mediante la combinación de los gastos de publicidad adicionales y de la pérdida ingresos por ventas anuales, en este caso se puede predecir un total de $27,520.00 dólares en pérdidas indirectas.

3.1.2 EXPECTATIVA DE PÉRDIDA SIMPLE

La expectativa de pérdida simple es la cantidad total de ingresos que se pierde por una única incidencia del riesgo. Se trata de un importe monetario que se asigna a un único suceso que representa la cantidad de pérdida potencial de la empresa, en caso de que una amenaza específica aproveche una vulnerabilidad. (La expectativa de pérdida simple es similar a la repercusión de un análisis de riesgos cualitativo). Dicha expectativa se calcula multiplicando el valor del activo por el factor de exposición. Dicho factor representa el porcentaje de pérdida que una amenaza realizada podría suponer para un determinado activo. Si un conjunto de servidores Web tiene un valor de activo de $150,000.00 dólares y un incendio provoca daños estimados en el 25% de su valor, en este caso la expectativa de pérdida simple será de $37,500.00 dólares. No obstante se trata de un ejemplo muy simplificado, ya que es necesario tener en cuenta otros gastos.

3.1.3 LA FRECUENCIA ANUAL

La frecuencia anual es la cantidad razonable de veces que se espera que ocurra el riesgo durante el año. La elaboración de estas estimaciones resulta muy difícil; existen muy pocos datos actuariales disponibles. Lo que se ha recopilado hasta ahora parece ser información privada que poseen unas pocas empresas de seguros de bienes. Para estimar la frecuencia anual, es necesario recurrir a sus experiencias anteriores y consultar a expertos en administración de riesgos, además de consultores de negocios y de seguridad. La frecuencia anual es similar a la probabilidad de un análisis de riesgos cualitativo y va del 0% (nunca) al 100% (siempre).

3.1.4 EXPECTATIVA DE PÉRDIDA ANUAL

La expectativa de pérdida anual es la cantidad total de dinero que la organización perderá en un año si no se toman medidas para mitigar el riesgo. Para calcular este valor se multiplica la expectativa de pérdida simple por la frecuencia anual. La expectativa de pérdida anual es similar al intervalo relativo de un análisis de riesgo cualitativo.

Por ejemplo, si un incendio en el conjunto de servidores Web de la misma empresa provoca daños valorados en $37,500.00 dólares y la probabilidad, o frecuencia anual, de que se produzca un incendio tiene un valor 0.1 (lo que indica una vez cada diez años), el valor de frecuencia anual sería $3,750.00 dólares (37,500 x 0.1 = 3,750)

La expectativa de pérdida anual proporciona un valor con el que la organización puede trabajar para presupuestar cuánto costará establecer controles o protecciones para prevenir este tipo de daño (en este caso, $3,750.00 dólares o menos al año) y brindar un nivel adecuado de protección. Es importante cuantificar la posibilidad real de un riesgo y el daño, en términos monetarios, que puede causar la amenaza para determinar la cantidad que se puede destinar en la protección contra la posible consecuencia de la amenaza.

3.1.5 DETERMINACIÓN DEL COSTO DE LOS CONTROLES

Determinar el costo de los controles requiere estimaciones precisas de cuánto costará adquirir, probar, implementar, poner en funcionamiento y mantener cada control. Dichos costos deben incluir la compra o desarrollo de la solución de control, la implementación y configuración de la solución de control, el mantenimiento de la misma, la notificación de nuevas directivas o procedimientos relacionados con el nuevo control a los usuarios, los cursos para usuarios y personal de TI acerca de cómo utilizar y dar soporte al control, supervisarlo y combatir la pérdida de comodidad o productividad que el control pueda imponer. Por ejemplo, para reducir el riesgo de que un incendio dañe el conjunto de servidores Web, la organización puede implementar un sistema de extinción de incendios automatizado. Será necesario contratar a un contratista para que diseñe e instale el sistema y, después, se tiene que supervisar continuamente. También será necesario comprobar el sistema periódicamente y, en ocasiones, recargarlo con los químicos que utilice.

3.1.6 RENDIMIENTO DE LA INVERSIÓN EN SEGURIDAD

Finalmente, una vez calculados todos los datos anteriores, podemos calcular el ahorro final que nos daría las inversiones en seguridad, para ello, usamos la siguiente ecuación:

(expectativa de pérdida anual antes del control) – (expectativa de pérdida anual después del control) – (costo anual del control) = rendimiento de la inversión en seguridad

Por ejemplo, la expectativa de pérdida anual de la amenaza de que un pirata informático inutilice un servidor Web es de $12,000.00 dólares y después de implementar la protección sugerida se valora en $3,000.00 dólares. El costo anual del mantenimiento de la protección es de $650.00 dólares, por lo que el rendimiento de la inversión en seguridad es de $8,350.00 dólares al año, tal como se expresa en la siguiente ecuación: 12,000 – 3,000 - 650 = 8,350.

Todos estos cálculos se basan en estimaciones subjetivas. Las cifras clave que proporcionan los resultados no se obtienen de ecuaciones objetivas o de conjuntos de datos actuariales bien definidos, sino de las opiniones de los que realizan la evaluación. El valor del activo, la expectativa de pérdida simple, la frecuencia anual y el costo de los controles son cifras que incorporan los propios participantes

3.2 EVALUACIÓN CUALITATIVA

Las escalas cualitativas permiten avanzar con rapidez, posicionando el valor de cada activo en un orden relativo respecto de los demás, en términos de alto, medio o bajo. Es frecuente plantear estas escalas como "órdenes de magnitud" y, en consecuencia, derivar estimaciones del orden de magnitud del riesgo.

La limitación de las valoraciones cualitativas es que no permiten comparar valores más allá de su orden relativo. No se pueden sumar valores.

El análisis cualitativo de riesgos, rechaza el elemento de probabilidad, concentrándose en las amenazas potenciales y en las características de la red o el sistema que pueden ser vulnerables ante dichas amenazas, para desarrollar posteriormente métodos para prevenir o reducir la probabilidad de las brechas, detectar cuando se producen y reducir y reparar los daños producidos en caso de que se materialicen las amenazas.

La diferencia entre la evaluación de riesgos cualitativa y la cuantitativa estriba en que en la primera no se intentan aplicar valores financieros puros a los activos, pérdidas previstas y costo de controles. En su lugar se calculan valores relativos. El análisis de riesgos normalmente se lleva a cabo mediante la combinación de cuestionarios y talleres colaborativos que implican a personas de varios grupos de la organización, como expertos en seguridad de información, responsables y personal de tecnología de la información, responsables y usuarios de activos de negocios y directivos. Los cuestionarios están diseñados para descubrir los activos y controles que ya están implementados, y la información recopilada puede resultar muy útil durante los talleres

posteriores. En los talleres, los participantes identifican los activos y estiman sus valores relativos. A continuación, intentan determinar las amenazas a las que se enfrenta cada activo y los tipos de vulnerabilidades que pueden aprovechar dichas amenazas en el futuro.

Como se puede comprobar, el proceso básico de las evaluaciones cualitativas es muy similar a lo que sucede en el enfoque cuantitativo. La diferencia se encuentra en los detalles. Las comparaciones entre el valor de un activo y otro son relativas, y los participantes no dedican demasiado tiempo en intentar calcular cifras financieras exactas para la valoración de activos. Lo mismo sucede en el cálculo de las repercusiones posibles si se produce un riesgo y el costo de la implementación de controles.

Las ventajas de un enfoque cualitativo estriban en que se supera la dificultad de calcular cifras exactas para el valor de activos, costo de control, etc., y el proceso exige menos personal. Los proyectos de administración de riesgos cualitativa normalmente empiezan a arrojar resultados importantes al cabo de pocas semanas, mientras que en las organizaciones que optan por un enfoque cuantitativo se aprecian pocas ventajas durante meses, y en ocasiones años, de esfuerzos. El inconveniente de un enfoque cualitativo reside en que las cifras resultantes son vagas.

3.3 COMPARACIÓN ENTRE EL ENFOQUE CUANTITATIVO Y EL ENFOQUE CUALITATIVO

Los enfoques cualitativo y cuantitativo tienen sus ventajas e inconvenientes. Determinadas situaciones pueden demandar que las organizaciones adopten el enfoque cuantitativo. Por el contrario, las organizaciones de pequeño tamaño o con recursos limitados normalmente encontrarán más adecuado el enfoque cualitativo. En la siguiente tabla se resumen las ventajas y los inconvenientes de cada enfoque:

Tabla 7 - Ventajas y desventajas de la evaluación de los enfoques cualitativo y cuantitativo.

	Cuantitativo	Cualitativo
Ventajas	– Se asignan prioridades a los riesgos según las repercusiones financieras; se asignan prioridades de los activos según los valores financieros. – Los resultados facilitan la administración del riesgo por el rendimiento de la inversión en seguridad. – Los resultados se pueden expresar en terminología específica de administración (por ejemplo, los valores	– Permite la visibilidad y la comprensión de la clasificación de riesgos. – Resulta más fácil lograr el consenso. – No es necesario cuantificar la frecuencia de las amenazas. – No es necesario determinar los valores financieros de los activos. – Resulta más fácil involucrar a personas que no sean expertas en seguridad o en informática.

	monetarios y la probabilidad expresados como un porcentaje específico). – La precisión tiende a ser mayor con el tiempo a medida que la organización crea un registro de historial de los datos mientras gana experiencia.	
Desventajas	– Los valores de repercusión asignados a los riesgos se basan en las opiniones subjetivas de los participantes. – El proceso para lograr resultados creíbles y el consenso es muy lento. – Los cálculos pueden ser complejos y lentos. – Los resultados sólo se presentan en términos monetarios y pueden ser difíciles de interpretar por parte de personas sin conocimientos técnicos. – El proceso requiere experiencia, por lo que los participantes no pueden recibir cursos fácilmente durante el mismo.	– No hay una distinción suficiente entre los riesgos importantes. – Resulta difícil invertir en la implementación de controles porque no existe una base para un análisis de costo-beneficio. – Los resultados dependen de la calidad del equipo de administración de riesgos que los hayan creado.

En el pasado, los enfoques cuantitativos parecían dominar la administración de riesgos de seguridad; sin embargo, esto ha cambiado recientemente a medida que cada vez más especialistas admiten que el seguimiento estricto de los procesos de administración de riesgos cuantitativa da lugar a proyectos difíciles y de larga duración que muestran pocas ventajas tangibles.

3.4 ENFOQUE HIBRIDO

El proceso de evaluación de riesgos híbrido combina los mejores elementos de los dos enfoques tradicionales. Este enfoque es considerablemente más rápido que un enfoque cuantitativo tradicional. Sin embargo, proporciona resultados que son más detallados y fácilmente justificables a los ejecutivos que un enfoque cualitativo típico. Mediante la combinación de la simplicidad y la elegancia del enfoque cualitativo con parte del rigor del enfoque cuantitativo. En forma general primero se hace una evaluación cualitativa y para los elementos que se consideren críticos se realiza una evaluación cuantitativa, de este modo, se logra tener una evaluación rápida

y para los elementos críticos se obtienen los elementos financieros para tener bases para las inversiones correspondientes.

3.5 CONCLUSIONES

En este capítulo se describió a detalle el proceso de evaluación de riesgos. La evaluación de riesgos es el proceso de identificar y asignar prioridades a los riesgos. Este proceso consiste en identificar los activos, determinar las amenazas a las que se están expuestos los activos y finalmente determinar el impacto de cada una de las amenazas. Tradicionalmente se tienen dos enfoques para evaluar riesgos: el cuantitativo y el cualitativo. El enfoque cuantitativo busca llevar la evaluación de riesgos hacia términos de pesos y centavos, lo que podría indicar que es un enfoque más preciso al determinar que el activo con riesgo mas alto es aquel que tiene una pérdida estimada más alta. Sin embargo muchos de los cálculos son estimaciones subjetivas. El otro enfoque es el enfoque cualitativo, en donde se determina el riesgo a partir de valores relativos. Básicamente se cuestiona el impacto de cada amenaza en términos relativos (alto, medio o bajo), su probabilidad de ocurrencia (alta, media, o baja) y al hacer una combinación de estas nos da una valoración que sirve como guía para determinar el nivel de riesgo. Finalmente, también se han desarrollado enfoques híbridos que tratan de obtener los beneficios de ambos enfoques.

4. NUEVA PROPUESTA DE EVALUACIÓN DE RIESGOS.

Los esquemas de evaluación de riesgos actuales tratan de determinar el riesgo a un nivel estratégico que permita definir un plan de acción para tratar de administrar de la mejor forma posible este riesgo y conseguir el menor riesgo residual con los recursos que se tengan. Sin embargo los dos esquemas de evaluación de riesgos son poco prácticos en la operación diaria de las organizaciones. Por un lado el esquema cualitativo es demasiado ambiguo a la hora de determinar el nivel de riesgo, ya que se reduce a preguntar a los dueños de los procesos que tan importante creen que sea su proceso (alto, medio o bajo) y también preguntar que tan posible creen que las amenazas se puedan materializar (alto, medio o bajo). Esto al fin y al cabo da una evaluación, pero como vemos es demasiado subjetiva y solo sirve para un acercamiento inicial hacia la administración de riesgos en sí.

Por otro lado, la evaluación de riesgos cuantitativa busca determinar el riesgo en términos de pesos y centavos, tomando en cuenta todos los posibles costos y valores de los activos. Esto es los costos del activo: hardware, software, instalaciones, mano de obra para implantar y mantener el sistema; el impacto si la amenaza se materializa: el costo de oportunidad si el sistema no esta en operación, el costo de la imagen publica, los costos legales (en caso de faltar a un contrato), etc. Este esquema de evaluación es mucho mas objetivo que el esquema cualitativo, sin embargo, suele ser extremadamente complicado obtener todos los costos asociados, ya que implica tomar en cuenta otras muchas variables de las que difícilmente se puede obtener información veraz y oportuna. Por ejemplo, si un atacante logra obtener el password de un usuario ¿Cómo determinar el impacto para la organización de este hecho? Es claro, que depende de que usuario tuvo el atacante la suerte de conseguir, es decir, no es lo mismo que obtenga el password de un empleado común, que de un directivo y mucho menos que de un administrador de sistemas. Además, puede que el atacante use este password para afectar a uno o varios sistemas. A la hora evaluar un riesgo en particular, nos topamos con que es difícil determinar en forma exacta muchos de los costos asociados y normalmente se estiman estos valores según la experiencia. Por lo tanto, además de que este esquema es difícil de implementar resulta también subjetivo e inexacto.

Ninguno de los dos esquemas toma en cuenta la operación del día a día, tan solo suponen la probabilidad de que una amenaza se materialice, suponiendo que se tienen registrados y mapeados los eventos para un riesgo en particular (lo que no sucede, en el ITESM-CEM).

A nivel estratégico pudiera ser que uno u otro esquema sirva para planear la mejor forma de administrar el riesgo y conlleve a una buena planeación para implantar las salvaguardas que permitan mitigar el riesgo. Sin embargo, todo este esfuerzo no suele llevarse a nivel operativo, ya que solo se sabe que se tienen cierta cantidad de servidores que hay que administrar y buscar que todos estén operando de la mejor forma. Solo se cuenta con el sentido común de los administradores para determinar la prioridad de los problemas que se enfrentan día a día.

El problema consiste en determinar la prioridad de un incidente tomando en cuenta el entorno. Es decir, determinar de forma objetiva que atender primero de manera que se atienda lo que pueda tener un mayor impacto al negocio. Si en algún momento hay una serie de problemas con el servidor de correo de Exchange, con los servidores Web, y con uno de los ruteadores es necesario determinar que problema atender primero de manera que se resuelva primero lo que implique un mayor impacto para el negocio.

Nuestra propuesta consiste en definir un nivel de riesgo buscando tener un enfoque holístico, es decir, buscar tomar en cuenta tanto la parte técnica como la parte de negocio. Este enfoque debe abarcar por lo menos los tres componentes de la definición de riesgo: el valor del activo, el nivel de amenaza y el nivel de vulnerabilidad. Sin embargo, la definición de que el riesgo es igual al producto del valor del activo, el nivel de amenaza y el nivel de vulnerabilidad, es prácticamente tan ambigua como una evaluación de riesgos cualitativa común, tan solo estamos agregando una variable mas, el valor del activo.

Es necesario tener una valoración más objetiva, es decir, que no este en términos de alto, medio o bajo (como lo hace la evaluación cualitativa), pero que tampoco sea tan estricta y difícil de conseguir como en el caso de expresar en términos económicos (como lo hace la evaluación cuantitativa). Lo que se busca es tener un indicador intermedio que se encuentre entre estos extremos de evaluación de riesgos.

Para lograr esto, se propone descomponer cada componente en términos que sean fáciles de conseguir y que permitan comparar a los distintos activos. Estos componentes se denominan criterios y a su descomposición le llamaremos factores. Debido a que los distintos factores y criterios varían en importancia estos serán ponderados.

El nivel de riesgo se define como la suma ponderada de criterios. A su vez cada criterio se define como la suma ponderada de factores. Matemáticamente el nivel de riesgo esta definido como:

$$N.R = \sum_{i=0}^{n} X_i C_i \qquad \text{y} \qquad C_i = \sum_{j=0}^{m} Y_j F_j$$

Donde:

N.R. es el Nivel de Riesgo.

X_i= Es la ponderación para el i-ésimo criterio.

C= Es el i-ésimo criterio.

Y_j= Es la ponderación para el j-ésimo factor.

F= Es el j-ésimo factor.

La idea de definir el nivel de riesgo como una suma ponderada ya había sido usada antes por Amrit Tiwana y Mark Keil en su herramienta de evaluación de riesgos de un minuto [30]. Esta herramienta solo se enfoca en el proceso de desarrollo de software y a diferencia del enfoque aquí presentado, solo define el riesgo como la suma ponderada en un solo nivel (solo criterios, sin factores).

Lo relevante en esta propuesta de evolución de riesgos es la descomposición de los criterios en factores, ya que esto permite definir cada criterio en términos simples. Por ejemplo, en vez de preguntarse ¿Qué tan vulnerable o expuesto esta un activo?, podemos hacernos unas preguntas más simples como ¿Cuántos puertos tiene abiertos? , ¿Cuántas vulnerabilidades tiene identificadas?, ¿Qué salvaguardas tiene implementadas?, ¿Tiene una IP homologada?, etc. Una vez contestadas todas estas preguntas podemos ponderarlas y sumarlas para así poder contestar con un mayor grado de certeza el nivel de vulnerabilidad o de exposición de un equipo específico.

La idea detrás de esta propuesta es la una de las máximas en programación: "Divide y vencerás", es decir, es necesario partir una idea compleja en varias simples y enfocarse a resolver las ideas más simples, para que una vez resueltas estas, se unan las distintas partes simples para resolver la idea más compleja.

A continuación se describirán los criterios que consideran relevantes para obtención del nivel de riesgo. Es importante resaltar que estos criterios tan solo son una propuesta y se pretende que en un futuro entren en debate para determinar cuales son los criterios más adecuados para determinar el nivel de riesgo.

4.1 CRITERIOS A CONSIDERAR EN LA EVALUACIÓN DE RIESGOS.

Recordemos que el riesgo es definido como "La combinación de que una amenaza explote una vulnerabilidad que pueda causar un daño a un activo"[9]. Si descomponemos esta definición veremos que tiene tres componentes principales: la amenaza, la vulnerabilidad y el activo. Cada uno de los componentes del riesgo aumenta o disminuye el nivel del riesgo.

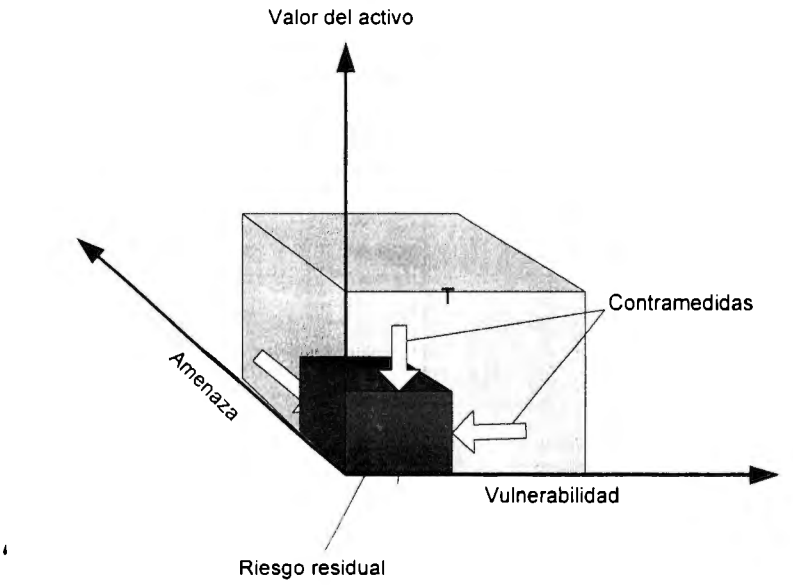

Fig. 12 - El nivel del riesgo

Para cada uno de estos componentes describiremos los factores y salvaguardas que se toman en cuenta para definirlos.

4.1.1 NIVEL DE AMENAZA

Una amenaza se define como la probabilidad de algo o alguien cause daño a un activo, es decir, todo aquello que pueda afectar la integridad, confidencialidad o disponibilidad del activo. En otras palabras, las amenazas serían los posibles ataques hacia los activos. Por ello, se propone que los factores y salvaguardas a considerar sean los siguientes:

- Factores :
  - Nivel de intromisión
  - Nivel de Ataque hacia los equipos.
- Salvaguardas:
  - Respaldos automatizados
  - Redundancia en el servicio

4.1.1.1 Factores del nivel de amenaza

Los factores del nivel de amenaza buscan medir el grado de daño que pueda afectar a los equipos. Por ello se sugiere que se mida tanto el nivel de intromisión como el nivel de ataque.

4.1.1.1.1 Nivel de intromisión

Este nivel es el grado de certeza de que un atacante ha podido introducirse al equipo. Este nivel aumenta en caso de que: la victima responda al intento de conexión del atacante, aumenta más si la conexión persiste por un periodo prolongado y aun más si la victima se vuelve atacante.

4.1.1.1.2 Nivel de Ataque hacia los equipos.

Es el nivel de ataque detectado hacia el equipo independientemente si el ataque es efectivo o no, es decir la cantidad de patrones de ataques detectados hacia el equipo.

4.1.1.2 Salvaguardas del nivel de amenaza

Las salvaguardas sugeridas en este criterio, buscan garantizar la disponibilidad o la rápida recuperación de los servicios, por ello sugerimos evaluar si se cuenta o no con:

4.1.1.2.1 Respaldos automatizados

Una copia de respaldo permite recuperar información en caso de que el sistema falle, sin embargo, esta copia debe ser lo más actualizada posible, es por ello que es necesario que los respaldos se automaticen.

4.1.1.2.2 Redundancia en el servicio

Si se cuenta con redundancia en la afectación de un servicio, es muy probable que se pueda seguir operando aun cuando uno de sus componentes falle o se le deba dar mantenimiento.

4.1.2 NIVEL DE VULNERABILIDAD

Una vulnerabilidad es una debilidad que un atacante puede aprovechar para comprometer el sistema, por ello, en este criterio estarán todos los factores que de alguna u otra manera expongan al equipo. Los factores y salvaguardas a considerar son:

- Factores :
  - Número de puertos abiertos

- Número de vulnerabilidades de nivel alto
- Número de vulnerabilidades de nivel medio
- Número de vulnerabilidades de nivel bajo
- Dirección IP homologada.

- Salvaguardas
  - Antivirus actualizado
  - Firewall
  - Instalación de parches automatizado.
  - Control de acceso exclusivo

4.1.2.1 Factores del nivel de vulnerabilidad

Los factores del nivel de vulnerabilidad buscan medir el grado de exposición de los equipos, por lo que se proponen los siguientes indicadores:

4.1.2.1.1 Número de puertos abiertos

Dado que cada puerto al fin y al cabo es un canal que acepta alguna comunicación, este puede ser usado para generar un buffer overflow del equipo y de este modo lograr penetrarlo o afectar la disponibilidad del equipo.

4.1.2.1.2 Número de vulnerabilidades de nivel alto

Una vulnerabilidad de nivel alto es aquella que permite el acceso remoto inmediato o la ejecución inmediata de comandos.

4.1.2.1.3 Número de vulnerabilidades de nivel medio

Una vulnerabilidad de nivel medio es aquella que tiene el potencial de permitir el acceso o la ejecución de código por medio de un procedimiento complicado.

4.1.2.1.4 Número de vulnerabilidades de nivel bajo

Una vulnerabilidad de nivel bajo es aquella que deniega el servicio o provee información que puede ser usada en un ataque más estructurado, pero por si misma no puede usarse para ganar el acceso sin autorización.

4.1.2.1.5 Dirección IP homologada.

Una dirección IP homologada es una dirección que puede ser identificada en todo Internet, es decir, es una dirección pública, de forma que cualquiera pueda acceder al equipo (Ya sea directamente asignada o mediante NAT (Network Access Translation).

4.1.2.2 Salvaguardas del nivel de vulnerabilidad

Las salvaguardas del nivel de vulnerabilidad son todos aquellos mecanismos que de alguna forma disminuyan el grado de exposición del equipo, por lo que se proponen los siguientes:

4.1.2.2.1 Antivirus actualizado

Debido a que es común que los ataques sean ejecutados por virus informáticos, es crucial que los equipos tengan implementado un buen antivirus, a fin de que este detecte y elimine estos virus. Sin embargo, no es suficiente con que ese instalado este software, ya que la detección se realiza generalmente mediante patrones conocidos como las firmas de los virus. Es necesario asegurarse que el sistema siempre cuente con las últimas firmas publicadas.

4.1.2.2.2 Firewall

Un firewall es un sistema que evita conexiones no permitidas. El firewall puede ser instalado en el mismo equipo o afuera de este.

4.1.2.2.3 Instalación de parches automatizado.

Las vulnerabilidades de los equipos son fallas en los sistemas o en las configuraciones de los mismos. En caso de que sean fallas de los propios sistemas la forma de corregirlos es

mediante la aplicación del parche correspondiente, por ello, es necesario que el equipo se mantenga siempre al último nivel de parches.

4.1.2.2.4 Control de acceso exclusivo

Este sistema de control garantiza que el sistema pueda ser acezado exclusivamente por aquellos que tienen que hacerlo.

4.1.3 VALOR DEL ACTIVO

El valor del un activo no solo se refiere al costo económico del mismo, sino al valor del mismo, es decir a el nivel de importancia para la organización del mismo. Por ello en este criterio estarán todos aquellos factores que de alguna u otra manera representen la criticidad del activo.

- Factores:
  - Nivel de la información.
  - Número de usuarios.
  - Nivel de los usuarios.
  - Nivel de dependencia.
  - Criticidad según el dueño.
- Salvaguarda:
  - Porcentaje de canales cifrados en las comunicaciones

4.1.3.1 Factores del valor del activo

Los factores del valor del activo son aquellos que permiten medir la importancia del equipo para la organización, por ello se sugieren los siguientes:

4.1.3.1.1 Nivel de la información

El nivel de información es el etiquetamiento que se le da a la misma para identificar su nivel de confidencialidad. Estos niveles de información se describen en la siguiente tabla:

Tabla 8 - Nivel de la información según su confidencialidad.

Nivel de la información	Descripción
Ultra secreto	Información restringida a empleados con una predeterminada base de quienes necesiten conocerla, donde un estricto mantenimiento y registro histórico de acceso es necesario.
Secreto	Información personal, técnica o de negocio con mayor sensibilidad, donde el daño a la organización podría resultar en un impacto serio de divulgación fuera de la organización. La información es restringida a quienes necesiten saberla.
Confidencial	Información personal, técnica o de negocio con mayor sensibilidad, donde la divulgación debe ser restringida a aquellos empleados que necesitan conocerla para hacer sus labores.
De uso interno	Información de tipo personal, médica, técnica o de negocio restringida para usar dentro de la organización y para propósitos relacionados con la organización.
Pública	Información aprobada para divulgación pública

4.1.3.1.2 Número de usuarios.

El número de usuarios representa la cantidad de usuarios que tienen acceso al sistema.

4.1.3.1.3 Nivel de los usuarios.

Representa el público al que esta enfocado el sistema y define el nivel de seguridad de los usuarios según la política de seguridad del ITESM-CEM. Estos niveles se describen en la siguiente tabla:

Tabla 9 - Nivel de seguridad de los usuarios según la política de seguridad del ITESM-CEM

Nivel de los usuarios	Descripción
1ra Línea, Admón. Gral.	Directores de Primera línea, Administrador de Seguridad, Administrador General de Sistemas
2da Línea, Informática	Directores de Segunda línea, Personal de la dirección de informática
Empleados	Personal del ITESM-CEM
Alumnos	Alumnos
Usuarios temporales	Cuentas provisionales, usuarios temporales
Usuarios degradados	Usuarios degradados

4.1.3.1.4 Nivel de dependencia.

Determina el grado de dependencia que tienen unos sistemas con respecto a otros, a fin de conservar la disponibilidad de los mismos. Estos grados se describen en la siguiente tabla:

Tabla 10 - Nivel de dependencia de los servicios

Nivel de dependencia	Descripción
Núcleo de Ruteo	Ruteo central. Es el corazón de la red.
Ruteo	Equipo activo que permite el funcionamiento de la red
DHCP	Dinamic Host Configuration Protocol. Permite que una maquina obtenga su dirección IP automáticamente
DNS	DNS (Servicio de nombres de dominio) Traducen un nombre a una dirección IP
Red Externa	Permiten o controlan el acceso hacia Internet Pueden ser : -Firewalls de frontera -Servidores NAT (Network Address Traslation)
Autenticación	Sistemas que proveen el servicio de autenticación. Estas pueden ser: -Directorios -Controladores de dominio -Servidores de correo -Servidores RAS -Servidores NIS -Servidores de Kerberos
Base de Datos	Contenedor de información que usan las aplicaciones que dan servicio
Aplicación	Aplicación específica que presta un determinado servicio final, es decir, no hay ningún sistema que dependa del mismo.

4.1.3.1.5 Criticidad según el dueño.

Representa el nivel de importancia del equipo para el dueño, de acuerdo a la siguiente ponderación:

Tabla 11 - Nivel de criticidad según el dueño

Nivel de criticidad	Descripción
Alta	La operación de la institución se detiene
Media alta	El daño es considerable, servicios críticos dejan de funcionar.
Media	El daño es mínimo, algunos servicios se dan de manera intermitente
Media baja	Algunos servicios con dependencias se ven afectados pero se da el servio con relativa normalidad.

Baja	Solamente algunos usuarios se ven afectados pero el servicio continua
Nula	El servidor pasa desapercibido si desaparece o se interrumpe el servicio.

4.1.3.2 Salvaguardas del valor del activo

Las salvaguardas sugeridas en este criterio buscan de alguna forma garantizar la confidencialidad e integridad de la información, entre estos se sugiere:

4.1.3.2.1 Porcentaje de canales cifrados en las comunicaciones

Este mecanismo evita que el atacante pueda entender las comunicaciones en caso de que las intercepte.

4.2 EVALUACIÓN DE CRITERIOS Y FACTORES.

Al fin de poder medir el nivel de riesgo de los equipos es necesario transformar la evaluación de cada factor y salvaguarda de manera que permita conjuntar estas evaluaciones y poder tener un indicador final.

Se propone que cada factor o salvaguarda sea mapeado a un valor entre cero y uno el cual indique el porcentaje de cumplimiento del mismo. Dado que un factor es una medida que indica un cierto grado de riesgo, sus valores serán positivos, mientras que una salvaguarda busca reducir el riesgo, por lo que sus valores serán negativos.

Una vez que se haya hecho el mapeo del factor (o salvaguarda) este será ponderado, buscando que para cada criterio la suma de las ponderaciones sea uno (un 100 por ciento). De esta forma, este proceso de ponderación se puede volver a aplicar para determinar el nivel de riesgo total.

En este punto nos enfrentamos con el problema que la forma de evaluación de cada factor o salvaguarda varia pudiéndose agrupar en:

- Evaluaciones de verdadero/falso
- Evaluaciones proporcionales al máximo del mismo factor.
- Evaluaciones por selección, aplicando una escala.
- Evaluaciones por porcentaje directo.

4.2.1 EVALUACIONES DE VERDADERO/FALSO

En este tipo de evaluación se cuestiona si se cuenta o no con el factor. Y estos valores se traducen a un uno (en el caso de ser verdadero) o a un cero (en el caso de ser falso). Un ejemplo de este tipo de evaluación seria si el hecho de que el equipo cuenta o no con una dirección IP homologada.

4.2.2 EVALUACIONES PROPORCIONALES AL MÁXIMO DEL MISMO FACTOR.

Las evaluaciones proporcionales, se calculan bajo la siguiente fórmula:

$$F = \frac{X}{MAX(X)}$$

Donde:

F es la evaluación proporcional del factor.

X es evaluación del factor.

MAX(X) es el valor máximo de todas las evaluaciones del mismo factor.

Un ejemplo de este tipo de evaluación correspondería a la cantidad de usuarios que atiende el equipo. Es decir, si tenemos dos equipos, uno atendiendo a 100 usuarios y otro atendiendo a 40 usuarios, entonces el primer equipo tendría una evaluación de 1 (100/100), mientras que el segundo de 0.4 (40/100). De esta manera, determinamos que el primer equipo presenta un mayor riesgo por que afecta a más usuarios.

4.2.3 EVALUACIONES POR SELECCIÓN, APLICANDO UNA ESCALA.

Este tipo de evaluaciones, sencillamente traducen un conjunto de valores previamente definidos a una escala. Un ejemplo de este tipo de evaluaciones correspondería a definir el nivel de información que maneja un equipo. En este caso se podría aplicar la siguiente tabla:

Tabla 12 - Escala del factor nivel de la información

Nivel de la información	Escala
Ultra secreto	100 %
Secreto	80 %

Confidencial	60 %
De uso interno	40 %
Pública	20%

4.2.4 EVALUACIONES POR PORCENTAJE DIRECTO.

En este tipo de evaluaciones se cuestiona directamente que tanto se aplica el factor o salvaguarda. Un ejemplo de este tipo de evaluaciones sería el porcentaje de canales cifrados en las comunicaciones

4.3 EJEMPLO DE EVALUACIÓN DE RIESGOS.

Para fines ilustrativos supongamos que en una organización cuenta con dos servidores y se quiere saber cual de ellos representa un mayor riesgo para la organización. El primer servidor es un servidor Windows cuya función principal es la de controlador de dominio. El segundo servidor es el servidor de correo de alumnos que corre bajo una plataforma Linux.

Lo primero que hay que hacer para determinar el nivel del riesgo de los equipos informáticos es establecer las ponderaciones para cada uno de los criterios, factores y salvaguardas. En este caso, se hizo una reunión directiva y se establecieron las siguientes ponderaciones:

Tabla 13 - Ejemplo de ponderación de criterios y factores

Criterio	Ponderación del criterio.	Factor / Salvaguarda	Tipo de Factor o Salvaguarda	Ponderación
Nivel de amenaza	30 %	Nivel de intromisión	Proporcional al máximo	80 %
		Nivel de ataque	Proporcional al máximo	20 %
		Respaldos Automatizados	Verdadero / Falso	- 20 %
		Redundancia en el servicio	Verdadero /Falso	- 30 %
Nivel de vulnerabilidad	30 %	Número de puertos abiertos	Proporcional al máximo	10 %
		Número de vulnerabilidades de alto nivel	Proporcional al máximo	50 %
		Número de vulnerabilidades de nivel medio	Proporcional al máximo	20 %
		Número de vulnerabilidades	Proporcional al máximo	5 %

		de bajo nivel		
		Dirección IP homologada	Verdadero /Falso	15%
		Antivirus actualizado	Verdadero /Falso	- 10 %
		Firewall	Verdadero /Falso	- 15 %
		Instalación de parches automatizado	Verdadero /Falso	- 10 %
		Control de acceso exclusivo	Verdadero /Falso	- 5 %
Valor del activo	40 %	Nivel de la información	Por selección, aplicando una escala	20 %
		Número de usuarios	Proporcional al máximo	15 %
		Nivel de los usuarios	Por selección, aplicando una escala	20 %
		Nivel de dependencia	Por selección, aplicando una escala	20 %
		Criticidad según el dueño	Por selección, aplicando una escala	25 %
		Porcentaje de canales cifrados	Por porcentaje directo	-10 %

Una vez establecido las ponderaciones para cada factor y salvaguarda es simple evaluar el nivel de riesgo. Para este caso supongamos los siguientes datos:

Los datos presentados en la tabla 14 son hipotéticos, pero después veremos que es posible automatizar la obtención de algunos de ellos, al interrogar directamente a la base de datos de alguna aplicación que permita su medición. Entre los datos que se pueden automatizar son: El nivel de intromisión (mediante OSSIM[1]), el nivel de ataque (mediante OSSIM), el número de puertos abiertos (mediante OSSIM) y la cantidad de vulnerabilidades de niveles bajo, medio y alto (mediante ISS[2]). El resto de factores o criterios se realiza mediante un cuestionario directamente al responsable del equipo.

[1] OSSIM (Open Source Security Information Manager), es un sistema de código abierto que integra varias herramientas como SNORT, NESSUS, NMAP, entre otras, para determinar el nivel de ataque y compromiso de los equipos. http://www.ossim.net

[2] ISS (Internet Security Scanner), es una herramienta comercial para el monitoreo de vulnerabilidades. http://www.iss.net.

Tabla 14 - Datos para la evaluación de riesgo

Factor/salvaguarda	*Servidor de dominio (S1)*	*Servidor de correo de alumnos (S2)*
Nivel de intromisión	2	3
Nivel de Ataque	25	15
Respaldos Automatizados	Verdadero (1)	Falso(0)
Redundancia en el servicio	Falso(0)	Verdadero (1)
Número de puertos abiertos	9	7
Número de vulnerabilidades de alto nivel	3	1
Número de vulnerabilidades de nivel medio	5	5
Número de vulnerabilidades de bajo nivel	11	2
Dirección IP homologada	Falso(0)	Verdadero (1)
Antivirus actualizado	Verdadero (1)	Falso(0)
Firewall	Verdadero (1)	Verdadero (1)
Instalación de parches automatizado	Verdadero (1)	Falso(0)
Control de acceso exclusivo	Falso(0)	Falso(0)
Nivel de la información	De uso interno (40 %)	Confidencial (60%)
Número de usuarios	3000	2000
Nivel de los usuarios	Usuarios degradados (16.6 %)	Alumnos (50 %)
Nivel de dependencia	Autenticación (75 %)	Autenticación (75 %)
Criticidad según el	Alta (100 %)	Media alta

dueño		(80 %)
Porcentaje de canales cifrados	70 %	40%

Y finalmente podemos hacer la evaluación.

Tabla 15 - Ejemplo de evaluación de riesgo.

		S1	S2	S1	S2	S1	S2	S1	S2
Nivel de amenaza (20 %)	Nivel de intromisión (80 %)	67%	100%	53%	80%	53%	62%	16%	19%
	Nivel de Ataque (20%)	100%	60%	20%	12%				
	Respaldos (-20%)	100%	0%	-20%	0%				
	Redundancia (- 30%)	0%	100%	0%	-30%				
Nivel de vulnerabilidad (30 %)	Puertos abiertos (10%)	100%	78%	10%	8%	50%	45%	15%	14%
	Vulnerabilidades de alto nivel (50%)	100%	33%	50%	17%				
	Vulnerabilidades de nivel medio (20 %)	100%	100%	20%	20%				
	Vulnerabilidades de bajo nivel (5 %)	100%	18%	5%	1%				
	Dirección IP homologada (15%)	0%	100%	0%	15%				
	Antivirus actualizado (-10%)	100%	0%	-10%	0%				
	Firewall (- 15 %)	100%	100%	-15%	-15%				
	Instalación de parches (-10%)	100%	0%	-10%	0%				
	Control de acceso exclusivo (-5%)	0%	0%	0%	0%				
Valor del activo (40%)	Nivel de la información (-20%)	40%	60%	8%	12%	59%	63%	24%	25%
	Número de usuarios (15%)	100%	67%	15%	10%				
	Nivel de los usuarios (20%)	17%	50%	3%	10%				
	Nivel de dependencia (20%)	75%	75%	15%	15%				
	Criticidad según el dueño (25%)	100%	80%	25%	20%				
	Porcentaje de canales cifrados (-10%)	70%	40%	-7%	-4%				
							Nivel de riesgo	55%	57%

Una vez hecha la evaluación es sencillo observar que el servidor S2 (correo de alumnos) representa una mayor riesgo, teniendo un enfoque holístico, por lo que debe tener mayor prioridad al momento de determinar los nuevos controles a implantar.

4.4 VENTAJAS Y DESVENTAJAS DE LA EVALUACIÓN DE RIESGOS PONDERADA.

La evaluación de riesgos presenta varias ventajas. En primer lugar se tiene un enfoque holístico, lo que permite asignar prioridades según las repercusiones de negocio. Esto es debido a que se toman en cuenta todo un conjunto de criterios, factores y salvaguardas, que juntos permiten tener una valoración general del nivel de riesgo. En segundo lugar, define en forma sencilla cada uno de los criterios al descomponerlos en factores. Esta descomposición permite aterrizar un concepto complejo en conceptos sencillos y mesurables que al juntarlos nuevamente nos permiten definir el concepto complejo. En tercer lugar, la evaluación de riesgos es sencilla y rápida. Una vez que se definen los criterios, factores, salvaguardas y sus respectivas ponderaciones, la evaluación de riesgos consiste en contestar un simple cuestionario. Esta facilidad de evaluación permite que este proceso sea económico, lo que a su vez permite que el proceso se pueda realizar continuamente y así observar el progreso de cada equipo. En cuarto lugar, permite identificar objetivamente cuales son los equipos con mayor riesgo de forma consensuada, pues es claro y transparente cual fue el proceso de medición. Además, no es necesario conseguir información histórica o financiera, lo facilita enormemente la medición del riesgo.

La evaluación de riesgos ponderada solo tiene dos desventajas. En primer lugar, los resultados de la evaluación de riesgos son relativos al mismo ambiente que se esta evaluando, por lo que no se deben comparar ambientes diferentes, en este caso, mas bien se deben unir ambos ambientes y hacer una evaluación global. En segundo lugar, resulta difícil invertir en la implementación de controles porque no existe una base para un análisis de costo-beneficio.

En general la evaluación de riesgos ponderada preserva todas las ventajas de la evaluación de riesgos cualitativa común, pero dándole solidez a sus resultados, pues se basa en criterios y factores específicos, en vez de solo tomar en cuenta la opinión subjetiva de unos cuantos.

Por otro lado este enfoque puede ser tan simple o complejo como se requiera (se agregan o eliminan criterios y factores) y es completamente adaptable a cualquier organización.

Su desventaja principal radica en no contar con una base financiera. Pero como ya hemos visto, el enfoque cuantitativo aunque si estable una base económica realmente los cálculos son muy subjetivos, pues se abusa de las proyecciones.

4.5 CONCLUSIONES

Los enfoques de evaluación de riesgos tradicionales son ineficientes. Por un lado en el enfoque cuantitativo es difícil conseguir los datos de los costos indirectos y con respecto a los costos directos suelen proyecciones poco reales. Por otro lado el enfoque cualitativo es demasiado subjetivo ya que se basa solo en cuestionar en términos relativos (alto, bajo o alto) el nivel de la

amenaza y su probabilidad de ocurrencia. Ante esta situación se propone un nuevo enfoque de evaluación de riesgos. El enfoque ponderado de evaluación de riesgos consiste en definir el nivel de riesgos como la suma ponderada de criterios, a su vez cada criterio se define como la suma ponderada de factores. Esta doble ponderación permite definir un criterio abstracto (como puede ser el nivel de vulnerabilidad) en factores y salvaguardas específicos (puertos abiertos, cantidad de vulnerabilidades detectadas, etc.) los cuales sean sencillos de evaluar para que después de ponderarlos y sumarlos obtengamos el nivel de riesgo.

La idea de evaluar el riesgo como una suma ponderada ya había sido expuesta por Tiwana y Keil [30] en su herramienta para evaluar de riesgos en un minuto. Tiwana y Keil desarrollaron una herramienta para medir el riesgo de los proyectos de desarrollo de software buscando ponderar la parte técnica del desarrollo con el conocimiento de las necesidades del cliente. Su idea solo se concentra en hacer una suma ponderada, es decir, su nivel de riesgo esta definido por:

Donde:

N.R. es el Nivel de Riesgo.

X_i= Es la ponderación para el i-ésimo criterio.

C_i= Es el i-ésimo criterio.

Es decir, definen una serie de criterios, les asignan una ponderación y finalmente evalúan cada criterio, para que al sumarlos obtengan el nivel de riesgo.

A diferencia de esta tesis, Tiwana y Keil no descomponen cada criterio en factores, por lo que la evaluación de cada criterio es totalmente subjetiva y difícil de calcular.

5. SISTEMA DE DEFINICIÓN Y EVALUACIÓN DE RIESGOS INFORMÁTICOS PARA EL ITESM CEM (SDERI)

A fin de poder manejar de una mejor forma las evaluaciones de riesgo ponderadas, se creo un sistema para administrar esta información.

Este sistema tiene tres objetivos. El primero es la creación de una herramienta que permita definir el nivel de riesgo. El segundo es la evaluación de equipos en base a la definición del nivel de riesgo establecida. Y el tercero es tener la capacidad de evaluar automáticamente ciertos factores que, mediante un query, permitan realizar esta evaluación.

5.1 Análisis de SDERI

La herramienta desarrollada recibió el nombre de SDERI (Sistema de Definición y Evaluación de Riesgos Informáticos)

5.1.1 DEFINICIÓN DEL NIVEL DE RIESGO.

El principal objetivo de SDERI es definir el nivel de riesgo. Recordemos que el riesgo es la probabilidad de que ocurra un evento adverso que cause daño a un activo. Por lo tanto, el riesgo es un indicador que combina por lo menos tres criterios: el valor del activo, el nivel de amenaza y el nivel de vulnerabilidad.

En forma más genérica podemos decir que el nivel de riesgo esta definido como una suma ponderada de criterios.

A fin de que sea más sencillo definir cada criterio, se propone descomponerlos en términos más simples que se puedan ponderar y sumar para integrar el criterio nuevamente. A esta

descomposición de los criterios le llamaremos factores. Por lo tanto, matemáticamente el nivel de riesgo esta definido como:

y

Donde:

N.R= Es el Nivel de Riesgo.

Xi= Es la ponderación para el i-ésimo criterio.

C_i= Es el i-ésimo criterio.

Y_j= Es la ponderación para el j-ésimo factor.

F_j= Es el j-ésimo factor.

5.1.2 EVALUACIÓN DEL NIVEL DE RIESGO DE EQUIPOS.

Una vez que se tenga definido el nivel de riesgo, procedemos a hacer una evaluación del mismo que no es otra cosa que asignar un valor a cada uno de los factores para así poder ponderarlos y sumarlos para poder calcular el nivel de riesgo para un equipo en específico.

La evaluación de cada factor debe indicar el grado de cumplimiento del mismo. Es importante que este grado de evaluación sea en forma porcentual (un valor entre cero y uno), ya que permite normalizar los distintos factores y de esta manera da pié a que se puedan ponderar y sumar. Si no se hiciera esta normalización estaríamos en el caso de querer sumar y ponderar peras y manzanas, lo que no es correcto. En cambio si hacemos esta normalización, estamos sumando y ponderando porcentajes los cuales ya están en una misma escala, lo que permite su manejo matemático.

Además, es necesario asegurarse que para un criterio en específico la suma de ponderaciones de sus factores sea uno (100 %) ya que esto permite realizar nuevamente la ponderación y suma a nivel de criterio para así determinar el nivel de riesgo.

5.1.3 EVALUACIÓN AUTOMÁTICA DE FACTORES.

En el entorno informático la única constante es el cambio. Frecuentemente se agregan y quitan equipos, se descubren y parchan nuevas vulnerabilidades, se crean nuevos mecanismos de ataques y defensas, etc. Por ello es necesario tener una evaluación continua que realmente permita determinar el nivel de riesgo en tiempo real y no solo una fotografía en el tiempo. Para ello, se pretende automatizar la evaluación de algunos factores mediante la ayuda de las herramientas respectivas que determinen estas evaluaciones. Por ejemplo, se desea que se le pregunte a un detector de intrusos el nivel de ataque hacia los equipos; a un detector de vulnerabilidades la cantidad de vulnerabilidades detectadas, etc.

Para llevar a cabo esta automatización lo único que se hará es hacer un query a la base de datos de la herramienta para el factor que se desea automatizar. Se propone usar la dirección IP del equipo como llave para consultar el valor deseado. Esta automatización permitirá que el nivel de riesgo varíe en tiempo real, y de esta forma tener un indicador mucho más real del entorno.

5.2 DISEÑO DE SDERI

Para poder realizar cada uno de los objetivos de este sistema se requiere la administración de su información relacionada, es decir, se requieren hacer altas, bajas y cambios. Para poder definir el nivel de riesgo, se requiere hacer una administración de criterios, factores y salvaguardas; para poder evaluar el nivel de riesgo de equipos, necesitamos hacer una administración tanto de equipos como de estas evaluaciones; y para poder automatizar la evaluación de algunos factores, se requiere administrar la información relacionada con los querys. A fin de hacer una programación más eficiente se optó por crear una infraestructura que permita hacer esta administración de manera genérica. Es decir, se generó una infraestructura que permita a partir de un query SQL generar de forma automatizada las formas de captura para las operaciones de alta, baja y cambio de este query.

Una vez teniendo las clases de la administración genérica de información, sencillamente se derivan las clases que implementen en si los objetivos del sistema SDERI.

5.2.1 DISEÑO DE LA BASE DE DATOS.

Para cumplir con cada uno de los objetivos del sistema SDERI, necesitamos almacenar toda la información necesaria para su implementación.

Para el objetivo de la definición del riesgo, necesitamos almacenar: los criterios, factores y salvaguardas (que en realidad son factores que reducen el riesgo). Para evaluar el nivel de riesgo, necesitamos almacenar los equipos y las evaluaciones de los factores, así como un conjunto de valores posibles para los factores que sean de tipo de selección aplicando una escala. Finalmente, para las evaluaciones automáticas, necesitamos almacenar los querys y las conexiones a las bases de datos que permitan hacer esta automatización.

Una vez habiendo desglosado y normalizado toda la información necesaria para el sistema SDERI, se concluyo en el diagrama de entidad relación presentado en la figura 13.

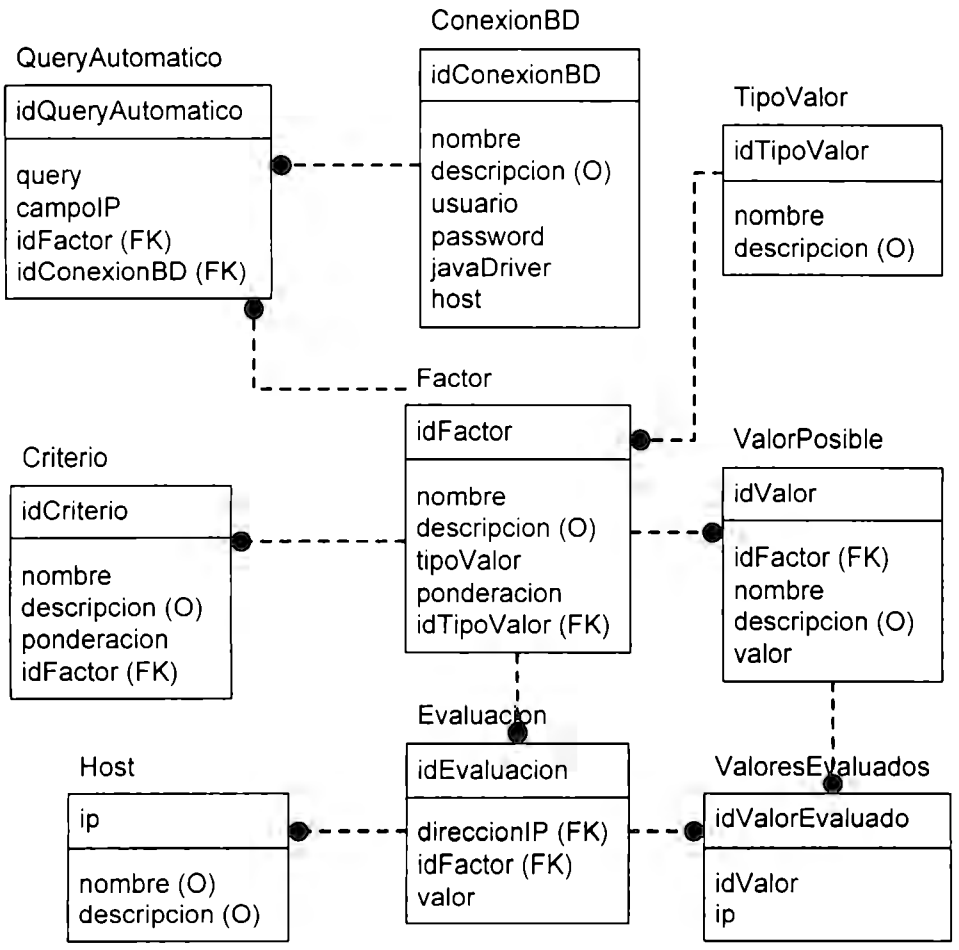

Fig. 13 - Diagrama E-R de SDERI

5.3 IMPLEMENTACIÓN DE SDERI

Al tomar la decisión de generar una clase genérica para implementar la administración de la información, se tuvo que evaluar la plataforma bajo la cual desarrollar el sistema. Por un lado es necesario que el lenguaje fuera orientado a objetos y por otro lado se requiere que el desarrollo de la interfaz del usuario fuera sencilla de realizar. Se decidió por un lenguaje orientado a objetos debido a que era necesario crear una clase genérica que hiciera la mayor parte de la administración de la información y desarrollar clases derivadas que en si implementaran la administración de un tipo de objeto en particular (criterio, factor, equipo o evaluación).

Por otro lado, es necesario que el desarrollo de la interfaz para el usuario no implique un gran esfuerzo sin que esto demerite la funcionalidad y amigabilidad del sistema. Por ello, se opto por utilizar una interfase web ya que cubría con estos requisitos.

Dados los requisitos anteriores se decidió implantar el sistema usando el lenguaje de programación java. Por otro lado se tenían dos opciones de tecnología Java para la implementación los servlets y por otro lado los denominados JSP (Java Server Pages).

Un servlet es la implementación en java de un CGI (Common Gateway Interface). La idea básica del CGI es usar a un servidor web como medio de comunicación para permitirle a un usuario invocar un procedimiento y el procedimiento utiliza al servidor web como medio para hacerle llegar la respuesta al cliente (generalmente en HTML). Un JSP es un estándar para desarrollar de manera más fácil un servlet. Lo que hace un JSP es incrustar en documento HTML código de java y esta incrustación la realiza al convertir este documento HTML modificado en un servlet que después compila y ejecuta. Sin embargo, la dificultad con los JSP es al momento del desarrollo pues esta traducción intermedia dificulta la depuración ya que este se realiza sobre el código compilado, es decir, sobre el servlet traducido.

Dada la complejidad de depuración en el desarrollo de JSPs, el sistema esta basado en servlets. Además, se contó con acceso a la infraestructura de desarrollo con que cuenta el ITESM-CEM, la cual esta principalmente en servlets. De esta infraestructura, destaca la clase *PageGenerator*, que remplaza patrones dentro de un archivo de texto (documentos HTML), lo cual permite separar la lógica de negocio de la interfaz de usuario.

El sistema esta basado en dos clases: *DbWeb* y *Esquema*. La clase *Esquema* es la implantación genérica de una interfase web para la manipulación de una base de datos haciendo sus tres operaciones básicas: alta, baja y cambio de un registro de una tabla especifica.

La clase *DbWeb* acumula todas las características que puede necesitar una página web como son una lista de características de los campos de la tabla en particular (definida por la clase *FieldInfo*), una lista de atributos ocultos (usado para migrar información entre diferentes servelts) y una lista de información a sustituir mediante la clase *PageGenerator*.

A partir de la clase *DbWeb*, se derivan las clases *DbWebAlta*, *DbWebBaja*, *DbWebCambio*, *DbWebExecCambio*, *DbWebMenu* y *DbWebSubMenu*. La clase *DbWebAlta* crea una forma para un alta de información. La clase *DbWebBaja* crea una forma para seleccionar los registros a eliminar (permitiendo una selección multiple). La clase *DbWebCambio* crea una forma para seleccionar un registro a modificar (permitiendo una selección simple). La clase *DbWebExecCambio* crea una forma para el cambio (actualización) de un registro. La clase *DbWebMenu*, extiende la clase DbWeb para guardar la información relativa a un menú. Finalmente, la clase *DbWebSubMenu*, permite invocar otro servlet.

La clase *Esquema* es un servlet, que verifica la variable *accion* (del Query String) mediante la cual se determina el procedimiento a invocar que corresponde a un proceso de administración de información (alta, baja o cambio). Adicionalmente, para cada uno de los procedimientos de administración de información, estos son subdivididos en tres etapas: Una inicialización, una modificación de propiedades en tiempo de ejecución y la ejecución propiamente. El propósito de hacer esta división es que las clases derivadas no tengan que ejecutar y solo inicialicen sus datos y modifiquen los las propiedades que necesiten para ejecutar las formas correspondientes, olvidándose de los detalles de la misma ejecución. La inicialización se encarga de asignar valores

que sean independientes de la ejecución, por ejemplo el titulo de las páginas, el nombre de los campos, etc. La modificación de propiedades en ejecución, actualiza valores que requieran de información relacionada de alguna manera con la interactividad del usuario.

De la clase *Esquema* se derivan las clases *EsquemaCriterio*, *EsquemaEval*, *EsquemaFactor*, *EsquemaHost* y *EsquemaValor*. Estas clases, manipulan la información referente a las distintas tablas según su terminación.

Para cumplir con el objetivo de realizar las evaluaciones de los niveles de riesgo se desarrollaron las clases *AnalisisRiesgo*, *EvaluacionEquipo*, *CriterioEvaluado* y *FactorEvaluado*. Estas clases calculan el riesgo en cada uno de los niveles del mismo. Para mostrar el resultado de estas clases se desarrollaron las clases *EquipoEvaluado* y *LstEquiposEvaluados*. La clase *EquipoEvaluado* muestra a detalle la evaluación del riesgo de un equipo en específico, mientras que la clase de *LstEquiposEvaluados* muestra una lista ordenada de equipos con su respectivo nivel de riesgo y sus calificaciones en cada uno de los criterios (datos guardados en la clase *AnalisisRiesgo*).

Para cumplir con el objetivo de evaluaciones automáticas de factores, la clase de *FactorEvaluado*, verifica el tipo de factor, en caso de que este se haya definido como "automático" entonces manda invocar el query (que se haya capturado) para obtener la evaluación automática de ese factor.

5.4 SDERI Y OTROS SISTEMAS

Uno de los puntos más relevantes de SDERI es el hecho de contar con la capacidad de explotar información de otros sistemas para que en conjunto se pueda tener una mejor visión del nivel de riesgo de los equipos. Por ello, se buscaron herramientas que de alguna u otra forma evalúen factores para automatizarlos. Se encontraron dos herramientas: OSSIM e ISS.

OSSIM es un conjunto de herramientas de seguridad que busca correlacionar una serie de eventos de cada una de las distintas herramientas para calcular un indicador que muestre el grado de intromisión y ataque hacia los equipos. SDERI usa a OSSIM como base primordial para determinar los factores del criterio del nivel de amenaza de los equipos, en particular los factores del nivel de intromisión y el nivel de ataque. Además SDERI usa a OSSIM para determinar el factor del número de puertos abiertos. OSSIM no es un paquete sencillo de instalar, ya que en realidad es una colección de herramientas que hay que hacer que convivan y cooperen, debido a esta dificultad de instalación se especifica en el anexo A.

ISS es una herramienta para el monitoreo de vulnerabilidades, lo que se conoce como un scanner de vulnerabilidades. SDERI usa a ISS como base para determinar los factores del número de vulnerabilidades de nivel alto, medio y bajo, los cuales son los factores primordiales del criterio del nivel de vulnerabilidad de los equipos.

5.4.1 OSSIM

Sistema de administración de información de seguridad de código abierto (Open Source Security Information Management). OSSIM integra actualmente 22 productos de seguridad (entre los que destacan: Snort, Nessus, Ntop, Acid,RRD, Nmap, P0f, Arpwatch, etc.) para crear una consola de seguridad distribuida.

Es el primer proyecto de código abierto de consolas de seguridad y muy probablemente la consola de seguridad mayormente usada en el mundo con aproximadamente 50,000 descargas al año

5.4.1.1 Arquitectura de OSSIM

OSSIM crea una arquitectura de cuatro niveles:

- **Agentes** internos al equipo(host)
- Un numero de **Sensores/Colectores** distribuidos en la red
- Una **consola central de administración**
- Una **base de datos forence** (Forensics)

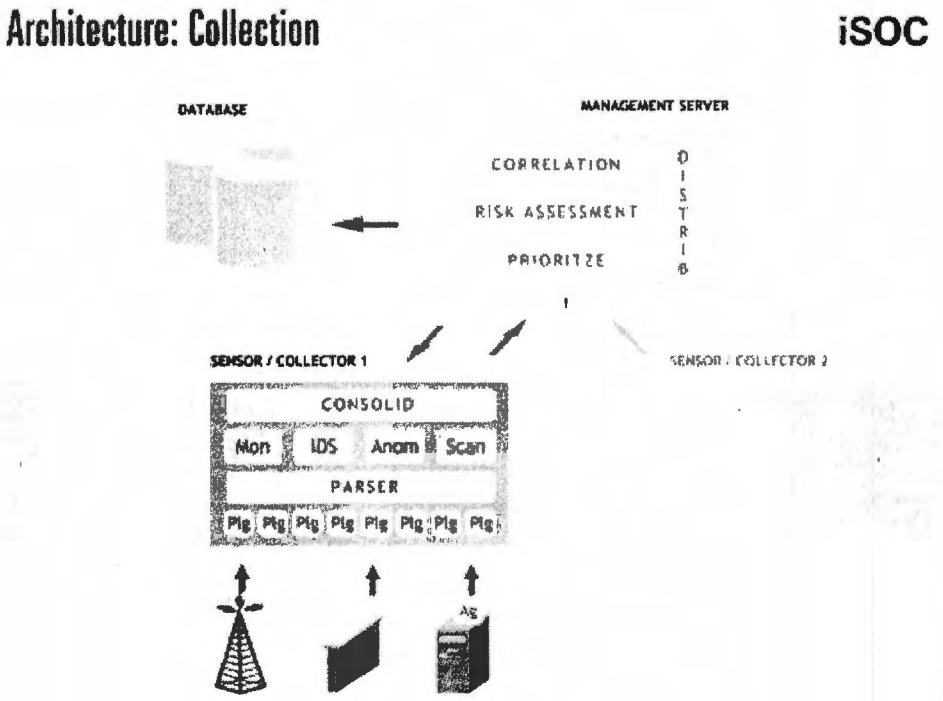

Fig. 14 - Arquitectura de OSSIM

Como se ve en la figura 14, OSSIM es capas de recibir datos de maquinas externas como son servidores Unix, ruteadores cisco o firewalls checkpoint1. Cuando es posible, la información de estas máquinas externas es almacenada a través de sus protocolos naturales como puede ser OPSEC de Checkpoint o syslog para una máquina Unix. Esto es conveniente ya que evita la instalación de agentes en las máquinas fuente. Si los protocolos naturales no existen, entones se debe instalar un agente primero. Se puede escoger instalar entre el agente de OSSIM o algún otro de código abierto.

5.4.1.1.1 Sensores y colectores de OSSIM

OSSIM cuenta con sensores y colectores. Los sensores son procesos que implementan la detección y auditoria de eventos (el objeto central en la figura 15).

Los colectores son programas que tienen la capacidad de coleccionar, analizar, normalizar y consolidar los eventos y enviarlos a la consola.

La funcionalidad del sensor integra herramientas de seguridad líderes, cubriendo el ciclo de seguridad completo como se muestra en la figura 15

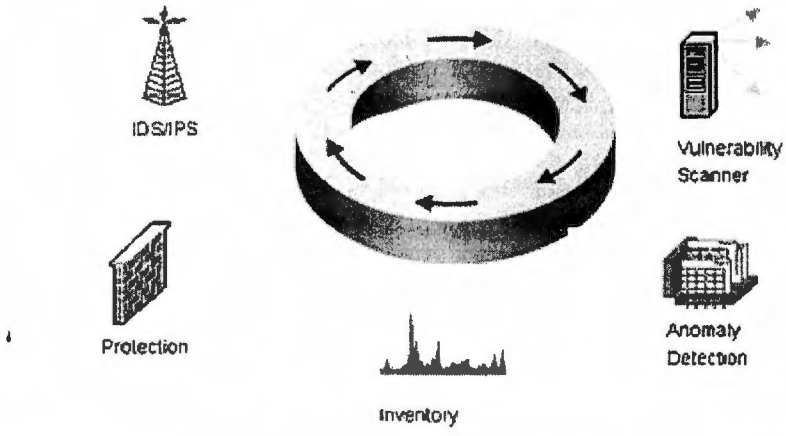

Fig. 15 - Sensores y colectores de OSSIM

Estas herramientas son líderes en sus categorías como por ejemplo Snort como IDS, Nessus como scaneador de vulnerabilidades, Ntop como monitor de red o Nmap como scaneador de puertos. Lo más interesante de todas estas herramientas es que son de código abierto, por lo que podemos empaquetarlos y compilarlos en un solo producto que se convierte en un arsenal de seguridad.

5.4.1.1.2 Consola de administración de OSSIM

Los datos recolectados de todos los sensores son almacenados en la consola, donde se llevan a cabo principalmente dos procesos automáticos: la medición de riesgo y la correlación

5.4.1.1.2.1 Medición del riesgo

La consola de OSSIM conoce la importancia de cada una de las maquinas de la red, lo que en el calculo de riesgo se conoce como valor del activo, conoce la amenaza que cada evento implica y puede calcular gracias a la correlación la confiabilidad del ataque. Todos estos valores permiten a OSSIM hacer la evaluación de riesgo de cada incidente y reportarlo.

5.4.1.1.2.2 Correlación

La correlación es hecha en diferentes maneras:

- correlación de diferentes eventos (correlación lógica),
- correlación de eventos y vulnerabilidades (correlación cruzada) y
- correlación de eventos y el inventario.

La correlación da la inteligencia a la consola, lo que permite reportar alarmas abstractas mas entendibles para los humanos, pero también permite reducir considerablemente la cantidad de alarmas reportadas, pudiendo darse el caso que se tienen millones de eventos, pero en realidad solo son reportados (debido a su importancia y su la verificación usando alguna correlación) una docena de ellas.

OSSIM determina la confiabilidad de un evento mediante su función de correlación. La lógica de la función de correlación de OSSIM tiene como base la comprobación de cada evento. Mientras tengamos millones de alarmas al día, no podemos confiar en ellas a menos que las verifiquemos, el motor de correlación buscará evidencias o síntomas que pruebe que el ataque es real o es un falso positivo.

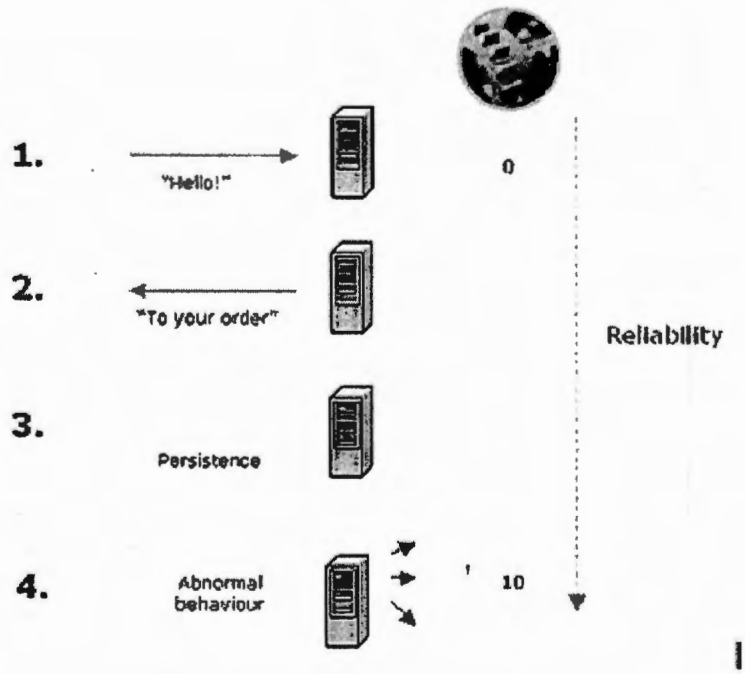

Fig. 16 - El motor de correlación de OSSIM

En el ejemplo de la figura 16, un atacante trata de introducirse a un equipo victima. Esto puede pasar miles de veces en un ambiente perimetral, el nivel de confiabilidad inicial será bajo o incluso cero. La función de correlación asignará un valor cuando encuentre una respuesta al ataque, dará un mayor valor si persiste la conexión anormal, un valor aun mas alto si el blanco se convierte en un atacante en si mismo que envía conexiones anormales.

5.4.1.1.3 Consola forense

OSSIM utiliza una extensión de ACID[3] como su consola forense, la cual permite explotar la base de datos de eventos. Esta base no solo almacena los datos de snort, también puede almacenar datos de Firewall-1, Cisco, Apache, etc).

La consola forense de OSSIM permite almacenar y buscar los siguientes tipos de información que normalmente no son incluidos:

- Riesgo acumulado al equipo en el momento del ataque.
- Riesgo instantáneo representado por una alerta
- Valor del activo al momento del ataque
- Nivel de intromisión del evento asignado por el motor de correlación.

Adicionalmente OSSIM implementa un pizarrón para mostrar el nivel de riesgo, que permite tener una vista de alto nivel de la situación de seguridad de los diferentes objetos y dominios en la red.

Las métricas son calculadas al acumular valores infinitesimales de riesgo en tiempo real de cada alerta que afecte un objeto, los dominios de seguridad son creados permitiendo una medición jerárquica o un "termómetro de seguridad" desde una maquina en particular hasta la red completa. El pizarrón permite a los administradores de seguridad seguir la tendencia en diferentes rangos de tiempo y compararlos con los umbrales establecidos.

---

[3] ACID. Analysis Console for Intrusion Databases. ACID es una consola web para el analisis de intrusiones. http://acidlab.sourceforge.net/

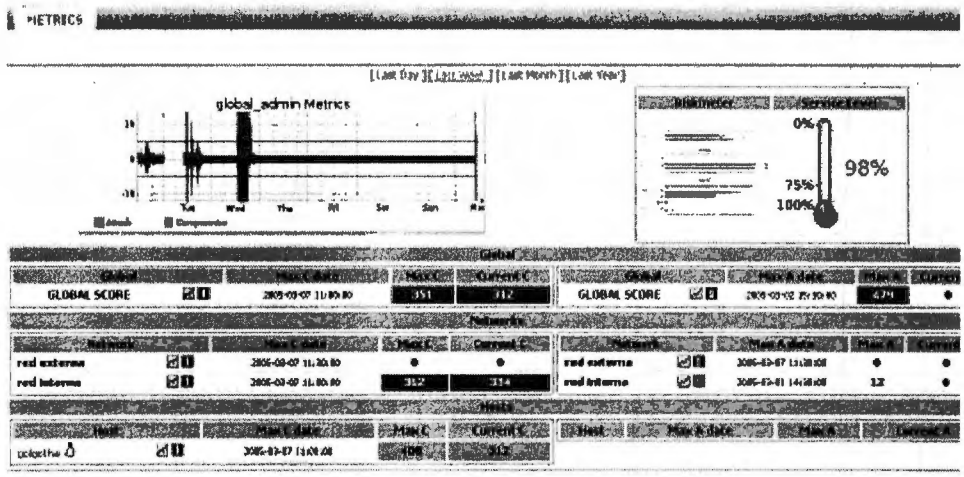

Fig. 17 - Pizarrón del riesgo de OSSIM

5.4.1.2 Integración de OSSIM con SDERI

SDERI aprovecha la excelente solución de OSSIM al extraer de esta dos indicadores primordiales para cada equipo. Estos indicadores son el nivel de ataque (identificado como "A") y el nivel de intromisión (identificado como "C" debido a que en ingles este indicador se le conoce como compromise). Estos indicadores corresponden con los factores del criterio del nivel de amenaza.

5.4.1.2.1 Análisis de la base de datos de OSSIM

Para poder extraer la información de OSSIM fue necesario analizar la compleja base de datos de OSSIM de 77 tablas. Dicho análisis se llevo a cabo a través de las llaves y tipos de datos. Como resultado del análisis se desarrollo el diagrama entidad-relacion presentado en la figura 18.

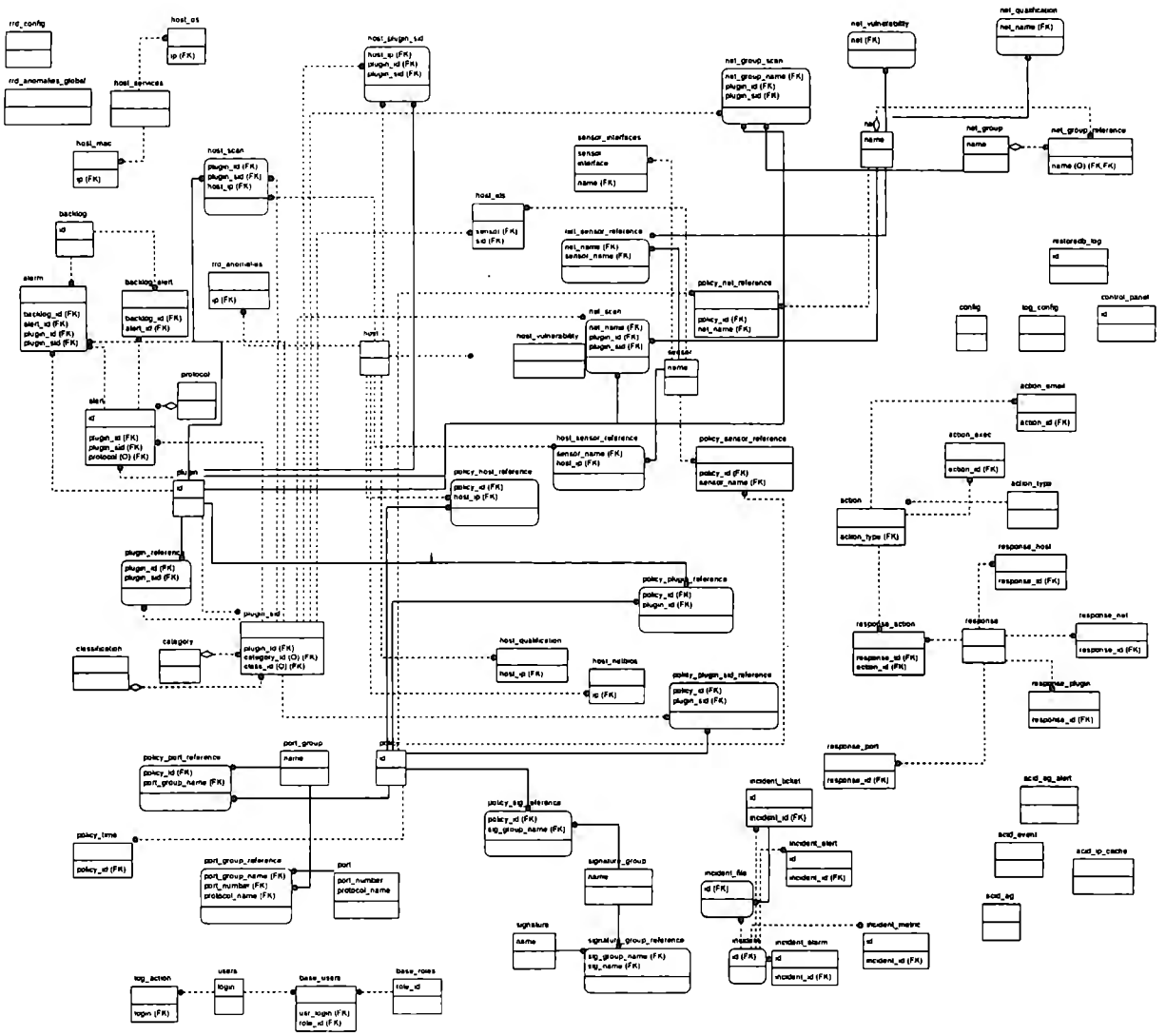

Fig. 18 - Diagrama E-R de OSSIM

A partir de esté diagrama es posible obtener los datos y tablas para determinar los factores del nivel de ataque y el nivel de intromisión que a continuación explicaremos en detalle.

5.4.1.2.1.1 Automatización del Nivel de Ataque.

A partir del diagrama E-R de la base de datos de OSSIM, se determino que se necesitaba la tabla *host_qualification*, para obtener el nivel de ataque que calcula OSSIM

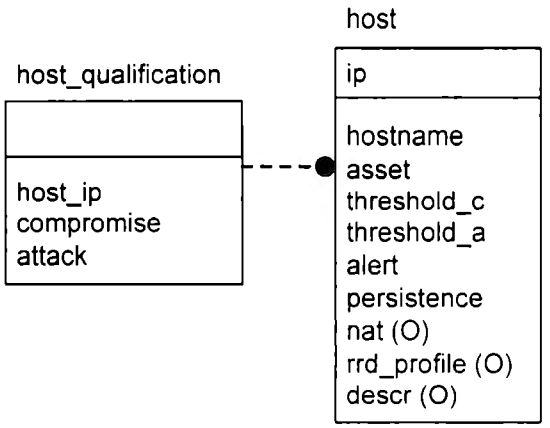

Fig. 19 - Diagrama E-R del factor nivel de ataque e intromisión basado en OSSIM

Y consecuentemente podemos determinar que el query para obtener el nivel del ataque es:

SELECT attack FROM host_qualification WHERE host_ip = %IP_EQUIPO%

5.4.1.2.1.2 Automatización del Nivel de Intromisión

Por fortuna, el mismo diagrama usado en el nivel de ataque puede ser usado para obtener el nivel de intromisión (compromise). A partir de este podemos determinar el query para obtener el nivel de intromisión:

SELECT compromise FROM host_qualification WHERE host_ip = %IP_EQUIPO%

5.4.1.2.1.3 Automatización del factor del número de puertos abiertos

A partir del diagrama E-R de OSSIM, se determino que la tabla *host-services* contiene toda la información necesaria para conocer la cantidad de puertos abiertos de los equipos. Esta tabla es alimentada por algún proceso dentro de OSSIM que usa NMAP para hacer estos barridos de puertos.

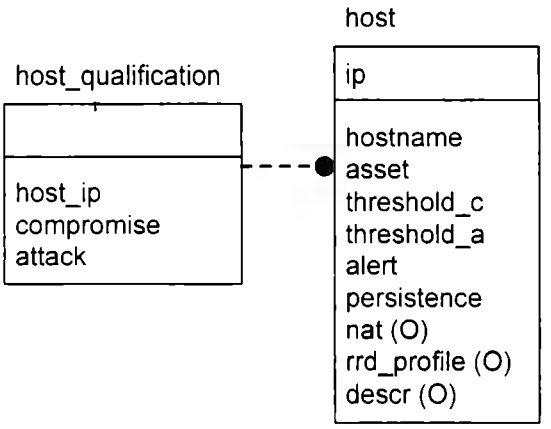

Fig. 20 - Diagrama ER del factor de cantidad de puertos abiertos

Para que finalmente se pueda generar el siguiente query

SELECT ip, COUNT(port) AS contPort FROM host_services GROUP BY ip HAVING ip = %IP_EQUIPO%

En este caso es importante aclarar que debido a la estructura de la base de la tabla host_services, el IP del equipo debe ser trasformada de texto a una representación de un entero de 32 bits.

5.4.1.2.2 ISS

Monitor de seguridad de Internet (Internet Security Scanner). ISS permite buscar vulnerabilidades en forma automatizada en servidores, desktops, sistemas operativos, routeadores, switches, firewalls, entre otros, y así identificar los riesgos potenciales que afectarán a estos sistemas.

5.4.1.3 Integración de ISS con SDERI

Para poder extraer la información de ISS fue necesario analizar la base de datos de ISS que consiste de 17 tablas. A través de las llaves y tipos de datos, se concluyo que esta base corresponde con el siguiente diagrama entidad-relación.

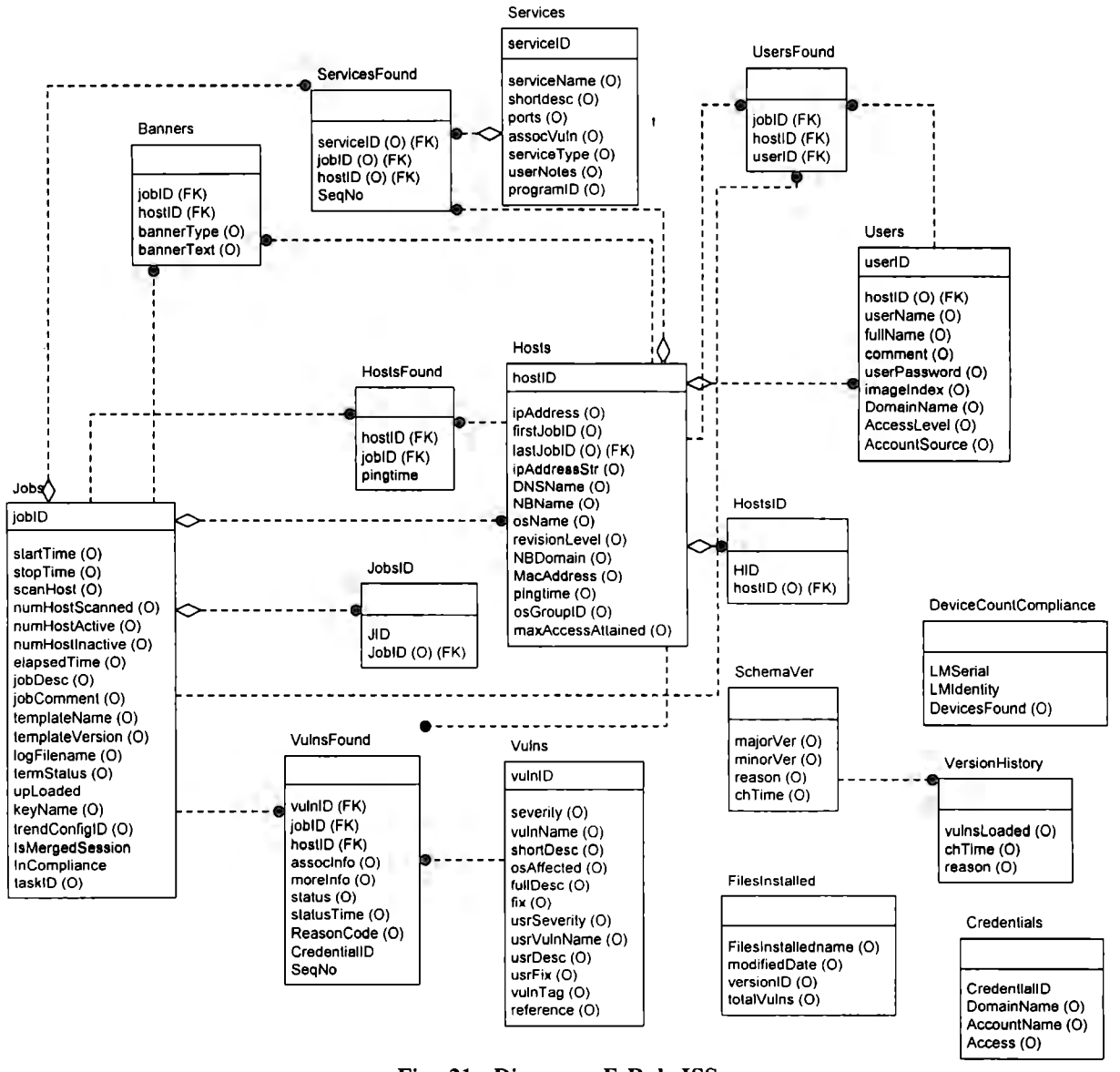

Fig. 21 - Diagrama E-R de ISS

A partir de esté diagrama podemos concluir los datos y tablas para determinar los factores del número de vulnerabilidades de niveles alto, medio y bajo de ataque que a continuación explicaremos en detalle.

5.4.1.3.1 Automatización del número de vulnerabilidades de nivel alto, medio y bajo

A partir del diagrama entidad-relación de ISS hubo que crear dos nuevas vistas para facilitar la automatización de la medición de estos factores.

Primero se creo una vista para obtener el último scaneo de cada IP, basándose en la tabla de Hosts de ISS

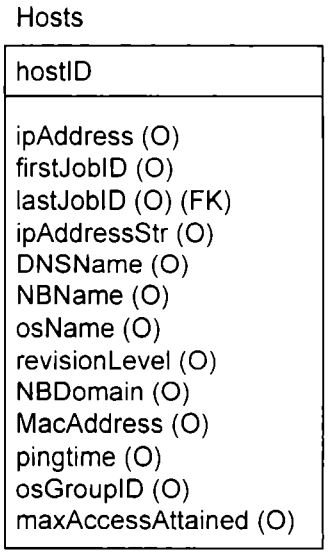

Fig. 22 - Tabla Hosts de ISS

La vista se crea con el siguiente comando directamente en la base de datos de ISS

```
CREATE VIEW MDGUERRA_LastScanByIP AS
        SELECT Hosts.ipAddressStr, Max(Hosts.lastJobID) AS MaxOflastJobID
FROM  Hosts
GROUP BY Hosts.ipAddressStr
GO
```

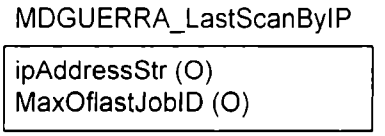

Fig. 23 - Vista (de BD) de el ultimo Job por IP de ISS

Posteriormente se creo otra vista para determinar la cantidad de vulnerabilidades de los distintos niveles para cada IP. Esta vista se basó en la vista *MDGUERRA_LastScanByIP* y en las tablas *VulnsFound* y *Vulns* de ISS.

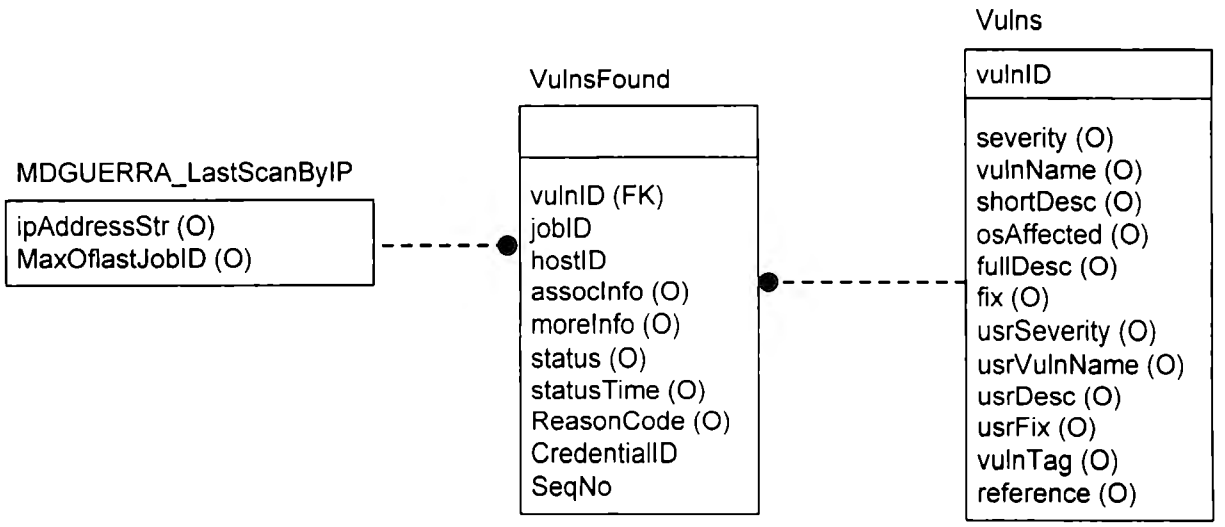

Fig. 24 - Vistas y tablas para crear la vista del conteo de vulnerabilidades por nivel por IP de ISS

Esta vista se crea con el siguiente comando:

```
CREATE VIEW MDGUERRA_VulnLastScan AS
SELECT MDGUERRA_LastScanByIP.ipAddressStr, Vulns.severity,
        Count(Vulns.severity) AS CountOfseverity
FROM MDGUERRA_LastScanByIP INNER JOIN
(VulnsFound INNER JOIN Vulns ON VulnsFound.vulnID=Vulns.vulnID) ON
MDGUERRA_LastScanByIP.MaxOflastJobID=VulnsFound.jobID
GROUP BY MDGUERRA_LastScanByIP.ipAddressStr, Vulns.severity
GO
```

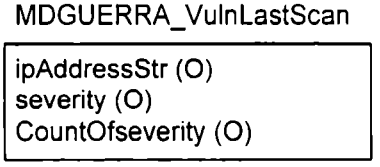

Fig. 25 - Vista (de BD) del numero de vulnerabilidades por nivel por IP de ISS

Una vez que se tienen las vistas el query necesario para consultar las vulnerabilidades por cada nivel es realmente sencillo.

5.4.1.3.1.1 Automatización del factor del número de vulnerabilidades de nivel bajo

Tan solo es necesario ejecutar el siguiente query:

SELECT CountOfSeverity FROM MDGUERRA_VulnLastScan WHERE severity =1 AND ipAddressStr= %IP_EQUIPO%

5.4.1.3.1.2 Automatización del factor del número de vulnerabilidades de nivel medio

Tan solo es necesario ejecutar el siguiente query:

SELECT CountOfSeverity FROM MDGUERRA_VulnLastScan WHERE severity =2 AND ipAddressStr= %IP_EQUIPO%

5.4.1.3.1.3 Automatización del factor del número de vulnerabilidades de nivel alto

Tan solo es necesario ejecutar el siguiente query:

SELECT CountOfSeverity FROM MDGUERRA_VulnLastScan WHERE severity =3 AND ipAddressStr= %IP_EQUIPO%

5.5 Conclusión

En este capitulo se describió el sistema SDERI. Sistema de Definición y Evaluación de Riesgos. El cual es la implementación de un sistema para administrar la evaluación de riesgos ponderada. Uno de los puntos más relevantes de esta implementación fue la determinación de crear una clase genérica para la administración de información (altas, bajas cambios). Adicionalmente, este sistema tiene la capacidad de explotar información de otras fuentes lo que permite automatizar factores lo que a su pez permite obtener el nivel de riesgo en tiempo real. Se propone a ISS y OSSIM como sistemas externos para explotar información que ayude a automatizar criterios.

Una vez desarrollado el sistema se uso para evaluar el nivel de riesgo de los servidores del ITESM-CEM, lo cual es expuesto en el siguiente capítulo.

6. RESULTADOS DEL SISTEMA SDERI EN EL ITESM-CEM

Se realizo una evaluación del nivel de riesgo de 34 equipos usando SDERI con los siguientes resultados (ordenados de acuerdo al nivel de riesgo total):

Tabla 16 - Nivel de riesgo de los servidores del ITESM-CEM

Equipo	Criterios			Nivel de Riesgo Total
	(40.0%) Valor del activo	(30.0%) Nivel de vulnerabilidad	(30.0%) Nivel de amenaza	
wdb-rep.cem.itesm.mx	55.93%	38.33%	0.00%	33.87%
cid01	50.33%	46.58%	-30.00%	25.11%
cid02	50.33%	46.58%	-30.00%	25.11%
ivr1	49.90%	8.63%	0.00%	22.55%
webcem01	39.83%	17.63%	0.00%	21.22%
authac.cem.itesm.mx	39.53%	15.00%	0.00%	20.31%
wdb-new	67.33%	-7.50%	-20.00%	18.68%
dsc04	47.17%	-5.00%	0.00%	17.37%
excem02	48.23%	-8.75%	0.00%	16.67%
fundes	27.13%	15.00%	0.00%	15.35%
alis	34.51%	3.25%	0.00%	14.78%
smserver01	26.56%	11.88%	0.00%	14.19%
galactica	38.85%	-5.00%	0.00%	14.04%
proyectcentral	51.18%	-23.75%	0.00%	13.35%
siass	32.30%	0.00%	0.00%	12.92%
smserver04	41.43%	-14.38%	0.00%	12.26%
sidi4	39.34%	-13.75%	0.00%	11.61%
aas	12.34%	21.17%	0.00%	11.29%
apserver04	34.56%	-10.00%	0.00%	10.82%
authac1.cem.itesm.mx	26.40%	0.00%	0.00%	10.56%
Equipo		Criterios		Nivel

	(40.0%) Valor del activo	(30.0%) Nivel de vulnerabilidad	(30.0%) Nivel de amenaza	
bioinformatics	31.51%	-8.75%	0.00%	9.98%
academ	24.43%	-1.38%	0.00%	9.36%
lpserver	30.89%	-12.88%	0.00%	8.50%
warehouse01.cem.itesm.mx	33.40%	-18.75%	0.00%	7.74%
aditesm11	52.18%	-19.75%	-30.00%	5.95%
seguridad	12.85%	0.00%	0.00%	5.14%
aditesm01	49.84%	-20.38%	-30.00%	4.82%
dktopctrl01.cem.itesm.mx	41.84%	-20.38%	-20.00%	4.62%
phoebe.cem.itesm.mx	46.34%	-20.00%	-30.00%	3.54%
pandora.cem.itesm.mx	46.34%	-20.00%	-30.00%	3.54%
app01	34.90%	-6.00%	-30.00%	3.16%
devel01	34.84%	-12.50%	-30.00%	1.19%
devel02	34.84%	-17.50%	-30.00%	-0.31%
app02	28.23%	-10.00%	-30.00%	-0.71%
app03	27.73%	-10.00%	-30.00%	-0.91%

Los resultados de la tabla 16 son un resumen del nivel de riesgo de los servidores del ITESM-CEM, ya que únicamente se muestra el resultado de la evaluación de cada uno de los criterios (valor del activo, nivel de vulnerabilidad y nivel de amenaza) y la evaluación del nivel de riesgo total.

Para una mejor compresión de los resultados presentados en la tabla 16, se tomara el renglón correspondiente al servidor wdb-rep.cem.itesm.mx. Este servidor tiene una evaluación del valor del activo de 55.93%, una evaluación del nivel de vulnerabilidad de 38.33% y una evaluación del nivel de amenaza de 0.00%, al sumar ponderadamente de cada uno de estos criterios obtenemos el nivel de riesgo total del equipo que es de 33.87%.

SDERI también permite observar el detalle de la evaluación de cada uno de los equipos. En la tabla 17 se muestra el desglose de la evaluación del servidor wdb-rep.cem.itesm.mx.

Tabla 17 - Evaluación del nivel de riesgo de wdb-rep

Criterio	Factor	Factor Capturado	Factor Evaluado	Factor Ponderado	Criterio Evaluado	Criterio Ponderado	Evaluación Total
(40.0%) Valor del activo	(20.0%) Nivel de información	Confidencial	60.000%	12.000%	55.933%	22.373%	33.873%
	(15.0%) Número de usuarios	1000.0	4.000%	0.600%			
	(20.0%) Nivel de los usuarios	Nivel 3 - Empleados	66.667%	13.333%			
	(20.0%) Nivel de dependencia	Base de Datos	25.000%	5.000%			
	(25.0%) Criticidad según el dueño	Alta	100.000%	25.000%			
	-(10.0%) % de canales cifrados en las comunic.	0.0	0.000%	-0.000%			
(30.0%) Nivel de	-(10.0%) Antivirus actualizado	Falso	0.000%	-0.000%	38.333%	11.500%	

vulnerabilidad	(15.0%) Dirección IP homologada	Falso	0.000%	0.000%			
	(5.0%) Número de vulnerabilidades de nivel bajo	8.0	100.000%	5.000%			
	(50.0%) Número de vulnerabilidades de nivel alto	2.0	66.667%	33.333%			
	(20.0%) Número de vulnerabilidades de nivel medio	0.0	0.000%	0.000%			
	(10.0%) Número de puertos abiertos	0.0	0.000%	0.000%			
	-(15.0%) Firewall	Falso	0.000%	-0.000%			
	-(10.0%) Instalación de parches automatizado	Falso	0.000%	-0.000%			
	-(5.0%) Control de acceso exclusivo	Falso	0.000%	-0.000%			
(30.0%) Nivel de amenaza	(80.0%) Nivel de intromisión	0.0	0.000%	0.000%	0.000%	0.000%	
	(20.0%) Nivel de Ataque	0.0	0.000%	0.000%			
	-(20.0%) Respaldos automatizados	Falso	0.000%	-0.000%			
	-(30.0%) Redundancia en el servicio	Falso	0.000%	-0.000%			

Al haber realizado esta evaluación podemos observar que el equipos que presenta un mayor nivel de riesgo para el ITESM-CEM es wdb-rep.cem.itesm.mx lo cual es entendible, ya que este servidor es un servidor de base de datos donde se tiene almacenado la información sensible para la empresa. Además se observa que tiene una alta cantidad de vulnerabilidades aunque por fortuna ninguna de ellas críticas.

7. CONCLUSIONES SOBRE EVALUACIÓN DE RIESGOS INFORMÁTICOS

Es innegable que los sistemas son un elemento imprescindible para la operación y estrategia de las empresas, en la cual el ITESM-CEM no es la excepción. Por ello, es necesario contar un proceso que permita manejar el riesgo que implica tener estos sistemas. Uno de los puntos clave para poder hacer correctamente esta administración de riesgos, es la propia evaluación de riesgos. Sin embargo, los esquemas actuales para hacer esta evaluación son ineficientes. La evaluación de riesgos cualitativa es demasiado subjetiva y solo sirve como un acercamiento inicial. La evaluación de riesgos cuantitativa es difícil de llevar a la práctica debido a que se deben sacar costos que debido a su complejidad se convierten en estimaciones subjetivas. Por lo tanto, fue necesario proponer un esquema de evaluación de riesgos que permitiese ser mucho mas objetivo y que al mismo tiempo fuera sencilla de calcular. Este nuevo enfoque de evolución de riesgos es la evaluación de riesgos ponderada. Este nuevo enfoque fue implementado en el sistema SDERI (Sistema para la Definición y Evaluación de Riesgos Informáticos). SDERI permite evaluar el riesgo en un nivel operativo y no solo administrativo, pues permite consultar el riesgo de los equipos en tiempo real debido a que tiene la capacidad de evaluar factores directamente de fuentes externas.

Lo más sobresaliente de este trabajo es la automatización de la evaluación de riesgos. Esto permite moverse de una administración de riesgos reactiva a una proactiva pues por un lado determina claramente cuales son los equipos que implican un mayor riesgo, que al analizar más a fondo (al observar las evaluaciones de cada uno de los factores) permite determinar sencillamente la causa del alto nivel de riesgo y así poder proponer la solución de manera directa. Adicionalmente al tener una evaluación continua, permite tener un indicador que sea la base para priorizar la resolución de problemas del día a día, lo que nos garantiza no perdernos y enfocarse en aquello que sea lo más importante para el negocio.

Bibliografía

[1] "Objetivos de Control para la Información y Tecnologías Relacionadas (CObIT)". http://www.comip.mendoza.gov.ar/cobit.doc . (Dic, 2005)

[2] Bruce, Moulton. "Administración de la TI y seguridad de la información". Publicado en EEUU 30 de Marzo de 2004. Article ID:3522

[3] Jøsang, Audun y otros. "Belief-based risk analysis". Ene, 2004. http://0-portal.acm.org.millenium.itesm.mx/citation.cfm?id=976448&coll=portal&dl=ACM&CFID=63174147&CFTOKEN=8157980 (Dic, 2005)

[4] "MAGERIT – versión 2. Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información". Mar, 2005. http://www.csi.map.es/csi/pg5m20.htm. (Dic.,2005)

[5] "Diccionario de Uso del Español d America Vox Larousse".2002. http://www.diccionarios.com/consultas.php . (Feb, 2006)

[6] Blakley, Bob y otros. "Session 5: less is more: Information security is information risk management". Sept 2001. http://0-portal.acm.org.millenium.itesm.mx/ft_gateway.cfm?id=508187&type=pdf&coll=portal&dl=ACM&CFID=63174147&CFTOKEN=8157980 (Dic, 2005)

[7] McNamee, David. "Glosario de Evaluación de Riesgo". http://www.mc2consulting.com/riesgo.htm. (Dic, 2005)

[8] "Guía de administración de riesgos de seguridad". 15 Oct. 2004. http://www.microsoft.com/latam/technet/articulos/adminriesgos/default.mspx (Dic, 2005)

[9] "First Measure Your Risk". Enero 2003. http://www.gammassl.co.uk/inforisk/riskpart1.html (2003)

[10] Cooper, Dale F."Tutorial notes: The Australian and New Zealand Standard on Risk Management 4360 :1999". 1999. http://www.rudnicki.com.pl/pub/RMStd_Austral_comment.pdf . (Marzo 2006)

[11] Yates, John C. y Paul H. Arne. "Balancing the Scales: Managing Risks in IT Projects". Jul, 2004. http://0-web.lexis-nexis.com.millenium.itesm.mx/universe/document?_m=4cc9b4422a847f2edca3c2b594c66372&_docnum=14&wchp=dGLzVlz-zSkVA&_md5=8086f2595de60566835a3866e817a6f3.(Dic., 2005)

[12] Purdy, Grant. "AS/NZS 4360: 2004 and Safety Risk Management". 2005. http://www.tcci.com.au/content/events/Events%202005/PPT%20Presentations/Grant%20Purdy%20-%20Standards%20Aust.ppt .(Marzo 2006)

[13] Blakley, Bob y otros. "Session 5: less is more: Information security is information risk management". Septiembre 2001. http://0-portal.acm.org.millenium.itesm.mx/ft_gateway.cfm?id=508187&type=pdf&coll=portal&dl=ACM&CFID=63174147&CFTOKEN=8157980 (Dic 2005)

[14] "CRAMM". Enero 2003. http://www.gammassl.co.uk/topics/hot5.html (Marzo 20006)

[15] Yazar, Zeki. "A qualitative risk analysis and management tool – CRAMM". http://www.sans.org/rr/whitepapers/auditing/83.php (Marzo 2006)

[16] Marquis, Hank. "10 STEPS TO DO IT YOURSELF CRAMM". Febrero, 2006. http://www.itsmsolutions.com/newsletters/DITYvol2iss8.htm. (Marzo 2006)

[17] Rojas Córsico, Ivana Soledad. "Trabajo de Auditoria: Normas COBIT". http://www.monografias.com/trabajos14/auditoriasistemas/auditoriasistemas.shtml (Diciembre 2005)

[18] "Ti-Solutions - FAQ". http://www.ti-solutions.com/faq_tisolutions.htm. (Diciembre 2005).

[19] Gauldentops, Erik y otros. "Aligning COBIT, ITIL and ISO 17799 for Business Benefit: Management Summary". 2005. http://www.itsmf.com/images/news/ITIL-COBiT.pdf (Febrero 2006)

[20] "COBIT 4.0".2005 https://www.isaca.org/Template.cfm?Section=COBIT6&Template=/MembersOnly.cfm&ContentID=23325 (Enero 2006)

[21] "COBIT Objetivos de Control, 3ra edición".2000 http://www.isaca.org/AMTemplate.cfm?Section=Downloads&Template=/MembersOnly.cfm&ContentFileID=5076 (Enero 2006)

[22] Izquierdo Duarte, Fernando. "Administración de Riesgos de TI". http://www.latinbanking.com/memorias_congreso_clain_2005/10izquierdo_f_adm_riesgo_ti.pdf (Diciembre 2005)

[23] Cao Avellaneda, Javier . "Análisis y gestión de riesgos de la seguridad de los sistemas de la información". Mar, 2005. http://www.cii-murcia.es/informas/abr05/articulos/Analisis_gestion_riesgos_seguridad_sistemas_informacion.php . Dic., 2005

[24] Hayes, Ian y William Ulrich. "It's all abaut managing risk". Abril 1998. http://0-proquest.umi.com.millenium.itesm.mx/pqdweb?index=11&did=28857633&SrchMode=1&sid=9&Fmt=4&VInst=PROD&VType=PQD&RQT=309&VName=PQD&TS=1134779430&clientId=23693 . (Diciembre 2005)

[25] Alberts, Christopher, y otros. "Introduction to the OCTAVE Approach". Agosto 2003. http://www.cert.org/octave/approach_intro.pdf. (Febrero 2006)

[26] Stoneburner, Gary y otros. "Risk Management Guide for Information Technology Systems." Julio 2002. http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf (Febrero 2006)

[27] "Estándares de gerencia de riesgos".2003. http://www.ferma-asso.org/Risk%20Managers%20section/Entreprise%20Risk%20Management%20Corporate%20Governance/RMS%20pour%20FORUM%202005%20final/RMS%20pdf%20ESP.pdf (Marzo 2006)

[28] "Business Impact Analysis/ Risk Assessment for Technology assets". 2004. http://www.security.vt.edu/playitsafe/riskanalysis/RA-2004-Section%201.doc (marzo 2006)

[29] METAGruop. "SELECTING THE RISK ASSESSMENT METHOD OF CHOICE". 21 Julio 2004. http://searchcio.techtarget.com/generic/0,295582,sid19_gci1049908,00.html (marzo 2006)

[30] Tiwana, Amrit y Mark Keil. "The One-Minute Risk Assessment Tool". Noviembre 2004. http://0-portal.acm.org.millenium.itesm.mx/ft_gateway.cfm?id=1029497&type=pdf&coll=portal&dl=ACM&CFID=63174147&CFTOKEN=8157980. (diciembre 2005).

[31] "More about OSSIM". http://ossim.itdeusto.com/about.htm (abril 2006)

[32] Casal, Julio. "OSSIM fast guide". 8 febrero 2004. http://www.ossim.net/docs/OSSIM-fastguide.pdf (marzo 2004)

[33] Casal, Julio y otros. "OSSIM. Descripción general del sistema". 21 octubre 2003. http://www.ossim.net/docs/OSSIM-desc-es.pdf (marzo 2004)

[34] "Internet Scanner". Octubre 2004. http://www.iss.net/products_services/enterprise_protection/vulnerability_assessment/scanner_internet.php (mayo 2006)

[35] "ISS". http://www.etek.cl/iss.htm . (mayo 2006)

[36] Internet Scanner 7.0 technical overview. 2003. http://www.ciac.org/cstc/iss/IS_TechOverview70.pdf (mayo 2006)

[37] "Encarta World English Dictionary".2006. http://ebcarta.msn.com/dictionary_/threat.html (febrero 2006)

[38] Keating, Gina. "Ciberataques alimentan mercado de seguros contra 'hackers'". http://www.terra.com/finanzas/articulo/html/fin926.htm. (junio 2006)

[39] "RSA Labortories Challenges". http://www.rsasecurity.com/rsalabs/node.asp?id=2091 . (Julio 2006)

[40] "Global Security Challenge". http://www.globalsecuritychallenge.com/. (Julio 2006)

[41] Passori, Al. "SELECTING THE RISK ASSESSMENT METHOD OF CHOICE". Septimbre 2004. http://searchcio.techtarget.com/originalContent/0,289142,sid19_gci994851,00.html . (Febrero 2005).

Anexo A. Guía de instalación para OSSIM en Debian GNU/Linux

A.0. Antes de instalar OSSIM

A.0.1. Configuración conveniente

Configurar el archivo `/etc/apt/sources.list` para fijar los repositorios Debian Etch y de OSSIM

```
deb http://ftp.debian.org/debian/ etch main contrib non-free
deb http://security.debian.org etch/updates main contrib non-free
deb http://www.ossim.net/download/ debian/
```

Crear un archivo de `/etc/apt/preferences`

```
Package: *
Pin: release o=ossim
Pin-Priority: 995
```

Esta configuración forzará una prioridad más alta a los paquetes de OSSIM y a sus dependencias.

A.1. Instalar la base de datos de OSSIM

Instalar:

```
# apt-get install ossim-mysql
```

Establecer la contraseña de root para tu base de datos:

```
# mysqladmin -u root password your_secret_password
```

Editar `/etc/mysql/my.cnf` para modificar la dirección de enlace

```
[/etc/mysql/my.cnf]
...
bind-address            = <server_ip>
```

Crear las siguientes bases de datos:

```
# mysql -u root -p
```

```
mysql>create database ossim;
mysql>create database ossim_acl;
mysql>create database snort;
mysql>exit;
```

Cree las tablas en las bases de datos:

```
# zcat /usr/share/doc/ossim-mysql/contrib/create_mysql.sql.gz \
    /usr/share/doc/ossim-mysql/contrib/ossim_config.sql.gz \
    /usr/share/doc/ossim-mysql/contrib/ossim_data.sql.gz \
    /usr/share/doc/ossim-mysql/contrib/realsecure.sql.gz | \
    mysql -u root ossim -p

# zcat /usr/share/doc/ossim-mysql/contrib/create_snort_tbls_mysql.sql.gz \
    /usr/share/doc/ossim-mysql/contrib/create_acid_tbls_mysql.sql.gz \
    | mysql -u root snort -p
```

A.2. 2. Instalar el servidor de OSSIM

Instalarlo con apt:

```
# apt-get install ossim-server
```

Se definirán las propiedades de red y de la base dedatos. Use dpkg-reconfigure `ossim-server` si se quiere actualizar la información (no se debe editar a mano el archivo `/etc/ossim/server/config.xml`).

A.3. 3. Instalar los pluggins de OSSIM

A.3.2. Snort

Instalar el snort:

```
# apt-get install snort-mysql
```

La base de datos de snort ha sido creada en la sección 1. (Instalar la base de datos de OSSIM). Utilice snort como nombre de la base de datos. Borre el archivo de `/etc/snort/db-pending-config`.

```
rm - f /etc/snort/db-pending-config
```

El archivo `/etc/snort/snort.conf` de la configuración del snort debe parecer esto:

```
..
var HOME_NET [192.168.0.0/16]
var EXTERNAL_NET !$HOME_NET
..
# Variables needed by the bleeding snort rules
include $RULE_PATH/bleeding.conf
..
# OSSIM output:
output database: alert, mysql, user=root password=yourdbpass dbname=snort
host=yourdbhost sensor_name=your_sensor_ip logfile=alert
..
# Si se quiere usar spade, obtenga un spade.ossim.conf (del paquete ossim-
contrib)
include spade.ossim.conf
..
```

Obtenta las reglas de snort de Bleeding Edge. Las reglas de Bleeding Edge, tienen una tasa muy baja de falsos positivos, por lo cual es muy util para OSSIM.

```
# cd /etc/snort/rules/
# wget http://www.bleedingsnort.com/bleeding-all.rules
```

```
# echo "include \$RULE_PATH/bleeding-all.rules" >> /etc/snort/snort.conf
```

Actualice la base de datos de OSSIM con las nuevas reglas. (Se necesita el paquete ossim-utils):

```
# /usr/share/ossim/scripts/create_sidmap.pl /etc/snort/rules | \
    mysql -u root ossim -p
```

A.3.3. Ntop

Instalar Ntop:

```
# apt-get install librrd2 ntop
```

Definir la contraseña para el usuario del admin:

```
# ntop -u ntop
>>Please enter the password for the admin user:
# ^C
# /etc/init.d/ntop start
```

Ir a http://yourhost:3000/ para ver Ntop en la acción. Activar el rrdPlugin en Admin□plugins. Habilite Host en Data Dump y especifique su máscara de red en Host Filter.

Para hacer RRD plugin del ntop que trabaje con el IPs en vez de los MACs (necesitados por el agente), edite el archivo de `/etc/default/ntop` y agregar `-no-mac a GETOPT=" "`.

A.3.4. Nagios

Instalar los paquetes siguientes:

```
# apt-get install nagios-mysql
```

Configure nagios usando siguiente el documento: `/usr/share/doc/nagios-mysql/README.mysql`

Es necesario editar `/etc/nagios/hosts.cfg` y otros archivos debajo de `/etc/nagios`.

Afortunadamente hay mucha documentación al respecto.

A.3.5. otros plugins

Tan simple como:

```
# apt-get install p0f arpwatch pads tcptrack
```

No arraque pads y arpwatch en el arranque, deje que el ossim-agent haga el trabajo:

```
# update-rc.d -f arpwatch remove
# update-rc.d -f pads remove
```

A.4. Instalar el agente de OSSIM

Instalar el ossim-agent:

```
# apt-get install ossim-agent
```

Use dpkg-reconfigure ossim-agent para actualizar la configuración del agente.

Es necerio editar manualemente `/etc/ossim/agent/config.xml` y especificar el ossim_db.

A.5. Instalar el marco de OSSIM

A.5.6. Web server

Instalar el paquete de apache con la soporte para php. Puedes utilizar apache, apache-SSL o apache2. También, puedes utilizar php4 o php5. Una instalación estándar:

```
# apt-get install apache2 php4 libapache2-mod-php4
```

A.5.7. phpGACL

Instalar el paquete de phpgacl:

```
# apt-get install phpgacl
```

Use `ossim_acl` como nombre para la base de datos del acl, nose debe utilizar el nombre del phpgacl del defecto.

A.5.8. Marco de Ossim

Instalar el `ossim-framework` y todas tus dependencias.

```
# apt-get install ossim-framework
```

Utilice `dpkg-reconfigure ossim-utils` y `dpkg-reconfigure ossim-framework` para `actualizar` la configuración del marco (no se debe editar `/etc/ossim/framework/ossim.conf` a mano).

Es necesario configurar los paquetes de php para habilitar las extensiones, ya que son inhabilitados por el defecto. Para php4, tendrás ejecute:

```
# dpkg-reconfigure php4-cli php4-domxml php4-gd php4-mysql php4-xslt
```

`Acceda al marco [http://yourhost/ossim/] e ir al menú Configuration→Main y verifique que todos los parámetros sean correctos.`

`Arranque el deamon de ossim-framework`

```
#/etc/init.d/ossim-framework  start
```

A.6. Instalar el contrib de OSSIM (opcional)

El ossim-contrib del paquete contiene un sistema de remiendos, de ejemplos y de archivos de la configuración usados por la distribución del ossim. Este paquete es solamente útil para los propósitos del desarrollo.

Bibliografía

- Karg, Dominique. "OSSIM Install Guide for Debian GNU/Linux". Diciembre 2005. (mayo 2006)
- OSSIM Install Guide for Debian GNU/Linux: http://www.ossim.net/docs/INSTALL.Debian.html
- OSSIM Debian Development: http://alioth.debian.org/projects/pkg-ossim/
- OSSIM 0.9.8 & Debian 3.1 Notes: Debian Sarge Notes

Anexo B. Manual de Usuario del Sistema para la Definición y Evaluación de Riesgos Informáticos (SDERI).

B.1. Introducción

El sistema para la definición y evaluación de riesgos informáticos (SDERI) fue desarrollado para implantar un esquema de evaluación de riesgos ponderada, en donde los distintos criterios de evaluación son divididos en factores.

B.2. Acceso a SDERI

SDERI fue desarrollado en java, bajo la tecnología de servlets, por lo cual para acceder a este debe ser mediante un browser, accediendo a la siguiente dirección:

http://<servidor >:<puerto>/sderi

Para propósitos de este trabajo, se usa el equipo calypso.cem.itesm.mx en el puerto 8888, por lo que su acceso esta en:

http://calypso.cem.itesm.mx:8888/sderi/

Al entrar a SDERI, este pide una autenticación la cual hace uso de un módulo de login desarrollado en el ITESM-CEM

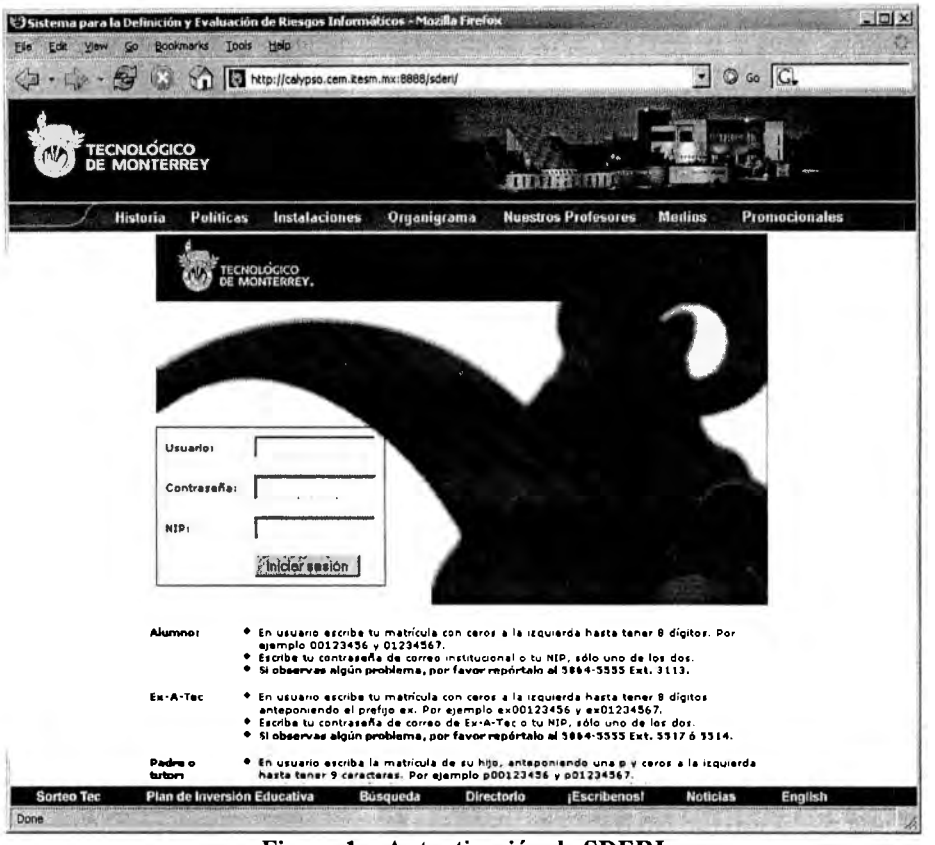

Figura 1 – Autenticación de SDERI

B.3. Funcionamiento general

SDERI fue conceptuado en un modelo orientado a objetos, buscando optimizar el desarrollo del mismo. Por ello, la mayor parte del sistema tiene el mismo ambiente y opciones.

SDERI esta conformado de varios módulos, donde cada modulo de administración presenta la siguiente estructura en su navegación

1. Menú de administración del módulo
   - 1.1. Alta
     - 1.1.1. Captura del alta
       - 1.1.1.1. Ejecución del alta
   - 1.2. Baja
     - 1.2.1. Selección de los elementos a dar de baja
       - 1.2.1.1. Ejecución de la baja
   - 1.3. Cambio
     - 1.3.1. Selección del elemento a hacer el cambio
       - 1.3.1.1. Captura del cambio
         - 1.3.1.1.1. Ejecución del cambio
   - 1.4. Sub-menú

B.3.9. Descripción genérica de los módulos del sistema

A continuación describiremos de forma general cada modulo.

B.3.9.1. Menú de administración del módulo

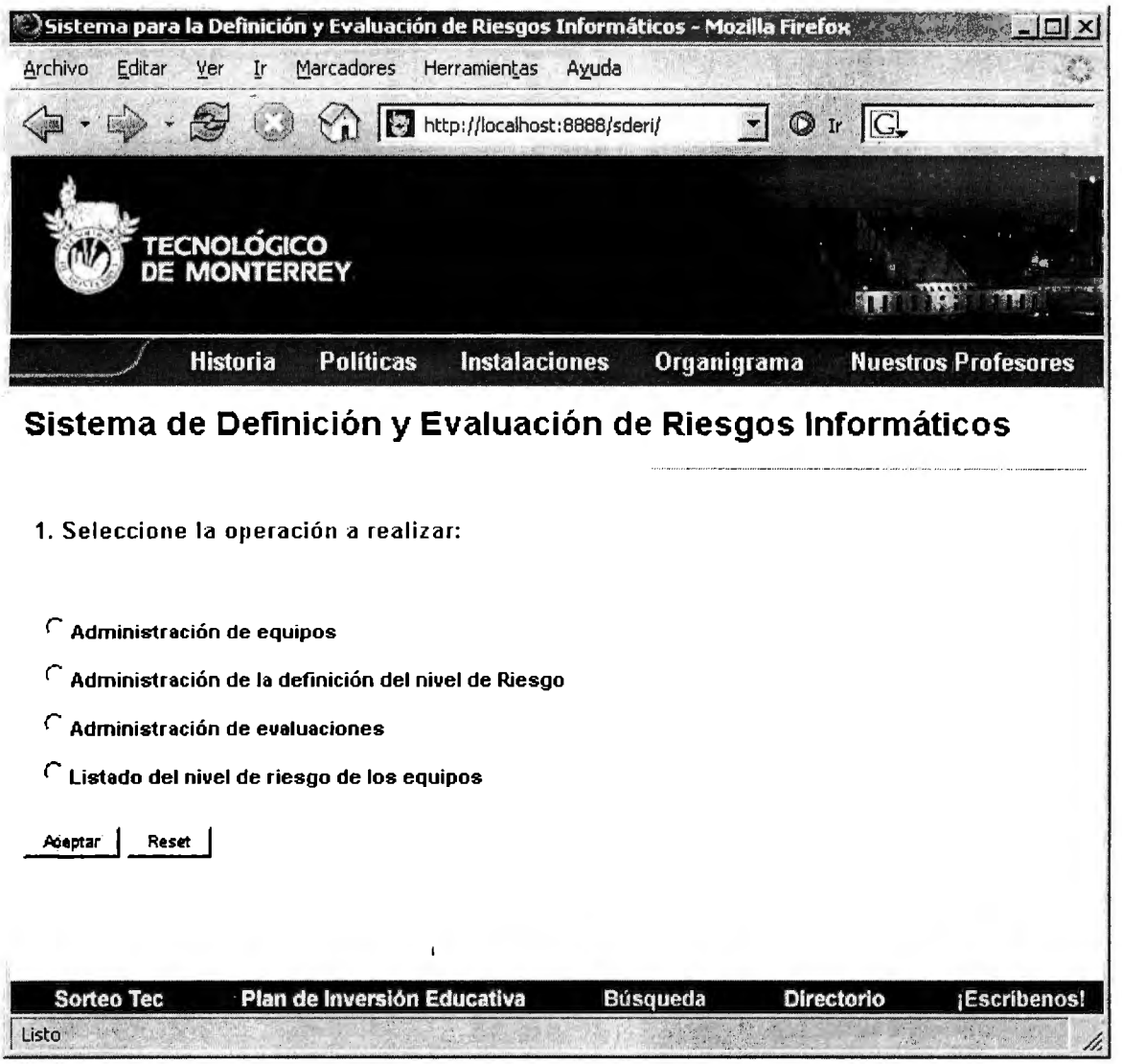

Figura 2- Ejemplo de menú de SDERI

En general para cada una de las partes de SDERI se cuenta con un menú, el cual permite acceder a una función para dar de alta, dar de baja o hacer un cambio del módulo en el que nos encontremos (criterios, factores, valores posibles, equipos, etc.). El menú también permite opcionalmente acceder a otro modulo.

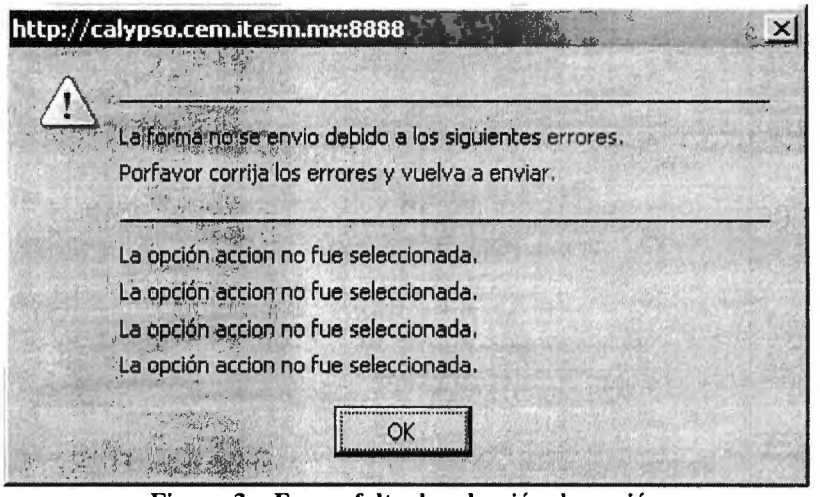

Figura 3 – Error, falta de selección de opción

El menú se asegura que todas las opciones hayan sido seleccionadas. En caso de que falte seleccionar alguna opción marcara un error parecido a la figura 3.

B.3.9.1.1. Alta

B.3.9.1.1.1. Captura del alta

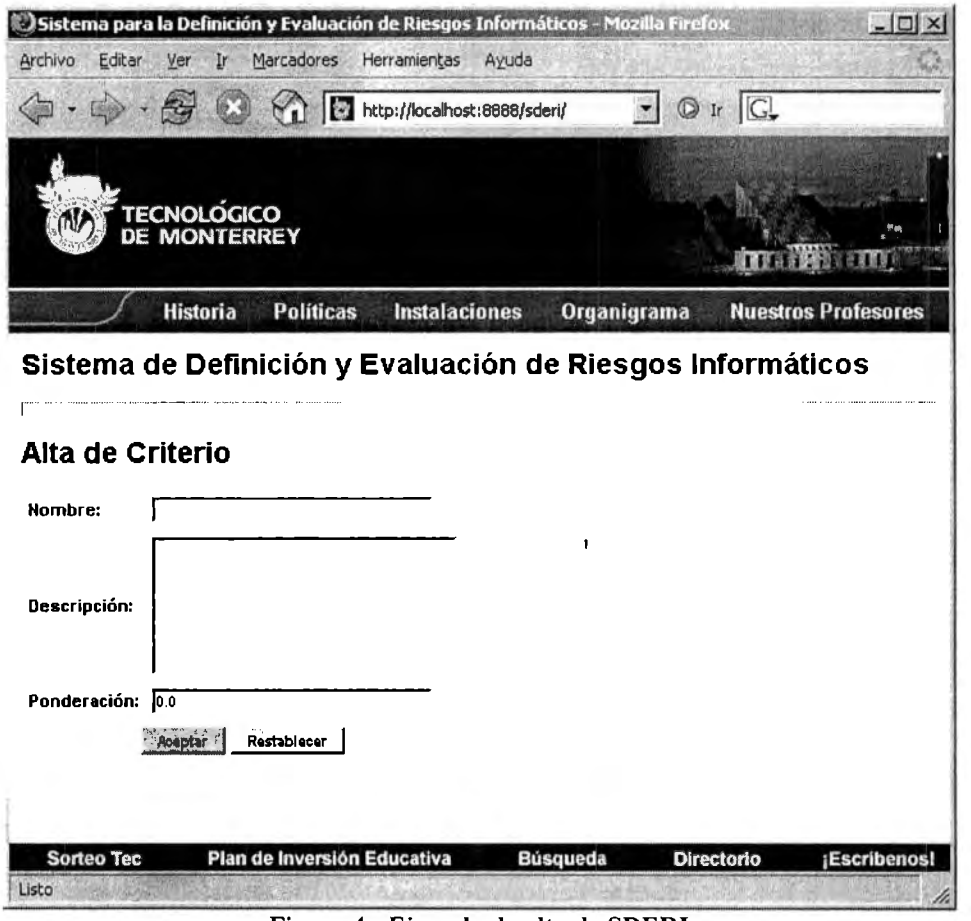

Figura 4 - Ejemplo de alta de SDERI

El modulo de alta permite capturar información. Y se asegura que todos los campos no estén vacios, a menos que alguno de ellos haya sido definido como opcional (por lo general la descripción es opcional). En caso de que se encuentre un campo vacio, manda el error de la figura 5.

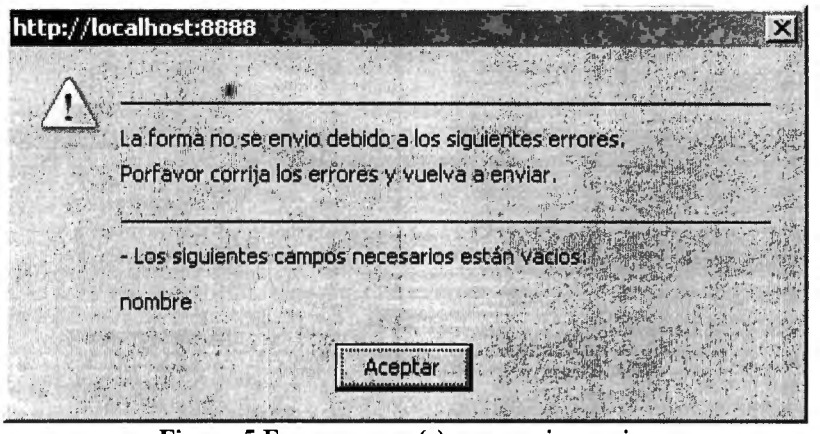

Figura 5 Error, campo(s) necesarios vacios

B.3.9.1.1.2. Ejecución del alta

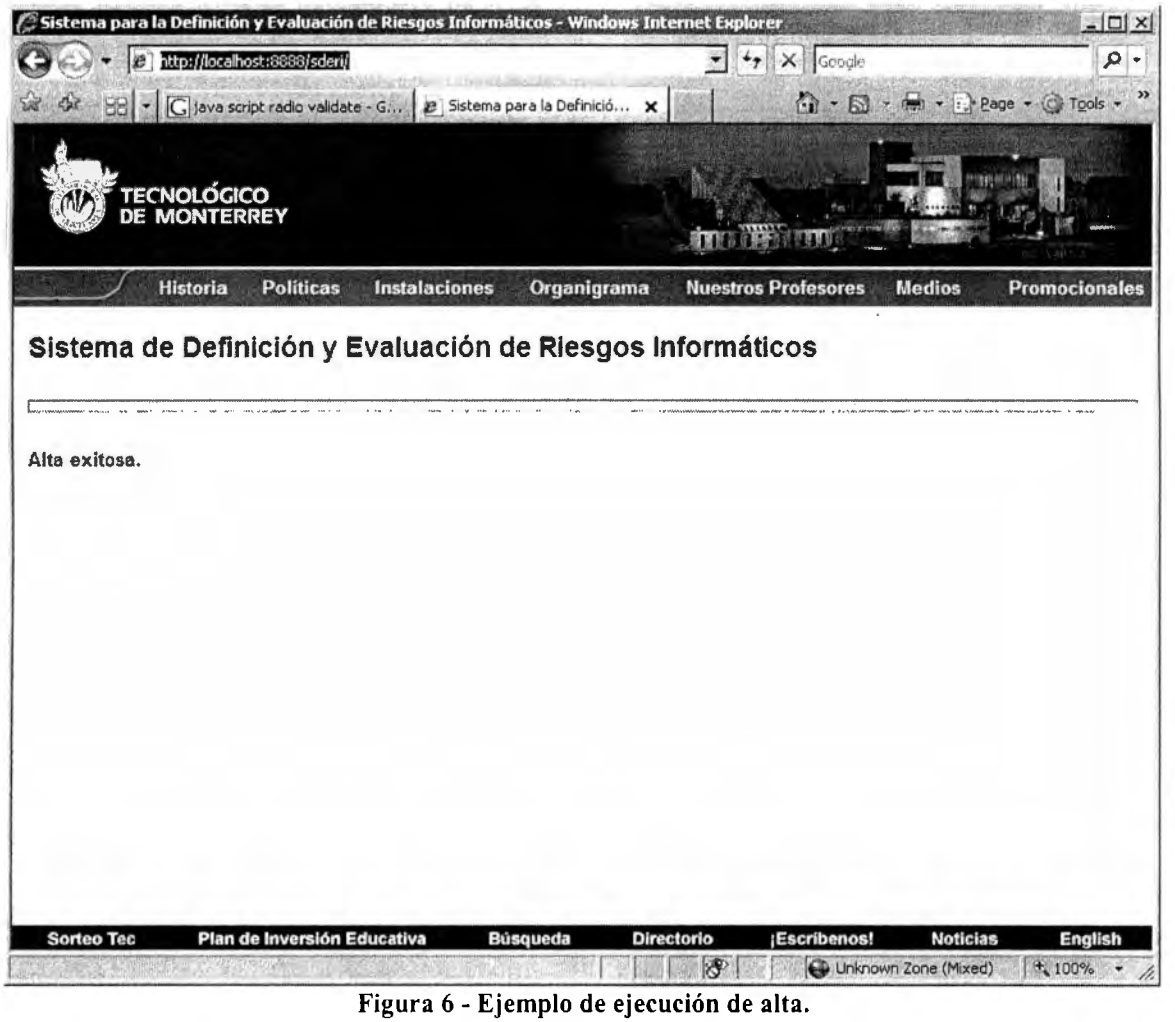

Figura 6 - Ejemplo de ejecución de alta.

La ejecución de alta, indica el status de este proceso.

B.3.9.2. Baja

B.3.9.2.2. Selección de los elementos a dar de baja

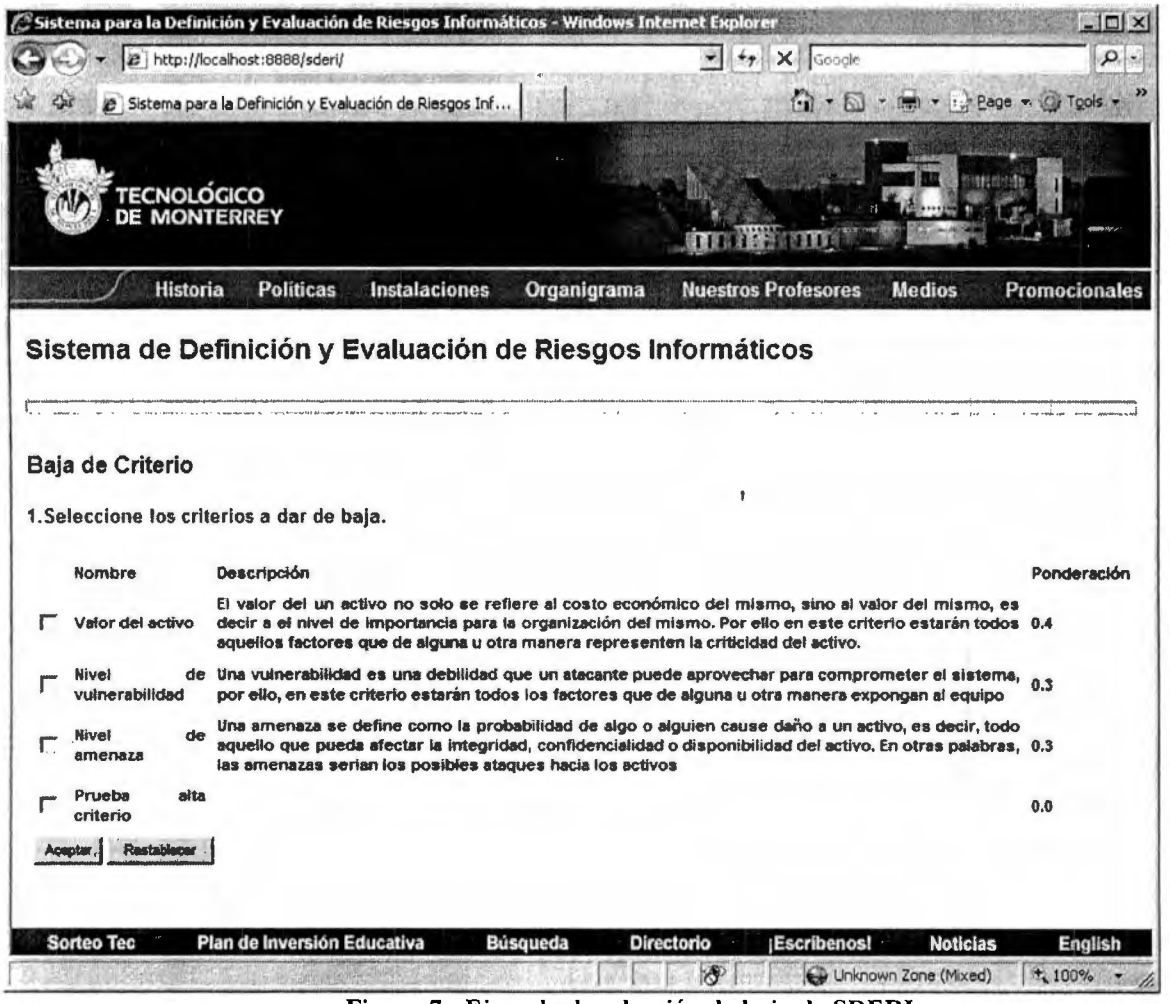

Figura 7 - Ejemplo de selección de baja de SDERI.

La ejecución de la baja es muy sencilla, tan solo se seleccionan los elementos que se deseen borrar.

B.3.9.2.2.3. Ejecución de la baja

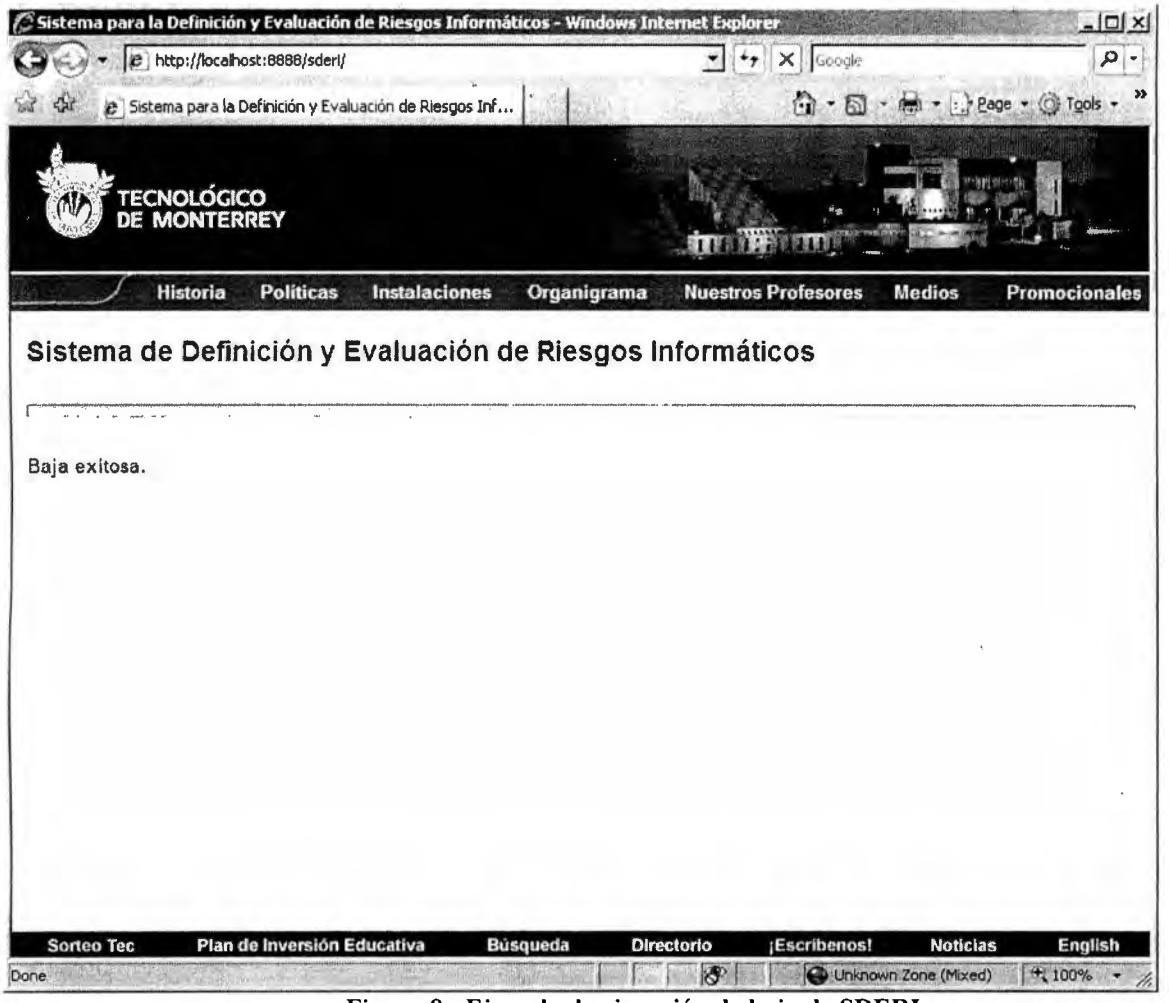

Figura 8 - Ejemplo de ejecución de baja de SDERI

Este modulo, notifica el status de la baja del elemento.

B.3.9.3. Cambio

B.3.9.3.3. Selección del elemento a hacer el cambio

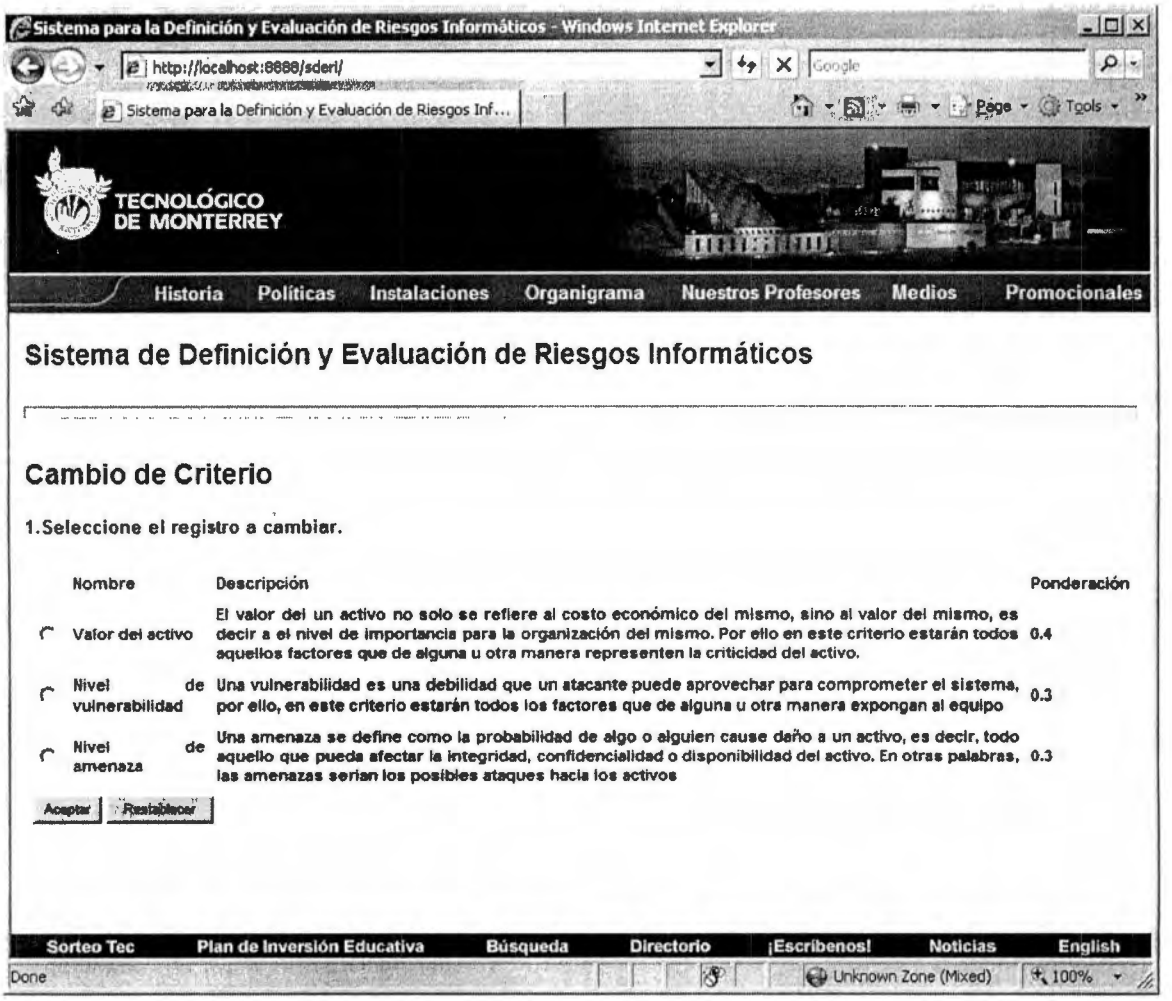

Figura 9 - Ejemplo de selección de cambio de SDERI

Para hacer algún cambio de un elemento, primero es necesario seleccionar el elemento a cambiar. En caso de que no se seleccione ningún elemento, entonces aparecerá un error parecido a la figura 10.

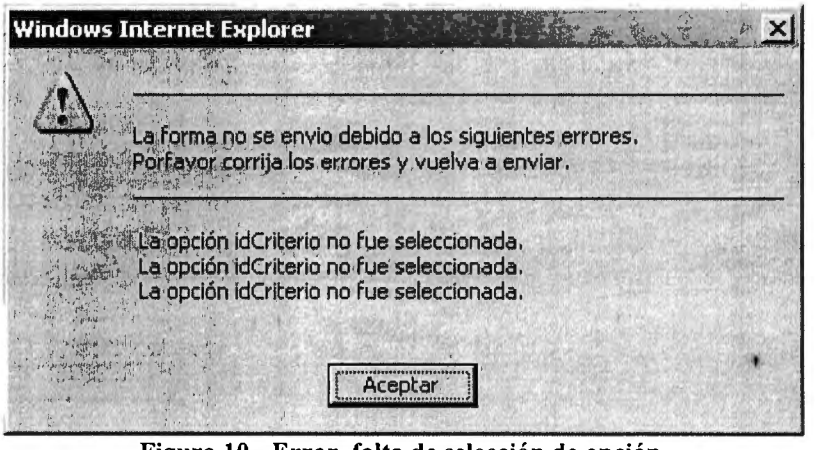

Figura 10 - Error, falta de selección de opción.

B.3.9.3.3.4. Captura del cambio

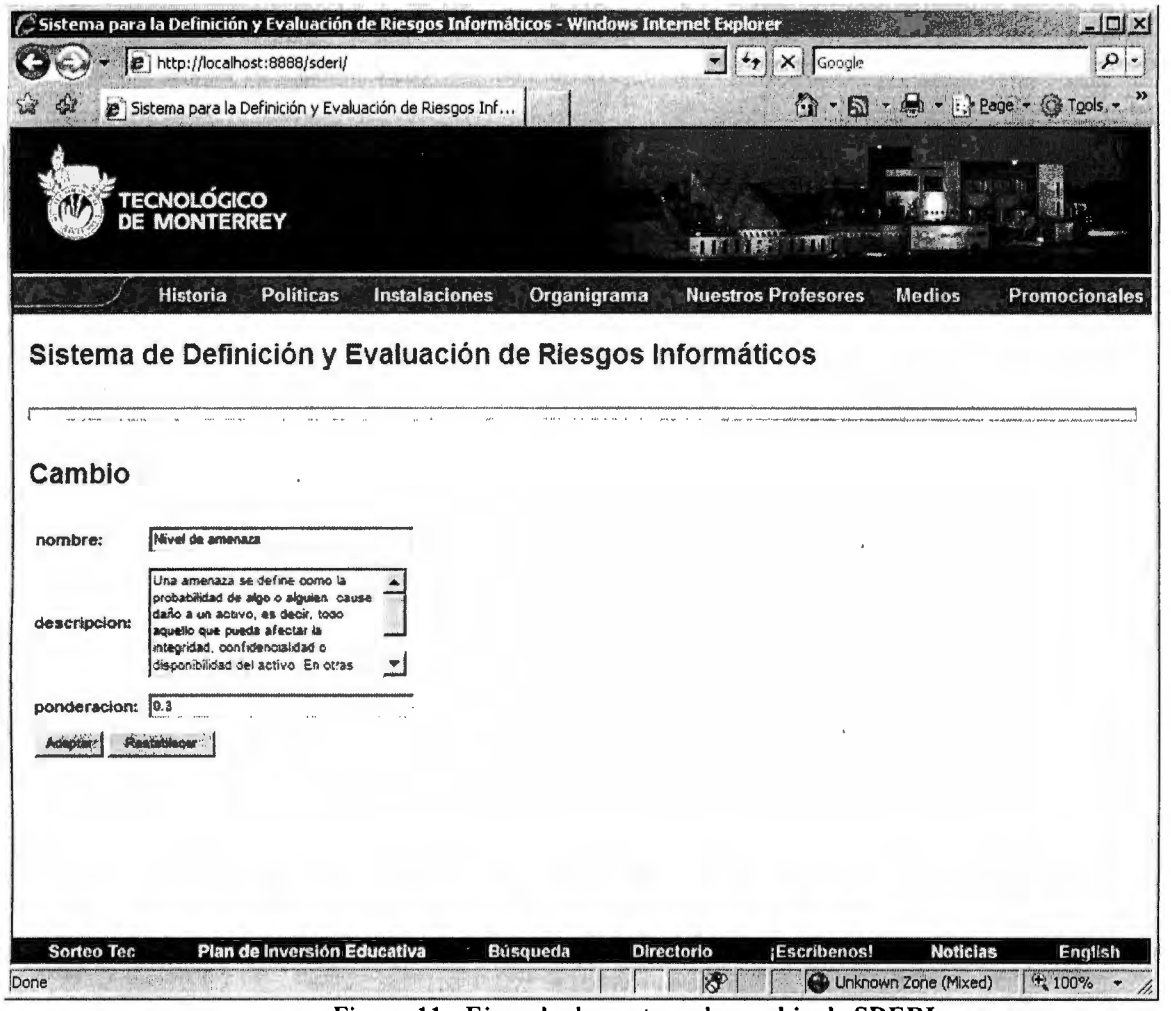

Figura 11 - Ejemplo de captura de cambio de SDERI

La captura del cambio, permite modificar un elemento previamente dado de alta. Este modulo mostrará el valor capturado en cada uno de los campos.

B.3.9.3.3.5. Ejecución del cambio

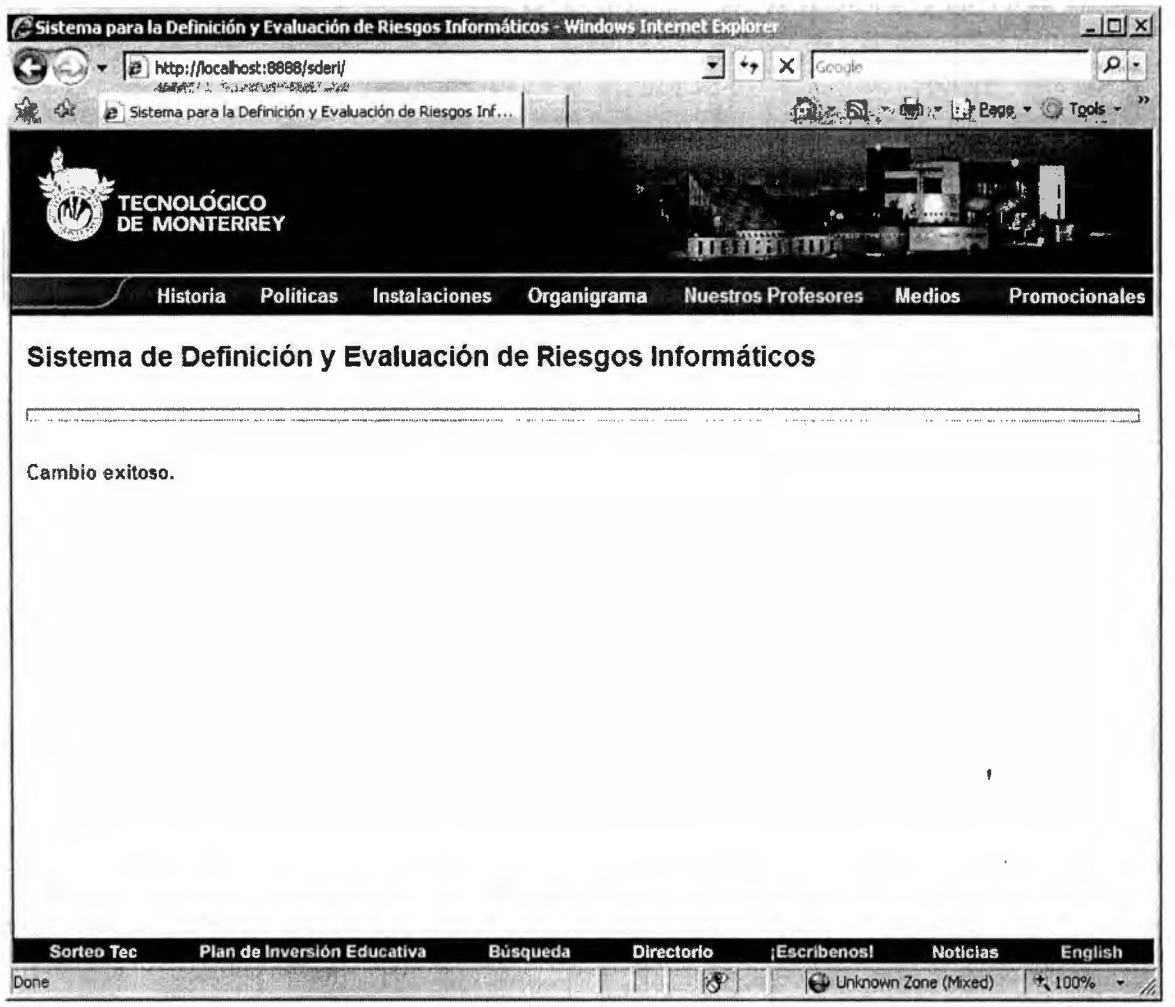

Figura 12 - Ejemplo de ejecución de cambio de SDERI.

Este modulo, notifica el status del cambio del elemento.

B.3.9.4. Sub-menú

El submenú, suele ser la última opción dentro de un menú. En realidad este modulo permite invocar otro servlet.

B.4. Mapa de Navegación

A continuación se enlista los diferentes módulos que conforman SDERI, así como los sub módulos de cada uno de ellos:

1) Inicio->Login->Menu General
    1) Administración de equipos
        a) Alta
            i) Captura del alta

(1) Ejecución del alta
b) Baja
i) Selección del(os) equipo(s) a dar de baja
(1) Ejecución de la baja
c) Cambio
i) Selección del equipo a hacer el cambio
(1) Captura del cambio
(a) Ejecución del cambio
2) Administración de la definición del nivel de Riesgo (Administración de criterios)
a) Alta
i) Captura del alta
(1) Ejecución del alta
b) Baja
i) Selección del(os) equipo(s) a dar de baja
(1) Ejecución de la baja
c) Cambio
i) Selección del equipo a hacer el cambio
(1) Captura del cambio
(a) Ejecución del cambio
d) Administración de factores
i) Alta
(1) Captura del alta
(a) Ejecución del alta
(b) Si es factor automático
(i) Captura de query para factor automático
1. Ejecución de alta de query para factor automático
ii) Baja
(1) Selección del(os) equipo(s) a dar de baja
(a) Ejecución de la baja
iii) Cambio
(1) Selección del equipo a hacer el cambio
(a) Captura del cambio
(i) Ejecución del cambio
iv) Administración de valores posibles
(1) Alta
(a) Captura del alta
(i) Ejecución del alta
(2) Baja
(a) Selección del(os) equipo(s) a dar de baja
(i) Ejecución de la baja
(3) Cambio
(a) Selección del equipo a hacer el cambio
(i) Captura del cambio
1. Ejecución del cambio

3) Administración de evaluaciones
a) Alta
i) Captura del alta
(1) Ejecución del alta

b) Baja
   i) Selección del(os) equipo(s) a dar de baja
      (1) Ejecución de la baja
c) Cambio
   i) Selección del equipo a hacer el cambio
      (1) Captura del cambio
         (a) Ejecución del cambio

4) Listado del nivel de riesgo de los equipos
   a) Listado detallado del nivel de riesgo de un equipo en particular
5) Administración de querys automáticos
   a) Alta
      i) Captura del alta
         (1) Ejecución del alta
   b) Baja
      i) Selección del(os) equipo(s) a dar de baja
         (1) Ejecución de la baja
   c) Cambio
      i) Selección del equipo a hacer el cambio
         (1) Captura del cambio
            (a) Ejecución del cambio
   d) Administración de conexiones a bases de datos
      i) Alta
         (1) Captura del alta
            (a) Ejecución del alta
      ii) Baja
         (1) Selección del(os) equipo(s) a dar de baja
            (a) Ejecución de la baja
      iii) Cambio
         (1) Selección del equipo a hacer el cambio
            (a) Captura del cambio
               (i) Ejecución del cambio

A partir de este mapa podemos apreciar que casi todo SDERI se comporta de la misma manera, lo que cambia es el elemento que se quiera modificar (Criterio, Factor, Valor posible, etc.).

B.5. Listado del nivel de riesgo de los equipos

Este modulo permite ver concentradamente cual es el nivel de riesgo de todos los equipos registrados en SDERI. En este módulo se muestran la evaluación de cada uno los equipos, especificando la evaluación de los criterios así como la evaluación total del nivel de riesgo (la

cual es la suma ponderada de la evaluación de los criterios). Un ejemplo de la evaluación la podemos ver en la figura 13.

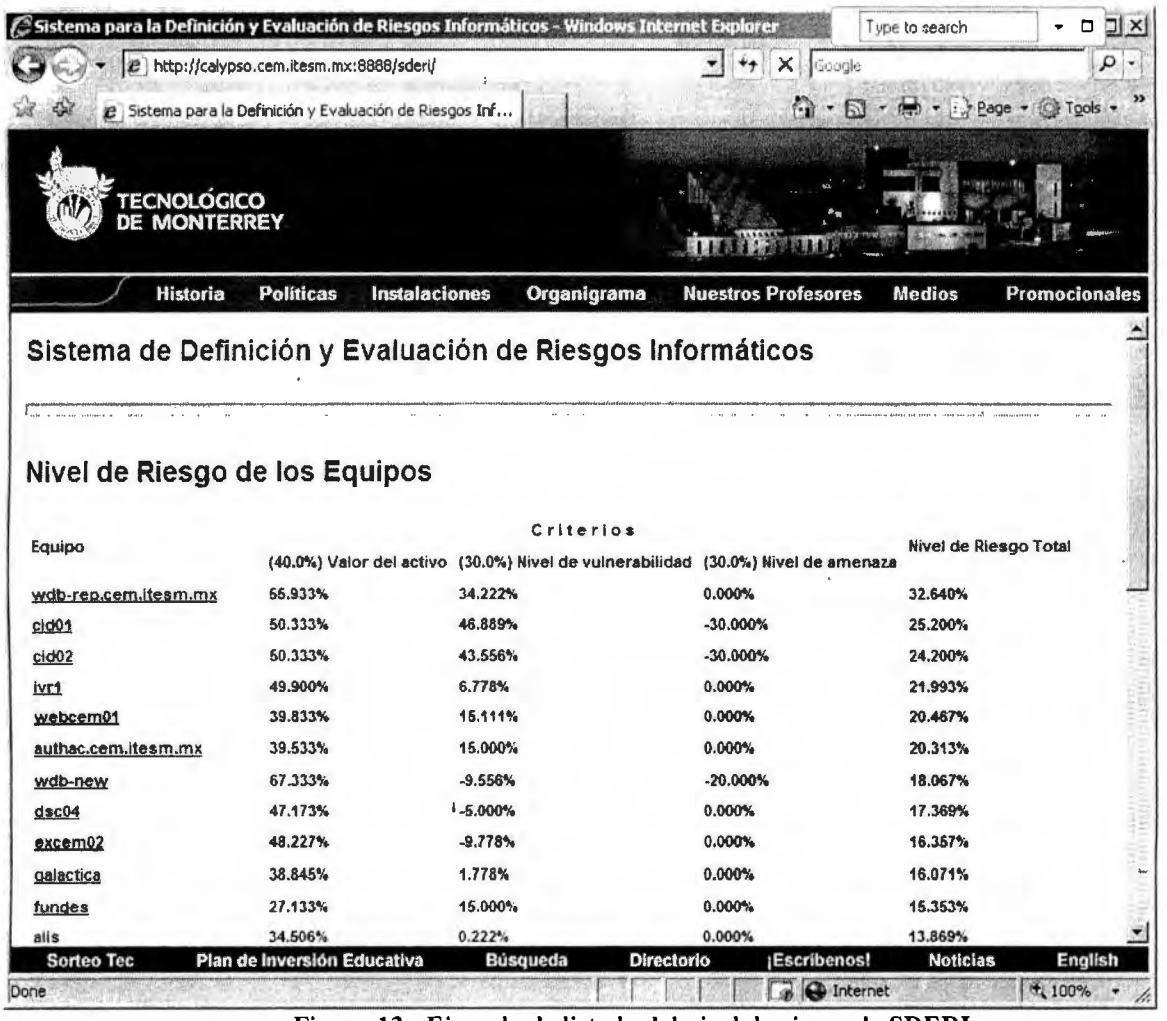

Figura 13 - Ejemplo de listado del nivel de riesgo de SDERI

Se puede ver el detalle de cada una de las evaluaciones al seleccionar la liga del equipo específico (Mostrado en la figura 14).

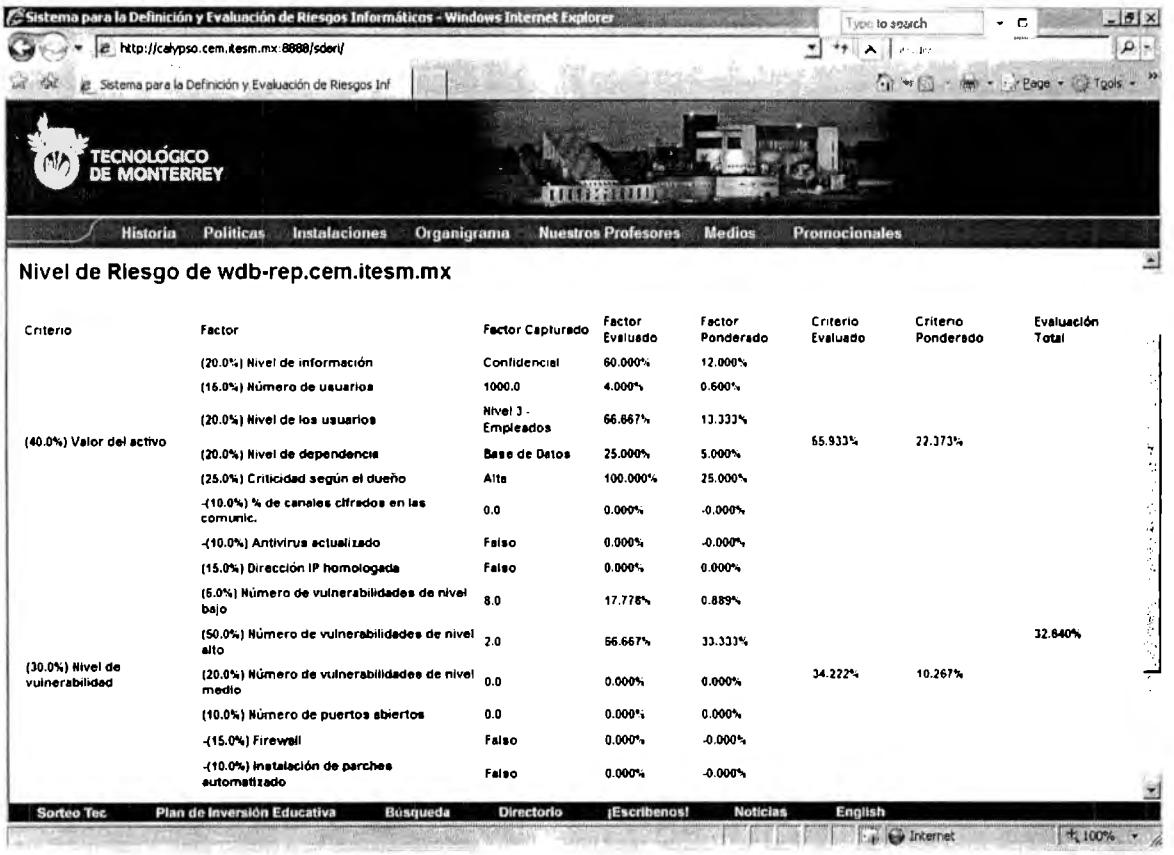

Figura 14 - Ejemplo de evaluación de riesgo de un equipo de SDERI

Este módulo permite ver a detalle la evaluación del nivel de riesgo, mostrando todos los criterios y factores que definen el nivel de riesgo. Mediante este modulo se puede determinar cuales son los factores que elevan el nivel de riesgo, de manera que se pueda tomar una decisión para administrar este riesgo.

Anexo C. Código Fuente del Sistema para la Definición y Evaluación de Riesgos Informáticos (SDERI).

```
package itesm.seguridad;
/*
 * Created on 25/10/2005
 * @author     Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *     Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Vector;
/**
 * Esta clase representa el calculo del nivel de riesgo de todos los equipos.
 * Los equipos se ordenan en base al nivel de riesgo.
 */
public class AnalisisRiesgo {
        DB db; // referencia a  la base de datos
        Vector equipos; // lista de equipos ordenados.
        /**
         * Único constructor que automaticamente realiza el calculo del
         * nivel de riesgo de todos los equipos.
         * @param db
         */
        AnalisisRiesgo(DB db) {
                this.db=db;
                equipos=  new Vector(4);
                getEquiposEvaluados();
        }

        /**
         * Cuenta la cantidad de equipos que se encuentren en la base de datos.
         * (En la tabla host)
         */
        public int getNumHosts() {
                int numHosts=0;
                String query="SELECT * FROM host ";
                try {
                        numHosts = db.contRows(query);
                }
                catch (BDException e) {
                        System.err.println("Query="+query);
                        e.printStackTrace();
                }
                return numHosts;
        }

        /**
         * Añade y ordena un equipo a la lista de equipos evaluados,
         * @param equipo
         */
        @SuppressWarnings("unchecked")
        private void addAndSort(EvaluacionEquipo equipo) {
                EvaluacionEquipo aux;
                int numEquipos = equipos.size(), i=0;
                for (i=0; i< numEquipos; i++) {
                        aux=(EvaluacionEquipo) equipos.get(i);
                        if (equipo.evaluacion >aux.evaluacion) {
                                break;
                        }
                }
                equipos.insertElementAt(equipo, i);
        }

        /**
         * Busca los equipos registrados y manda evaluar el nivel de riesgo
         * de cada uno de ellos
         */
```



```
        protected void getEquiposEvaluados() {
                String query="SELECT * FROM host";
                Statement stmt;
                ResultSet rs;
                Connection con;
                EvaluacionEquipo equipo;
                try {
                        con = db.newConection();
                        stmt = con.createStatement();
                        rs = stmt.executeQuery(query);
                        while(rs.next()) {
                                equipo=new EvaluacionEquipo(
                                                rs.getString("ip"), db);
                                addAndSort(equipo);
                        }
                        rs.close();
                        stmt.close();
                        con.close();
                }
                catch (SQLException e) {
                        System.err.println("Query="+query);
                        e.printStackTrace();
                }
                catch (BDException e) {
                        System.err.println("Query="+query);
                        e.printStackTrace();
                }
        }
}
```

```
package itesm.seguridad;
/*
 * Created on May 1, 2005
 * @author     Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import java.sql.SQLException;
/**
 * Esta clase representa una excepcion con la base de datos. La diferencia entre
 * esta clase y su superclase en que en el error tambien imprime el query que lo
 * ejecuto.
 */
public class BDException extends Exception {
        private static final long serialVersionUID = 784731053607522283 5L;
        //    Query Ejecutado.
        String query;
        // La expcion SQL que en si genera el problema.
        SQLException sqle;
        //String de conexion a la base de datos.
        String urlJDBC;

        /**
         * Obtiene el string de conexion a la base de datos
         * @return Returns the urlJDBC.
         */
        public String getUrlJDBC() {
                return urlJDBC;
        }

        /**
         * Define el string de conexion a la base de datos,
         * @param urlJDBC
         *        The urlJDBC to set.
         */
        public void setUrlJDBC(String urlJDBC) {
                this.urlJDBC = urlJDBC;
        }

        /**
         * Nos aseguramos que se cree la superclase.
         */
        public BDException() {
                super();
        }

        /**
         * Constructor para inicializar el query y la excepcion SQL
         * @param s Exception SQL
         * @param q Query
         */

        public BDException(SQLException s, String q) {
                this.sqle = s;
                this.query = q;
        }

        /**
         * Regresa la excepción original
         * @return Returns the sqle.
         */
        public SQLException getSqle() {
                return sqle;
        }

        /**
         * Establece la excepción original
         * @param sqle
```



```
         *            The sqle to set.
         */
        public void setSqle(SQLException sqle) {
                this.sqle = sqle;
        }

        /**
         * Obtiene el query que genera la excepcion
         * @return Returns the query.
         */
        public String getQuery() {
                return query;
        }

        /**
         * Establece el query que genera la excepción
         * @param query
         *            The query to set.
         */
        public void setQuery(String query) {
                this.query = query;
        }
}
```

```
package itesm.seguridad;
/*
 * @author    Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *     Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
/**
 * Esta clase representa un registro de la tabla conexionBD. Usada para
 * almacenar los datos de conexión para la automatización de factores.
 */
public class ConexionBd {
        public int idConexionBD;
        public String nombre;
        public String descripcion;
        public String usuario;
        public String password;
        public String javaDriver;
        public String servidor;
        public String puerto;
        public String bd;
        public String url;
        public int tipoBD;

        /**
         * Constructor sin argumentos que no inincializa nada.
         */
        public ConexionBd() {
        }

        /**
         * Constructor que especifica todos los atributos de la clase.
         */
        public ConexionBd(int tipoBD, int idConexionBD, String nombre,
                String descripcion, String usuario, String password,
                String javaDriver, String servidor, String puerto,
                String bd, String url) {
                this.tipoBD=tipoBD;
                this.idConexionBD = idConexionBD;
                this.nombre = nombre;
                this.descripcion = descripcion;
                this.usuario = usuario;
                this.password = password;
                this.javaDriver = javaDriver;
                this.servidor = servidor;
                this.puerto = puerto;
                this.bd = bd;
                this.url = url;
        }

        /**
         * Constructor que especifica los elementos esenciales de la clase.
         */
        public ConexionBd(String usuario, String password, String javaDriver,
                String servidor, String puerto, String bd, String url) {
                idConexionBD=0;
                nombre = "Default";
                descripcion="";
                this.usuario = usuario;
                this.password = password;
                this.javaDriver = javaDriver;
                this.servidor = servidor;
                this.puerto = puerto;
                this.bd = bd;
                this.url = url;
        }
}
```

```
package itesm.seguridad;
/*
 * Created on 19/05/2005
 * @author    Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *     Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
/**
 * Esta clase representa un registro de la tabla Criterio, que en forma genérica
 * es un conjunto de Factores  a los cuales se les da una ponderación.
 */
public class Criterio {
        int idCriterio;
        String nombre;
        String descripcion;
        double ponderacion;
        /**
         * Busca un criterio a partir del id del mismo.
         * @param idCriterio    el identificador del criterio.
         * @param db    referencia a la base
         * @return
         */
        public static Criterio searchCriterio (String idCriterio, DB db){
                Statement stmt = db.getStmt();
                ResultSet rs;
                Criterio c = new Criterio();
                String query = "SELECT * FROM Criterio WHERE idCriterio ="
                        +idCriterio;
                try {
                        rs = stmt.executeQuery(query);
                        if (rs.next()) {
                                c.idCriterio= rs.getInt("idCriterio");
                                c.nombre=rs.getString("nombre");
                                c.descripcion =rs.getString("descripcion");
                                c.ponderacion= rs.getDouble("ponderacion");
                        } else {
                                c = null;
                        }
                } catch (SQLException e) {
                        System.err.println("Query =" + query);
                        System.err.println("SQLException: " + e.getMessage());
                        e.printStackTrace();
                }
                return c;
        }
}
```

```
package itesm.seguridad;
/*
 * Created on 23/10/2005
 * @author    Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *     Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
/**
 * Esta clase representa la evaluación de un criterio de un equipo.
 */
public class CriterioEvaluado {
        Criterio criterio;
        FactorEvaluado factores[];
        double evaluacion;
        String ip;
        DB db;

        /**
         * Único costructor que inicializa las varibles criterio,
         * ip y db y además manda calcular la evalución del criterio.
         * @param ip
         * @param c
         * @param db
         */
        public CriterioEvaluado(String ip, Criterio c, DB db) {
                criterio = c;
                this.db = db;
                this.ip = ip;
                setEvaluacionCriterio();
        }

        /**
         * Cuenta la cantidad de factores que tenga el criterio en cuestion.
         * @return
         */
        public int getNumFactores() {
                int numFactores = 0;
                String query = "SELECT * FROM Factor Where idCriterio="
                        + criterio.idCriterio;
                try {
                        numFactores = db.contRows(query);
                } catch (BDException e) {
                        System.err.println("Query=" + query);
                        e.printStackTrace();
                }
                return numFactores;
        }

        /**
         * Manda evaluar cada uno de los factores que comprende el criterio.
         */
        private void getFactoresEvaluados() {
                int numFactores = getNumFactores();
                Factor f;
                factores = new FactorEvaluado[numFactores];
                String query = "SELECT * FROM Factor Where idCriterio="
                        + criterio.idCriterio;
                Statement stmt;
                ResultSet rs;
                Connection con;
                try {
                        con = db.newConection();
                        stmt = con.createStatement();
                        rs = stmt.executeQuery(query);
```



```
                        int i = 0;
                        while (rs.next()) {
                                f = Factor.searchFactor(rs.getInt("idFactor")
                                                + "", db);
                                factores[i] = new FactorEvaluado(ip, f, db);
                                i++;
                        }
                        rs.close();
                        stmt.close();
                        con.close();
                } catch (SQLException e) {
                        System.err.println("Query=" + query);
                        e.printStackTrace();
                } catch (BDException e) {
                        System.err.println("Query=" + query);
                        e.printStackTrace();
                }
        }

        /**
         * Manda evaluar los factores y despues calcular el nivel de riesgo del
         * criterio al hacer una suma ponderada cada uno de los factores
         * evaluados.
         */
        public void setEvaluacionCriterio() {
                evaluacion = 0;
                getFactoresEvaluados();
                for (int i = 0; i < factores.length; i++) {
                        evaluacion += factores[i].evaluacion
                                        * factores[i].factor.ponderacion;
                }
        }
}
```

```
package itesm.seguridad;
/*
 * Created on 15/02/2004
 * @author     Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *     Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import itesm.utilsdsc.*;
import java.sql.*;
import java.util.Enumeration;
/**
 * Clase para el manejo de la base de datos.
 */
public class DB {
        public static final int DBTYPE_MYSQL=1;
        public static final int DBTYPE_MSSQL=2;
        // Variables de bases de datos , la coneccion y el statment (usado para
        // ejecutar querys)
        Connection con;
        private String driverDB = "com.mysql.jdbc.Driver";
        private Statement stmt;
        private String urlJDBC =
                "jdbc:mysql://localhost/sderi?user=root&password=am33awal";

        /**
         * constructor general que iniciliza la base de datos (al llamar init)
         */
        public DB() throws BDException {
                init();
        }

        /**
         * Añade un registro a partir de:
         * @param tabla
         * @param data  - datos a añadir
         */
        public int add(String tabla, SortedHashtable data) throws BDException {
                String query;
                int temp = 0;
                query = Utils.createAddQuery(tabla, data);
                try {
                        temp = stmt.executeUpdate(query);
                } catch (SQLException e) {
                        System.err.println("Query =" + query);
                        System.err.println("SQLException: " + e.getMessage());
                        e.printStackTrace();
                        BDException bde = new BDException(e, query);
                        throw bde;
                }
                return temp;
        }

        /**
         * Cuenta la cantidad de registros de un query
         * @param tabla
         * @param data  - datos a añadir
         */
        public int contRows(String query) throws BDException {
                Connection myCon = newConection();
                Statement myStmt;
                int aux = 0;
                ResultSet rs;
                try {
                        myStmt = myCon.createStatement();
                        rs = myStmt.executeQuery(query);
                        while (rs.next()) {
                                aux++;
                        }
```



```
                        rs.close();
                        myStmt.close();
                        myCon.close();
                } catch (SQLException e) {
                        System.err.println("Query =" + query);
                        System.err.println("SQLException: " + e.getMessage());
                        e.printStackTrace();
                        BDException bde = new BDException(e, query);
                        throw bde;
                }
                return aux;
        }

        /**
         * Añade un registro a partir de:
         * @param tabla
         * @param campos
         * @param valores
         */
        public int add(String tabla, String[] campos, String[] valores)
                throws BDException {
                String query;
                int temp = 0;
                query = Utils.createAddQuery(tabla, campos, valores);
                try {
                        temp = stmt.executeUpdate(query);
                } catch (SQLException e) {
                        System.err.println("Query =" + query);
                        System.err.println("SQLException: " + e.getMessage());
                        e.printStackTrace();
                        BDException bde = new BDException(e, query);
                        throw bde;
                }
                return temp;
        }

        /**
         * Cambia un registro a partir de:
         * @param tabla
         * @param data
         * @param donde
         */
        public int change(String tabla, SortedHashtable data, String donde)
                throws BDException {
                String query;
                int temp = 0;
                query = Utils.createUpdateQuery(tabla, data, donde);
                try {
                        temp = stmt.executeUpdate(query);
                } catch (SQLException e) {
                        System.err.println("Query =" + query);
                        System.err.println("SQLException: " + e.getMessage());
                        e.printStackTrace();
                        BDException bde = new BDException(e, query);
                        throw bde;
                }
                return temp;
        }

        /**
         * Cambia un registro a partir de:
         * @param tabla
         * @param campos
         * @param valores
         * @param donde
         */
        public int change(String tabla, String[] campos, String[] valores,
                String donde) throws BDException {
                String query;
```

```
                int temp = 0;
                query = Utils.createUpdateQuery(tabla, campos, valores, donde);
                try {
                        temp = stmt.executeUpdate(query);
                } catch (SQLException e) {
                        System.err.println("Query =" + query);
                        System.err.println("SQLException: " + e.getMessage());
                        e.printStackTrace();
                        BDException bde = new BDException(e, query);
                        throw bde;
                }
                return temp;
        }

        /**
         * Borra  registro(s) a partir de:
         * @param tabla
         * @param data - datos a añadir
         */
        public int del(String tabla, SortedHashtable data) throws BDException {
                int temp = 0;
                String query, key, val[];
                Enumeration keys = data.keys();
                while (keys.hasMoreElements()) {
                        key = (String) keys.nextElement();
                        val = (String[]) data.get(key);
                        for (int i = 0; i < val.length; i++) {
                                query = Utils.createDelQuery(tabla, key, val[i])
;
                                try {
                                        temp = stmt.executeUpdate(query);
                                } catch (SQLException e) {
                                        System.err.println("Query =" + query);
                                        System.err.println("SQLException: "
                                                        + e.getMessage());
                                        e.printStackTrace();
                                        BDException bde =
                                                new BDException(e, query);
                                        throw bde;
                                }
                        }
                }
                return temp;
        }

        /**
         * Elimina registros a partir la TABLA donde CAMPO sea igual a VALOR:
         * @param tabla
         * @param campo
         * @param valor
         */
        public int del(String tabla, String campo, String valor)
                throws BDException {
                String query;
                int temp = 0;
                query = Utils.createDelQuery(tabla, campo, valor);
                try {
                        temp = stmt.executeUpdate(query);
                } catch (SQLException e) {
                        System.err.println("Query =" + query);
                        System.err.println("SQLException: " + e.getMessage());
                        e.printStackTrace();
                        BDException bde = new BDException(e, query);
                        throw bde;
                }
                return temp;
        }

        /**
```



```
         * Se desconecta de la base de datos.
         */
        public void destroy() throws BDException {
                try {
                        stmt.close();
                        con.close();
                } catch (SQLException e) {
                        System.err.println("SQLException: " + e.getMessage());
                        BDException bde = new BDException(e, "");
                        throw bde;
                }
        }

        /**
         * Regresa el statement, sobre el cual se pueden ejecutar
         * querys directamete.
         * @return
         */
        public Statement getStmt() {
                return stmt;
        }

        /**
         * Obtiene el primer valor del "Campo" de la "Tabla" que cumpla
         * con "Donde"
         * @param tabla
         * @param campos
         * @param donde
         */
        public String getStringValue(String tabla, String campo, String donde)
                throws BDException {
                String query, res = "";
                ResultSet rs;
                query = Utils.createSelectQuery(tabla, campo, donde);
                try {
                        rs = stmt.executeQuery(query);
                        boolean haveElements = false;
                        if (rs.next()) {
                                haveElements = true;
                        }
                        if (haveElements) {
                                res = rs.getString(campo);
                        } else {
                                res = null;
                        }
                } catch (SQLException e) {
                        System.err.println("Query =" + query);
                        System.err.println("SQLException: " + e.getMessage());
                        e.printStackTrace();
                        BDException bde = new BDException(e, query);
                        throw bde;
                }
                return res;
        }

        /**
         * Obtiene el primer valor del "Campo" de la "Tabla" que cumpla
         * con "Donde"
         * @param tabla
         * @param campos
         * @param donde
         */
        public int getIntegerValue(String tabla, String campo, String donde)
                throws BDException {
                String query;
                int res = 0;
                ResultSet rs;
                query = Utils.createSelectQuery(tabla, campo, donde);
                try {
```

```
                rs = stmt.executeQuery(query);
                boolean haveElements = false;
                if (rs.next()) {
                        haveElements = true;
                }
                if (haveElements) {
                        res = rs.getInt(campo);
                } else {
                        res = 0;
                }
        } catch (SQLException e) {
                System.err.println("Query =" + query);
                System.err.println("SQLException: " + e.getMessage());
                e.printStackTrace();
                BDException bde = new BDException(e, query);
                throw bde;
        }
        return res;
}

/**
 * Inicializa la base de datos (se conecta)
 */
public void init() throws BDException {
        // Leemos el driver de la base de datos.
        try {
                Class.forName(driverDB);
        } catch (java.lang.ClassNotFoundException e) {
                System.err.print("ClassNotFoundException: ");
                System.err.println(e.getMessage());
        }
        // Nos conectamos con la base de datos.
        try {
                // con = DriverManager.getConnection
                // (urlJDBC, userDB, userPasswdDB);
                con = DriverManager.getConnection(urlJDBC);
                stmt = con.createStatement();
        } catch (SQLException e) {
                System.err.println("SQLException: " + e.getMessage());
                System.err.println("urlJDBC=" + urlJDBC);
                BDException bde = new BDException(e, "");
                bde.setUrlJDBC(urlJDBC);
                throw bde;
        }
}

/**
 * Crea una nueva conexion con la BD.
 */
public Connection newConection() throws BDException {
        Connection myCon;
        // Leemos el driver de la base de datos.
        try {
                Class.forName(driverDB);
        } catch (java.lang.ClassNotFoundException e) {
                System.err.print("ClassNotFoundException: ");
                System.err.println(e.getMessage());
        }
        // Nos conectamos con la base de datos.
        try {
                myCon = DriverManager.getConnection(urlJDBC);
        } catch (SQLException e) {
                System.err.println("SQLException: " + e.getMessage());
                System.err.println("urlJDBC=" + urlJDBC);
                BDException bde = new BDException(e, "");
                bde.setUrlJDBC(urlJDBC);
                throw bde;
        }
        return myCon;
```



```
}

/**
 * Crea una nueva conexion a la BD, especificando todos los parametros.
 */
public Connection newConection(int tipoBD, String javaDriver,
        String servidor, String puerto, String db, String url,
        String usuario, String password) throws BDException {
        Connection myCon;
        String urlJDBC = getConnectionUrl(tipoBD, url,
                        servidor, puerto, db);
        // Leemos el driver de la base de datos.
        try {
                Class.forName(javaDriver);
        } catch (java.lang.ClassNotFoundException e) {
                System.err.print("ClassNotFoundException: ");
                System.err.println(e.getMessage());
        }
        // Nos conectamos con la base de datos.
        try {
                myCon = DriverManager.getConnection(
                                urlJDBC, usuario, password);
        } catch (SQLException e) {
                System.err.println("SQLException: " + e.getMessage());
                System.err.println("urlJDBC=" + urlJDBC);
                System.err.println("Driver="+javaDriver);
                BDException bde = new BDException(e, "");
                bde.setUrlJDBC(urlJDBC);
                throw bde;
        }
        return myCon;
}

/**
 * Crea una nueva conexion pasando como paramegtros una ConexionBd
 */
public Connection newConection (ConexionBd conBd) throws BDException{
        return newConection(conBd.tipoBD, conBd.javaDriver,
                conBd.servidor, conBd.puerto, conBd.bd, conBd.url,
                conBd.usuario,  conBd.password );
}

/**
 * Crea el string de conexion, depediendo del tipo de base de datos.
 */
private String getConnectionUrl(int tipoBD, String url, String servidor,
        String puerto, String db) {
        String res="";
        switch (tipoBD) {
                case DBTYPE_MSSQL:
                        res= url + servidor + ":"
                                + puerto + ";databaseName=" + db;
                        break;
                case DBTYPE_MYSQL:
                        res= url+servidor+":"+puerto+"/"+db;
                        break;
        }
        return res;
}
/**
 * Crea una nueva conexion de base de datos,
 * obteniendo la información de la base de datos.
 */
public Connection newConectionFromdb (int idConexionBD)
        throws BDException{
        Connection con1=null, con2=null;
        ResultSet rs=null;
        Statement stmt=null;
        String query = "SELECT * FROM conexionBd WHERE idConexionBD="
```

```
                + idConexionBD;
        con1=newConection();
        ConexionBd condb=new ConexionBd();
        try {
                stmt =con1.createStatement();
                rs = stmt.executeQuery(query);
                boolean haveElements = false;
                if (rs.next()) {
                        haveElements = true;
                }
                if (haveElements) {
                        condb.idConexionBD = idConexionBD;
                        condb.tipoBD = rs.getInt("tipoBD");
                        condb.nombre=rs.getString("nombre");
                        condb.descripcion=rs.getString("descripcion");
                        condb.usuario=rs.getString("usuario");
                        condb.password=rs.getString("password");
                        condb.javaDriver=rs.getString("javaDriver");
                        condb.servidor=rs.getString("servidor");
                        condb.puerto=rs.getString("puerto");
                        condb.bd=rs.getString("bd");
                        condb.url=rs.getString("url");
                        con2 = newConection( condb);
                }

        } catch (SQLException e) {
                System.err.println("Query =" + query);
                System.err.println("SQLException: " + e.getMessage());
                e.printStackTrace();
                BDException bde = new BDException(e, query);
                throw bde;
        }
        return con2;
}

/*
 * Despliega los detalles de la conexion. Metodo usado para debuguear.
 */
public void displayDbProperties(Connection con) {
        DatabaseMetaData dm = null;
        ResultSet rs = null;
        try {
                if (con != null) {
                        dm = con.getMetaData();
                        System.out.println("Driver Information");
                        System.out.println("\tDriver Name: "
                                        + dm.getDriverName());
                        System.out.println("\tDriver Version: "
                                        + dm.getDriverVersion());
                        System.out.println("\nDatabase Information ");
                        System.out.println("\tDatabase Name: "
                                + dm.getDatabaseProductName());
                        System.out.println("\tDatabase Version: "
                                + dm.getDatabaseProductVersion());
                        System.out.println("Avalilable Catalogs ");
                        rs = dm.getCatalogs();
                        while (rs.next()) {
                                System.out.println("\tcatalog: "
                                                + rs.getString(1));
                        }
                        rs.close();
                        rs = null;
                } else
                        System.out.println("Error: "
                                        +"No active Connection");
        } catch (Exception e) {
                e.printStackTrace();
        }
        dm = null;
```



```
        }
        /**
         * Metodo principal, usado para pruebas.
         */
        public static void main(String[] args) {
                DB db = null;
                Connection con= null;
                String javaDriver =
                                "com.microsoft.jdbc.sqlserver.SQLServerDriver";
                String servidor = "10.48.17.103";
                String puerto = "1433";
                String db2 = "scan7db";
                String url = "jdbc:microsoft:sqlserver://";
                String usuario = "sderi";
                String password = "SD3r1c#m";
                ConexionBd conBd=null;
                if (conBd!=null){puerto=puerto+"123";}
                try {
                        conBd =new ConexionBd(usuario, password, javaDriver ,
                                servidor, puerto,  db2,url);
                        db = new DB();
                        con = db.newConectionFromdb(1);
                        db.displayDbProperties(con);
                        con.close();
                        con = db.newConection();
                        db.displayDbProperties(con);
                        con.close();
                } catch (BDException e) {
                        e.printStackTrace();
                } catch (SQLException e) {
                        e.printStackTrace();
                }
        }
} // class DB
```

```
package itesm.seguridad;
/*
 * @author    Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *     Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import java.sql.*;
import java.util.*;
import itesm.utilsdsc.*;
/**
 * Clase generica para la manipulacion de una base de datos a través de
 * una interfaz web usando servlets.
 */
public class DbWeb {

        //*********************************************
        // CONSTANTES
        //*********************************************
        /**
         * public static final int NUMERIC_TYPE Clasificación general para datos
         * numericos
         */
        public static final int NUMERIC_TYPE = 1;

        /**
         * public static final int STRING_TYPE Clasificación general para datos
         * string
         */
        public static final int STRING_TYPE = 0;

        /**
         * public static void main(String args[])
         * Metodo de prueba de la clase
         *
         * @param args
         */
        public static void main(String args[]) {
                DbWeb dbweb = new DbWeb();
                dbweb.setQuery("SELECT idCriterio, nombre, descripcion "
                                +"FROM Criterio");
                System.out.println(dbweb.analyzeTable());
        }

        //*********************************************
        // VARIABLES GLOBALES
        //*********************************************

        /**
         * Representa la el servet a invocar cuando la forma se ejecute
         */
        protected String action;

        /**
         * Representa la referencia a la base de datos.
         */
        protected DB db;

        /**
         * Conjunto de fieldInfo los cuales contienen todos los datos
         * de los campos.
         */
        private Vector fields;

        /**
         * Conjunto datos a guardarse como campos ocultos (<INPUT TYPE =HIDDEN>)
         */
        protected SortedHashtable hiddenFields;
```



```
        /**
         * Conjunto de datos a sustiturise el el pageGenerator correspondiente
         * (*LLAVE*) = valor
         */
        protected SortedHashtable otherProperties;

        /**
         * Conjunto de datos a sustituirse a la hora de invocar el javaScript
         * verificador (el que valide la forma). El objetivo es crear algo
         * parecido a: this.firstname.optional = true;
         * this.phonenumber.optional = true;
         * this.zip.min = 0; this.zip.max = 99999;
         */
        protected SortedHashtable javaScriptProperties;

        /**
         * Query a ejecutar.
         */
        private String query;

        /**
         * Bandera que dermina si se esta buscando valores,
         * para diferenciar si se esta haciendo una alta o un cambio
         */
        private boolean searchingValues = false;

        /**
         * Título de pagina a generar.
         */
        protected String title;

        //Constructores
        /**
         * Inicializa con los datos por default (sin datos, pero no nulos)
         */
        DbWeb() {
                try {
                        db = new DB();
                } catch (BDException e) {
                        System.err.print("Error al inicializar la base "
                                        +"de datos: \n");
                        System.err.print("UrlJDBC =" + e.getUrlJDBC() + "\n");
                        System.err.println("Mensaje ="
                                        + e.sqle.getMessage() + "\n");
                        e.printStackTrace();
                }
                hiddenFields = new SortedHashtable();
                otherProperties = new SortedHashtable();
                javaScriptProperties = new SortedHashtable();
                query = "";
                title = "";
                action = "";
        }

        /**
         * Inicializa especificando el query.
         */
        DbWeb(String q) {
                query = q;
                try {
                        db = new DB();
                } catch (BDException e) {
                        System.err.print("Error al inicializar la base "+
                                        "de datos: \n");
                        System.err.print("UrlJDBC =" + e.getUrlJDBC() + "\n");
                        System.err.println("Mensaje ="
                                        + e.sqle.getMessage() + "\n");
                        e.printStackTrace();
                }
```

```
                hiddenFields = new SortedHashtable();
                otherProperties = new SortedHashtable();
                javaScriptProperties = new SortedHashtable();
                fields = analyzeTable();
                title = "";
                action = "";
        }

        /**
         * Inicializa especificando el titulo y la accion
         */
        DbWeb(String title, String action) {
                this.title = title;
                this.action = action;
                try {
                        db = new DB();
                } catch (BDException e) {
                        System.err.print("Error al inicializar la base "
                                        +"de datos: \n");
                        System.err.print("UrlJDBC =" + e.getUrlJDBC() + "\n");
                        System.err.println("Mensaje =" +
                                        e.sqle.getMessage() + "\n");
                        e.printStackTrace();
                }
                hiddenFields = new SortedHashtable();
                otherProperties = new SortedHashtable();
                javaScriptProperties = new SortedHashtable();
                query = "";
                title = "";
                action = "";
        }

        //+++++++++++++++++++++++++++++++++++++++++++
        // Metodos
        //+++++++++++++++++++++++++++++++++++++++++++

        /**
         * Agrega un campo oculto
         */
        public void addHiddenField(String fieldName, String value) {
                hiddenFields.put(fieldName, value);
        }

        /**
         * Agrega una propiedad (para ser remplazada dentro de un PageGenerator.
         * @param propertie
         * @param value
         */
        public void addPropertie(String propertie, String value) {
                otherProperties.put(propertie, value);
        }

        /**
         * Agrega una propiedad del java script
         * @param propertie
         * @param value
         */
        public void addJavaScriptPropertie(String propertie, String value) {
                javaScriptProperties.put(propertie, value);
        }

        /**
         * Regresa un vector que contiene FieldInfo, que tienen a su vez
         * la información de los campos especificados en un query.
         */
        @SuppressWarnings("unchecked")
        public Vector analyzeTable() {
                ResultSet rs;
                Vector res;
```



```
                FieldInfo info;
                boolean haveElements = false;
                ResultSetMetaData rsmd;
                Statement stmt;
                int cols = 0;
                res = new Vector(8, 4);
                stmt = db.getStmt();
                try {
                        rs = stmt.executeQuery(query);
                        rsmd = rs.getMetaData();
                        cols = rsmd.getColumnCount();
                        if (isSearchingValues()) {
                                if (rs.next()) {
                                        haveElements = true;
                                }
                        }
                        for (int i = 1; i <= cols; i++) {
                                info = new FieldInfo();
                                info.setName(rsmd.getColumnName(i));
                                info.setType(rsmd.getColumnTypeName(i));
                                info.setTitle(rsmd.getColumnLabel(i));
                                info.setEditable(true);
                                info.setVisible(!(rsmd.isAutoIncrement(i)));
                                info.setIdType(rsmd.getColumnType(i));
                                if (isSearchingValues() & haveElements) {
                                        switch (info.getIdType()) {
                                                case FieldInfo.INTEGER_TYPE:
                                                case FieldInfo.BIGINT_TYPE:
                                                    info.setValue(rs.getInt(
                                                        info.getName()) + "");
                                                    break;
                                                case FieldInfo.DOUBLE_TYPE:
                                                     info.setValue(rs.getDouble(
                                                        info.getName()) + "");
                                                     break;
                                                case FieldInfo.LONGCHAR_TYPE:
                                                case FieldInfo.VARCHAR_TYPE:
                                                     info.setValue(rs.getString(
                                                        info.getName()));
                                                     break;
                                        }
                                }
                                res.add(info);
                        }
                } catch (SQLException e) {
                        System.err.println("Query =" + query);
                        System.err.println("SQLException: " + e.getMessage());
                        e.printStackTrace();
                }
                return res;
        }

        /**
         * Cuenta cuantos campos(del query) se tienen.
         * @return
         */
        public int countFileds() {
                return fields.size();
        }

        /**
         * Borra un campo oculto
         * @param fieldName
         */
        public void delHiddenField(String fieldName) {
                hiddenFields.remove(fieldName);
        }

        /**
```

```
	 * Borra una propiedad
	 * @param propertie
	 */
	public void delPropertie(String propertie) {
		otherProperties.remove(propertie);
	}

	/**
	 * Borra una propiedad del javaScript
	 * @param propertie
	 */
	public void delJavaScriptPropertie(String propertie) {
		javaScriptProperties.remove(propertie);
	}

	/**
	 * Obtiene el servlet a invocar cuando la forma se ejecute
	 * @return
	 */
	public String getAction() {
		return action;
	}

	/**
	 * Regresa la lista de FieldInfo una vez procesados.
	 * @return
	 */
	public Vector getFields() {
		return fields;
	}

	/**
	 * Regresa la lista de variables que se desean pasar como campos ocultos

	 * @return
	 */
	public SortedHashtable getHiddenFields() {
		return hiddenFields;
	}

	/**
	 * Regresa el query a ejecutar.
	 * @return
	 */
	public String getQuery() {
		return query;
	}

	/**
	 * Obtiene el titulo de la pagina
	 * @return
	 */
	public String getTitle() {
		return title;
	}

	/**
	 * Determina si el campo es Numérico (NUMERIC_TYPE)o String(STRING_TYPE)
	 * @param fieldName
	 * @return
	 */
	public int getTypeOf(String fieldName) {
		FieldInfo aux = searchField(fieldName);
		switch (aux.getIdType()) {
			case FieldInfo.TYNY_INT:
			case FieldInfo.BIGINT_TYPE:
			case FieldInfo.COUNTER_TYPE:
			case FieldInfo.DOUBLE_TYPE:
				//case FieldInfo.INTEGER_TYPE:
```



```
					return NUMERIC_TYPE;
					//case FieldInfo.LONGCHAR_TYPE :
			case FieldInfo.TEXT_TYPE:
			case FieldInfo.VARCHAR_TYPE:
					return STRING_TYPE;
		}
		//error
		return -1;
	}

	/**
	 * Regresa la bandera para saber si se estan buscando valores.
	 * @return
	 */
	public boolean isSearchingValues() {
		return searchingValues;
	}

	/**
	 * Busca el FieldInfo a partir de su nombre (del campo)
	 * @param fieldName
	 * @return
	 */
	private FieldInfo searchField(String fieldName) {
		FieldInfo res = null;
		for (int i = 0; i <= fields.size(); i++) {
			res = (FieldInfo) fields.elementAt(i);
			if (res.getName().equalsIgnoreCase(fieldName))
				return res;
		}
		return null;
	}

	/**
	 * Establece la accion a ejecutar (al apretar el submit de la forma
	 * @param action
	 */
	public void setAction(String action) {
		this.action = action;
	}

	/**
	 * Establece cual va a ser la lista de variables (nombre, valor) a ser
	 * transformadas en campos ocultos
	 * @param hiddenFields
	 */
	public void setHiddenFields(SortedHashtable hiddenFields) {
		this.hiddenFields = hiddenFields;
	}

	/**
	 *  Metodo que establece propiamente las propiedades en el pageGenerator
	 *  (a partir del atributo otherProperites)
	 *
	 *  Este metodo supone que el .pg tiene al menos los siguientes strings
	 *  a remplazar: (*Title*) -- para el título de la página (*Action*)
	 *  -- para el nombre del servlet a invocar (al apretar SUBMIT)
	 * @param page .PageGenerator a modificar
	 */
	public void setPropertiesTo(PageGenerator page) {
		Enumeration aux;
		page.setItem("(*Title*)", title);
		page.setItem("(*Action*)", action);
		if (!otherProperties.isEmpty()) {
			aux = otherProperties.keys();
			while (aux.hasMoreElements()) {
				String key, val;
				key = (String) aux.nextElement();
				val = (String) otherProperties.get(key);
```

```
                        page.setItem(key, val);
                    }
                }
        }

        /**
         * Establece el query a ejecutar y lo analiza.
         * @param query
         */
        public void setQuery(String query) {
                this.query = query;
                fields = analyzeTable();
        }

        /**
         * Establece la bandera que indica si se estan buscando valores.
         * @param searchingValues
         */
        public void setSearchingValues(boolean searchingValues) {
                this.searchingValues = searchingValues;
        }

        /**
         * Establece si un campo es selecionable y el query para selecionar los
         * datos de esta selección.
         * @param fieldName
         * @param selectQuery
         */
        public void setSelectionableField(String fieldName,String selectQuery) {
                FieldInfo aux = searchField(fieldName);
                aux.setSelectionable(true);
                aux.setSelectionQuery(selectQuery);
        }

        /**
         * Establece el titulo de la página
         * @param title
         */
        public void setTitle(String title) {
                this.title = title;
        }

        /**
         * Metodo para establecer el Titulo (nombre a publicar) de un campo
         * @param fieldName
         * @param title
         */
        public void setTitleField(String fieldName, String title) {
                FieldInfo aux = searchField(fieldName);
                aux.setTitle(title);
        }

        /**
         * Especifica que un campo va a ser visible
         * @param fieldName
         */
        public void setVisibleField(String fieldName) {
                FieldInfo aux = searchField(fieldName);
                aux.setVisible(true);
        }

        /**
         * Indica que un campo deja de ser selecionable
         * @param fieldName
         */
        public void unSetSelectionableField(String fieldName) {
                FieldInfo aux = searchField(fieldName);
                aux.setSelectionable(false);
                aux.setSelectionQuery(null);
```



```
        }

        /**
         * Especifica que un campo va a ser editable
         * @param fieldName
         */
        public void setEditableField(String fieldName) {
                FieldInfo aux = searchField(fieldName);
                aux.setEditable(true);
        }

        /**
         * Indica que un campo deja de ser editable
         * @param fieldName
         */
        public void unSetEditableField(String fieldName) {
                FieldInfo aux = searchField(fieldName);
                aux.setEditable(false);
        }

        /**
         * Indica que un campo deja de ser visible
         * @param fieldName
         */
        public void unSetVisibleField(String fieldName) {
                FieldInfo aux = searchField(fieldName);
                aux.setVisible(false);
        }
        /**
         * regresa el string que invoca un validador generico en Java Script.
         */
        public String createJavaScriptValidateFunction() {
                StringBuffer res = new StringBuffer();
                //> language=\"JavaScript\"
                res.append("   <script type=\"text/javascript\" "
                        +"src=\"./seguridad/verifyForm.js\"> \n");
                res.append("   </script> \n");
                return res.toString();
        }

        /**
         * Obtiene un Vector donde cada elemento es un arreglo de Strings que
         * contiene el valor de una tupla de un query.
         * @return
         */
        @SuppressWarnings("unchecked")
        public Vector getVectorOfQueryResult() {
                Vector res = new Vector(5, 5);
                ResultSet rs;
                Statement stmt = db.getStmt();
                FieldInfo fieldInfo;
                try {
                        rs = stmt.executeQuery(query);
                        while (rs.next()) { //Para cada tupla
                                String fieldValues[] =new String[fields.size()];
                                //Para cada campo
                                for (int i = 0; i < fields.size(); i++) {
                                        fieldInfo = (FieldInfo) fields.get(i);
                                        switch (fieldInfo.getIdType()) {
                                                case FieldInfo.INTEGER_TYPE:
                                                case FieldInfo.BIGINT_TYPE:
                                                     fieldValues[i] = rs.getInt(
                                                       fieldInfo.getName())+"";
                                                       break;
                                                case FieldInfo.DOUBLE_TYPE:
                                                   fieldValues[i]=rs.getDouble(
                                                         fieldInfo.getName())+"";
                                                         break;
                                                case FieldInfo.LONGCHAR_TYPE:
```

```
                                        case FieldInfo.VARCHAR_TYPE:
                                            fieldValues[i]=rs.getString(
                                                fieldInfo.getName())+"";
                                            break;
                                    } //switch
                                }//for
                                res.add(fieldValues);
                            }//while
                        } catch (SQLException e) {
                            e.printStackTrace();
                        }
                        return res;
                }
}
```

```
package itesm.seguridad;
/*
 * @author    Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import java.util.*;
/**
 * Crea una forma de entrada a partir de una tabla
 */
public class DbWebAlta extends DbWeb {
        /**
         * Metodo de prueba para ver si funciona la clase.
         */
        public static void main(String args[]) {
                DbWebAlta dbwebA = new DbWebAlta();
                dbwebA.addHiddenField("idCriterio", "1");
                dbwebA.setQuery("SELECT * FROM Criterio");
                System.out.println(dbwebA.createDBFormAlta());
        }

        //Constructores
        DbWebAlta() {
                super();
        }
        DbWebAlta(String query) {
                super(query);
        }
        DbWebAlta(String title, String accion) {
                super(title, accion);
        }

        /**
         * Metodo que crea la forma de alta.
         */
        public String createDBFormAlta() {
                StringBuffer res = new StringBuffer();
                Vector fields = getFields();
                res.append( Utils.createHTMLFormTag("form1", action,"verify",
                        javaScriptProperties));
                if (hiddenFields != null) {
                        res.append(Utils.createHTMLHiddenFields(hiddenFields));
                }
                res.append("\t<TABLE BORDER=\"1\">\n");
                for (int i = 0; i < fields.size(); i++) {
                        FieldInfo fieldInfo;
                        fieldInfo = (FieldInfo) fields.get(i);
                        if (fieldInfo.isVisible() && fieldInfo.isEditable()) {
                                res.append("\t\t<TR>\n");
                                res.append("\t\t\t<TD>");
                                res.append(fieldInfo.getTitle());
                                res.append(":</TD>\n");
                                res.append("\t\t\t<TD>");
                                if (fieldInfo.isSelectionable()) {
                                        res.append(Utils.
                                           createHTMLSelectOptionTag(fieldInfo
                                                .getName(), db, fieldInfo.
                                                        getSelectionQuery()));
                                }
                                else {
                                        switch (fieldInfo.getIdType()) {
                                                case FieldInfo.TYNY_INT:
                                                  res.append(Utils.
                                                        createTrueFalseOption(
                                                                fieldInfo.
                                                                getName()));
                                                        break;
                                                case FieldInfo.VARCHAR_TYPE:
                                                        res.append(Utils
                                                          .createHTMLInputTag(
                                                          fieldInfo.getName()));
                                                        break;
                                                case FieldInfo.LONGCHAR_TYPE:
                                                // case FieldInfo.TEXT_TYPE:
                                                        res.append(Utils.
                                                          createHTMLTextAreaTag(
                                                          fieldInfo.getName()));
                                                        break;
                                                //  case FieldInfo.COUNTER_TYPE:
                                                case FieldInfo.BIGINT_TYPE:
                                                case FieldInfo.INTEGER_TYPE:
                                                        res.append(Utils.
                                                        createHTMLInputTag(
                                                        fieldInfo.getName(),
                                                                "0"));
                                                        break;
                                                case FieldInfo.DOUBLE_TYPE:
                                                        res.append(Utils.
                                                        createHTMLInputTag(
                                                        fieldInfo.getName(),
                                                                "0.0"));
                                                        break;
                                        }
                                }
                                res.append("</TD>\n");
                                res.append("\t\t</TR>\n");
                        }
                }
                res.append("\n\t\t<TR ALIGN=\"CENTER\">\n");
                res.append("\t\t\t<TD COLSPAN=\"2\">" +
                        "<INPUT TYPE=\"SUBMIT\" VALUE=\"Aceptar\"> ");
                res.append("<INPUT TYPE=\"RESET\" VALUE=\"Restablecer\">"
                                +" </TD>\n");
                res.append("\t\t</TR>\n");
                res.append("\t</TABLE>\n");
                res.append("</FORM> ");
                return res.toString();
        }
}
```

```
package itesm.seguridad;
/*
 * @author    Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *     Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import java.util.*;
/**
 * Crea una forma de selección para dar de baja elementos a partir de un query
 */
public class DbWebBaja extends DbWeb{
        /**
         * Metodo de prueba para ver si la clase funciona.
         */
        public static void main (String args[]){
                DbWebBaja dbwebB= new DbWebBaja();
                dbwebB.addHiddenField("idCriterio", "1" );
                dbwebB.setQuery("SELECT idCriterio, nombre, "+
                        " descripcion FROM Criterio");
                dbwebB.setTitleField("idCriterio", "");
                System.out.println( dbwebB.createDBFormBaja());
        }
        //Constructores
        DbWebBaja(){
                super();
        }
        DbWebBaja(String query){
                super(query);
        }
        DbWebBaja(String title, String accion){
                super(title, accion);
        }

        /**
         * Crea la forma a dar de baja, con sus respectivas validaciones.
         */
        public String createDBFormBaja(){
                StringBuffer res = new StringBuffer();
                Vector fields = getFields();
                FieldInfo fieldInfo;
                res.append(Utils.createHTMLFormTag("form1", action,"verify",
                        javaScriptProperties));
                if (hiddenFields!=null){
                        res.append(Utils.createHTMLHiddenFields(hiddenFields));
                }
                res.append("\t <TABLE BORDER=\"1\"> \n");
                //Titulo de los campos
                res.append("\t\t<TR>");
                for(int i=0; i< fields.size(); i++){
                        fieldInfo=(FieldInfo) fields.get(i);
                        res.append("<TD>");
                        res.append(fieldInfo.getTitle());
                        res.append("</TD>");
                }
                res.append("</TR>\n");
                //Desplegamos los datos a ser borrados.
                res.append(Utils.createCheckBoxRows(fields, db,  getQuery()));
                res.append("\n\t\t <TR> \n");
                res.append("\t\t\t <TD COLSPAN=\""+fields.size()
                        +"\"><INPUT TYPE=\"SUBMIT\" VALUE=\"Aceptar\"> ");
                res.append("<INPUT TYPE=\"RESET\" VALUE=\"Restablecer\">"
                                +"</TD>\n");
                res.append("\t\t </TR> \n");
                res.append("\t</TABLE> \n");
                res.append("</FORM> ");
                return res.toString();
        }
}
```

```
package itesm.seguridad;
/*
 * @author    Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *     Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import java.util.*;
/**
 * Crea una forma de seleccion de cambio a partir de un query
 */
public class DbWebCambio extends DbWebAlta {
        /**
         * Metodo de prueba de la clase.
         */
        @SuppressWarnings("unchecked")
        public static void main(String args[]) {
                DbWebCambio dbwebC = new DbWebCambio();
                Hashtable ht = new Hashtable();
                ht.put("idCriterio", "1");
                dbwebC.setQuery("SELECT idCriterio, nombre, descripcion "
                                +"FROM Criterio");
                dbwebC.setTitleField("idCriterio", "");
                System.out.println(dbwebC.createDBFormCambio());
        }
        //Constructores
        DbWebCambio() {
                super();
        }
        DbWebCambio(String query) {
                super(query);
        }
        DbWebCambio(String title, String accion) {
                super(title, accion);
        }

        /**
         * Crea la forma del cambio.
         */
        public String createDBFormCambio() {
                StringBuffer res = new StringBuffer();
                Vector fields = getFields();
                res.append(Utils.createHTMLFormTag("form1", action,"verify",
                        javaScriptProperties));
                if (hiddenFields != null) {
                        res.append(Utils.createHTMLHiddenFields(hiddenFields));
                }
                res.append("\t <TABLE BORDER=\"1\">\n");
                res.append(Utils.createRadioBottonRows(fields, db, getQuery()));
                res.append("\n\t\t <TR> \n");
                res.append("\t\t\t <TD COLSPAN=\"" + fields.size()
                        + "\"><INPUT TYPE=\"SUBMIT\" VALUE=\"Aceptar\"> ");
                res.append("<INPUT TYPE=\"RESET\" VALUE=\"Restablecer\">"
                                +"</TD>\n");
                res.append("\t\t </TR> \n");
                res.append("\t</TABLE> \n");
                res.append("</FORM> ");
                return res.toString();
        }
}
```

```
package itesm.seguridad;
/*
 * @author     Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import java.util.*;
/**
 * Crea una forma de ejecución del cambio a partir de una tabla
 */
public class DbWebExecCambio extends DbWeb {
        /**
         * Metodo de prueba de la clase.
         */
        public static void main(String args[]) {
                DbWebExecCambio dbwebec = new DbWebExecCambio();
                dbwebec.addHiddenField("idCriterio", "1");
                dbwebec.setQuery("SELECT * FROM Criterio");
                System.out.println(dbwebec.createDBFormExecCambio());
        }

        //Constructores
        DbWebExecCambio() {
                super();
        }
        DbWebExecCambio(String query) {
                super(query);
        }
        DbWebExecCambio(String title, String accion) {
                super(title, accion);
        }

        /**
         * Crea la forma del cambio de valores, buscando los que se
         * encuentren en la base de datos
         */
        public String createDBFormExecCambio() {
                StringBuffer res = new StringBuffer();
                Vector fields = getFields();
                res.append(Utils.createHTMLFormTag("form1", action,"verify",
                        javaScriptProperties));
                if (hiddenFields != null) {
                        res.append(Utils.createHTMLHiddenFields(hiddenFields));
                }
                res.append("\t<TABLE BORDER=\"1\">\n");
                for (int i = 0; i < fields.size(); i++) {
                        FieldInfo fieldInfo;
                        fieldInfo = (FieldInfo) fields.get(i);
                        if (!isSearchingValues()) {
                                if (fieldInfo.isVisible() &&
                                    fieldInfo.isEditable()) {
                                        res.append("\t\t<TR>\n");
                                        res.append("\t\t\t<TD>");
                                        res.append(fieldInfo.getTitle());
                                        res.append(":</TD>\n");
                                        res.append("\t\t\t<TD>");
                                        if (fieldInfo.isSelectionable()) {
                                                res.append(Utils.
                                                   createHTMLSelectOptionTag(
                                                       fieldInfo.getName(), db,
                                                       fieldInfo.
                                                       getSelectionQuery()));
                                        } else {
                                                switch (fieldInfo.getIdType()) {
                                                   case FieldInfo.TYNY_INT:
                                                       res.append(Utils.
                                                       createTrueFalseOption(
                                                          fieldInfo.getName()));
                                                       break;
                                                   case FieldInfo.INTEGER_TYPE:
                                                   case FieldInfo.BIGINT_TYPE:
                                                       res.append(Utils.
                                                       createHTMLInputTag(
                                                          fieldInfo.getName(),
                                                          "0"));
                                                       break;
                                                    case FieldInfo.DOUBLE_TYPE:
                                                       res.append(Utils.
                                                       createHTMLInputTag(
                                                        fieldInfo.getName(),
                                                        "0.0"));
                                                       break;
                                                   case FieldInfo.LONGCHAR_TYPE:
                                                       res.append(Utils.
                                                       createHTMLTextAreaTag(
                                                         fieldInfo.getName()));
                                                       break;
                                                   case FieldInfo.VARCHAR_TYPE:
                                                       res.append(Utils.
                                                       createHTMLInputTag(
                                                       fieldInfo.getName()));
                                                       break;
                                                }
                                        }
                                }
                        } else { // is searching values
                                if (fieldInfo.isVisible() &&
                                        fieldInfo.isEditable()) {
                                        res.append("\t\t<TR>\n");
                                        res.append("\t\t\t<TD>");
                                        res.append(fieldInfo.getTitle());
                                        res.append(":</TD>\n");
                                        res.append("\t\t\t<TD>");
                                        if (fieldInfo.isSelectionable()) {
                                                res.append(Utils.
                                                   createHTMLSelectOptionTag(
                                                      fieldInfo.getName(), db,
                                                      fieldInfo.
                                                          getSelectionQuery(),
                                                      fieldInfo.getValue()));
                                        } else {
                                                switch (fieldInfo.getIdType()) {
                                                   case FieldInfo.TYNY_INT:
                                                     res.append(Utils.
                                                     createTrueFalseOption(
                                                           fieldInfo.
                                                            getName(),
                                                           fieldInfo.
                                                            getValue()));
                                                           break;
                                                   case FieldInfo.
                                                           INTEGER_TYPE:
                                                   case FieldInfo.
                                                           BIGINT_TYPE:
                                                   case FieldInfo.
                                                           DOUBLE_TYPE:
                                                     res.append(
                                                        Utils.
                                                        createHTMLInputTag(
                                                          fieldInfo
                                                          .getName(),
                                                          fieldInfo.
                                                          getValue()));
                                                           break;
                                                   case FieldInfo.
                                                           LONGCHAR_TYPE:
                                                     res.append(Utils.
```

```
                                                createHTMLTextAreaTag(
                                                  fieldInfo.getName(),
                                                  fieldInfo.
                                                  getValue()));
                                                      break;
                                                case FieldInfo.
                                                      VARCHAR_TYPE:
                                                   res.append(Utils.
                                                    createHTMLInputTag(
                                                    fieldInfo.getName(),
                                                    fieldInfo.
                                                    getValue()));
                                                      break;
                                            }
                                        }
                                        res.append("</TD>\n");
                                        res.append("\t\t </TR> \n");
                                    }else{
                                        res.append (Utils.createHTMLHiddenField(
                                                fieldInfo.getName(),
                                                fieldInfo.getValue()));
                                    }
                                }
                            }
                            res.append("\n\t\t <TR> \n");
                            res.append("\t\t\t <TD COLSPAN=\"2\"><INPUT TYPE=\"SUBMIT\" "+
                                    " VALUE=\"Aceptar\"> ");
                            res.append("<INPUT TYPE=\"RESET\"  VALUE=\"Restablecer\">"
                                    +" </TD> \n");
                            res.append("\t\t </TR> \n");
                            res.append("\t</TABLE> \n");
                            res.append("</FORM> ");
                            return res.toString();
                        }
                    }
```

```
package itesm.seguridad;
/*
 * Created on Apr 25, 2004
 * @author    Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *     Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import itesm.utilsdsc.*;

/**
 * Crea una forma de un menu a partir de un hash table que define los titulos-
 * valores
 */
public class DbWebMenu extends DbWeb {
        SortedHashtable menu;
        /**
         * Constructures
         */
        public DbWebMenu() {
                super();
                menu = new SortedHashtable();
        }

        /**
         * Constructor que espefica el query.
         * @param q
         */
        public DbWebMenu(String q) {
                super(q);
                menu = new SortedHashtable();
        }

        /**
         * Constructor que especifica el titulo y la accion.
         * @param title
         * @param action
         */
        public DbWebMenu(String title, String action) {
                super(title, action);
                menu = new SortedHashtable();
        }

        /**
         * Obtiene el menu.
         * @return
         */
        public SortedHashtable getMenu() {
                return menu;
        }

        /**
         * Establece el menu.
         * @param menu
         */
        public void setMenu(SortedHashtable menu) {
                this.menu = menu;
        }
}
```

```
package itesm.seguridad;
/*
 * Created on May 7, 2004
 * @author     Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */

/**
 * DbWeb cuyo unico proposito es redirecionarse a otro servlet
 * (newServlet)
 */
public class DbWebSubMenu extends DbWeb {
        String newServlet;
        /**
         * Obtiene el servlet a invocar
         * @return
         */
        public String getNewServlet() {
                return newServlet;
        }

        /**
         * Establece el servlet a invocar
         * @param newServlet
         */
        public void setNewServlet(String newServlet) {
                this.newServlet = newServlet;
        }

        /**
         * Constructores que solo hacen referencia a la superclase.
         */
        public DbWebSubMenu() {
                super();
        }

        /**
         * Constructor que establece el query.
         * @param q
         */
        public DbWebSubMenu(String q) {
                super(q);
        }

        /**
         * Constructor que establce el titulo y la action.
         * @param title
         * @param action
         */
        public DbWebSubMenu(String title, String action) {
                super(title, action);
        }
}
```

```
package itesm.seguridad;
/*
 * Created on 26/09/2005
 * @author      Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import itesm.utilsdsc.*;
import java.io.IOException;
import java.io.PrintWriter;
import java.text.DecimalFormat;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet cuyo proposito es desplegar en forma detallada la evaluación de
 * riesgo de un equipo (host) en especifico.
 */
public class EquipoEvaluado extends Esquema {
        private static final long serialVersionUID = -3658314267032706734L;
        //Propiedades del servlet
        DbWebMenu dwm;
        //Equipo a evaluar
        Host host;
        //Para dar formato a los numeros.
        DecimalFormat  df,df2;

        /**
         * Único constgructor que invoca la superclase e inicializa las
         * variables locales(df y df2)
         */
        public EquipoEvaluado() {
                super();
                df= new DecimalFormat ("0.000%");
                df2 = new DecimalFormat ("(0.0%)");
        }

        /**
         * Inicializa parametros de la pagina en tiempo de inicialización.
         * @see itesm.seguridad.Esquema#initPageMenu()
         */

        protected void initPageMenu() {
                dwm = (DbWebMenu) pageProperties[MENU];
        }

        /**
         * int contDbRows(String query)
         *   Cuenta la cantidad de registros de un query
         * @param query - Query a invocar
         * @return - el numero de registros.
         */
        int contDbRows(String query) {
                int res = 0;
                try {
                        res = db.contRows(query);
                } catch (BDException e) {
                        e.printStackTrace();
                }
                return res;
        }

        /**
         * Crea una tabla que contiene toda la información detallada de la
         * evaluación del nivel de riesgo de un equipo.
         * La estructura de la tabla es:
         *      <TABLE BORDER="1">
         <TR>
```



```
         <TD> Criterio </TD>
         <TD> Factor</TD>
         <TD> Factor Capturado</TD>
         <TD> Factor Evaluado</TD>

         <TD> Factor Ponderado</TD>
         <TD> Criterio Evaluado</TD>
         <TD> Criterio Ponderado</TD>
         <TD> Evaluación Total</TD>
         </TR>

         <TR>
         <TD>(POND%) Criterio 1 ....</TD>
         <TD>(POND%) Factor 1...</TD>
         <TD> Factor Capturado 1 </TD>
         <TD> Factor Evaluado 1 </TD>
         <TD> Factor Ponderado 1 </TD>

         <TD>Criterio Evaluado 1</TD>
         <TD>Criterio Ponderado 1</TD>
         <TD>Evaluacion Total</TD>
         </TR>
         </TABLE>
         * @param req
         * @return
         */
        protected String createHostTableData(HttpServletRequest req) {
                StringBuffer res = new StringBuffer();
                String aux;
                double valorEval = 0.0;
                double ponderacionFact = 0;
                //Calculamos los valores
                EvaluacionEquipo evalEquipo = new EvaluacionEquipo(host.ip, db);
                evalEquipo.setEvaluacionEquipo();
                //Primer renglon global, es necesario poner los rowspan tanto
                //de las columnas referentes al criterio como el de la
                //evaluacion total.
                String ponderacion =
                     df2.format(evalEquipo.criterios[0].criterio.ponderacion);
                String pondFactor =
                    df2.format(evalEquipo.criterios[0].factores[0]
                                    .factor.ponderacion);
                res.append("\t<TR>\n");
                //Criterio
                res.append("\t\t<TD ROWSPAN=\""
                        + evalEquipo.criterios[0].factores.length + "\">");
                res.append(ponderacion);
                res.append(evalEquipo.criterios[0].criterio.nombre);
                res.append("</TD>\n");
                //Factor
                res.append("\t\t<TD>");
                res.append(pondFactor);
                res.append(evalEquipo.criterios[0].factores[0].factor.nombre);
                res.append("</TD>\n");
                //Factor capturado
                res.append("\t\t<TD>");
                res.append(evalEquipo.criterios[0].factores[0].valorCapturado);
                res.append("</TD>\n");
                //Factor evaluado
                res.append("\t\t<TD>");
                valorEval = evalEquipo.criterios[0].factores[0].evaluacion;
                aux=df.format(valorEval);
                res.append(aux);
                res.append("</TD>\n");
                //Factor ponderado
                res.append("\t\t<TD>");
                ponderacionFact=evalEquipo.criterios[0].
                        factores[0].factor.ponderacion;
                aux=df.format(valorEval * ponderacionFact);
```

```
res.append(aux);
res.append("</TD>\n");
//Criterio evaluado
res.append("\t\t<TD ROWSPAN=\""
        + evalEquipo.criterios[0].factores.length + "\">");
aux=df.format(evalEquipo.criterios[0].evaluacion);
res.append(aux);
res.append("</TD>\n");
//Criterio ponderado
res.append("\t\t<TD ROWSPAN=\""
        + evalEquipo.criterios[0].factores.length + "\">");
aux=df.format(evalEquipo.criterios[0].evaluacion
        * evalEquipo.criterios[0].criterio.ponderacion);
res.append(aux);
res.append("</TD>\n");
//Evaluacion total del equipo
res.append("\t\t<TD ROWSPAN=\""
        + contDbRows("SELECT * FROM Factor")
        + "\">");
res.append(df.format(evalEquipo.evaluacion));
res.append("</TD>\n");
res.append("\t</TR>\n");
// Para el primer criterio completamos sus factores.
for (int j=1;j<evalEquipo.criterios[0].factores.length; j++) {
        res.append("\t<TR>\n");
        //Factor
        pondFactor = df2.format(
                evalEquipo.criterios[0].
                        factores[j].factor.ponderacion);
        res.append("\t\t<TD>\n");
        res.append(pondFactor);
        res.append(evalEquipo.criterios[0].
                        factores[j].factor.nombre);
        res.append("\t\t</TD>\n");
        //Factor capturado
        res.append("\t\t<TD>\n");
        res.append(evalEquipo.criterios[0].
                        factores[j].valorCapturado);
        res.append("\t\t</TD>\n");
        //Factor evaluado
        res.append("\t\t<TD>");
        valorEval = evalEquipo.criterios[0].
                        factores[j].evaluacion;
        aux = df.format(valorEval);
        res.append(aux);
        res.append("</TD>\n");
        //Factor ponderado
        res.append("\t\t<TD>");
        ponderacionFact =
                evalEquipo.criterios[0].
                factores[j].factor.ponderacion;
        aux = df.format(valorEval * ponderacionFact);
        res.append(aux);
        res.append("</TD>\n");
        res.append("\t</TR>\n");
}// for de los factores del primer criterio
//Para el resto de los criterios
for (int i = 1; i < evalEquipo.criterios.length; i++) {
        res.append("\t<TR>\n");
        //Criterio
        res.append("\t\t<TD ROWSPAN=\""
             +evalEquipo.criterios[i].factores.length + "\">");
        res.append(df2.format(
                evalEquipo.criterios[i].criterio.ponderacion));
        res.append(evalEquipo.criterios[i].criterio.nombre);
        res.append("</TD>\n");
        //Factor
        res.append("\t\t<TD>");
        res.append(df2.format(
```



```
                evalEquipo.criterios[i].factores[0].
                        factor.ponderacion));
        res.append(evalEquipo.criterios[i].factores[0].
                        factor.nombre);
        res.append("</TD>\n");
        //Factor capturado
        res.append("\t\t<TD>");
        res.append(evalEquipo.criterios[i].factores[0].
                        valorCapturado);
        res.append("</TD>\n");
        //Factor evaluado
        res.append("\t\t<TD>");
        valorEval = evalEquipo.criterios[i].factores[0].
                        evaluacion;
        aux= df.format(valorEval);
        res.append(aux);
        res.append("</TD>\n");
        //Factor ponderado
        res.append("\t\t<TD>");
        ponderacionFact =
                evalEquipo.criterios[i].factores[0].
                factor.ponderacion;
        aux = df.format(valorEval * ponderacionFact);
        res.append(aux);
        res.append("</TD>\n");
        //Criterio evaluado
        res.append("\t\t<TD ROWSPAN=\""
                +evalEquipo.criterios[i].factores.length+"\">");
        aux = df.format(evalEquipo.criterios[i].evaluacion);
        res.append(aux);
        res.append("</TD>\n");
        //Criterio ponderado
        res.append("\t\t<TD ROWSPAN=\""
                +evalEquipo.criterios[i].factores.length+"\">");
        aux = df.format(evalEquipo.criterios[i].evaluacion
                * evalEquipo.criterios[i].criterio.ponderacion);
        res.append(aux);
        res.append("</TD>\n");
        res.append("\t</TR>\n");
        //para el resto de los factores a partir del segundo
        //criterio
        for (int j=1;j<evalEquipo.criterios[i].factores.length;
                        j++) {
                res.append("\t<TR>\n");
                //Factor
                res.append("\t\t<TD>\n");
                pondFactor = df2.format(
                        evalEquipo.criterios[i].
                        factores[j].factor.ponderacion);
                res.append(pondFactor);
                res.append(evalEquipo.criterios[i].
                        factores[j].factor.nombre);
                res.append("\t\t</TD>\n");
                //Factor capturado
                res.append("\t\t<TD>\n");
                res.append(evalEquipo.criterios[i].
                        factores[j].valorCapturado);
                res.append("\t\t</TD>\n");
                //Factor evaluado
                res.append("\t\t<TD>");
                valorEval = evalEquipo.criterios[i].
                        factores[j].evaluacion;
                aux= df.format(valorEval);
                res.append(aux);
                res.append("</TD>\n");
                //Factor ponderado
                res.append("\t\t<TD>");
                ponderacionFact =
                        evalEquipo.criterios[i].
```

```
                                factores[j].factor.ponderacion;
                            aux = df.format(valorEval * ponderacionFact);
                            res.append(aux);
                            res.append("</TD>\n");
                            res.append("\t</TR>\n");
                        }//for del resto de los factores a partir del
                         //segundo criterio
                }//for para el segundo criterio en adelante.
                return res.toString();
        }

        /**
         * Modifica las propiedades del menú generico en tiempo de ejecución.
         * @param req
         */
        protected void modifyPropertiesPgMenu(HttpServletRequest req) {
                host = Host.searchHost(req.getParameter("ip"),db);
                dwm.addPropertie("(*Titulo*)", "Nivel de Riesgo de "
                                + host.hostname);
                dwm.addPropertie("(*TablaEquipo*)", createHostTableData(req));
        }

        /**
         * Método que despliga la información de esta página.
         * @see itesm.seguridad.Esquema#pgMenu(
         *      javax.servlet.http.HttpServletRequest,
         *      javax.servlet.http.HttpServletResponse)
         */
        protected void pgMenu(HttpServletRequest req, HttpServletResponse res)
                throws IOException {
                PrintWriter out = res.getWriter();
                String archivo = PATH + "EquipoEvaluado.pg";
                res.setContentType("text/html; charset=iso-8859-1");
                PageGenerator page = initPageGenerator(archivo);
                modifyPropertiesPgMenu(req);
                dwm.setPropertiesTo(page);
                out.println(page.render());
                out.close();
        }
}
```

```
package itesm.seguridad;
/*
 * @date 2004
 * @author    Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *     Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import itesm.utilerias.*;
import itesm.utilsdsc.*;

/**
 * Clase Generica que permite hacer una interface web para
 * la manipulacion de una base de datos haciedo sus tres opeaciones básicas
 *  -ALTA
 *  -BAJA
 *  -CAMBIO
 * Además de los menus correspondientes
 *
 * El objetivo de esta clase es facilitar el desarrollo al generalizar la
 * interface necesaria para manipular una base de datos.
 *
 * En forma general, esta clase esta divida en tres secciones:
 *
 *  ** Inicialización
 *  ** Modificacion de propiedades en ejecución
 *  ** Ejecucion
 *
 * El proposito de hacer esta división es que las clases derivadas
 * no tengan que ejecutar y solo inicialicen sus datos y modifiquen los
 * las propiedades que necesiten para ejecutar las formas correspondientes.
 * Olvidandese de los detalles de la misma ejecución.
 *
 * @version 1.1
 * v1.0  formas validadas con JavaScript.
 * v1.1  Encapsulamiento de la ejecución, ahora se tienen los métodos
 *          modifyProperties* que tienen el proposito de modificar todos los
 *          datos que sea necesario modifcar en tiempo de ejecución. Los metodos
 *          pg* tendran la logica de ejeción en si.
 *
 */
public class Esquema extends HttpServlet {
        private static final long serialVersionUID = 1893096505211457924L;
        //Constantes para el manejo de las posibles páginas a manipular
        protected static final int ALTA = 1;
        protected static final int ALTAEXEC = 5;
        protected static final int BAJA = 2;
        protected static final int BAJAEXEC = 6;
        protected static final int CAMBIO = 3;
        protected static final int CAMBIOEXEC = 7;
        protected static final int CAMBIOEXECUPDATE = 8;
        protected static final int GENERAL = 4;
        protected static final int MENU = 0;
        protected static final int SUBMENU = 9;
        protected static int NUM = 10; //Numero de páginas genericas totales
        static protected String PATH; // Path del archivo de ejecucion (esta cla
se)

        /**
         * Arreglo de los nombres de las "páginas" (.pg)
         */
        protected static String PG[] = { "menu.pg", "alta.pg", "baja.pg",
                "cambio.pg", "mensajeGeneral.pg", "mensajeGeneral.pg",
                "mensajeGeneral.pg", "cambio.pg", "mensajeGeneral.pg",
                "mensajeGeneral.pg" };
```



```
        /**
         * Arreglo de DbWeb (caracteristicas de páginas)
         */
        protected DbWeb pageProperties[] = new DbWeb[NUM];
        protected DB db; //La base de datos.
        protected String table = "Criterio"; //Tabla a modificar
        protected String URL = "/sderi/Esquema"; //Servlet a llamar
        protected String error=""; //Mensaje de error, en caso que exista.

        /**
         * Inicializa este esquema, mandando inicializar las páginas
         * (Menu,Altas,Bajas,Cambios, y las ejecuciones del ABC)
         */
        public Esquema() {
                super();
                try {
                        db = new DB();
                } catch (BDException e) {
                        error +="Error al inicializar la base de datos: <BR>\n"+
                                "UrlJDBC =" + e.getUrlJDBC() +"<BR>\n"+
                                "Mensaje =" + e.sqle.getMessage() +"<BR>\n"+
                                e.getStackTrace();
                }
                initPagesProperties();
        }

        /**
         * Obtiene la dirección del servlet a la que se va a llamar.
         * @return
         */
        public String getURL() {
                return URL;
        }

        /**
         * Proceso que se ejecuta la primera vez que se ejecuta
         * el servlet. Define el path de ejecucion.
         */
        public void init() {
                PATH = Util.stringAsegurarSufijo(
                        Util.stringAsegurarPrefijo(Util.obtenerPaqueteDeLaClase(
                        Esquema.class).replace('.', '/'), "/"), "/");
        }

        /**
         * Inicializa la página de alta.
         */
        protected void initPageAlta() {
                DbWebAlta dwa;
                dwa = (DbWebAlta) pageProperties[ALTA];
                dwa.setQuery("SELECT * FROM " + table);
                dwa.addHiddenField("accion", "execalta");
                dwa.addPropertie("(*Titulo*)", "Alta ");
        }

        /**
         * Inicializa la página de la forma de baja.
         */
        protected void initPageBaja() {
                DbWebBaja dwb = (DbWebBaja) pageProperties[BAJA];
                dwb.addHiddenField("accion", "execBaja");
                dwb.addPropertie("(*Titulo*)", "Baja ");
                dwb.addPropertie("(*Intrucciones1*)",
                        "1.Seleccione los registros a dar de baja.");
                dwb.setQuery("SELECT idCriterio, nombre, descripcion FROM " +
                        table);
        }

        /**
```

```
     * Inicializa la página de la forma de baja.
     */
    protected void initPageCambio() {
            DbWebCambio dwc = (DbWebCambio) pageProperties[CAMBIO];
            dwc.addPropertie("(*Titulo*)", "Cambio");
            dwc.addPropertie("(*Instrucciones1*)",
                    "1.Seleccione el registro a cambiar.");
            dwc.setQuery("SELECT idCriterio, nombre, descripcion FROM "
                    + table);
            dwc.setTitleField("idCriterio", "");
            dwc.setTitleField("nombre", "Nombre");
            dwc.setTitleField("descripcion", "Descripción");
            dwc.addHiddenField("accion", "execCambio");
    }

    /**
     * Inicializa la página de captura del cambio.
     */
    protected void initPageCambioExec() {
            DbWebExecCambio dwce;
            dwce = (DbWebExecCambio) pageProperties[CAMBIOEXEC];
            dwce.setQuery("SELECT * FROM " + table);
            dwce.addHiddenField("accion", "execCambioUpdate");
            dwce.addPropertie("(*Titulo*)", "Cambio ");
            dwce.addPropertie("(*Instrucciones1*)", "");
    }

    /**
     * PageGenerator initPageGenerator(String archivo) Inicializa en forma
     * general un PageGenerator
     *
     * @param archivo archivo a inicilizar
     * @return - página inicializada
     */
    protected PageGenerator initPageGenerator(String archivo) {
            PageGenerator page = new PageGenerator();
            page.setCacheEnabled(false);
            page.setSource(archivo);
            page.clear();
            page.setType(PageGenerator.RESOURCE);
            page.read();
            return page;
    }

    /**
     * Inicializa el menú de este esquema, básicamente indica cuales son las
     * opciones que se puedan ejecutar.
     */
    protected void initPageMenu() {
            DbWebMenu dwm;
            SortedHashtable menu;
            dwm = (DbWebMenu) pageProperties[MENU];
            menu = dwm.getMenu();
            menu.put("Alta", "alta");
            menu.put("Cambio", "cambio");
            menu.put("Baja", "baja");
            menu.put("SubMenu", "subMenu");
            dwm.addPropertie("(*Menu*)",
                            Utils.createOptionButtonMenu("accion",    menu));
            dwm.addPropertie("(*OptMenu*)", "");
            dwm.addPropertie("(*Instrucciones1*)", "");
            dwm.addPropertie("(*Instrucciones2*)", "");
    }

    /**
     * Inicializa las propiedades de las páginas, indicando su titulo
     * y accion
     */
    protected void initPagesProperties() {
```



```
            pageProperties[MENU] = new DbWebMenu("Menu", getURL());
            pageProperties[ALTA] = new DbWebAlta("Alta", getURL());
            pageProperties[BAJA] = new DbWebBaja("Baja", getURL());
            pageProperties[CAMBIO] = new DbWebCambio("Cambio", getURL());
            pageProperties[GENERAL] = new DbWeb("General", getURL());
            pageProperties[ALTAEXEC]=
                    new DbWeb("Ejecución de Alta",getURL());
            pageProperties[BAJAEXEC] =
                    new DbWeb("Ejecución de Baja", getURL());
            pageProperties[CAMBIOEXEC] =
                    new DbWebExecCambio("Ejecución de Cambio",getURL());
            pageProperties[CAMBIOEXECUPDATE] =
                    new DbWeb("Ejecucion de Cambio",getURL());
            pageProperties[SUBMENU] = new DbWebSubMenu("SubMenu", getURL());
            initPageMenu();
            initPageAlta();
            initPageAltaExec();
            initPageBaja();
            initPageBajaExec();
            initPageCambio();
            initPageCambioExec();
            initPageCambioExecUpdate();
            initPageSubMenu();
    }

    /**
     * Inidica el servlet a invocar.
     */
    protected void initPageSubMenu() {
            DbWebSubMenu dw = (DbWebSubMenu) pageProperties[SUBMENU];
            dw.setNewServlet("EsquemaFactor");
    }

    /**
     * Modifica las propiedades de la forma de alta en tiempo de ejecución.
     * @param req
     */
    protected void modifyPropertiesPgAlta(HttpServletRequest req){
            DbWebAlta dwa = (DbWebAlta) pageProperties[ALTA];
            dwa.addJavaScriptPropertie("this.descripcion.optional","true");
    }
    /**
     * Forma de captura de Alta
     * @param req
     * @param res
     * @throws IOException
     */
    protected void pgAlta(HttpServletRequest req, HttpServletResponse res)
            throws IOException {
            res.setContentType("text/html");
            DbWebAlta dwa = (DbWebAlta) pageProperties[ALTA];
            String archivo = PATH + PG[ALTA];
            PageGenerator page = initPageGenerator(archivo);
            modifyPropertiesPgAlta(req);
            //Modificaciones genericas en todas las formas de Alta.
            dwa.addPropertie("(*FormaAlta*)", dwa.createDBFormAlta());
            dwa.addPropertie("(*javaScript*)",
                    dwa.createJavaScriptValidateFunction());
            dwa.setPropertiesTo(page);
            PrintWriter out = res.getWriter();
            out.println(page.render());
            out.close();
    }

    /**
     * Modifica las propiedades de la forma de baja en tiempo de ejecución.
     * @param req
     */
    protected void modifyPropertiesPgBaja(HttpServletRequest req){
```

```
        //DbWebBaja dwb = (DbWebBaja) pageProperties[BAJA];
    }

    /**
     * Forma de captura de baja.
     * @param req
     * @param res
     * @throws IOException
     */
    protected void pgBaja(HttpServletRequest req, HttpServletResponse res)
            throws IOException {
        res.setContentType("text/html");
        DbWebBaja dwb = (DbWebBaja) pageProperties[BAJA];
        String archivo = PATH + PG[BAJA];
        PageGenerator page = initPageGenerator(archivo);
        modifyPropertiesPgBaja(req);
        dwb.addPropertie("(*FormaBaja*)", dwb.createDBFormBaja());
        dwb.addPropertie("(*javaScript*)",
                dwb.createJavaScriptValidateFunction());
        dwb.setPropertiesTo(page);
        PrintWriter out = res.getWriter();
        out.println(page.render());
        out.close();
    }

    /**
     * Cambia las propiedades de la forma de cambio en tiempo de ejecución.
     * @param req
     */
    protected void modifyPropertiesPgCambio(HttpServletRequest req){
        //DbWebCambio dwc = (DbWebCambio) pageProperties[CAMBIO];
    }

    /**
     * Forma de selección del cambio.
     * @param req
     * @param res
     * @throws IOException
     */
    protected void pgCambio(HttpServletRequest req, HttpServletResponse res)
            throws IOException {
        res.setContentType("text/html");
        DbWebCambio dwc = (DbWebCambio) pageProperties[CAMBIO];
        String archivo = PATH + PG[CAMBIO];
        PageGenerator page = initPageGenerator(archivo);
        modifyPropertiesPgCambio(req);
        dwc.addPropertie("(*FormaCambio*)", dwc.createDBFormCambio());
        dwc.addPropertie("(*javaScript*)",
                dwc.createJavaScriptValidateFunction());
        dwc.setPropertiesTo(page);
        PrintWriter out = res.getWriter();
        out.println(page.render());
        out.close();
    }

    /**
     * Modifica las propiedades de la forma de ejecución del
     * cambio en tiempo de ejecución.
     * @param req
     */
    protected void modifyPropertiesPgCambioExec(HttpServletRequest req){
        SortedHashtable data;
        String key, val;
        DbWebExecCambio dwce =
                (DbWebExecCambio) pageProperties[CAMBIOEXEC];
        data = WebUtils.getFieldsAndValues(req);
        data.remove("accion");
        key = Utils.hashTableGetKeyAt(data, 0);
        val = Utils.hashTableGetValueAt(data, 0);
```



```
        dwce.addHiddenField(key,val);
        dwce.setSearchingValues(true);
        dwce.setQuery("SELECT * FROM " + table + " WHERE "
                + key + "=" + val);
    }

    /**
     * Forma de captura de cambio.
     * @param req
     * @param res
     * @throws IOException
     */
    protected void pgCambioExec(HttpServletRequest req,
            HttpServletResponse res) throws IOException {
        res.setContentType("text/html");
        DbWebExecCambio dwce =
                (DbWebExecCambio) pageProperties[CAMBIOEXEC];
        String archivo = PATH + PG[CAMBIOEXEC];
        PageGenerator page = initPageGenerator(archivo);
        modifyPropertiesPgCambioExec(req);
        dwce.addPropertie("(*FormaCambio*)",
                dwce.createDBFormExecCambio());
        dwce.addPropertie("(*javaScript*)",
                dwce.createJavaScriptValidateFunction());
        dwce.setPropertiesTo(page);
        PrintWriter out = res.getWriter();
        out.println(page.render());
        out.close();
    }

    /**
     * Inicializa la página de ejecución de alta.
     */
    protected void initPageAltaExec() {
        DbWeb dw = pageProperties[ALTAEXEC];
        dw.addPropertie("(*Titulo*)", "Ejecución de Alta ");
    }

    /**
     * Modifica las propiedades de la forma de ejecución de alta
     * en tiempo de ejecución.
     *
     * @param req
     */
    protected void modifyPropertiesPgExecAlta(HttpServletRequest req)
            throws BDException{
        SortedHashtable data;
        data = WebUtils.getFieldsAndValues(req);
        data.remove("accion");
        data = WebUtils.formatDataforDB("SELECT * FROM " + table, data);
        db.add(table, data);
    }
    /**
     * Ejecuta el Alta
     *
     * @param req
     * @param res
     * @throws IOException
     */
    protected void pgExecAlta(HttpServletRequest req,
            HttpServletResponse res) throws IOException {
        res.setContentType("text/html; charset=iso-8859-1");
        DbWeb dw = pageProperties[ALTAEXEC];
        String archivo = PATH + PG[ALTAEXEC];
        try{
            modifyPropertiesPgExecAlta(req);
        } catch (BDException e) {
            error +="Error al ejecutar el alta: <BR>\n"+
                    "Query =" + e.getQuery() +"<BR>\n"+
```

```
                        "msgError=" + e.sqle.getMessage() +"<BR>\n"+
                        e.getStackTrace();
                }
                if (error.equalsIgnoreCase("")){
                        dw.addPropertie("(*Mensaje*)", "Alta exitosa.");
                }else{
                        dw.addPropertie("(*Mensaje*)", error);
                        error="";
                }
                PageGenerator page = initPageGenerator(archivo);
                dw.setPropertiesTo(page);
                PrintWriter out = res.getWriter();
                out.println(page.render());
                out.close();
        }

        /**
         * Inicializa la página de ejecución de baja.
         */
        protected void initPageBajaExec() {
                DbWeb dw = pageProperties[BAJAEXEC];
                dw.addPropertie("(*Titulo*)", "Ejecución de Baja ");
        }

        /**
         * Modifica las propiedades de la forma de ejecucion de baja
         * en tiempo de ejecución.
         *
         * @param req
         */
        protected void modifyPropertiesPgExecBaja(HttpServletRequest req)
                throws BDException{
                SortedHashtable data;
                data = WebUtils.getFieldsAndValues(req);
                data.remove("accion");
                data = WebUtils.formatDataforDB("SELECT * FROM " + table, data);
                db.del(table, data);
        }
        /**
         * Ejecuta la Baja
         *
         * @param req
         * @param res
         * @throws IOException
         */
        protected void pgExecBaja(HttpServletRequest req,
                        HttpServletResponse res) throws IOException {
                res.setContentType("text/html; charset=iso-8859-1");
                DbWeb dw = pageProperties[BAJAEXEC];
                String archivo = PATH + PG[BAJAEXEC];
                try{
                        modifyPropertiesPgExecBaja(req);
                } catch (BDException e) {
                        error +="Error al ejecutar la baja: <BR>\n"+
                                "Query=" + e.getQuery() +"<BR>\n"+
                                "msgError=" + e.sqle.getMessage() +"<BR>\n"+
                                e.getStackTrace();
                }
                if (error.equalsIgnoreCase("")){
                        dw.addPropertie("(*Mensaje*)", "Baja exitosa.");
                }else{
                        dw.addPropertie("(*Mensaje*)", error);
                        error="";
                }
                PageGenerator page = initPageGenerator(archivo);
                dw.setPropertiesTo(page);
                PrintWriter out = res.getWriter();
                out.println(page.render());
                out.close();
```



```
        }

        /**
         * Inicializa la página de ejecución de alta.
         */
        protected void initPageCambioExecUpdate() {
                DbWeb dw = pageProperties[CAMBIOEXECUPDATE];
                dw.addPropertie("(*Titulo*)", "Ejecución de Cambio ");
        }

        /**
         * Modifica las propiedades de la forma de ejecución de actualización
         * del cambio tiempo de ejecución.
         * @param req
         */
        protected void modifyPropertiesPgExecCambioUpdate(
                        HttpServletRequest req) throws BDException{
                String val;
                SortedHashtable data;
                val = req.getParameter("idCriterio");
                data = WebUtils.getFieldsAndValues(req);
                data.remove("accion");
                data.remove("idCriterio");
                data = WebUtils.formatDataforDB("SELECT * FROM " + table, data);
                db.change(table, data, "idCriterio=" + val);
        }

        /**
         * Ejecuta el cambio.
         * @param req
         * @param res
         * @throws IOException
         */
        protected void pgExecCambioUpdate(HttpServletRequest req,
                HttpServletResponse res) throws IOException {
                res.setContentType("text/html; charset=iso-8859-1");
                DbWeb dw = pageProperties[CAMBIOEXECUPDATE];
                String archivo = PATH + PG[CAMBIOEXECUPDATE];
                try{
                        modifyPropertiesPgExecCambioUpdate(req);
                } catch (BDException e) {
                        error +="Error al ejecutar el cambio: <BR>\n"+
                                "Query=" + e.getQuery() +"<BR>\n"+
                                "msgError=" + e.sqle.getMessage() +"<BR>\n"+
                                e.getStackTrace();
                }
                if (error.equalsIgnoreCase("")){
                        dw.addPropertie("(*Mensaje*)", "Cambio exitoso.");
                }else{
                        dw.addPropertie("(*Mensaje*)", error);
                        error="";
                }
                PageGenerator page = initPageGenerator(archivo);
                dw.setPropertiesTo(page);
                PrintWriter out = res.getWriter();
                out.println(page.render());
                out.close();
        }

        /**
         * Modifica las propiedades del menú generico en tiempo de ejecución.
         * @param req
         */
        protected void modifyPropertiesPgMenu(HttpServletRequest req){
                //DbWebMenu dwm = (DbWebMenu) pageProperties[MENU];
        }

        /**
         * Menu Generico : Alta Baja Cambio  subMenu
```

```
     * @param req
     * @param res
     * @throws IOException
     */
    protected void pgMenu(HttpServletRequest req, HttpServletResponse res)
            throws IOException {
            PrintWriter out = res.getWriter();
            String archivo = PATH + PG[MENU];
            res.setContentType("text/html; charset=iso-8859-1");
            PageGenerator page = initPageGenerator(archivo);
            DbWebMenu dwm= (DbWebMenu) pageProperties[MENU];
            modifyPropertiesPgMenu(req);
            dwm.addPropertie("(*InitForm*)",
                    Utils.createHTMLFormTag("form1", dwm.action,"verify",
                    dwm.javaScriptProperties));
            dwm.addPropertie("(*javaScript*)",
                    dwm.createJavaScriptValidateFunction()  );
            dwm.setPropertiesTo(page);
            out.println(page.render());
            out.close();
            out.println(page.render());
    }

    /**
     * Redirecciona el servlet a invocar con sus repectivos parametros.
     * @param req
     * @param res
     */
    protected void pgSubMenu(HttpServletRequest req,
                    HttpServletResponse res) throws IOException {
            DbWebSubMenu dw = (DbWebSubMenu) pageProperties[SUBMENU];
            res.setContentType("text/html; charset=iso-8859-1");
            HttpURL url = new HttpURL(req);
            url.setFile("/" + dw.getNewServlet());
            res.sendRedirect(url + "");
    }

    /**
     *  Metodo generico que basicamente determina que "accion" debe tomar.
     *  El "accion" es una variable que debe contener el QueryString
     *  del servlet.
     */
    public void service(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {
            String action = req.getParameter("accion");
            req.removeAttribute("accion");
            try {
                    // Primer página del servlet
                    if (action == null || action.equals("")) {
                            pgMenu(req, res);
                            return;
                    }
                    if (action.equalsIgnoreCase("alta")) {
                            pgAlta(req, res);
                            return;
                    }
                    if (action.equalsIgnoreCase("execAlta")) {
                            pgExecAlta(req, res);
                            return;
                    }
                    if (action.equalsIgnoreCase("baja")) {
                            pgBaja(req, res);
                            return;
                    }
                    if (action.equalsIgnoreCase("execBaja")) {
                            pgExecBaja(req, res);
                            return;
                    }
                    if (action.equalsIgnoreCase("cambio")) {
```



```
                            pgCambio(req, res);
                            return;
                    }
                    if (action.equalsIgnoreCase("execCambio")) {
                            pgCambioExec(req, res);
                            return;
                    }
                    if (action.equalsIgnoreCase("execCambioUpdate")) {
                            pgExecCambioUpdate(req, res);
                            return;
                    }
                    if (action.equalsIgnoreCase("subMenu")) {
                            pgSubMenu(req, res);
                            return;
                    }
            } catch (IOException e) {
                    e.printStackTrace();
            }
    }

    /**
     * Establece la dirección del servlet a invocar
     * @param url
     */
    public void setURL(String url) {
            this.URL = url;
    }

    /**
     * Cambia el url a todas las páginas.
     * @param url
     */
    public void setURLtoAllPages(String url) {
            for (int i = 0; i < pageProperties.length; i++) {
                    pageProperties[i].setAction(url);
            }
    }
}
```

```
package itesm.seguridad;
/*
 * @author      Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import javax.servlet.http.HttpServletRequest;
import itesm.utilsdsc.*;
/**
* Esta clase es la interfase para base de datos de los equipos a evaluar.
*/
public class EsquemaConexionBD extends Esquema {
        private static final long serialVersionUID = -45236359115099926l5L;
        /**
         * Especifca el servlet a invocar.
         */
        public EsquemaConexionBD() {
                super();
                setURLtoAllPages("/sderi/EsquemaConexionBD");
                table = "ConexionBD";
        }

        /**
         * Especifica el menu y el titulo del mismo.
         */
        protected void initPageMenu() {
                DbWebMenu ppm;
                ppm = (DbWebMenu) pageProperties[MENU];
                SortedHashtable menu = ppm.getMenu();
                menu.put("Alta", "alta");
                menu.put("Baja", "baja");
                menu.put("Cambio", "cambio");
                ppm.setTitle("Administración de conexiones a Bases de Datos");
                ppm.addPropertie("(*Menu*)",
                        Utils.createOptionButtonMenuRows("accion",menu));
                ppm.addPropertie("(*Instrucciones1*)",
                        "1. Seleccione la operación a realizar:");
                ppm.addPropertie("(*OptMenu*)", "");
                ppm.addPropertie("(*Instrucciones2*)", "");
        }

        /**
         *   Inicializa la página de alta.
         */
        protected void initPageAlta() {
                DbWebAlta ppa;
                table = "ConexionBD";
                ppa = (DbWebAlta) pageProperties[ALTA];
                ppa.setQuery("SELECT nombre, descripcion ,usuario, password," +
                        " javaDriver, servidor, puerto, bd, url FROM " + table);
                ppa.addHiddenField("accion", "execalta");
                ppa.addPropertie("(*Titulo*)",
                        "Alta de conexión a base de datos");
                ppa.setTitleField("nombre", "Nombre");
                ppa.setTitleField("descripcion", "Descripción");
                ppa.setTitleField("usuario", "Usuario");
                ppa.setTitleField("password", "Password");
                ppa.setTitleField("javaDriver", "Java Driver");
                ppa.setTitleField("servidor", "Servidor");
                ppa.setTitleField("puerto", "Puerto");
                ppa.setTitleField("bd", "Base de datos");
                ppa.setTitleField("url", "Prefijo del URL");
        }

        /**
         * Inicializa la página de la forma de baja
         */
        protected void initPageBaja() {
```



```
                DbWebBaja dwb = (DbWebBaja) pageProperties[BAJA];
                dwb.addPropertie("(*Titulo*)",
                        "Baja de conexiones a bases de datos");
                dwb.addPropertie("(*Instrucciones1*)",
                        "1. Seleccione las conexiones de bases de datos a "
                        +"dar de baja.");
        }

        /**
         * Inicializa la página de ejecución de baja, especifica el mensaje
         */
        protected void initPageBajaExec() {
                DbWeb dw = pageProperties[BAJAEXEC];
                dw.addPropertie("(*Titulo*)",
                        "Ejecución de baja de conexiones a bases de datos ");
                dw.addPropertie("(*Mensaje*)", "Baja exitosa.");
        }


        /**
         * Modifica las propiedades de la forma de baja en tiempo de
         * ejecución.
         * @param req
         */
        protected void modifyPropertiesPgBaja(HttpServletRequest req){
                DbWebBaja dwb = (DbWebBaja) pageProperties[BAJA];
                String query = "SELECT idConexionBD, nombre, descripcion "
                        +"FROM ConexionBD";
                dwb.setQuery(query);
                dwb.setTitleField("idConexionBD", "");
                dwb.setTitleField("nombre", "Nombre");
                dwb.setTitleField("descripcion", "Descripción");
                dwb.addHiddenField("accion", "execBaja");
        }

        /**
         * Inicializa la página de la forma de cambio.
         */
        protected void initPageCambio() {
                DbWebCambio dwc = (DbWebCambio) pageProperties[CAMBIO];
                dwc.addPropertie("(*Titulo*)",
                                "Cambio de conexion a bases de datos");
                dwc.addPropertie("(*Instrucciones1*)",
                        "1.Selecciónе la conexion a cambiar.");
        }

        /**
         * Cambia las propiedades de la forma de cambio en tiempo de ejecución.
         * @param req
         */
        protected void modifyPropertiesPgCambio(HttpServletRequest req){
                DbWebCambio dwc = (DbWebCambio) pageProperties[CAMBIO];
                String query = "SELECT idConexionBD, nombre, descripcion "
                        +"FROM ConexionBD ";
                dwc.setQuery(query);
                dwc.setTitleField("idConexionBD", "");
                dwc.setTitleField("nombre", "Nombre");
                dwc.setTitleField("descripcion", "Descripción");
                dwc.addHiddenField("accion", "execCambio");
        }

        /**
         * Inicializa la página de captura del cambio
         */
        protected void initPageCambioExec() {
                DbWebExecCambio dwce =(DbWebExecCambio) pageProperties[CAMBIOEXE
C];
                dwce.setQuery("SELECT * FROM " + table);
                dwce.addHiddenField("accion", "execCambioUpdate");
```

```
                dwce.addPropertie("(*Titulo*)", "Cambio de Equipo");
                dwce.addPropertie("(*Instrucciones!*)", "");
        }

        /**
         * Modifica las propiedades de la forma de ejecución del cambio en
         * tiempo de ejecución.
         * @param req
         */
        protected void modifyPropertiesPgCambioExec(HttpServletRequest req){
                SortedHashtable data;
                String key, val;
                DbWebExecCambio dwce =
                        (DbWebExecCambio) pageProperties[CAMBIOEXEC];
                data = WebUtils.getFieldsAndValues(req);
                data.remove("accion");
                key = Utils.hashTableGetKeyAt(data, 0);
                val = Utils.hashTableGetValueAt(data, 0);
                dwce.setSearchingValues(true);
                dwce.addHiddenField("idConexionBD",
                                req.getParameter("idConexionBD"));
                dwce.setQuery("SELECT * FROM ConexionBD WHERE "
                                + key + "=" + val+"");
                dwce.setTitleField("nombre", "Nombre");
                dwce.setTitleField("descripcion", "Descripción");
                dwce.setTitleField("usuario", "Usuario");
                dwce.setTitleField("password", "Password");
                dwce.setTitleField("javaDriver", "Java Driver");
                dwce.setTitleField("servidor", "Servidor");
                dwce.setTitleField("puerto", "Puerto");
                dwce.setTitleField("bd", "Base de datos");
                dwce.setTitleField("url", "Prefijo del URL");
        }

        /**
         * Inicializa la página de ejecución de alta.
         */
        protected void initPageCambioExecUpdate() {
                DbWeb dw = pageProperties[CAMBIOEXECUPDATE];
                dw.addPropertie("(*Titulo*)",
                        "Ejecución de Cambio de conexión a base de datos.");
                dw.addPropertie("(*Mensaje*)", "Cambio exitoso.");
        }

        /**
         * Modifica las propiedades de la forma de ejecución de actualización
         * del cambio tiempo de ejecución.
         * @param req
         */
        protected void modifyPropertiesPgExecCambioUpdate(
                        HttpServletRequest req) throws BDException{
                String val;
                SortedHashtable data;
                val = req.getParameter("idConexionBD");
                data = WebUtils.getFieldsAndValues(req);
                data.remove("accion");
                data.remove("idConexionBD");
                data = WebUtils.formatDataforDB("SELECT * FROM " + table, data);
                db.change(table, data, "idConexionBD=" + val+ "");
        }
}
```

```
package itesm.seguridad;
/*
 * @author      Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import itesm.utilsdsc.*;
/**
 * Esta clase es la interfaz para definir el conjunto de criterios para definir
 * el nivel de seguridad (como la suma ponderada de los mismos)
 *
 * Nota: los datos de esta clase se tomaron como base para la creación de
 * la superclase.
 */
public class EsquemaCriterio extends Esquema {
        private static final long serialVersionUID = -8362701699493355400L;
        /**
         * Constructor que inicializa la clase (Define cual va a ser el
         * path para  invocarla)
         */
        public EsquemaCriterio() {
                super();
                setURLtoAllPages("sderi/EsquemaCriterio");
        }

        /**
         * Inicializa la pagina de alta, especifica el query que se usa para
         * generar la forma de entrada.
         */
        protected void initPageAlta() {
                DbWebAlta dwa;
                dwa = (DbWebAlta) pageProperties[ALTA];
                dwa.setQuery("SELECT * FROM " + table);
                dwa.addHiddenField("accion", "execalta");
                dwa.addPropertie("(*Titulo*)", "Alta de Criterio");
                dwa.setTitleField("nombre", "Nombre");
                dwa.setTitleField("descripcion","Descripción");
                dwa.setTitleField("ponderacion","Ponderación");
        }

        /**
         * Inicializa la pagina de la forma de baja
         */
        protected void initPageBaja() {
                DbWebBaja dwb = (DbWebBaja) pageProperties[BAJA];
                dwb.setQuery("SELECT idCriterio, nombre, descripcion, "
                                +"ponderacion FROM "+ table);
                dwb.addPropertie("(*Titulo*)", "Baja de Criterio");
                dwb.addPropertie("(*Instrucciones1*)",
                        "1.Seleccione los criterios a dar de baja.");
                dwb.addHiddenField("accion", "execBaja");
                dwb.setTitleField("idCriterio", "");
                dwb.setTitleField("nombre", "Nombre");
                dwb.setTitleField("descripcion", "Descripción");
                dwb.setTitleField("ponderacion", "Ponderación");
        }

        /**
         * Inicializa la pagina de la forma de cambio
         */
        protected void initPageCambio() {
                DbWebCambio dwc = (DbWebCambio) pageProperties[CAMBIO];
                dwc.addPropertie("(*Titulo*)", "Cambio de Criterio");
                dwc.addPropertie("(*Instrucciones1*)",
                        "1.Seleccione el registro a cambiar.");
                dwc.setQuery("SELECT idCriterio, nombre, descripcion, "
                                +"ponderacion FROM "+ table);
                dwc.setTitleField("idCriterio", "");
```



```
                dwc.setTitleField("nombre", "Nombre");
                dwc.setTitleField("descripcion", "Descripción");
                dwc.setTitleField("ponderacion", "Ponderación");
                dwc.addHiddenField("accion", "execCambio");
        }

        /**
         * Especifica el menu y el titulo del mismo.
         */
        protected void initPageMenu() {
                DbWebMenu dwm;
                dwm = (DbWebMenu) pageProperties[MENU];
                SortedHashtable menu = dwm.getMenu();
                menu.put("Alta", "alta");
                menu.put("Baja", "baja");
                menu.put("Cambio", "cambio");
                menu.put("Administración de Factores", "subMenu");
                dwm.setTitle("Administración de Criterios");
                dwm.addPropertie("(*Menu*)",
                        Utils.createOptionButtonMenuRows("accion",menu));
                dwm.addPropertie("(*OptMenu*)", "");
                dwm.addPropertie("(*Instrucciones1*)", "");
                dwm.addPropertie("(*Instrucciones2*)",
                                "1. Seleccione la operación a realizar: ");
        }
}
```

```
package itesm.seguridad;
/*
 * @author      Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Statement;
import java.sql.ResultSet;
import java.util.Vector;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import itesm.utilsdsc.*;
/**
 * public class EsquemaEval extends Esquema
 *
 * Esta clase es la interfaz para administrar las evaluaciones de riesgo de los
 * equipos (hosts)
 */
public class EsquemaEval extends Esquema {
        private static final long serialVersionUID = 712094127581868684 4L;
        /**
         * Consturctor que básicamente espefica el servlet a invocar.
         */
        public EsquemaEval() {
                super();
                setURLtoAllPages("/sderi/EsquemaEval");
        }

        /**
         * Especifica el menu y el titulo del mismo.
         */
        protected void initPageMenu() {
                String query = "SELECT ip,nombre,ip ip2, descripcion FROM host";
                DbWebMenu ppm = (DbWebMenu) pageProperties[MENU];
                SortedHashtable menu = ppm.getMenu();
                menu.put("Alta", "alta");
                menu.put("Baja", "execBaja");
                menu.put("Cambio", "cambio");
                menu.put("Administración de Criterios", "subMenu");
                ppm.setTitle("Administración de evaluaciones");
                ppm.addPropertie("(*Menu*)",
                        Utils.createOptionButtonMenuRows("accion",menu));
                ppm.setQuery(query);
                ppm.setTitleField("ip", "");
                ppm.setTitleField("nombre", "Equipo");
                ppm.setTitleField("ip2", "Dirección IP");
                ppm.setTitleField("descripcion", "Descripción");
                ppm.addPropertie("(*OptMenu*)", Utils.createRadioBottonRows(ppm
                        .getFields(), db, query));
                ppm.addPropertie("(*Instrucciones1*)",
                        "1. Seleccione el equipo donde hará la operación:");
                ppm.addPropertie("(*Instrucciones2*)",
                        "2. Seleccione la operación a realizar:");
        }

        /**
         * Especifica el servlet (EsquemaCriterio) a invocar si se escoje la
         * opción del submenu
         */
        protected void initPageSubMenu() {
                DbWebSubMenu dw = (DbWebSubMenu) pageProperties[SUBMENU];
                dw.setNewServlet("EsquemaCriterio");
        }

        ////////////////////////////////////////////////////////////////////////
        // ALTA
```



```
        ////////////////////////////////////////////////////////////////////////

        /**
         * Inicializa la página de alta de evaluaciones.
         */
        protected void initPageAlta() {
                DbWebAlta ppa = (DbWebAlta) pageProperties[ALTA];
                ppa.action = "execalta";
                ppa.addHiddenField("accion", "execalta");
                ppa.addPropertie("(*Titulo*)", "Alta de evaluación");
        }

        /**
         * Forma de captura de Alta
         * @param req
         * @param res
         * @throws IOException
         * TODO: Revisar este metodo, en especial la forma de acceder
         * a los datos de  los criterios y factores.
         */
        protected void pgAlta(HttpServletRequest req, HttpServletResponse res)
                throws IOException {
                Factor factor;
                res.setContentType("text/html");
                String ip = req.getParameter("ip");
                DbWebAlta dwa = (DbWebAlta) pageProperties[ALTA];
                String archivo = PATH + "evaluacion.pg";
                PageGenerator page = initPageGenerator(archivo);
                String host = "";
                String queryCriterios = "SELECT * FROM Criterio";
                DbWeb dwCriterios = new DbWeb();
                String queryFactores = "SELECT * FROM Factor";
                DbWeb dwFactores = new DbWeb();
                String queryValores = "SELECT * FROM Valor";
                Vector qrCriterios, qrFactores;
                dwa.addHiddenField("ip", ip);
                try {
                        host = db.getStringValue("host", "nombre", "ip='"
                                        + ip + "'");
                } catch (BDException e) {
                        e.printStackTrace();
                }
                dwa.addPropertie("(*Equipo*)", host);
                dwa.addPropertie("(*DatosForma*)",
                        Utils.createHTMLFormTag("form1", dwa.action,
                                "verify", dwa.javaScriptProperties));
                dwa.addPropertie("(*javaScript*)", dwa
                        .createJavaScriptValidateFunction());
                if (dwa.hiddenFields != null) {
                        dwa.addPropertie("(*hidenFields*)", Utils
                                .createHTMLHiddenFields(dwa.hiddenFields));
                } else {
                        dwa.addPropertie("(*hidenFields*)", "");
                }
                PageGenerator pCriterio = page.getPG("(*PG_Criterio*)");
                pCriterio.clear();
                dwCriterios.setQuery(queryCriterios);
                qrCriterios = dwCriterios.getVectorOfQueryResult();
                //Para cada Criterio
                for (int i = 0; i < qrCriterios.size(); i++) {
                        //Nombre del Criterio
                        pCriterio.setItem("(*Criterio*)",
                                ((String[]) qrCriterios.get(i))[1]);
                        PageGenerator pFactor = page.getPG("(*PG_Factor*)");
                        pFactor.clear();
                        //idCriterio
                        dwFactores.setQuery(queryFactores +" WHERE idCriterio ="
                                + ((String[]) qrCriterios.get(i))[0]);
                        qrFactores = dwFactores.getVectorOfQueryResult();
```

```
                    //Para cada Factor
                    for (int j = 0; j < qrFactores.size(); j++) {
                            //Nombre del Factor)
                            pFactor.setItem("(*Factor*)",
                                    ((String[]) qrFactores.get(j))[2]);
                            PageGenerator pValor=page.getPG("(*PG_Valor*)");
                            pValor.clear();
                            String idFactor =((String[])
                                    (qrFactores.get(j)))[0];
                            factor = Factor.searchFactor(idFactor, db);
                            switch (factor.tipoValor) {
                                    case TipoValor.PROP_MAX:
                                    case TipoValor.NUMERICO_DIRECTO:
                                        pValor.setItem("(*Valor*)",
                                            Utils.createHTMLInputTag(
                                                factor.idFactor + "",
                                                10));
                                    case TipoValor.FUNCION_NUMERICA:
                                        pValor.setItem("(*Valor*)",
                                            Utils.createHTMLInputTag(
                                                factor.idFactor+"",10));
                                        break;
                                    case TipoValor.CONTADOR_NUMERICO:
                                        pValor.setItem("(*Valor*)",Utils
                                            .createCheckBoxRowsForContadorNu
merico(
                                                factor.idFactor + "", db,
                                                queryValores
                                                + " WHERE idFactor="
                                                + idFactor));
                                        break;
                                    case TipoValor.NUMERICO_SELECCIONABLE:
                                        pValor.setItem("(*Valor*)",Utils
                                            .createHTMLSelectOptionTag(
                                             factor.idFactor + "",db,
                                             "SELECT valor,nombre FROM Valor "
                                             + "WHERE idFactor=" + idFactor));
                                        break;
                                    case TipoValor.VERDADERO_FALSO:
                                        pValor.setItem("(*Valor*)", Utils
                                            .createTrueFalseOption(
                                              factor.idFactor + ""));
                                        break;
                                    case TipoValor.AUTOMATICO:
                                       double val = Utils.
                                         getValueFromQueryAutomatico(factor,
                                             db,ip);
                                       String hf =Utils.
                                         createHTMLHiddenField(
                                             factor.idFactor+"", val + "");
                                       pValor.setItem("(*Valor*)", hf
                                             + val);
                                       break;
                                    default:
                                            break;
                            }
                            pValor.append();
                            pFactor.append();
                    }
                    pCriterio.append();
            }
            dwa.setPropertiesTo(page);
            PrintWriter out = res.getWriter();
            out.println(page.render());
            out.close();
    }

    /**
     * Inicializa la página de ejecución de alta.
```



```
     */
    protected void initPageAltaExec() {
            DbWeb dw = pageProperties[ALTAEXEC];
            dw.addPropertie("(*Titulo*)", "Alta de evaluacion de criterios");
    }

    /**
     * Modifica las propiedades de la forma de ejecución de alta en tiempo
     * de ejecución.
     * @param req
     */
    protected void modifyPropertiesPgExecAlta(HttpServletRequest req)
            throws BDException {
            SortedHashtable data;
            data = WebUtils.getFieldsAndValues(req);
            data.remove("accion");
            String ip = ((String[]) data.get("ip"))[0];
            data.remove("ip");
            /*
             * Algoritmo:
             *      Para cada factor:
             *              1 indentificar el tipo de factor
             *              2 procesarlo segun sea necesario
             *              3 guardar los datos en su caso.
             */
            String idFactorActual, valores[];
            java.util.Enumeration idFactores = data.keys();
            Factor factor;
            double valFactor = 0.0;
            for (int i = 0; idFactores.hasMoreElements(); i++) {
                    idFactorActual = (String) idFactores.nextElement();
                    valores = (String[]) data.get(idFactorActual);
                    factor = Factor.searchFactor(idFactorActual, db);
                    switch (factor.tipoValor) {
                            case TipoValor.AUTOMATICO:
                                    break;
                            case TipoValor.CONTADOR_NUMERICO:
                                    valFactor = valores.length;
                                    for (int j=0; j < valores.length; j++) {
                                        String key[] = { "idValor", "ip" };
                                        String val[] = new String[2];
                                        val[0] = valores[j];
                                        val[1] = "'" + ip + "'";
                                        db.add("ValoresEvaluadosCapturados",
                                             key, val);
                                    }
                                    break;
                            case TipoValor.PROP_MAX:
                            case TipoValor.FUNCION_NUMERICA:
                            case TipoValor.NUMERICO_DIRECTO:
                            case TipoValor.NUMERICO_SELECCIONABLE:
                            case TipoValor.VERDADERO_FALSO:
                                valFactor=Double.parseDouble(valores[0]);
                                    break;
                    }
                    String campos[] = { "ip", "idCriterio", "idFactor",
                            "valor","valorPonderado" };
                    String val[] = new String[campos.length];
                    val[0] = "'" + ip + "'";
                    val[1] = factor.idCriterio + "";
                    val[2] = factor.idFactor + "";
                    val[3] = valFactor + "";
                    val[4] = (valFactor * factor.ponderacion) + "";
                    if ((factor.tipoValor != TipoValor.AUTOMATICO)) {
                            db.add("Evaluacion", campos, val);
                    }
            }
    }
```

```
////////////////////////////////////////////////////////////////////////
//      CAMBIO
////////////////////////////////////////////////////////////////////////

/**
 * Inicializa los datos de la página de cambio de evaluaciones.
 */
protected void initPageCambio() {
        DbWebCambio ppc = (DbWebCambio) pageProperties[CAMBIO];
        ppc.action = "execcambio";
        ppc.addHiddenField("accion", "execCambioUpdate");
        ppc.addPropertie("(*Titulo*)", "Cambio de evaluación");
}

/**
 * Forma de captura de Cambio
 * @param req
 * @param res
 * @throws IOException
 * TODO: Revisar la forma en que se acceden a los datos de los criterios
 * y factores.
 */
protected void pgCambio(HttpServletRequest req, HttpServletResponse res)
        throws IOException {
        Factor factor;
        res.setContentType("text/html");
        String ip = req.getParameter("ip");
        DbWebCambio dwa = (DbWebCambio) pageProperties[CAMBIO];
        String archivo = PATH + "evaluacion.pg";
        PageGenerator page = initPageGenerator(archivo);
        String host = "";
        String queryCriterios = "SELECT * FROM Criterio";
        DbWeb dwCriterios = new DbWeb();
        String queryFactores = "SELECT * FROM Factor";
        DbWeb dwFactores = new DbWeb();
        String queryValores = "SELECT * FROM Valor";
        Vector qrCriterios, qrFactores;
        dwa.addHiddenField("ip", ip);
        try {
                host = db.getStringValue("host", "nombre", "ip='"
                                + ip + "'");
        } catch (BDException e) {
                e.printStackTrace();
        }
        dwa.addPropertie("(*Equipo*)", host);
        dwa.addPropertie("(*DatosForma*)",
                Utils.createHTMLFormTag("form1",
                dwa.action, "verify", dwa.javaScriptProperties));
        dwa.addPropertie("(*javaScript*)", dwa
                .createJavaScriptValidateFunction());
        if (dwa.hiddenFields != null) {
                dwa.addPropertie("(*hidenFields*)", Utils
                        .createHTMLHiddenFields(dwa.hiddenFields));
        } else {
                dwa.addPropertie("(*hidenFields*)", "");
        }
        PageGenerator pCriterio = page.getPG("(*PG_Criterio*)");
        pCriterio.clear();
        dwCriterios.setQuery(queryCriterios);
        qrCriterios = dwCriterios.getVectorOfQueryResult();
        //Para cada Criterio
        for (int i = 0; i < qrCriterios.size(); i++) {
               //Nombre del Criterio
                pCriterio.setItem("(*Criterio*)",
                        ((String[]) qrCriterios.get(i))[1]);
                PageGenerator pFactor = page.getPG("(*PG_Factor*)");
                pFactor.clear();
                //idCriterio
                dwFactores.setQuery(queryFactores+" WHERE idCriterio ="
```



```
                        + ((String[]) qrCriterios.get(i))[0]);
                qrFactores = dwFactores.getVectorOfQueryResult();
                //Para cada Factor
                for (int j = 0; j < qrFactores.size(); j++) {
                        //Nombre del Factor
                        pFactor.setItem("(*Factor*)",
                                ((String[]) qrFactores.get(j))[2]);
                        PageGenerator pValor=page.getPG("(*PG_Valor*)");
                        pValor.clear();
                        String idFactor = ((String[])
                                (qrFactores.get(j)))[0];
                        factor = Factor.searchFactor(idFactor, db);
                        String queryEval =
                                "SELECT * FROM Evaluacion WHERE ip ='"
                                + ip + "' AND " + "idFactor="+ idFactor;
                        Statement stmt = db.getStmt();
                        ResultSet rs;
                        try {
                                rs = stmt.executeQuery(queryEval);
                                if (!rs.next()) {
                                    switch (factor.tipoValor) {
                                            case TipoValor.PROP_MAX:
                                       case TipoValor.NUMERICO_DIRECTO:
                                         pValor.setItem("(*Valor*)",
                                           Utils.createHTMLInputTag(
                                             factor.idFactor + "",10));
                                       case TipoValor.FUNCION_NUMERICA:
                                         pValor.setItem("(*Valor*)",
                                           Utils.createHTMLInputTag(
                                             factor.idFactor + "",10));
                                         break;
                                       case TipoValor.CONTADOR_NUMERICO:
                                         pValor.setItem("(*Valor*)",
                                           Utils.createCheckBoxRowsForCo
ntadorNumerico(
                                              factor.idFactor + "", db,
                                              queryValores
                                              + " WHERE idFactor="
                                              + idFactor));
                                         break;
                                        case TipoValor.
                                                NUMERICO_SELECCIONABLE:
                                          pValor.setItem("(*Valor*)",
                                            Utils
                                             .createHTMLSelectOptionTag(
                                                factor.idFactor+"", db,
                                                "SELECT valor, nombre "
                                                +"FROM Valor WHERE "
                                                +"idFactor="+idFactor));
                                         break;
                                       case TipoValor.VERDADERO_FALSO:
                                         pValor.setItem("(*Valor*)",
                                           Utils.createTrueFalseOption(
                                                factor.idFactor+ ""));
                                         break;
                                        case TipoValor.AUTOMATICO:
                                         double val = Utils.
                                          getValueFromQueryAutomatico(
                                                factor, db, ip);
                                         String hf=Utils.
                                                createHTMLHiddenField(
                                                  factor.idFactor + "",
                                                  val + "");
                                                pValor.setItem(
                                                  "(*Valor*)",hf+val);
                                         break;
                                       default:
                                         break;
                                       }
```

```
                                } else {//if rs
                                  switch (factor.tipoValor) {
                                        case TipoValor.PROP_MAX:
                                        case TipoValor.NUMERICO_DIRECTO:
                                          pValor.setItem("(*Valor*)",
                                            Utils.createHTMLInputTag(
                                              factor.idFactor + "",
                                              rs.getInt("valor")
                                              +"",10));
                                        case TipoValor.FUNCION_NUMERICA:
                                            pValor.setItem("(*Valor*)",
                                              Utils.createHTMLInputTag(
                                                factor.idFactor + "",
                                                rs.getInt("valor")
                                                + "", 10));
                                           break;
                                        case TipoValor.
                                                CONTADOR_NUMERICO:
                                           pValor.setItem("(*Valor*)",
                                             Utils.createCheckBoxRowsFor
ContadorNumerico(
                                                factor.idFactor + "",
                                                db,queryValores
                                                + " WHERE idFactor="
                                                + idFactor, ip));
                                           break;
                                        case TipoValor
                                                .NUMERICO_SELECCIONABLE:
                                          pValor.setItem("(*Valor*)",
                                            Utils
                                            .createHTMLSelectOptionTag(
                                               factor.idFactor + "", db,
                                                "SELECT valor, nombre "
                                                +"FROM Valor WHERE "
                                                +"idFactor="+ idFactor,
                                               rs.getInt("valor")
                                               + ""));
                                          break;
                                        case TipoValor.VERDADERO_FALSO:
                                          pValor.setItem("(*Valor*)",
                                            Utils.createTrueFalseOption(
                                              factor.idFactor + "",
                                              rs.getInt("valor")+ ""));
                                          break;
                                        case TipoValor.AUTOMATICO:
                                          double val=
                                           Utils.getValueFromQueryAutoma
tico(
                                              factor, db, ip);
                                           String hf = Utils.
                                             createHTMLHiddenField(
                                               factor.idFactor + "",
                                               val + "");
                                             pValor.setItem("(*Valor*)",
                                               hf + val);
                                           break;
                                        default:
                                           break;
                                        }
                                  } catch (Exception e) {
                                        e.printStackTrace();
                                  }
                                  pValor.append();
                                  pFactor.append();
                          }
                          pCriterio.append();
                  }
                  dwa.setPropertiesTo(page);
```



```
                PrintWriter out = res.getWriter();
                out.println(page.render());
                out.close();
        }

        /**
         * Modifica las propiedades de la forma de ejecución de cambio en
         * tiempo de ejecución.
         * @param req
         */
        protected void modifyPropertiesPgExecCambioUpdate(
                        HttpServletRequest req) throws BDException {
                SortedHashtable data;
                data = WebUtils.getFieldsAndValues(req);
                data.remove("accion");
                String ip = ((String[]) data.get("ip"))[0];
                data.remove("ip");
                /*
                 * Algoritmo:
                 *      Para cada factor:
                 *      1 indentificar el tipo de factor
                 *      2 procesarlo segun sea necesario
                 *      3 guardar los datos en su caso.
                 */
                String idFactorActual, valores[];
                java.util.Enumeration idFactores = data.keys();
                Factor factor;
                int valFactor = 0;
                for (int i = 0; idFactores.hasMoreElements(); i++) {
                        idFactorActual = (String) idFactores.nextElement();
                        valores = (String[]) data.get(idFactorActual);
                        factor = Factor.searchFactor(idFactorActual, db);
                        switch (factor.tipoValor) {
                                case TipoValor.AUTOMATICO:
                                        break;
                                case TipoValor.CONTADOR_NUMERICO:
                                        valFactor = valores.length;
                                        db.del("ValoresEvaluadosCapturados",
                                                        "ip", "'" + ip + "'");
                                        for (int j=0; j < valores.length; j++) {
                                                String key[]={"idValor", "ip" };
                                                String val[] = new String[2];
                                                val[0] = valores[j];
                                                val[1] = "'" + ip + "'";
                                                db.add(
                                                  "ValoresEvaluadosCapturados",
                                                  key, val);
                                        }
                                        break;
                                case TipoValor.FUNCION_NUMERICA:
                                        break;
                                case TipoValor.NUMERICO_DIRECTO:
                                case TipoValor.NUMERICO_SELECCIONABLE:
                                case TipoValor.VERDADERO_FALSO:
                                case TipoValor.PROP_MAX:
                                        valFactor =Integer.parseInt(valores[0]);
                                        break;
                        }

                        String campos[] = { "ip", "idCriterio", "idFactor",
                                        "valor", "valorPonderado" };
                        String val[] = new String[campos.length];
                        val[0] = "'" + ip + "'";
                        val[1] = factor.idCriterio + "";
                        val[2] = factor.idFactor + "";
                        val[3] = valFactor + "";
                        val[4] = (valFactor * factor.ponderacion) + "";
                        if ((factor.tipoValor != TipoValor.AUTOMATICO)
                            &&factor.tipoValor!=TipoValor.FUNCION_NUMERICA) {
```

```
                                String donde = "idFactor=" + factor.idFactor
                                        + " AND ip='" + ip + "'";
                                db.change("Evaluacion", campos, val, donde);
                        }
                }
        }

        /**
         * Modifica las propiedades de la forma de ejecucion de baja en tiempo
         * de ejecución.
         * @param req
         */
        protected void modifyPropertiesPgExecBaja(HttpServletRequest req)
                throws BDException {
                SortedHashtable data;
                data = WebUtils.getFieldsAndValues(req);
                data.remove("accion");
                String ip = ((String[]) data.get("ip"))[0];
                db.del("Evaluacion", "ip", "'" + ip + "'");
                db.del("ValoresEvaluadosCapturados", "ip", "'" + ip + "'");
        }
}
```

```
package itesm.seguridad;
/*
 * @author      Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import java.io.*;
import javax.servlet.http.*;
import itesm.utilsdsc.*;
/**
 * Esta clase es la interfaz para definir los fatores que forman los
 * criterios  (Un criterio esta formado por un conjunto de facores, el valor de
 * un criterio es la suma poderada de un conjunto de fatores).
 */
public class EsquemaFactor extends Esquema {
        private static final long serialVersionUID = -6566477394170222425L;
        private DbWebAlta dwaq, dwaf;
        private DbWebExecCambio dwecq, dwecf;
        /**
         * Consturctor que básicamente espefica el servlet a invocar.
         */
        public EsquemaFactor(){
                super();
                setURLtoAllPages("/sderi/EsquemaFactor");
        }

        /**
         * Metodo para guardar en un los hidenfields de un DbWeb los datos de
         * un factor.
         * @param req
         * @param pp
         */
        protected void getFactorFormData(HttpServletRequest req, DbWeb pp){
                pp.addHiddenField(
                        "idCriterio", req.getParameter("idCriterio"));
                pp.addHiddenField(
                        "nombre", req.getParameter("nombre"));
                pp.addHiddenField(
                        "descripcion", req.getParameter("descripcion"));
                pp.addHiddenField(
                        "tipoValor", req.getParameter("tipoValor"));
                pp.addHiddenField(
                        "ponderacion", req.getParameter("ponderacion"));
                pp.addHiddenField("factorCapturado", "1");
        }
        /**
         * Inidica el servlet a invocar
         */
        protected void initPageSubMenu(){
                DbWebSubMenu dw=(DbWebSubMenu)pageProperties[SUBMENU];
                dw.setNewServlet("EsquemaValor");
        }
        /**
         * Redirecciona el servlet a  invocar con sus repectivos parametros
         * @param req
         * @param res
         */
        protected void pgSubMenu(HttpServletRequest req,HttpServletResponse res)
                throws IOException {
                DbWebSubMenu dw=(DbWebSubMenu)pageProperties[SUBMENU];
                HttpURL url= new HttpURL(req);
                String idCriterio = req.getParameter("idCriterio");
                url.setFile("/"+ dw.getNewServlet());
                url.addVar("idCriterio", idCriterio);
                res.sendRedirect(url+"");
        }
        /**
         * Especifica el menu y el titulo del mismo.
```



```
         */
        protected void initPageMenu(){
                DbWeb dbweb = new DbWeb();
                String  query = "SELECT * FROM Criterio";
                DbWebMenu  ppm;
                ppm = (DbWebMenu)pageProperties[MENU];
                SortedHashtable menu = ppm.getMenu();
                menu.put("Alta", "alta");
                menu.put("Baja", "baja");
                menu.put("Cambio", "cambio");
                menu.put("Administración de Valores posibles ", "subMenu");
                ppm.setTitle("Administración de Factores");
                ppm.addPropertie("(*Menu*)",
                        Utils.createOptionButtonMenuRows("accion", menu));
                dbweb.setQuery(query);
                dbweb.setTitleField("idCriterio", "");
                dbweb.setTitleField("nombre", "Nombre");
                dbweb.setTitleField("descripcion", "Descripción");
                dbweb.setTitleField("ponderacion", "Ponderación");
                ppm.addPropertie("(*OptMenu*)",
                     Utils.createRadioBottonRows(dbweb.getFields(), db, query));
                ppm.addPropertie("(*Instrucciones1*)",
                                "1. Seleccione el criterio  donde hará la "
                                +"operación:");
                ppm.addPropertie("(*Instrucciones2*)",
                                "2. Seleccione la operación a realizar:");
        }

        ////////////////////////////////////////////////////////////////////////
        /////A L T A de Factor
        ////////////////////////////////////////////////////////////////////////
        /**
         * Inicializa la página de alta.
         */
        protected void initPageAlta(){
                DbWebAlta ppa;
                table="Factor";
                ppa=(DbWebAlta)pageProperties[ALTA];
                ppa.setQuery("SELECT idFactor, idCriterio, nombre, " +
                        "descripcion, ponderacion, tipoValor FROM "+table);
                ppa.addHiddenField("accion", "execalta");
                ppa.addPropertie("(*Titulo*)", "Alta de Factor");
                ppa.unSetVisibleField("idCriterio");
                ppa.setSelectionableField("tipoValor",
                        "SELECT idTipoValor,nombre FROM TipoValor");
                ppa.setTitleField("nombre", "Nombre");
                ppa.setTitleField("descripcion","Descripción");
                ppa.setTitleField("tipoValor", "Tipo de Valor");
                ppa.setTitleField("ponderacion", "Ponderación");
                initPageAltaQueryAutomatico();
                initPageAltaFuncion();
        }
        /**
         * Modifica las propiedades de la forma de alta en tiempo de ejecución.
         */
        protected void modifyPropertiesPgAlta(HttpServletRequest req){
                DbWebAlta dwa = (DbWebAlta) pageProperties[ALTA];
                dwa.addHiddenField("idCriterio",
                                req.getParameter("idCriterio"));
        }
        /**
         * Ejecuta el Alta de factor, verificando si el tipo de dato es
         * "Automático". Si es asi, entonces solicita el query por medio del
         * cual se va a hacer el  calculo.
         * También verifica si el tipo es "Función numérica", si es asi
         * solicita los* datos de la función numérica.
         * @param req
         * @param res
         * @throws IOException
```

```
     */
    protected void pgExecAlta(HttpServletRequest req,
                    HttpServletResponse res) throws IOException {
            String factorCapturado="";
            res.setContentType("text/html; charset=iso-8859-1");
            int tipoValor = Integer.parseInt(req.getParameter("tipoValor"));
            factorCapturado = req.getParameter("factorCapturado");
            switch (tipoValor){
                    case TipoValor.AUTOMATICO :
                            if (factorCapturado == null)
                                    pgAltaQueryAutomatico(req, res);
                            else
                                    pgAltaExecQueryAutomatico(req,res);
                            break;
                    case TipoValor.FUNCION_NUMERICA:
                            if (factorCapturado == null)
                                    pgAltaFuncion(req, res);
                            else
                                    pgAltaExecFuncion(req,res);

                            break;
                    default:
                            super.pgExecAlta(req,res);

            }
    }
    /**
     * Inicializa la página alta del query automatico.
     */
    protected void initPageAltaQueryAutomatico(){
            dwaq =new DbWebAlta ("Alta de query para factor automatico",
                    "/sderi/EsquemaFactor");
            dwaq.setQuery("SELECT nombre1, descripcion1, query, maxQuery,"
                            +" cond, campoIP, idConexionBD, stringIP "
                            +" FROM QueryAutomatico" );
            dwaq.addHiddenField("accion", "execalta");
            dwaq.addPropertie("(*Titulo*)",
                    "Alta de querys para la automatización de factores");
            dwaq.setTitleField("nombre1", "Nombre");
            dwaq.setTitleField("descripcion1", "Descripción");
            dwaq.setTitleField("query",
                    "SELECT para obtener el valor del factor");
            dwaq.setTitleField("maxQuery",
                  "SELECT para obtener el valor maximo evaluado del factor");
            dwaq.setTitleField("cond", "Condicional (Clausula WHERE) ");
            dwaq.setTitleField("campoIP", "Campo llave (dirección IP)");
            dwaq.setTitleField("idConexionBD", "Conexion de base datos");
            dwaq.setTitleField("stringIP",
                            "La dirección IP es de tipo string ?");
            dwaq.setSelectionableField("idConexionBD",
                            "SELECT idConexionBD,nombre FROM ConexionBD");
    }

    /**
     * Inicializa la página alta del query automatico.
     */
    protected void initPageAltaFuncion(){
            dwaf = new DbWebAlta ("Alta de función para factor",
                    "/sderi/EsquemaFactor");
            dwaf.setQuery("SELECT variable, funcion FROM FuncionFactor ");
            dwaf.addPropertie("(*Titulo*)", "Alta de funcion para factor.");
            dwaf.addHiddenField("accion", "execalta");
            dwaf.setTitleField("variable","Variable");
            dwaf.setTitleField("funcion", "Función");
    }

    /**
     * Forma de captura de Alta de un query de automatizacion para medir un
     * factor.
```



```
     * @param req
     * @param res
     * @throws IOException
     */
    protected void pgAltaQueryAutomatico(HttpServletRequest req,
            HttpServletResponse res) throws IOException {
            res.setContentType("text/html");
            String archivo = PATH + "query.pg";
            PageGenerator page = initPageGenerator(archivo);
            getFactorFormData(req, dwaq);
            dwaq.addPropertie("(*Forma*)",
                    dwaq.createDBFormAlta());
            dwaq.addPropertie("(*javaScript*)",
                    dwaq.createJavaScriptValidateFunction());
            dwaq.setPropertiesTo(page);
            PrintWriter out = res.getWriter();
            out.println(page.render());
            out.close();
    }

    /**
     * Forma de captura de Alta de una función para medir un factor.
     * @param req
     * @param res
     * @throws IOException
     */
    protected void pgAltaFuncion(HttpServletRequest req,
            HttpServletResponse res) throws IOException {
            res.setContentType("text/html");
            String archivo = PATH + "funcion.pg";
            PageGenerator page = initPageGenerator(archivo);
            getFactorFormData(req, dwaf);
            dwaf.addPropertie("(*Forma*)",
                    dwaf.createDBFormAlta());
            dwaf.addPropertie("(*javaScript*)",
                    dwaf.createJavaScriptValidateFunction());
            dwaf.setPropertiesTo(page);
            PrintWriter out = res.getWriter();
            out.println(page.render());
            out.close();
    }

    /**
     * Agrega en la base de datos un registro de un factor y un registro de
     * un query automatico.
     * @param req
     * @throws BDException
     */
    protected void addFactorYQueryAutomatico(HttpServletRequest req)
            throws BDException{
            SortedHashtable datfactor, datqueryAutomatico, data;
            data = WebUtils.getFieldsAndValues(req);
            data.remove("accion");
            data.remove("factorCapturado");
            datfactor = (SortedHashtable) data.clone();
            datqueryAutomatico = (SortedHashtable) data.clone();
            //a factor le quitamos los atributos del query aumatico
            datfactor.remove("nombre1");
            datfactor.remove("descripcion1");
            datfactor.remove("query");
            datfactor.remove("maxQuery");
            datfactor.remove("cond");
            datfactor.remove("campoIP");
            datfactor.remove("idConexionBD");
            datfactor.remove("stringIP");
            //a query aumatico le quitamos los atributos del factor
            datqueryAutomatico.remove("idCriterio");
            datqueryAutomatico.remove("nombre");
            datqueryAutomatico.remove("descripcion");
```

```
            datqueryAutomatico.remove("tipoValor");
            datqueryAutomatico.remove("ponderacion");
            //insertamos el query a la base de datos.
            datqueryAutomatico = WebUtils.formatDataforDB(
                    "SELECT * FROM QueryAutomatico", datqueryAutomatico);
            db.add("QueryAutomatico", datqueryAutomatico);
            //obtemos el idQuery
            int idQuery= db.getIntegerValue("QueryAutomatico",
                    "idQueryAutomatico","nombre1='" +
                    req.getParameter("nombre1")+ "'" );
            String aux[] = new String[1];
            aux[0]= ""+idQuery;
            datfactor.put("idQueryTipoAutomatico", aux);
            //      agregamos el factor
            datfactor= WebUtils.formatDataforDB(
                            "SELECT * FROM Factor", datfactor);
            db.add("Factor", datfactor);
    }

    /**
     * Agrega en la base de datos un registro de un factor y funcion
     * @param req
     * @throws BDException
     */
    protected void addFactorYFuncion(HttpServletRequest req)
            throws BDException{
            SortedHashtable datfactor, datfuncion, data;
            data = WebUtils.getFieldsAndValues(req);
            data.remove("accion");
            data.remove("factorCapturado");
            datfactor = (SortedHashtable) data.clone();
            datfuncion = (SortedHashtable) data.clone();
            //a factor le quitamos los atributos de la  funcion
            datfactor.remove("query");
            datfactor.remove("variable");
            datfactor.remove("funcion");
            //a query aumatico le quitamos los atributos del factor
            datfuncion.remove("idCriterio");
            datfuncion.remove("nombre");
            datfuncion.remove("descripcion");
            datfuncion.remove("tipoValor");
            datfuncion.remove("ponderacion");
            //insertamos el query a la base de datos.
            datfuncion = WebUtils.formatDataforDB(
                    "SELECT * FROM FuncionFactor", datfuncion);
            db.add("FuncionFactor", datfuncion);
            //obtemos el idFuncion
            int idFuncion= db.getIntegerValue("FuncionFactor", "idFuncion",
                    "funcion='" +
                    req.getParameter("funcion")+
                    "'" );
            String aux[] = new String[1];
            aux[0]= ""+idFuncion;
            datfactor.put("idQueryTipoAutomatico", aux);
            //      agregamos el factor
            datfactor= WebUtils.formatDataforDB("SELECT * FROM Factor",
                            datfactor);
            db.add("Factor", datfactor);
    }

    /**
     * Pagina de resultados del Alta de una funcion del factor.
     * @param req
     * @param res
     * @throws IOException
     */
    protected void pgAltaExecFuncion(HttpServletRequest req,
            HttpServletResponse res) throws IOException {
            res.setContentType("text/html; charset=iso-8859-1");
```



```
            DbWeb dw = new DbWeb();
            String archivo = PATH + PG[ALTAEXEC];
            try{
                    addFactorYFuncion(req);
            } catch (BDException e) {
                    error +="Error al ejecutar el alta de la funcion " +
                            "del factor: <BR>\n"+
                            "Query =" + e.getQuery() +"<BR>\n"+
                            "msgError =" + e.sqle.getMessage() +"<BR>\n"+
                            e.getStackTrace();
            }
            if (error.equalsIgnoreCase("")){
                    dw.addPropertie("(*Mensaje*)", "Alta exitosa.");
            }else{
                    dw.addPropertie("(*Mensaje*)", error);
                    error="";
            }
            PageGenerator page = initPageGenerator(archivo);
            dw.setPropertiesTo(page);
            PrintWriter out = res.getWriter();
            out.println(page.render());
            out.close();
    }

    /**
     * Página de resultados del Alta de un query de automatización  y del
     * factor.
     * @param req
     * @param res
     * @throws IOException
     */
    protected void pgAltaExecQueryAutomatico(HttpServletRequest req,
            HttpServletResponse res) throws IOException {
            res.setContentType("text/html; charset=iso-8859-1");
            DbWeb dw = new DbWeb();
            String archivo = PATH + PG[ALTAEXEC];
            try{
                    addFactorYQueryAutomatico(req);
            } catch (BDException e) {
                    error +="Error al ejecutar el alta del query: <BR>\n"+
                            "Query =" + e.getQuery() +"<BR>\n"+
                            "msgError =" + e.sqle.getMessage() +"<BR>\n"+
                            e.getStackTrace();
            }
            if (error.equalsIgnoreCase("")){
                    dw.addPropertie("(*Mensaje*)", "Alta exitosa.");
            }else{
                    dw.addPropertie("(*Mensaje*)", error);
                    error="";
            }
            PageGenerator page = initPageGenerator(archivo);
            dw.setPropertiesTo(page);
            PrintWriter out = res.getWriter();
            out.println(page.render());
            out.close();
    }
    ////////////////////////////////////////////////////////////////////////
    /// B A J A de Factor
    ////////////////////////////////////////////////////////////////////////
    /**
     * protected void initPageBaja()
     *
     * Inicializa la página de la forma de baja
     */
    protected void initPageBaja(){
            DbWebBaja dwb=(DbWebBaja)pageProperties[BAJA];;
            dwb.addPropertie("(*Titulo*)", "Baja de Factores");
            dwb.addPropertie("(*Instrucciones1*)",
                    "1.Seleccione los factores a dar de baja.");
```

```
        }

        /**
         * Modifica las propiedades de la forma de baja en tiempo de
         * ejecución.
         * @param req
         */
        protected void modifyPropertiesPgBaja(HttpServletRequest req){
                DbWebBaja dwb = (DbWebBaja) pageProperties[BAJA];
                String idCriterio = req.getParameter("idCriterio");
                String query = "SELECT idFactor, nombre, descripcion "
                        +"FROM Factor WHERE idCriterio="
                        + idCriterio ;
                dwb.setQuery(query);
                dwb.addHiddenField("accion", "execBaja");
                dwb.setTitleField("idFactor", "");
                dwb.setTitleField("nombre", "Nombre");
                dwb.setTitleField("descripcion", "Descripción");
        }

        /**
         * Modifica las propiedades de la forma de ejecucion de baja
         * en tiempo de ejecución.
         * @param req
         */
        protected void modifyPropertiesPgExecBaja(HttpServletRequest req)
                throws BDException{
                SortedHashtable data;
                data = WebUtils.getFieldsAndValues(req);
                data.remove("accion");
                data = WebUtils.formatDataforDB("SELECT * FROM Factor", data);
                String idFactor = req.getParameter("idFactor");
                Factor factor= Factor.searchFactor(idFactor, db);
                switch (factor.tipoValor){
                        case TipoValor.AUTOMATICO:
                                db.del("QueryAutomatico", "idQueryAutomatico",
                                        factor.idQueryTipoAutomatico+"");
                                break;
                        case TipoValor.FUNCION_NUMERICA:
                                db.del("FuncionFactor", "idFuncion",
                                        factor.idQueryTipoAutomatico+"");
                                break;
                }
                db.del("Factor", data);
        }

        /**
         * Inicializa la página de ejecución de baja.
         *
         */
        protected void initPageBajaExec(){
                DbWeb dw=pageProperties[BAJAEXEC];
                dw.addPropertie("(*Titulo*)", "Ejecución de baja de factor ");
                dw.addPropertie("(*Mensaje*)", "Baja exitosa.");
        }

        ////////////////////////////////////////////////////////////////////////
        /// C A M B I O de Factor
        ////////////////////////////////////////////////////////////////////////
        /**
         * Inicializa la página de la forma de cambio.
         */
        protected void initPageCambio(){
                DbWebCambio dwc=(DbWebCambio)pageProperties[CAMBIO];
                dwc.addPropertie("(*Titulo*)", "Cambio de Factor");
                dwc.addPropertie("(*Instrucciones1*)",
                        "1.Seleccione el factor a cambiar.");
        }
        /**
```



```
         * Modifica las propiedades de la forma de cambio en tiempo de
         * ejecución.
         * @param req
         */
        protected void modifyPropertiesPgCambio(HttpServletRequest req){
                DbWebCambio dwc = (DbWebCambio) pageProperties[CAMBIO];
                String query="SELECT idFactor,nombre,descripcion,ponderacion" +
                        " FROM Factor WHERE " +
                        "idCriterio="+ req.getParameter("idCriterio");
                dwc.setQuery(query);
                dwc.setTitleField("idFactor","");
                dwc.setTitleField("nombre","Nombre");
                dwc.setTitleField("descripcion", "Descripción");
                dwc.setTitleField("ponderacion","Ponderación");
                dwc.addHiddenField("idCriterio",
                                req.getParameter("idCriterio"));
                dwc.addHiddenField("accion", "execCambio");
        }

        /**
         * Inicializa la página de captura del cambio
         */
        protected void initPageCambioExec(){
                DbWebExecCambio dwce;
                dwce=(DbWebExecCambio)pageProperties[CAMBIOEXEC];
                dwce.setQuery("SELECT * FROM "+table);
                dwce.addHiddenField("accion", "execCambioUpdate");
                dwce.addPropertie("(*Titulo*)", "Cambio de factor");
                dwce.addPropertie("(*Instrucciones1*)","");
                initPageCambioQueryAutomatico();
                initPageCambioFuncion();
        }

        /**
         * Modifica las propiedades de la forma de ejecucion del
         * cambio en tiempo de ejecución.
         * @param req
         */
        protected void modifyPropertiesPgCambioExec(HttpServletRequest req){
                DbWebExecCambio dwce =
                        (DbWebExecCambio)pageProperties[CAMBIOEXEC];
                SortedHashtable data;
                String key,val;
                data= WebUtils.getFieldsAndValues(req);
                data.remove("accion");
                key= "idFactor";
                val= req.getParameter("idFactor");
                dwce.setSearchingValues(true);
                dwce.setQuery("SELECT * FROM "+ table +" WHERE "+key+"="+val );
                dwce.addHiddenField("idFactor", req.getParameter("idFactor"));
                dwce.addHiddenField("idCriterio",
                                req.getParameter("idCriterio"));
                dwce.unSetVisibleField("idCriterio");
                dwce.setSelectionableField("tipoValor",
                        "SELECT idTipoValor,nombre FROM TipoValor");
                dwce.setTitleField("nombre", "Nombre ");
                dwce.setTitleField("descripcion", "Descripción ");
                dwce.unSetEditableField("tipoValor");
                dwce.setTitleField("tipoValor", "Tipo de Valor ");
                dwce.setTitleField("ponderacion", "Ponderación ");
                dwce.unSetVisibleField("idQueryTipoAutomatico");
        }

        /**
         * Inicializa la página cambio del query automatico.
         */
        protected void initPageCambioQueryAutomatico(){
                dwecq = new DbWebExecCambio("Cambio para query Automático",
                        "/sderi/EsquemaFactor");
```

```
            dwecq.setQuery("SELECT * FROM QueryAutomatico ");
            dwecq.addPropertie("(*Titulo*)", "Cambio de Query para " +
                    "factor Automático");
            dwecq.addHiddenField("accion", "execCambioUpdate");
            dwecq.setTitleField("nombre1", "Nombre");
            dwecq.setTitleField("descripcion1", "Descripción");
            dwecq.setTitleField("query",
                            "SELECT para obtener el valor del factor");
            dwecq.setTitleField("cond", "Condicional (Clausula WHERE) ");
            dwecq.setTitleField("maxQuery",
                            "SELECT para obtener el valor maximo evaluado"
                            +" del factor");
            dwecq.setTitleField("campoIP", "Campo llave (dirección IP)");
            dwecq.setTitleField("idConexionBD", "Conexion de base datos");
            dwecq.setTitleField("stringIP", "La dirección IP es de tipo "
                            +"string ?");
            dwecq.setSelectionableField("idConexionBD",
                            "SELECT idConexionBD,nombre FROM ConexionBD");
    }

    /**
     * Inicializa la página cambio de la funcion.
     */
    protected void initPageCambioFuncion(){
            dwecf = new DbWebExecCambio("Cambio para la funcion",
                    "/sderi/EsquemaFactor");
            dwecf.setSearchingValues(true);
            dwecf.setQuery("SELECT * FROM FuncionFactor ");
            dwecf.addPropertie("(*Titulo*)",
                            "Cambio de función para factor");
            dwecf.addHiddenField("accion", "execCambioUpdate");
            dwecf.setTitleField("variable","Variable");
            dwecf.setTitleField("funcion", "Función");
    }

    /**
     * Modifica las propiedades del Cambio del query automatico en tiempo
     * de ejecución.
     * @param req
     * @param dwecq
     */
    protected void modifyPropertiesPgCambioQueryAutomatico(
            HttpServletRequest req, DbWebExecCambio dwecq){
            SortedHashtable data;
            String key,val;
            data= WebUtils.getFieldsAndValues(req);
            data.remove("accion");
            key= "idQueryAutomatico";
            val= req.getParameter("idQueryAutomatico");
            dwecq.setSearchingValues(true);
            dwecq.setQuery("SELECT * FROM QueryAutomatico WHERE "
                            +key+"="+val);
            dwecq.setTitleField("nombre1", "Nombre");
            dwecq.setTitleField("descripcion1", "Descripción");
            dwecq.setTitleField("query",
                            "SELECT para obtener el valor del factor");
            dwecq.setTitleField("cond", "Condicional (Clausula WHERE) ");
            dwecq.setTitleField("maxQuery",
                  "SELECT para obtener el valor maximo evaluado del factor");
            dwecq.setTitleField("campoIP", "Campo llave (dirección IP)");
            dwecq.setTitleField("idConexionBD", "Conexion de base datos");
            dwecq.setTitleField("stringIP",
                            "La dirección IP es de tipo string ?");
            dwecq.setSelectionableField("idConexionBD",
                            "SELECT idConexionBD,nombre FROM ConexionBD");

    }

    /**
```



```
     * Modifica las propiedades del Cambio de la función numérica
     * en tiempo de ejecución.
     * @param req
     * @param dwecq
     */
    protected void modifyPropertiesPgCambioFuncion(
            HttpServletRequest req, DbWebExecCambio dwecq){
            SortedHashtable data;
            String key,val;
            data= WebUtils.getFieldsAndValues(req);
            data.remove("accion");
            key= "idFuncion";
            val= req.getParameter("idQueryTipoAutomatico");
            dwecf.setSearchingValues(true);
            dwecf.setQuery("SELECT * FROM FuncionFactor WHERE "
                            +key+"="+val );
            dwecf.setTitleField("variable","Variable");
            dwecf.setTitleField("funcion", "Función");
    }

    /**
     * Forma de captura de Alta de un query de automatizacion para medir un
     * factor.
     * @param req
     * @param res
     * @throws IOException
     */
    protected void pgCambioQueryAutomatico(HttpServletRequest req,
            HttpServletResponse res) throws IOException {
            res.setContentType("text/html");
            String archivo = PATH + "query.pg";
            PageGenerator page = initPageGenerator(archivo);
            getFactorFormData(req, dwecq);
            dwecq.addHiddenField("idQueryTipoAutomatico",
                    req.getParameter("idQueryTipoAutomatico"));
            modifyPropertiesPgCambioQueryAutomatico(req, dwecq);
            dwecq.addPropertie("(*Forma*)",
                    dwecq.createDBFormExecCambio());
            dwecq.addPropertie("(*javaScript*)",
                    dwecq.createJavaScriptValidateFunction());
            dwecq.setPropertiesTo(page);
            PrintWriter out = res.getWriter();
            out.println(page.render());
            out.close();
    }
    /**
     * Forma de captura de cambio de una funcion para medir un factor.
     * @param req
     * @param res
     * @throws IOException
     */
    protected void pgCambioFuncion(HttpServletRequest req,
            HttpServletResponse res) throws IOException {
            res.setContentType("text/html");
            String archivo = PATH + "funcion.pg";
            PageGenerator page = initPageGenerator(archivo);
            getFactorFormData(req, dwecf);
            dwecf.addHiddenField("idQueryTipoAutomatico",
                    req.getParameter("idQueryTipoAutomatico"));
            modifyPropertiesPgCambioFuncion(req, dwecf);
            dwecf.addPropertie("(*Forma*)",
                    dwecf.createDBFormExecCambio());
            dwecf.addPropertie("(*javaScript*)",
                    dwecf.createJavaScriptValidateFunction());
            dwecf.setPropertiesTo(page);
            PrintWriter out = res.getWriter();
            out.println(page.render());
            out.close();
    }
```

```
        /**
         * Ejecuta el Cambio de factor, verificando si el tipo de dato es
         * "Automatico". Si es asi, entonces solicita el query por medio del
         * cual se va a hacer el
         * calculo.
         * Tambien verifica si el factor es de tipo "función numérica" ,
         * alterando  los datos correspondientes.
         * @param req
         * @param res
         * @throws IOException
         */
        protected void pgExecCambioUpdate(HttpServletRequest req,
                HttpServletResponse res) throws IOException {
                res.setContentType("text/html; charset=iso-8859-1");
                int tipoValor = Integer.parseInt(req.getParameter("tipoValor"));
                String factorCapturado = req.getParameter("factorCapturado");
                switch (tipoValor){
                        case TipoValor.AUTOMATICO :
                                if (factorCapturado == null)
                                        pgCambioQueryAutomatico(req, res);
                                else
                                        pgCambioExecQueryAutomatico(req,res);
                                break;
                        case TipoValor.FUNCION_NUMERICA:
                                if (factorCapturado == null)
                                        pgCambioFuncion(req, res);
                                else
                                        pgCambioExecFuncion(req,res);
                                break;
                        default:
                                super.pgExecCambioUpdate(req,res);
                }
        }

        /**
         * Inicializa la página de ejecucion de alta, especifica el mensaje
         */
        protected void initPageCambioExecUpdate(){
                DbWeb dw=pageProperties[CAMBIOEXECUPDATE];
                dw.addPropertie("(*Titulo*)", "Ejecución de Cambio de Factor");
                dw.addPropertie("(*Mensaje*)", "Cambio exitoso.");
        }
        /**
         * Modifica las propiedades de la forma de ejecucion de actualización
         * del cambio tiempo de ejecución.
         * @param req
         */
        protected void modifyPropertiesPgExecCambioUpdate(
                        HttpServletRequest req) throws BDException{
                String  val;
                SortedHashtable data;
                val= req.getParameter("idFactor");
                data= WebUtils.getFieldsAndValues(req);
                data.remove("accion");
                data.remove("idFactor");
                data=WebUtils.formatDataforDB("SELECT * FROM "+table, data);
                db.change(table, data, "idFactor = "+val);
        }

        /**
         * Actualiza en la base de datos un registro de un factor y un
         * registro de un query automatico.
         * @param req
         * @throws BDException
         */
        protected void updateFactorYQueryAutomatico(HttpServletRequest req)
                throws BDException{
                SortedHashtable datfactor, datqueryAutomatico, data;
```



```
                data = WebUtils.getFieldsAndValues(req);
                data.remove("accion");
                data.remove("factorCapturado");
                datfactor = (SortedHashtable) data.clone();
                datqueryAutomatico = (SortedHashtable) data.clone();
                //a factor le quitamos los atributos del query aumatico
                datfactor.remove("query");
                datfactor.remove("campoCondicion");
                datfactor.remove("condicion");
                datfactor.remove("idQuery");
                datfactor.remove("maxQuery");
                //a query aumatico le quitamos los atributos del factor
                datqueryAutomatico.remove("idCriterio");
                datqueryAutomatico.remove("idFactor");
                datqueryAutomatico.remove("nombre");
                datqueryAutomatico.remove("descripcion");
                datqueryAutomatico.remove("tipoValor");
                datqueryAutomatico.remove("ponderacion");
                datqueryAutomatico.remove("idQueryTipoAutomatico");
                //Acutalizamos  el query en la base de datos.
                datqueryAutomatico = WebUtils.formatDataforDB(
                        "SELECT * FROM QueryTipoAutomatico", datqueryAutomatico);
                db.change("QueryTipoAutomatico", datqueryAutomatico,
                        "idQuery="+ req.getParameter("idQuery") );
                //      Actualizamos el factor
                datfactor= WebUtils.formatDataforDB(
                                "SELECT * FROM Factor", datfactor);
                db.change("Factor", datfactor,
                        "idFactor=" + req.getParameter("idFactor"));
        }

        /**
         * Actualiza en la base de datos un registro de un factor y un
         * registro de una funcion.
         * @param req
         * @throws BDException
         */
        protected void updateFactorYFuncion(HttpServletRequest req)
                throws BDException{
                SortedHashtable datfactor, datfuncion, data;
                data = WebUtils.getFieldsAndValues(req);
                data.remove("accion");
                data.remove("factorCapturado");
                datfactor = (SortedHashtable) data.clone();
                datfuncion= (SortedHashtable) data.clone();
                //a factor le quitamos los atributos del query aumatico
                datfactor.remove("variable");
                datfactor.remove("funcion");
                datfactor.remove("idFuncion");
                //a query aumatico le quitamos los atributos del factor
                datfuncion.remove("idCriterio");
                datfuncion.remove("idFactor");
                datfuncion.remove("nombre");
                datfuncion.remove("descripcion");
                datfuncion.remove("tipoValor");
                datfuncion.remove("ponderacion");
                datfuncion.remove("idQueryTipoAutomatico");
                //Acutalizamos  el query en la base de datos.
                datfuncion = WebUtils.formatDataforDB(
                        "SELECT * FROM FuncionFactor", datfuncion);
                db.change("FuncionFactor", datfuncion,
                        "idFuncion="+ req.getParameter("idFuncion") );
                //      Actualizamos el factor
                datfactor= WebUtils.formatDataforDB(
                                "SELECT * FROM Factor", datfactor);
                db.change("Factor", datfactor,
                        "idFactor=" + req.getParameter("idFactor"));
        }
```

```
        /**
         * Pagina de resultados del Alta de un query de automatizacion  y del
         * factor.
         * @param req
         * @param res
         * @throws IOException
         */
        protected void pgCambioExecQueryAutomatico(HttpServletRequest req,
                HttpServletResponse res) throws IOException {
                res.setContentType("text/html; charset=iso-8859-1");
                DbWeb dw = new DbWeb();
                String archivo = PATH + PG[CAMBIOEXECUPDATE];
                try{
                        updateFactorYQueryAutomatico(req);
                } catch (BDException e) {
                        error +="Error al ejecutar el cambio del query: <BR>\n"+
                                "Query =" + e.getQuery() +"<BR>\n"+
                                "msgError =" + e.sqle.getMessage() +"<BR>\n"+
                                e.getStackTrace();
                }
                if (error.equalsIgnoreCase("")){
                        dw.addPropertie("(*Mensaje*)", "Cambio exitoso.");
                }else{
                        dw.addPropertie("(*Mensaje*)", error);
                        error="";
                }
                PageGenerator page = initPageGenerator(archivo);
                dw.setPropertiesTo(page);
                PrintWriter out = res.getWriter();
                out.println(page.render());
                out.close();
        }

        /**
         * Pagina de resultados del Alta de un query de automatizacion  y del
         * factor.
         * @param req
         * @param res
         * @throws IOException
         */
        protected void pgCambioExecFuncion(HttpServletRequest req,
                HttpServletResponse res) throws IOException {
                res.setContentType("text/html; charset=iso-8859-1");
                DbWeb dw = new DbWeb();
                String archivo = PATH + PG[CAMBIOEXECUPDATE];
                try{
                        updateFactorYFuncion(req);
                } catch (BDException e) {
                    error +="Error al ejecutar el cambio de la funcion: <BR>\n"+
                "Query =" + e.getQuery() +"<BR>\n"+
                        "msgError =" + e.sqle.getMessage() +"<BR>\n"+
                        e.getStackTrace();
                }
                if (error.equalsIgnoreCase("")){
                        dw.addPropertie("(*Mensaje*)", "Cambio exitoso.");
                }else{
                        dw.addPropertie("(*Mensaje*)", error);
                        error="";
                }
                PageGenerator page = initPageGenerator(archivo);
                dw.setPropertiesTo(page);
                PrintWriter out = res.getWriter();
                out.println(page.render());
                out.close();
        }
}
/*
 * TODOs
 * Ver si para el QueryAutomatico y la FunciónNumerica se pueden enviar a
```



```
 * otras clases.
 */
```

```
package itesm.seguridad;
/*
 * @author      Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import javax.servlet.http.*;
import itesm.utilsdsc.*;
/**
 * Esta clase es la interfase para base de datos de valores POSIBLES, a fin de
 * que estos sean tomados a la hora de querer evaluar un host especifico para
 * determinar su nivel de seguridad.
 *
 * Un factor puede o no tener un conjunto de valores posibles.
 */
public class EsquemaFuncion extends Esquema {
        private static final long serialVersionUID = -6978002171356148938L;
        /**
         * Constructor.
         * Especifca el servlet a invocar.
         */
        public EsquemaFuncion() {
                super();
                setURLtoAllPages("/sderi/EsquemaFuncion");
                table = "FuncionFactor";
        }

        /**
         * Especifica el menu y el titulo del mismo.
         */
        protected void initPageMenu() {
                DbWebMenu ppm;
                ppm = (DbWebMenu) pageProperties[MENU];
                SortedHashtable menu = ppm.getMenu();
                menu.put("Alta", "alta");
                menu.put("Baja", "baja");
                menu.put("Cambio", "cambio");
                ppm.setTitle("Administración de Valores");
                ppm.addPropertie("(*Menu*)", Utils.
                                createOptionButtonMenuRows("accion",menu));
                ppm.addPropertie("(*Instrucciones1*)", "1. Seleccione el factor"
                                + " donde hará la operación:");
                ppm.addPropertie("(*Instrucciones2*)", "2. Seleccione la "
                                +"operación a realizar:");
        }

        /**
         * Modifica las propiedades del menú generico en tiempo de ejecución.
         * @param req
         */
        protected void modifyPropertiesPgMenu(HttpServletRequest req){
                DbWebMenu dwm = (DbWebMenu) pageProperties[MENU];
                String query = "SELECT idFactor, nombre, descripcion FROM "
                        +"Factor WHERE idCriterio="
                        + req.getParameter("idCriterio");
                dwm.setQuery(query);
                dwm.setTitleField("idFactor", "");
                dwm.setTitleField("nombre", "Nombre");
                dwm.setTitleField("descripcion", "Descripción");
                dwm.addPropertie("(*OptMenu*)", Utils.createRadioBottonRows(dwm
                        .getFields(), db, query));
        }

        /**
         * Inicializa la página de alta.
         */
        protected void initPageAlta() {
                DbWebAlta ppa;
```



```
                table = "Valor";
                ppa = (DbWebAlta) pageProperties[ALTA];
                ppa.setQuery("SELECT * FROM " + table);
                ppa.addHiddenField("accion", "execalta");
                ppa.addPropertie("(*Titulo*)", "Alta de Valor");
                ppa.unSetVisibleField("idFactor");
                ppa.setTitleField("nombre", "Nombre");
                ppa.setTitleField("valor", "Valor");
                ppa.setTitleField("descripcion", "Descripción");
        }

        /**
         * Modifica las propiedades de la forma de alta en tiempo de ejecución.
         * @param req
         */
        protected void modifyPropertiesPgAlta(HttpServletRequest req){
                DbWebAlta dwa = (DbWebAlta) pageProperties[ALTA];
                dwa.addHiddenField("idFactor", req.getParameter("idFactor"));
        }

        /**
         * Inicializa la página de la forma de baja
         */
        protected void initPageBaja() {
                DbWebBaja dwb = (DbWebBaja) pageProperties[BAJA];
                dwb.addPropertie("(*Titulo*)", "Baja de Valores");
                dwb.addPropertie("(*Instrucciones1*)",
                        "1. Seleccione los valores a dar de baja.");
                dwb.addHiddenField("accion", "execBaja");
        }

        /**
         * Modifica las propiedades de la forma de baja en tiempo de ejecución.
         * @param req
         */
        protected void modifyPropertiesPgBaja(HttpServletRequest req){
                DbWebBaja dwb = (DbWebBaja) pageProperties[BAJA];
                String query = "SELECT idValor, nombre, valor, descripcion "
                        +"FROM Valor WHERE idFactor="
                        + req.getParameter("idFactor");
                dwb.setQuery(query);
                dwb.setTitleField("idValor", "");
                dwb.setTitleField("nombre", "Nombre");
                dwb.setTitleField("descripcion", "Descripción");
                dwb.setTitleField("valor", "Valor");
        }

        /**
         * Inicializa la página de ejecución de baja, especifica el mensaje
         */
        protected void initPageBajaExec() {
                DbWeb dw = pageProperties[BAJAEXEC];
                dw.addPropertie("(*Titulo*)", "Ejecución de baja de valor ");
                dw.addPropertie("(*Mensaje*)", "Baja exitosa.");
        }

        /**
         *   Inicializa la página de la forma de cambio.
         */
        protected void initPageCambio() {
                DbWebCambio dwc = (DbWebCambio) pageProperties[CAMBIO];
                dwc.addPropertie("(*Titulo*)", "Cambio de Valor");
                dwc.addPropertie("(*Instrucciones1*)",
                        "1.Selecciónе el valor a cambiar.");
        }

        /**
         * Modifica las propiedades de la forma de cambio en tiempo de ejecución
         * @param req
```

```
     */
    protected void modifyPropertiesPgCambio(HttpServletRequest req){
            DbWebCambio dwc = (DbWebCambio) pageProperties[CAMBIO];
            String query = "SELECT idValor,nombre, valor, descripcion " +
                    "FROM Valor WHERE "
                    + "idFactor=" + req.getParameter("idFactor");
            dwc.setQuery(query);
            dwc.setTitleField("idValor", "");
            dwc.setTitleField("nombre", "Nombre");
            dwc.setTitleField("descripcion", "Descripción");
            dwc.setTitleField("valor", "Valor");
            dwc.addHiddenField("accion", "execCambio");
    }

    /**
     * Inicializa la página de captura del cambio
     */
    protected void initPageCambioExec() {
            DbWebExecCambio dwce;
            dwce = (DbWebExecCambio) pageProperties[CAMBIOEXEC];
            dwce.setQuery("SELECT * FROM " + table);
            dwce.addHiddenField("accion", "execCambioUpdate");
            dwce.addPropertie("(*Titulo*)", "Cambio de factor");
            dwce.addPropertie("(*Instrucciones1*)", "");
    }

    /**
     * Modifica las propiedades de la forma de ejecución del cambio en
     * tiempo de ejecución.
     * @param req
     */
    protected void modifyPropertiesPgCambioExec(HttpServletRequest req){
            SortedHashtable data;
            String key, val;
            DbWebExecCambio dwce =
                    (DbWebExecCambio) pageProperties[CAMBIOEXEC];
            data = WebUtils.getFieldsAndValues(req);
            data.remove("accion");
            key = Utils.hashTableGetKeyAt(data, 0);
            val = Utils.hashTableGetValueAt(data, 0);
            dwce.setSearchingValues(true);
            dwce.addHiddenField("idValor", req.getParameter("idValor"));
            dwce.setQuery("SELECT * FROM " + table
                            + " WHERE " + key + "=" + val);
            dwce.unSetVisibleField("idFactor");
            dwce.setTitleField("nombre", "Nombre ");
            dwce.setTitleField("descripcion", "Descripción ");
            dwce.setTitleField("valor", "Valor ");
    }

    /**
     * Inicializa la página de ejecución de alta.
     */
    protected void initPageCambioExecUpdate() {
            DbWeb dw = pageProperties[CAMBIOEXECUPDATE];
            dw.addPropertie("(*Titulo*)", "Ejecución de Cambio de Valor");
            dw.addPropertie("(*Mensaje*)", "Cambio exitoso.");
    }

    /**
     * Modifica las propiedades de la forma de ejecución de actualización
     * del cambio tiempo de ejecución.
     * @param req
     */
    protected void modifyPropertiesPgExecCambioUpdate(
                    HttpServletRequest req) throws BDException{
            String val;
            SortedHashtable data;
            val = req.getParameter("idValor");
```



```
            data = WebUtils.getFieldsAndValues(req);
            data.remove("accion");
            data.remove("idValor");
            data = WebUtils.formatDataforDB("SELECT * FROM " + table, data);
            db.change(table, data, "idValor=" + val);
        }
}
```

```
package itesm.seguridad;
/*
 * @author      Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import javax.servlet.http.HttpServletRequest;
import itesm.utilsdsc.*;
/**
* Esta clase es la interfase para base de datos de los equipos a evaluar.
*/
public class EsquemaHost extends Esquema {
        private static final long serialVersionUID = -4523635911509992615L;
        /**
         * Constructor.
         * Especifca el servlet a invocar.
         */
        public EsquemaHost() {
                super();
                setURLtoAllPages("/sderi/EsquemaHost");
                table = "Host";
        }

        /**
         * Especifica el menu y el titulo del mismo.
         */
        protected void initPageMenu() {
                DbWebMenu ppm;
                ppm = (DbWebMenu) pageProperties[MENU];
                SortedHashtable menu = ppm.getMenu();
                menu.put("Alta", "alta");
                menu.put("Baja", "baja");
                menu.put("Cambio", "cambio");
                ppm.setTitle("Administración de Equipos");
                ppm.addPropertie("(*Menu*)",
                        Utils.createOptionButtonMenuRows("accion",menu));
                ppm.addPropertie("(*Instrucciones1*)",
                                "1. Seleccione la operación a realizar:");
                ppm.addPropertie("(*OptMenu*)", "");
                ppm.addPropertie("(*Instrucciones2*)", "");
        }

        /**
         * Inicializa la página de alta.
         */
        protected void initPageAlta() {
                DbWebAlta ppa;
                table = "Host";
                ppa = (DbWebAlta) pageProperties[ALTA];
                ppa.setQuery("SELECT * FROM " + table);
                ppa.addHiddenField("accion", "execalta");
                ppa.addPropertie("(*Titulo*)", "Alta de Equipo");
                ppa.setTitleField("ip", "Dirección IP");
                ppa.setTitleField("nombre", "Hostname");
                ppa.setTitleField("descripcion", "Descripción");
        }

        /**
         * protected void initPageBaja()
         *
         * Inicializa la página de la forma de baja
         */
        protected void initPageBaja() {
                DbWebBaja dwb = (DbWebBaja) pageProperties[BAJA];
                dwb.addPropertie("(*Titulo*)", "Baja de Equipos");
                dwb.addPropertie("(*Instrucciones1*)",
                        "1. Seleccione los equipos a dar de baja.");
        }
```



```
        /**
         * Inicializa la página de ejecución de baja, especifica el mensaje
         */
        protected void initPageBajaExec() {
                DbWeb dw = pageProperties[BAJAEXEC];
                dw.addPropertie("(*Titulo*)", "Ejecución de baja de equipo ");
                dw.addPropertie("(*Mensaje*)", "Baja exitosa.");
        }

        /**
         * Modifica las propiedades de la forma de baja en tiempo de
         * ejecución.
         * @param req
         */
        protected void modifyPropertiesPgBaja(HttpServletRequest req){
                DbWebBaja dwb = (DbWebBaja) pageProperties[BAJA];
                String query = "SELECT * FROM Host";
                dwb.setQuery(query);

                dwb.setTitleField("ip", "Dirección IP");
                dwb.setTitleField("nombre", "Hostname");
                dwb.setTitleField("descripcion", "Descripción");
                dwb.addHiddenField("accion", "execBaja");
        }

        /**
         * Inicializa la página de la forma de cambio.
         */
        protected void initPageCambio() {
                DbWebCambio dwc = (DbWebCambio) pageProperties[CAMBIO];
                dwc.addPropertie("(*Titulo*)", "Cambio de Equipo");
                dwc.addPropertie("(*Instrucciones1*)",
                        "1.Selecciónе el equipo a cambiar.");
        }

        /**
         * Modifica las propiedades de la forma de cambio en tiempo de ejecución
         * @param req
         */
        protected void modifyPropertiesPgCambio(HttpServletRequest req){
                DbWebCambio dwc = (DbWebCambio) pageProperties[CAMBIO];
                String query = "SELECT ip, ip ip2, nombre,descripcion "+
                        "FROM Host ";
                dwc.setQuery(query);
                dwc.setTitleField("ip", "");
                dwc.setTitleField("ip2", "Dirección IP");
                dwc.setTitleField("nombre", "Hostname");
                dwc.setTitleField("descripcion", "Descripción");
                dwc.addHiddenField("accion", "execCambio");
        }

        /**
         * protected void initPageCambioExec()
         *
         * Inicializa la página de captura del cambio
         */
        protected void initPageCambioExec() {
                DbWebExecCambio dwce;
                dwce = (DbWebExecCambio) pageProperties[CAMBIOEXEC];
                dwce.setQuery("SELECT * FROM " + table);
                dwce.addHiddenField("accion", "execCambioUpdate");
                dwce.addPropertie("(*Titulo*)", "Cambio de Equipo");
                dwce.addPropertie("(*Instrucciones1*)", "");
        }

        /**
         * Modifica las propiedades de la forma de ejecución del cambio en
         * tiempo de ejecución.
```

```
         * @param req
         */
        protected void modifyPropertiesPgCambioExec(HttpServletRequest req){
                SortedHashtable data;
                String key, val;
                DbWebExecCambio dwce = (DbWebExecCambio)
                        pageProperties[CAMBIOEXEC];
                data = WebUtils.getFieldsAndValues(req);
                data.remove("accion");
                key = Utils.hashTableGetKeyAt(data, 0);
                val = Utils.hashTableGetValueAt(data, 0);
                dwce.setSearchingValues(true);
                dwce.addHiddenField("ip", req.getParameter("ip"));
                dwce.setQuery("SELECT * FROM Host WHERE " + key +
                                "='" + val+"'");
                dwce.setTitleField("ip", "Dirección IP ");
                dwce.setTitleField("nombre", "Hostname ");
                dwce.setTitleField("descripcion", "Descripción ");
        }

        /**
         * Inicializa la página de ejecución de alta.
         */
        protected void initPageCambioExecUpdate() {
                DbWeb dw = pageProperties[CAMBIOEXECUPDATE];
                dw.addPropertie("(*Titulo*)", "Ejecución de Cambio de Equipo");
                dw.addPropertie("(*Mensaje*)", "Cambio exitoso.");
        }

        /**
         * Modifica las propiedades de la forma de ejecución de actualización
         * del cambio tiempo de ejecución.
         * @param req
         */
        protected void modifyPropertiesPgExecCambioUpdate(
                        HttpServletRequest req)throws BDException{
                String val;
                SortedHashtable data;
                val = req.getParameter("ip");
                data = WebUtils.getFieldsAndValues(req);
                data.remove("accion");
                data.remove("ip");
                data = WebUtils.formatDataforDB("SELECT * FROM " + table, data);
                db.change(table, data, "ip='" + val+ "'");
        }
}
```

```
package itesm.seguridad;
/*
 * @author      Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import java.io.IOException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import itesm.utilsdsc.*;
/**
* Esta clase es la interfase para base de datos de los equipos a evaluar.
*/
public class EsquemaMenu extends Esquema {
        private static final long serialVersionUID = -4523635911509992615L;
        /**
         * Constructor.
         * Especifca el servlet a invocar.
         */
        public EsquemaMenu() {
                super();
                setURLtoAllPages("/sderi/EsquemaMenu");
                table = "Host";
        }

        /**
         * Especifica el menu y el titulo del mismo.
         */
        protected void initPageMenu() {
                DbWebMenu ppm;
                ppm = (DbWebMenu) pageProperties[MENU];
                SortedHashtable menu = ppm.getMenu();
                menu.put("Administración de equipos", "EsquemaHost");
                menu.put("Administración de la definición del nivel de Riesgo",
                                "EsquemaCriterio");
                menu.put("Administración de evaluaciones", "EsquemaEval");
                menu.put("Listado del nivel de riesgo de los equipos",
                                "LstEquiposEvaluados");
                ppm.setTitle("");
                ppm.addPropertie("(*Menu*)",
                        Utils.createOptionButtonMenuRows("newServlet",
                        menu));
                ppm.addPropertie("(*Instrucciones1*)",
                        "1. Seleccione la operación a realizar:");
                ppm.addPropertie("(*Instrucciones2*)"
                         Utils.createHTMLHiddenField("accion", "subMenu"));
                ppm.addPropertie("(*OptMenu*)", "");
        }
        /**
         * Redirecciona el servlet a  invocar con sus repectivos parametros
         * @param req
         * @param res
         */
        protected void pgSubMenu(HttpServletRequest req,HttpServletResponse res)
                throws IOException {
                //DbWebSubMenu dw=(DbWebSubMenu)pageProperties[SUBMENU];
                HttpURL url= new HttpURL(req);
                String newServlet = req.getParameter("newServlet");
                url.setFile("/"+ newServlet );
                res.sendRedirect(url+"");
        }
}
```

```
package itesm.seguridad;
/*
 * @author      Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import javax.servlet.http.HttpServletRequest;
import itesm.utilsdsc.*;
/**
* Esta clase es la interfase para base de datos de los querys que permiten
* automatizar factores.
*/

public class EsquemaQueryAutomatico extends Esquema {
        private static final long serialVersionUID = -4523635911509992615L;
        /**
         * Constructor.
         * Especifca el servlet a invocar.
         */
        public EsquemaQueryAutomatico() {
                super();
                setURLtoAllPages("/sderi-EsquemaQueryAutomatico");
                table = "QueryAutomatico";
        }

        /**
         * Especifica el menu y el titulo del mismo.
         */
        protected void initPageMenu() {
                DbWebMenu ppm;
                ppm = (DbWebMenu) pageProperties[MENU];
                SortedHashtable menu = ppm.getMenu();
                menu.put("Alta", "alta");
                menu.put("Baja", "baja");
                menu.put("Cambio", "cambio");
                menu.put ("Administración de conexiones de BD", "subMenu");
                ppm.setTitle("Administración de querys para la automatización"
                        +" de factores");
                ppm.addPropertie("(*Menu*)",
                        Utils.createOptionButtonMenuRows("accion",menu));
                ppm.addPropertie("(*Instrucciones1*)",
                        "1. Seleccione la operación a realizar:");
                ppm.addPropertie("(*OptMenu*)", "");
                ppm.addPropertie("(*Instrucciones2*)", "");
        }

        /**
         * Inicializa la página de alta.
         */
        protected void initPageAlta() {
                DbWebAlta ppa;
                table = "QueryAutomatico";
                ppa = (DbWebAlta) pageProperties[ALTA];
                ppa.setQuery("SELECT nombre1, descripcion1, query, maxQuery,"
                                +"cond, campoIP, idConexionBD, stringIP FROM "
                                + table);
                ppa.addHiddenField("accion", "execalta");
                ppa.addPropertie("(*Titulo*)",
                        "Alta de querys para la automatización de factores");
                ppa.setTitleField("nombre1", "Nombre");
                ppa.setTitleField("descripcion1", "Descripción");
                ppa.setTitleField("query",
                        "SELECT para obtener el valor del factor");
                ppa.setTitleField("maxQuery", "SELECT para obtener el valor"
                        +" maximo evaluado del factor");
                ppa.setTitleField("cond", "Condicional (Clausula WHERE) ");
                ppa.setTitleField("campoIP", "Campo llave (dirección IP)");
                ppa.setTitleField("idConexionBD", "Conexion de base datos");
```



```
                ppa.setTitleField("stringIP",
                        "La dirección IP es de tipo string ?");
                ppa.setSelectionableField("idConexionBD",
                        "SELECT idConexionBD,nombre FROM ConexionBD");
        }

        /**
         * Inicializa la página de la forma de baja
         */
        protected void initPageBaja() {
                DbWebBaja dwb = (DbWebBaja) pageProperties[BAJA];
                dwb.addPropertie("(*Titulo*)",
                        "Baja de querys para la automatización de factores");
                dwb.addPropertie("(*Instrucciones1*)",
                        "1. Seleccione las querys a dar de baja.");
        }

        /**
         * Inicializa la página de ejecución de baja, especifica el mensaje
         */
        protected void initPageBajaExec() {
                DbWeb dw = pageProperties[BAJAEXEC];
                dw.addPropertie("(*Titulo*)",
                        "Ejecución de baja de querys para la automatización "
                        +" de factores");
                dw.addPropertie("(*Mensaje*)", "Baja exitosa.");
        }

        /**
         * Inidica el servlet a invocar.
         */
        protected void initPageSubMenu() {
                DbWebSubMenu dw = (DbWebSubMenu) pageProperties[SUBMENU];
                dw.setNewServlet("EsquemaConexionBD");
        }

        /**
         * Modifica las propiedades de la forma de baja en tiempo de
         * ejecución.
         * @param req
         */
        protected void modifyPropertiesPgBaja(HttpServletRequest req){
                DbWebBaja dwb = (DbWebBaja) pageProperties[BAJA];
                String query = "SELECT idQueryAutomatico, nombre1, "
                        +"descripcion1 FROM QueryAutomatico";
                dwb.setQuery(query);
                dwb.setTitleField("idQueryAutomatico", "");
                dwb.setTitleField("nombre1", "Nombre");
                dwb.setTitleField("descripcion1", "Descripción");
                dwb.addHiddenField("accion", "execBaja");
        }

        /**
         *  Inicializa la página de la forma de cambio.
         */
        protected void initPageCambio() {
                DbWebCambio dwc = (DbWebCambio) pageProperties[CAMBIO];
                dwc.addPropertie("(*Titulo*)",
                        "Cambio de querys para la automatización de factores");
                dwc.addPropertie("(*Instrucciones1*)",
                        "1.Selecciónе query a cambiar.");
        }

        /**
         * Modifica las propiedades de la forma de cambio en tiempo de ejecución
         * @param req
         */
        protected void modifyPropertiesPgCambio(HttpServletRequest req){
                DbWebCambio dwc = (DbWebCambio) pageProperties[CAMBIO];
```

```
                String query = "SELECT idQueryAutomatico, nombre1, "
                        +"descripcion1 FROM QueryAutomatico ";
                dwc.setQuery(query);
                dwc.setTitleField("idQueryAutomatico", "");
                dwc.setTitleField("nombre1", "nombre1");
                dwc.setTitleField("descripcion1", "Descripción");
                dwc.addHiddenField("accion", "execCambio");
        }

        /**
         * Inicializa la página de captura del cambio
         */
        protected void initPageCambioExec() {
                DbWebExecCambio dwce;
                dwce = (DbWebExecCambio) pageProperties[CAMBIOEXEC];
                dwce.setQuery("SELECT * FROM QueryAutomatico" );
                dwce.addHiddenField("accion", "execCambioUpdate");
                dwce.addPropertie("(*Titulo*)",
                        "Cambio de querys para la automatización de factores");
                dwce.addPropertie("(*Instrucciones1*)", "");
        }

        /**
         * Modifica las propiedades de la forma de ejecución del cambio
         * en tiempo de ejecución.
         * @param req
         */
        protected void modifyPropertiesPgCambioExec(HttpServletRequest req){
                SortedHashtable data;
                String key, val;
                DbWebExecCambio dwce =
                        (DbWebExecCambio) pageProperties[CAMBIOEXEC];
                data = WebUtils.getFieldsAndValues(req);
                data.remove("accion");
                key = Utils.hashTableGetKeyAt(data, 0);
                val = Utils.hashTableGetValueAt(data, 0);
                dwce.setSearchingValues(true);
                dwce.addHiddenField("idQueryAutomatico",
                        req.getParameter("idQueryAutomatico"));
                dwce.setQuery("SELECT * FROM QueryAutomatico WHERE "
                        + key + "=" + val+"");
                dwce.setTitleField("nombre1", "Nombre");
                dwce.setTitleField("descripcion1", "Descripción");
                dwce.setTitleField("query",
                        "SELECT para obtener el valor del factor");
                dwce.setTitleField("maxQuery", "SELECT para obtener el valor"
                        +" maximo evaluado del factor");
                dwce.setTitleField("cond", "Condicional (Clausula WHERE) ");
                dwce.setTitleField("campoIP", "Campo llave (dirección IP)");
                dwce.setTitleField("idConexionBD", "Conexion de base datos");
                dwce.setTitleField("stringIP",
                                "La dirección IP es de tipo string ?");
        }

        /**
         * Inicializa la página de ejecución de alta.
         */
        protected void initPageCambioExecUpdate() {
                DbWeb dw = pageProperties[CAMBIOEXECUPDATE];
                dw.addPropertie("(*Titulo*)","Ejecución de Cambio de query "
                        +" para la automatización de factores.");
                dw.addPropertie("(*Mensaje*)", "Cambio exitoso.");
        }


        /**
         * Modifica las propiedades de la forma de ejecución de actualización
         * del cambio tiempo de ejecución.
         * @param req
```



```
         */
        protected void modifyPropertiesPgExecCambioUpdate(
                        HttpServletRequest req) throws BDException{
                String val;
                SortedHashtable data;
                val = req.getParameter("idQueryAutomatico");
                data = WebUtils.getFieldsAndValues(req);
                data.remove("accion");
                data.remove("idQueryAutomatico");
                data = WebUtils.formatDataforDB("SELECT * FROM " + table, data);
                db.change(table, data, "idQueryAutomatico=" + val+ "");
        }
}
```

```
package itesm.seguridad;
/*
 * @author      Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import java.io.IOException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import itesm.utilsdsc.*;
/**
* Esta clase es la interfase para base de datos de los equipos a evaluar.
*/
public class EsquemaSubMenuFactor extends Esquema {
        private static final long serialVersionUID = -4523635911509992615L;
        /**
         * Constructor.
         * Especifca el servlet a invocar.
         */
        public EsquemaSubMenuFactor() {
                super();
                setURLtoAllPages("/sderi/EsquemaSubMenuFactor");
        }

        /**
         * Modifica las propiedades del menú generico en tiempo de ejecución.
         * @param req
         */
        protected void modifyPropertiesPgMenu(HttpServletRequest req){
                DbWebMenu dwm = (DbWebMenu) pageProperties[MENU];
                String query = "SELECT idFactor, nombre, descripcion "
                        + "FROM Factor "
                        + "WHERE idCriterio=" + req.getParameter("idCriterio");
                dwm.setQuery(query);
                dwm.setTitleField("idFactor", "");
                dwm.setTitleField("nombre", "Nombre");
                dwm.setTitleField("descripcion", "Descripción");
                dwm.addPropertie("(*OptMenu*)", Utils.createRadioBottonRows(dwm
                        .getFields(), db, query));
        }

        /**
         * Especifica el menu y el titulo del mismo.
         */
        protected void initPageMenu() {
                DbWebMenu ppm;
                ppm = (DbWebMenu) pageProperties[MENU];
                SortedHashtable menu = ppm.getMenu();
                menu.put("Alta", "alta");
                menu.put("Baja ", "baja");
                menu.put("Cambio", "cambio");
                ppm.setTitle("Adminsitracion de valores posibles y querys "
                                +" para factores automaticos");
                ppm.addPropertie("(*Menu*)",
                        Utils.createOptionButtonMenuRows("newServlet",menu));
                ppm.addPropertie("(*Instrucciones1*)",
                        "1. Seleccione el factor donde hará la operación: "
                        + "a realizar:");
                ppm.addPropertie("(*Instrucciones2*)",
                        "2. Seleccione la operación a realizar:");
        }
        /**
         * Redirecciona el servlet a  invocar con sus repectivos parametros
         *
         * @param req
         * @param res
         */
        protected void pgSubMenu(HttpServletRequest req,HttpServletResponse res)
```



```
                throws IOException {
                //DbWebSubMenu dw=(DbWebSubMenu)pageProperties[SUBMENU];
                HttpURL url= new HttpURL(req);
                String newServlet = req.getParameter("newServlet");
                url.setFile("/"+ newServlet );
                res.sendRedirect(url+"");
        }
}
```

```
package itesm.seguridad;
/*
 * @author      Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import javax.servlet.http.*;
import itesm.utilsdsc.*;
/**
 * Esta clase es la interfase para base de datos de valores POSIBLES, a fin de
 * que estos sean tomados a la hora de querer evaluar un host especifico para
 * determinar su nivel de seguridad.
 *
 * Un factor puede o no tener un conjunto de valores posibles.
 */
public class EsquemaValor extends Esquema {
        private static final long serialVersionUID = -6978002171356148938L;
        /**
         * Constructor.
         * Especifca el servlet a invocar.
         */
        public EsquemaValor() {
                super();
                setURLtoAllPages("/sderi/EsquemaValor");
                table = "Valor";
        }

        /**
         * Especifica el menu y el titulo del mismo.
         */
        protected void initPageMenu() {
                DbWebMenu ppm;
                ppm = (DbWebMenu) pageProperties[MENU];
                SortedHashtable menu = ppm.getMenu();
                menu.put("Alta", "alta");
                menu.put("Baja", "baja");
                menu.put("Cambio", "cambio");
                ppm.setTitle("Administración de Valores");
                ppm.addPropertie("(*Menu*)",
                                Utils.createOptionButtonMenuRows("accion",
                                        menu));
                ppm.addPropertie("(*Instrucciones1*)",
                                "1. Seleccione el factor donde hará la operación:");
                ppm.addPropertie("(*Instrucciones2*)",
                                "2. Seleccione la operación a realizar:");
        }

        /**
         * Modifica las propiedades del menú generico en tiempo de ejecución.
         * @param req
         */
        protected void modifyPropertiesPgMenu(HttpServletRequest req){
                DbWebMenu dwm = (DbWebMenu) pageProperties[MENU];
                String query = "SELECT idFactor, nombre, descripcion "
                        + "FROM Factor "
                        + "WHERE idCriterio=" + req.getParameter("idCriterio");
                dwm.setQuery(query);
                dwm.setTitleField("idFactor", "");
                dwm.setTitleField("nombre", "Nombre");
                dwm.setTitleField("descripcion", "Descripción");
                dwm.addPropertie("(*OptMenu*)", Utils.createRadioBottonRows(dwm
                        .getFields(), db, query));
        }

        /**
         * Inicializa la página de alta.
         */
        protected void initPageAlta() {
```



```
                DbWebAlta ppa;
                table = "Valor";
                ppa = (DbWebAlta) pageProperties[ALTA];
                ppa.setQuery("SELECT * FROM " + table);
                ppa.addHiddenField("accion", "execalta");
                ppa.addPropertie("(*Titulo*)", "Alta de Valor");
                ppa.unSetVisibleField("idFactor");
                ppa.setTitleField("nombre", "Nombre");
                ppa.setTitleField("valor", "Valor");
                ppa.setTitleField("descripcion", "Descripción");
        }

        /**
         * Modifica las propiedades de la forma de alta en tiempo de ejecución.
         * @param req
         */
        protected void modifyPropertiesPgAlta(HttpServletRequest req){
                DbWebAlta dwa = (DbWebAlta) pageProperties[ALTA];
                dwa.addHiddenField("idFactor", req.getParameter("idFactor"));

        }

        /**
         * Inicializa la página de la forma de baja
         */
        protected void initPageBaja() {
                DbWebBaja dwb = (DbWebBaja) pageProperties[BAJA];
                dwb.addPropertie("(*Titulo*)", "Baja de Valores");
                dwb.addPropertie("(*Instrucciones1*)",
                        "1. Seleccione los valores a dar de baja.");
                dwb.addHiddenField("accion", "execBaja");
        }

        /**
         * Modifica las propiedades de la forma de baja en tiempo de ejecución.
         * @param req
         */
        protected void modifyPropertiesPgBaja(HttpServletRequest req){
                DbWebBaja dwb = (DbWebBaja) pageProperties[BAJA];
                String query = "SELECT idValor, nombre, valor, descripcion "
                        + " FROM Valor"
                        + " WHERE idFactor=" + req.getParameter("idFactor");
                dwb.setQuery(query);
                dwb.setTitleField("idValor", "");
                dwb.setTitleField("nombre", "Nombre");
                dwb.setTitleField("descripcion", "Descripción");
                dwb.setTitleField("valor", "Valor");
        }

        /**
         * Inicializa la página de ejecución de baja, especifica el mensaje
         */
        protected void initPageBajaExec() {
                DbWeb dw = pageProperties[BAJAEXEC];
                dw.addPropertie("(*Titulo*)", "Ejecución de baja de valor ");
                dw.addPropertie("(*Mensaje*)", "Baja exitosa.");
        }

        /**
         * Inicializa la página de la forma de cambio.
         */
        protected void initPageCambio() {
                DbWebCambio dwc = (DbWebCambio) pageProperties[CAMBIO];
                dwc.addPropertie("(*Titulo*)", "Cambio de Valor");
                dwc.addPropertie("(*Instrucciones1*)",
                        "1.Selecciónе el valor a cambiar.");
        }

        /**
```

```
        * Modifica las propiedades de la forma de cambio en tiempo de ejecución
        * @param req
        */
       protected void modifyPropertiesPgCambio(HttpServletRequest req){
               DbWebCambio dwc = (DbWebCambio) pageProperties[CAMBIO];
               String query = "SELECT idValor,nombre, valor, descripcion " +
                       "FROM Valor WHERE "
                       + "idFactor=" + req.getParameter("idFactor");
               dwc.setQuery(query);
               dwc.setTitleField("idValor", "");
               dwc.setTitleField("nombre", "Nombre");
               dwc.setTitleField("descripcion", "Descripción");
               dwc.setTitleField("valor", "Valor");
               dwc.addHiddenField("accion", "execCambio");
       }

       /**
        * Inicializa la página de captura del cambio
        */
       protected void initPageCambioExec() {
               DbWebExecCambio dwce;
               dwce = (DbWebExecCambio) pageProperties[CAMBIOEXEC];
               dwce.setQuery("SELECT * FROM " + table);
               dwce.addHiddenField("accion", "execCambioUpdate");
               dwce.addPropertie("(*Titulo*)", "Cambio de factor");
               dwce.addPropertie("(*Instrucciones1*)", "");
       }

       /**
        * Modifica las propiedades de la forma de ejecución del cambio en
        * tiempo de ejecución.
        * @param req
        */
       protected void modifyPropertiesPgCambioExec(HttpServletRequest req){
               SortedHashtable data;
               String key, val;
               DbWebExecCambio dwce =
                       (DbWebExecCambio) pageProperties[CAMBIOEXEC];
               data = WebUtils.getFieldsAndValues(req);
               data.remove("accion");
               key = Utils.hashTableGetKeyAt(data, 0);
               val = Utils.hashTableGetValueAt(data, 0);
               dwce.setSearchingValues(true);
               dwce.addHiddenField("idValor", req.getParameter("idValor"));
               dwce.setQuery("SELECT * FROM " + table
                               + " WHERE " + key + "=" + val);
               dwce.unSetVisibleField("idFactor");
               dwce.setTitleField("nombre", "Nombre ");
               dwce.setTitleField("descripcion", "Descripción ");
               dwce.setTitleField("valor", "Valor ");
       }

       /**
        * Inicializa la página de ejecución de alta.
        */
       protected void initPageCambioExecUpdate() {
               DbWeb dw = pageProperties[CAMBIOEXECUPDATE];
               dw.addPropertie("(*Titulo*)", "Ejecución de Cambio de Valor");
               dw.addPropertie("(*Mensaje*)", "Cambio exitoso.");
       }


       /**
        * Modifica las propiedades de la forma de ejecución de actualización
        * del cambio tiempo de ejecución.
        * @param req
        */
       protected void modifyPropertiesPgExecCambioUpdate(
                       HttpServletRequest req)
```



```
                       throws BDException{
                       String val;
                       SortedHashtable data;
                       val = req.getParameter("idValor");
                       data = WebUtils.getFieldsAndValues(req);
                       data.remove("accion");
                       data.remove("idValor");
                       data = WebUtils.formatDataforDB("SELECT * FROM " + table, data);
                       db.change(table, data, "idValor = " + val);
               }
}
```

```
package itesm.seguridad;
/*
* Created on 23/10/2005
* @author Ing. M. Damián Guerra Farías.
* ITESM-CEM
* Tesis:
* Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
*/
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
/**
* Esta clase representa el calculo de la evaluacion del nivel de riesgo de un
* equipo en especifico.
* Básicamente manda evaluar cada uno de los criterios y al final hace una suma
* ponderada de estos.
*/
public class EvaluacionEquipo{
	CriterioEvaluado criterios[];
	double evaluacion=0;
	String ip;
	DB db;
	Host host;
	/**
	 * Único constructor que inicializa las variables locales y manda
	 * evaluar el niviel de riesgo del equipo.
	 * @param ip //Equipo a evaluar
	 * @param db //Conexion de la base
	 */
	EvaluacionEquipo(String ip, DB db){
		this.ip=ip;
		this.db=db;
		host = Host.searchHost(ip, db);
		setEvaluacionEquipo();
	}

	/**
	 * Cuenta la cantidad de criterios que haya definidos en la base
	 * de datos.
	 * @return
	 */
	public int getNumCriterios(){
		int numCriterios=0;
		String query="SELECT * FROM Criterio ";
		try {
			numCriterios = db.contRows(query);
		} catch (BDException e) {
			System.err.println("Query="+query);
			e.printStackTrace();
		}
		return numCriterios;
	}

	/**
	 * Manda evaluar cada uno de los criterios que se tengan definidos en la
	 * base de datos y los almacena en un arreglo llamado criterios
	 */
	private void getCriteriosEvaluados(){
		int numCriterios= getNumCriterios();
		Criterio c;
		criterios = new CriterioEvaluado[numCriterios];
		String query="SELECT * FROM Criterio";
		Statement stmt;
		ResultSet rs;
		Connection con;
		try {
			con = db.newConection();
			stmt = con.createStatement();
```



```
			rs = stmt.executeQuery(query);
			int i=0;
			while(rs.next()){
				c= Criterio.searchCriterio(
					rs.getInt("idCriterio")+"", db);
				criterios[i]=new CriterioEvaluado(ip,c,db);
				i++;
			}
			rs.close();
			stmt.close();
			con.close();
		} catch (SQLException e) {
			System.err.println("Query="+query);
			e.printStackTrace();
		} catch (BDException e) {
			System.err.println("Query="+query);
			e.printStackTrace();
		}
	}

	/**
	 * Manda evaluar a los criterios, posteriormente hace la suma ponderada
	 * de los mismos para obtener el nivel de riesgo del equipo.
	 * El resultado final lo almacena en la variable "evaluacion"
	 */
	public void setEvaluacionEquipo(){
		evaluacion=0;
		getCriteriosEvaluados();
		for (int i=0; i< criterios.length; i++){
			evaluacion+= criterios[i].evaluacion *
				criterios[i].criterio.ponderacion;
		}

	}
}
```

```
package itesm.seguridad;
/*
 * Created on 19/05/2005
 * @author      Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
/**
 * Esta clase representa un registro de la tabla Factor.
 */
public class Factor {
        public static DB db;
        int idFactor;
        int idCriterio;
        String nombre;
        String descripcion;
        int tipoValor;
        double ponderacion;
        int idQueryTipoAutomatico;

        /**
         * Busca un factor en la base de datos a partir de su id y regresa un
         * objeto de tipo Factor con todos los datos del mismo.
         * @param idFactor
         * @param db
         * @return
         */
        public static Factor searchFactor (String idFactor, DB db){
                Statement stmt = db.getStmt();
                ResultSet rs;
                Factor f = new Factor();
                String query = "SELECT * FROM Factor WHERE idFactor ="+idFactor;
                try {
                        rs = stmt.executeQuery(query);
                        if (rs.next()) {
                                f.idFactor = rs.getInt("idFactor");
                                f.idCriterio= rs.getInt("idCriterio");
                                f.nombre=rs.getString("nombre");
                                f.descripcion =rs.getString("descripcion");
                                f.tipoValor=rs.getInt("tipoValor");
                                f.idQueryTipoAutomatico=
                                        rs.getInt("idQueryTipoAutomatico");
                                f.ponderacion= rs.getDouble("ponderacion");
                        } else {
                                f = null;
                        }
                } catch (SQLException e) {
                        System.err.println("Query =" + query);
                        System.err.println("SQLException: " + e.getMessage());
                        e.printStackTrace();
                }
                return f;
        }
}
```

```
package itesm.seguridad;
/*
 * Created on 23/10/2005
 * @author      Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
/**
 * Esta clase representa la evaluacion de un factor en especifico.
 * Basicamente a partir de un factor y un equipo, determina el tipo de factor
 * a partir del cual hace las operaciones necesarias para obtener la evaluación
 * del factor para el equipo.
 */
public class FactorEvaluado {
        //Datos del factor Evaluado
        Factor factor;
        //Equipo a evaluar
        String ip;
        //Valor de la evaluacion
        double evaluacion;
        //Conexion hacia la  base de datos.
        DB db;
        //Valor capturado
        String  valorCapturado="";

        /**
         * Único constructor que inicializa las varibles locales y manda
         * calcular la evaluacion del factor, asi como obtener el valor
         * capturado por el usuario.
         */
        public FactorEvaluado(String ip, Factor f, DB db) {
                this.ip = ip;
                factor = f;
                this.db = db;
                evaluacion = getEvaluacionFactor();
                valorCapturado = getValorCapturado();
        }

        /**
         * Obtiene la evaluación del factor (Verifica el tipo de factor y
         * luego hace los calculos necesarios para obtener su evaluación)
         * @return
         */
        double getEvaluacionFactor() {
                double res = 0;
                double aux, aux1;
                switch (factor.tipoValor) {
                        case TipoValor.AUTOMATICO:
                                aux = Utils.getValueFromQueryAutomatico(
                                                factor, db, ip);
                                aux1= Utils.getMaxValueFromQueryAutomatico(
                                                factor,db);
                                if (aux==0)
                                        res=0;
                                else
                                        res = aux/aux1;
                                break;
                        case TipoValor.NUMERICO_SELECCIONABLE:
                        case TipoValor.VERDADERO_FALSO:
                        case TipoValor.NUMERICO_DIRECTO:
                                res = getValor();
                                break;
                        case TipoValor.CONTADOR_NUMERICO:
                                aux =getValor();
```



```
                                aux1 = getSumValor();
                                if (aux==0)
                                        res=0;
                                else
                                        res = aux/aux1;
                                break;
                        case TipoValor.FUNCION_NUMERICA:
                                res = Utils.getEvaluatedValueFromFunctionFactor(
                                                factor, db, ip);
                                break;
                        case TipoValor.PROP_MAX:
                                aux =getValor();
                                aux1 = getMaxValor();
                                if (aux==0)
                                        res=0;
                                else
                                        res = aux/aux1;
                                break;
                }
                return res;
        }

        /**
         * Obtiene de la tabla Evaluacion el valor del factor evaluado
         * actualmente.
         * @return  la evaluacion del factor.
         */
        double getValor() {
                double res = 0;
                Connection con = null;
                Statement stmt = null;
                ResultSet rs = null;
                String query = "SELECT * FROM Evaluacion " + "WHERE idFactor="
                                + factor.idFactor + " AND ip='" + ip + "'";
                try {
                        con = db.newConection();
                        stmt = con.createStatement();
                        rs = stmt.executeQuery(query);
                        while (rs.next()) {
                                res = res + rs.getDouble("valor");
                        }
                        rs.close();
                        stmt.close();
                        con.close();
                } catch (BDException e) {
                        System.err.println("Query =" + query);
                        e.printStackTrace();
                } catch (SQLException e) {
                        System.err.println("Query=" + query);
                        e.printStackTrace();
                }
                return res;
        }

        /**
         * Obtiene el valor maximo para un factor en especifico, a fin de poder
         * tener una relacion del equipo actual contra los demas.
         * @return
         */
        double getMaxValor() {
                double res = 0;
                Connection con = null;
                Statement stmt = null;
                ResultSet rs = null;
                String query = "SELECT *, MAX(valor) maxvalor FROM Evaluacion "
                                +"WHERE idFactor="
                                + factor.idFactor
                                +" GROUP BY idFactor";
                try {
```

```
                con = db.newConection();
                stmt = con.createStatement();
                rs = stmt.executeQuery(query);
                while (rs.next()) {
                        res = res + rs.getDouble("maxvalor");
                }
                rs.close();
                stmt.close();
                con.close();
        } catch (BDException e) {
                System.err.println("Query=" + query);
                e.printStackTrace();
        } catch (SQLException e) {
                System.err.println("Query=" + query);
                e.printStackTrace();
        }
        return res;
}

/**
 * Obtiene la suma de un tipo de valor
 * @return
 */
double getSumValor() {
        double res = 0;
        Connection con = null;
        Statement stmt = null;
        ResultSet rs = null;
        String query = "SELECT *, SUM(valor) sumvalor FROM Valor " +
                "WHERE idFactor="
                + factor.idFactor  +
                " GROUP BY idFactor";
        try {
                con = db.newConection();
                stmt = con.createStatement();
                rs = stmt.executeQuery(query);
                while (rs.next()) {
                        res = res + rs.getDouble("sumvalor");
                }
                rs.close();
                stmt.close();
                con.close();
        } catch (BDException e) {
                System.err.println("Query=" + query);
                e.printStackTrace();
        } catch (SQLException e) {
                System.err.println("Query=" + query);
                e.printStackTrace();
        }
        return res;
}


/**
 * Obtiene el dato capturado directamente por el usuario del factor
 * especifico.
 * @return valor caputrado
 */
String getValorCapturado(){
        String res="";
        switch (factor.tipoValor) {
                case TipoValor.AUTOMATICO:
                        res = Utils.getValueFromQueryAutomatico(
                                        factor, db, ip)+"";
                        break;
                case TipoValor.CONTADOR_NUMERICO:
                        res = getValoresContadorNumerico();
                        break;
                case TipoValor.VERDADERO_FALSO:
```



```
                        res=getValorVerdaderoFalso();
                        break;
                case TipoValor.NUMERICO_SELECCIONABLE:
                        res=getValorNumericoSeleccionable();
                        break;
                case TipoValor.NUMERICO_DIRECTO:
                case TipoValor.PROP_MAX:
                case TipoValor.FUNCION_NUMERICA:
                        res=getValor()+"";
                        break;
        }
        return res;
}

/**
 * Para los factores tipo Contador Numerico, obtiene la lista de
 * factores capturados por el usuario.
 * @return
 */
String getValoresContadorNumerico(){
        String res="";
        Connection con=null;
        Statement stmt=null;
        ResultSet rs=null;
        String query= "SELECT * FROM ValoresEvaluadosCapturados a,"
                +" Valor b " +
                "WHERE a.idValor=b.idValor" +
                " AND idFactor="+factor.idFactor+
                " AND ip='"+ip+"'";
        try {
                con=db.newConection();
                stmt = con.createStatement();
                rs = stmt.executeQuery(query);
                res="<FONT SIZE=\"-1\"><UL>";
                while (rs.next()){
                        res = res +"<LI>" +rs.getString("nombre")
                                + "</LI>";
                }
                res=res+"</UL> </FONT>";
                rs.close();
                stmt.close();
                con.close();
        } catch (BDException e) {
                System.err.println("Query="+query);
                e.printStackTrace();
        } catch (SQLException e) {
                System.err.println("Query="+query);
                e.printStackTrace();
        }
        return res;
}

/**
 * Para los factores de tipo verdadero/falso, obtiene la evaluación del
 * factor
 * @return
 */
String getValorVerdaderoFalso(){
        String res="";
        Connection con=null;
        Statement stmt=null;
        ResultSet rs=null;
        int aux=0;
        String query= "SELECT * FROM Evaluacion " +
                " WHERE idFactor="+factor.idFactor+
                " AND ip='"+ip+"'";
        try {
                con=db.newConection();
                stmt = con.createStatement();
```

```
                        rs = stmt.executeQuery(query);
                        if (rs.next()){
                                aux = rs.getInt("valor");
                                if (aux==1){
                                        res="Verdadero";
                                }else{
                                        res="Falso";
                                }
                        }
                        rs.close();
                        stmt.close();
                        con.close();
                } catch (BDException e) {
                        System.err.println("Query ="+query);
                        e.printStackTrace();
                } catch (SQLException e) {
                        System.err.println("Query="+query);
                        e.printStackTrace();
                }
                return res;
        }

        /**
         * Para los factores de tipo Numérico Seleccionable, obtiene el nombre
         * del valor selecionado por el usario.
         * @return
         */
        String getValorNumericoSeleccionable(){
                String res="";
                Connection con=null;
                Statement stmt=null;
                ResultSet rs=null;
                String query= "SELECT * FROM Evaluacion a, Valor b " +
                        "WHERE a.idFactor= b.idFactor " +
                        "AND a.valor=b.valor" +
                        " AND a.idFactor="+factor.idFactor+
                        " AND ip='"+ip+"'";
                try {
                        con=db.newConection();
                        stmt = con.createStatement();
                        rs = stmt.executeQuery(query);
                        while (rs.next()){
                                res = res + rs.getString("nombre");
                        }
                        rs.close();
                        stmt.close();
                        con.close();
                } catch (BDException e) {
                        System.err.println("Query ="+query);
                        e.printStackTrace();
                } catch (SQLException e) {
                        System.err.println("Query="+query);
                        e.printStackTrace();
                }
                return res;
        }
}
```

```java
package itesm.seguridad;
/*
 * Created on 25/10/2005
 * @author      Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */

/**
 * Clase que obtien todos los metadatos de un query
 * (como puede ser el tipo de datos de los campos, su nombre, titulo, etc).
 */
public class FieldInfo {
        //Tipos de datos de los campos
        public static final int BIGINT_TYPE = -5;
        public static final int COUNTER_TYPE = 4;
        public static final int DOUBLE_TYPE = 8;
        public static final int INTEGER_TYPE = 4;
        public static final int LONGCHAR_TYPE = -1;
        public static final int TEXT_TYPE = -1;
        public static final int VARCHAR_TYPE = 12;
        public static final int TYNY_INT = -6;

        //propiedades
        private boolean editable;
        private int idType;
        private String name;
        private boolean selectionable;
        private String selectionQuery;
        private String title;
        private String type;
        private String value;
        private boolean visible;

        /**
         * Obtine el tipo del campo en forma numérica.
         * @return
         */
        public int getIdType() {
                return idType;
        }

        /**
         * Obtiene el nombre del campo, dato especialmente util, pues
         * en general se toma el nombre para hacer refencia este campo.
         * @return
         */
        public String getName() {
                return name;
        }

        /**
         * Obtiene el query de selecion (en caso de que campo sea selecionable)
         * @return
         */
        public String getSelectionQuery() {
                return selectionQuery;
        }

        /**
         * Obtiene el titulo del campo
         * @return
         */
        public String getTitle() {
                return title;
        }

        /**
```



```java
         * Obtiene el tipo del campo (como string)
         * @return
         */
        public String getType() {
                return type;
        }

        /**
         * Obtiene el valor del campo
         * @return
         */
        public String getValue() {
                return value;
        }

        /**
         * Bandera para saber si el campo es editable.
         * @return
         */
        public boolean isEditable() {
                return editable;
        }

        /**
         * Bandera para saber si el campo es selecionable
         * @return
         */
        public boolean isSelectionable() {
                return selectionable;
        }

        /**
         * Bandera para saber si el campo es visible
         * @return
         */
        public boolean isVisible() {
                return visible;
        }

        /**
         * Bandera para saber si el campo es editable
         * @param editable
         */
        public void setEditable(boolean editable) {
                this.editable = editable;
        }

        /**
         * Establece el tipo del campo (en formato numerico)
         * @param idType
         */
        public void setIdType(int idType) {
                this.idType = idType;
        }

        /**
         * Establece el nombre del campo
         * @param name
         */
        public void setName(String name) {
                this.name = name;
        }

        /**
         * Establce la bandera para saber si es selecionable
         * @param selectionable
         */
        public void setSelectionable(boolean selectionable) {
                this.selectionable = selectionable;
```

```
	}

	/**
	 * Establece el query de seleccion.
	 * @param selectionQuery
	 */
	public void setSelectionQuery(String selectionQuery) {
		this.selectionQuery = selectionQuery;
	}

	/**
	 * Establece el titulo del campo
	 * @param title
	 */
	public void setTitle(String title) {
		this.title = title;
	}

	/**
	 * Establece el tipo del campo (como string)
	 * @param type
	 */
	public void setType(String type) {
		this.type = type;
	}

	/**
	 * Establece el valor del campo
	 * @param value
	 */
	public void setValue(String value) {
		this.value = value;
	}

	/**
	 * Establece la bandera para saber si el campo es visible.
	 * @param visible
	 */
	public void setVisible(boolean visible) {
		this.visible = visible;
	}

	/**
	 * Permiete conocer el contido de esta estructura (al pasarla a string)
	 */
	public String toString() {
		String res;
		//res = "name\t type\t title\t idType\t editable \tvalue\n";
		res = name + "\t" + type + "\t" + title + "\t" + idType;
		if (editable) {
			res = res + "\t editable";
		} else {
			res = res + "\t NOT editable";
		}
		if (visible) {
			res = res + "\t visible";
		} else {
			res = res + "\t NOT visible";
		}
		if (selectionable) {
			res = res + "\t selectionable";
		} else {
			res = res + "\t NOT selectionable";
		}
		res = res + value;
		res = res + "\n";
		return res;
	}
}
```

```
package itesm.seguridad;
/*
 * Created on 19/05/2005
 * @author      Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

/**
 * public class Host Esta clase representa un registro de la tabla Host
 */
public class Host {
        public static DB db;
        String ip;
        String hostname;

        /**
         * Busca un host a partir de su identificador (ip)
         * @param ip
         * @param db
         * @return
         *
         * TODO:cambiar hostname x nombre y agregar descripcion
         *
         */
        public static Host searchHost(String ip, DB db) {
                Statement stmt = db.getStmt();
                ResultSet rs;
                Host h = new Host();
                String query = "SELECT * FROM host WHERE ip='" + ip + "'";
                try {
                        rs = stmt.executeQuery(query);
                        if (rs.next()) {
                                h.ip = rs.getString("ip");
                                h.hostname = rs.getString("nombre");
                        } else {
                                h = null;
                        }
                } catch (SQLException e) {
                        System.err.println("Query=" + query);
                        System.err.println("SQLException: " + e.getMessage());
                        e.printStackTrace();
                }
                return h;
        }
}
```

```
package itesm.seguridad;
/*
 * @author      Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import java.net.URLEncoder;
import java.util.Enumeration;
import java.io.*;
import javax.servlet.http.*;
import itesm.utilsdsc.*;
/**
 * Esta clase reprenta un URL, de la forma:
 *
 * protocol://host:port/path/file?queryString
 *
 */
public class HttpURL {
        //Atributos
        private String file;
        private String host;
        private String path;
        private int port;
        private String protocol;
        private String queryString;
        private SortedHashtable vars;
        /**
         * Constructor a partir de el request http, obiente todos los datos del
         * request exepto las variables.
         * @param req
         */
        public HttpURL(HttpServletRequest req) {
                protocol = req.getScheme();
                host = req.getServerName();
                port = req.getServerPort();
                path = req.getContextPath();
                file = req.getServletPath();
                vars = new SortedHashtable();
        }

        /**
         * Constructor con las opciones por default
         * @param host
         * @param file
         */
        public HttpURL(String host, String file) {
                new HttpURL("http", host, 80, "", file);
        }
        /**
         * Constructor donde se espefifican manualmente todos los parametros.
         * @param protocol
         * @param host
         * @param port
         * @param path
         * @param file
         */
        public HttpURL(String protocol, String host, int port, String path,
                String file) {
                this.protocol = protocol;
                this.host = host;
                this.port = port;
                this.path = path;
                this.file = file;
                vars = new SortedHashtable();
        }

        /**
         * Añade una variable
```



```
         * @param key
         * @param val
         */
        public void addVar(String key, String val) {
                vars.put(key, val);
        }

        /**
         * Quita una variable
         * @param key
         */
        public void delVar(String key) {
                vars.remove(key);
        }

        /**
         * Obtiene el nombre del archivo
         */
        public String getFile() {
                return file;
        }

        /**
         * Obtiene el host
         */
        public String getHost() {
                return host;
        }

        /**
         * Obtiene el path
         */
        public String getPath() {
                return path;
        }

        /**
         * Obtiene el puerto de conexion
         */
        public int getPort() {
                return port;
        }

        /**
         * Obtiene el protocolo, normalmente http
         */
        public String getProtocol() {
                return protocol;
        }

        /**
         * Obtiene el hashTable de variables
         */
        public SortedHashtable getVars() {
                return vars;
        }

        /**
         * Establece el archivo
         */
        public void setFile(String file) {
                this.file = file;
        }

        /**
         * Establece el host
         */
        public void setHost(String host) {
                this.host = host;
```

```
	}

	/**
	 *  Establce el path
	 */
	public void setPath(String path) {
		this.path = path;
	}

	/**
	 * Establce el puerto
	 */
	public void setPort(int port) {
		this.port = port;
	}

	/**
	 *  Establece el protocolo
	 */
	public void setProtocol(String protocol) {
		this.protocol = protocol;
	}
	/**
	 * Regresa un querySting formateado (UTF-8) a partir de ht Vars
	 * @return
	 */
	private String setQueryString() {
		String key, val, temp = "";
		StringBuffer res = new StringBuffer();
		Enumeration keys = vars.keys();
		int i = 1, num = vars.size();
		while (keys.hasMoreElements()) {
			key = (String) keys.nextElement();
			val = (String) vars.get(key);
			res.append(key);
			res.append("=");
			try {
				temp = URLEncoder.encode(val, "UTF-8");
			} catch (UnsupportedEncodingException e) {
				System.err
				     .println("UnsupportedEncodingException:"
				              +" URL= "+ res);
				e.printStackTrace();
			}
			res.append(temp);
			if (i != num) {
				res.append("&");
			}
			i++;
		}
		return res.toString();
	}

	/**
	 * Establece el HT de variables
	 */
	public void setVars(SortedHashtable vars) {
		this.vars = vars;
	}

	/**
	 *  Traduce el URL a texto
	 */
	public String toString() {
		String res;
		res = protocol + "://" + host + ":" + port + path + file;
		queryString = setQueryString();
		if (queryString == null || queryString.equalsIgnoreCase("")) {
			return res;
```



```
		} else {
			res = res + "?" + queryString;
		}
		return res;
	}
}
```

```
package itesm.seguridad;
/*
 * Created on 26/09/2005
 * @author      Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import itesm.utilsdsc.*;
import java.io.IOException;
import java.io.PrintWriter;
import java.text.DecimalFormat;
import java.util.Vector;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Esta clase lista de forma ordenada los equipos con sus respectivos niveles
 * de riesgo.
 */
public class LstEquiposEvaluados extends Esquema {
        private static final long serialVersionUID = 6426589533440859189L;
        DbWebMenu dwm;
        DecimalFormat  df,df2;
        /**
         *  Constructor generico que invoca a la superclase e inicializa los
         *  DecimalFormat
         */
        public LstEquiposEvaluados() {
                super();
                df= new DecimalFormat ("0.000%");
                df2 = new DecimalFormat ("(0.0%)");
        }

        /**
         * Establece el título de la página.
         */
        protected void initPageMenu() {
                dwm = (DbWebMenu) pageProperties[MENU];
                dwm.addPropertie("(*Titulo*)",
                                "Nivel de Riesgo de los Equipos");
        }


        /**
         * Modifica las propiedades del menú generico en tiempo de ejecución.
         * @param req
         */
        protected void modifyPropertiesPgMenu(HttpServletRequest req) {
                dwm.addPropertie("(*lstEquipos*)", getLstEquipos(req));
        }

        /**
         * Genera la lista de equipos con sus evaluaciones.
         * @param req
         * @return
         */
        protected String getLstEquipos(HttpServletRequest req){
                StringBuffer res= new StringBuffer();
                Vector equipos;
                String aux;
                AnalisisRiesgo ar = new AnalisisRiesgo(db);
                equipos = ar.equipos;
                EvaluacionEquipo evalEquipo =
                        (EvaluacionEquipo) equipos.firstElement();
                res.append("\t<TABLE BORDER=\"1\">\n");
                res.append("\t\t<TR>\n");
                res.append("\t\t\t<TD ROWSPAN=\"2\">Equipo </TD>\n");
                res.append("\t\t\t<TD COLSPAN=\""+
```



```
                                evalEquipo.criterios.length +"\">" +
                        "<CENTER> Criterios</CENTER></TD>\n");
                res.append("\t\t\t<TD ROWSPAN=\"2\"> "
                                +"Nivel de Riesgo Total </TD>\n");
                res.append("\t\t</TR>\n");
                res.append("\t\t<TR>\n");
                for(int i=0; i<evalEquipo.criterios.length; i++ ){
                        res.append("\t\t\t<TD >");
                        aux = df2.format(evalEquipo.criterios[i].
                                        criterio.ponderacion);
                        res.append(aux);
                        res.append(evalEquipo.criterios[i].criterio.nombre);
                        res.append("</TD>\n");
                }
                res.append("\t\t</TR>\n");
                //Equipo
                for(int j=0; j< equipos.size(); j++){
                        res.append("\t\t<TR>\n");
                        evalEquipo = (EvaluacionEquipo)equipos.get(j);
                        res.append("\t\t\t<TD >");
                        HttpURL url = new HttpURL(req);
                        url.setFile("/EquipoEvaluado");
                        url.addVar("ip", evalEquipo.host.ip);
                        res.append ("<A HREF=\""+url+"\">");
                        res.append(evalEquipo.host.hostname);
                        res.append("</A>");
                        res.append("</TD>\n");
                        //Criterios
                        for(int i=0; i<evalEquipo.criterios.length; i++ ){
                                res.append("\t\t\t<TD >");
                                aux = df.format(evalEquipo.criterios[i].
                                                evaluacion);
                                res.append(aux);
                                res.append("</TD>\n");
                        }
                        //Evaluacion total
                        res.append("\t\t\t<TD >");
                        aux= df.format(evalEquipo.evaluacion);
                        res.append(aux);
                        res.append("</TD>\n");
                        res.append("\t\t</TR>\n");
                }
                res.append("\t</TABLE>\n");
                return res.toString();
        }
        /**
         * Genera la pagina a mostrar, basicamente indica cual es el
         * .pg a utilizar
         */
        protected void pgMenu(HttpServletRequest req, HttpServletResponse res)
                throws IOException {
                PrintWriter out = res.getWriter();
                String archivo = PATH + "lstEquiposEvaluados.pg";
                res.setContentType("text/html; charset=iso-8859-1");
                PageGenerator page = initPageGenerator(archivo);
                modifyPropertiesPgMenu(req);
                dwm.setPropertiesTo(page);
                out.println(page.render());
                out.close();
        }
}
```

```
package itesm.seguridad;
/*
 * Created on 10/05/2005
 * @author      Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */

/**
 * Esta clase tan solo es una clasificacion de los posibles tipos de factores
 * que maneja el sistema.
 */
public class TipoValor {
        public static final int NUMERICO_DIRECTO = 1;
        public static final int FUNCION_NUMERICA =2;
        public static final int VERDADERO_FALSO =3;
        public static final int NUMERICO_SELECCIONABLE =4;
        public static final int CONTADOR_NUMERICO=5;
        public static final int AUTOMATICO=6;
        public static final int PROP_MAX=7;
}
```

```
package itesm.seguridad;
/*
 * Created on Feb 15, 2004
 * @author      Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */

import itesm.utilsdsc.*;
import java.util.*;
import java.sql.*;
import expeval.EvaluateError;
import expeval.ExpEval;
import expeval.UserExp;

/**
 *  Utilerias generales
 */
public class Utils {

        /**
         * Crea un query para agregar datos a una tabla.
         * @param tabla - nombre de la tabla
         * @param data  - relacion -variable-valor
         */
        public static String createAddQuery(String tabla, SortedHashtable data){
                String res, key = "", val[], aux = "", aux2 = "";
                Enumeration keys = data.keys();
                res = "INSERT INTO " + tabla + "(";
                for (int i = 0; keys.hasMoreElements(); i++) {
                        key = (String) keys.nextElement();
                        val = (String[]) data.get(key);
                        aux = aux + key;
                        aux2 = aux2 + val[0];
                        if (i != (data.size() - 1)) {
                                aux = aux + ",";
                                aux2 = aux2 + ",";
                        }
                        aux = aux + "";
                        aux2 = aux2 + "";
                }
                res = res + aux + ")VALUES(" + aux2;
                res = res + ");";
                return res;
        }

        /**
         * Crea un query para agregar datos a una tabla.
         * @param tabla
         * @param campos
         * @param valores
         */
        public static String createAddQuery(String tabla, String[] campos,
                String[] valores) {
                String res;
                res = "INSERT INTO " + tabla + "(";
                for (int i = 0; i < campos.length; i++) {
                        res = res + campos[i];
                        if (i != (campos.length - 1)) {
                                res = res + ",";
                        }
                        res = res + " ";
                }
                res = res + ")VALUES(";
                for (int i = 0; i < valores.length; i++) {
                        res = res + valores[i];
                        if (i != (valores.length - 1)) {
                                res = res + ",";
                        }
                        res = res + " ";
                }
                res = res + ");";
                return res;
        }

        /**
         * Crea un registro de checkbox en formato HTML
         * @param key
         * @param val
         * @return
         */
        public static String createCheckBoxField(String key, String val) {
                String res = "<input type=\"checkbox\" name=\""
                        + key + "\" value=\"" + val + "\">";
                return res;
        }

        /**
         * Crea renglones de checkbox
         * @param fields - espeficicaiones de los campos
         * @param db - la base de datos
         * @param query - query a mostrar
         * @return
         */
        public static String createCheckBoxRows(Vector fields, DB db,
                        String query) {
                FieldInfo fieldInfo;
                StringBuffer res = new StringBuffer();
                ResultSet rs;
                Statement stmt;
                stmt = db.getStmt();
                try {
                        rs = stmt.executeQuery(query);
                        while (rs.next()) {
                        //El primer campo debe ser el identificador del registro
                        //<input type="checkbox"name="checkbox"value="checkbox">
                                fieldInfo = (FieldInfo) fields.firstElement();
                                res.append("\n\t\t<TR>\n");
                                res.append("\t\t\t<TD>");
                                res.append("<INPUT TYPE=\"CHECKBOX\" NAME=\""
                                        + fieldInfo.getName() + "\"");
                                res.append(" VALUE=\"");
                                if (fieldInfo.getType()
                                                .equalsIgnoreCase("VARCHAR")
                                        || fieldInfo.getType()
                                                .equalsIgnoreCase("LONGCHAR"))
{
                                        res.append(rs.getString(1));
                                }
                                if (fieldInfo.getType()
                                                .equalsIgnoreCase("COUNTER")
                                        || fieldInfo.getType()
                                                .equalsIgnoreCase("INTEGER")
                                        || fieldInfo.getType()
                                                .equalsIgnoreCase("BIGINT")) {
                                        res.append(rs.getInt(1));
                                }
                                if (fieldInfo.getType()
                                                .equalsIgnoreCase("DOUBLE")) {
                                        res.append(rs.getDouble(1));
                                }
                                res.append("\"></TD>\n");

                                for (int i=1;
                                        i<rs.getMetaData().getColumnCount();
                                        i++) {
                                        int j = i + 1;
```

```
                                fieldInfo = (FieldInfo) fields.get(i);
                                res.append("\t\t\t<TD>");
                                if (fieldInfo.getType()
                                        .equalsIgnoreCase("VARCHAR")
                                        || fieldInfo.getType()
                                        .equalsIgnoreCase("LONGCHAR")
                                        || fieldInfo.getType()
                                        .equalsIgnoreCase("TEXT")) {
                                        res.append(rs.getString(j));
                                }
                                if (fieldInfo.getType()
                                        .equalsIgnoreCase("COUNTER")
                                        || fieldInfo.getType()
                                        .equalsIgnoreCase("INTEGER")
                                        || fieldInfo.getType()
                                        .equalsIgnoreCase("BIGINT")) {
                                        res.append(rs.getInt(j));
                                }
                                if (fieldInfo.getType()
                                        .equalsIgnoreCase("DOUBLE")) {
                                        res.append(rs.getDouble(j));
                                }
                                res.append("</TD>\n");
                        }
                        res.append("\t\t</TR>");
                }
        } catch (SQLException e) {
                System.err.println("Query =" + query);
                System.err.println("SQLException: " + e.getMessage());
                e.printStackTrace();
        }
        return res.toString();
}

/**
 * Crea una lista de checkBox a partir de un query.
 * @param name  -  nombre del checkbox
 * @param db    -  conexion con la base de datos
 * @param query -  query a ejecutar.
 * @return
 */
public static String createCheckBoxRowsForContadorNumerico(String name,
        DB db, String query) {
        return createCheckBoxRowsForContadorNumerico(
                        name, db, query, null);
}

/**
 * Crea una lista de checkBox, en los cuales se pueden especificar
 * cuales estan marcados (cheked) y cuales no para un equipo
 * en especifico.
 * @param name  - nombre del checkbox
 * @param db    - conexion con la base de datos
 * @param query - query a ejecutar
 * @param ip    - equipo
 * @return
 */
public static String createCheckBoxRowsForContadorNumerico(String name,
        DB db, String query, String ip) {
        StringBuffer res = new StringBuffer();
        ResultSet rs;
        Statement stmt;
        String queryValCap = "", idValor = "", nombre = "";
        int aux = 0;
        stmt = db.getStmt();
        try {
                rs = stmt.executeQuery(query);
                // <input type="checkbox" name="checkbox"
                // value="checkbox" checked>
```



```
                while (rs.next()) {
                        idValor = rs.getInt("idValor") + "";
                        nombre = rs.getString("nombre");
                        queryValCap = "SELECT * FROM "
                                + "ValoresEvaluadosCapturados "
                                + "WHERE ip='" + ip + "'"
                                + "AND idValor=" + idValor;
                        aux = db.contRows(queryValCap);
                        res.append("<INPUT TYPE=\"CHECKBOX\" NAME=\"");
                        res.append(name);
                        res.append("\" VALUE=\"" + idValor + "\"");
                        if (aux != 0) {
                                res.append(" \"CHECKED\"");
                        }
                        res.append(">");
                        res.append(nombre);
                        res.append("<BR>");
                }
        } catch (SQLException e) {
                System.err.println("Query =" + query);
                System.err.println("SQLException: " + e.getMessage());
                e.printStackTrace();
        } catch (BDException e) {
                System.err.println("Query =" + queryValCap);
                System.err.println("SQLException: " + e.getMessage());
                e.printStackTrace();
        }
        return res.toString();
}

/**
 * Crea un query para eliminar datos de una tabla.
 * @param tabla
 * @param campo -  campo condicional
 * @param valor -  valor a comparar
 */
public static String createDelQuery(String tabla, String campo,
                String valor) {
        String res;
        res = "DELETE FROM " + tabla + " WHERE " +
                campo + "=" + valor + ";";
        return res;
}

/**
 * Funcion de ayuda de HTML Crea un Hidden Field
 * Genera el siguiente string
 *      <INPUT NAME = "key" TYPE="HIDDEN" VALUE="val">
 * @param key
 * @param val
 * @return
 */
public static String createHTMLHiddenField(String key, String val) {
        String res = "";
        // <input name="accion" type="hidden" value="execBajaValor">
        res = res + "<INPUT TYPE=\"HIDDEN\" NAME=\"" +
                key + "\" VALUE=\"" + val + "\">\n";
        return res;
}

/**
 * Función de ayuda de HTML Crea un lista de HIDDEN Field a partir de un
 * Hastble
 * @param ht- relacion variable-valor
 * @return
 */
public static String createHTMLHiddenFields(Hashtable ht) {
        String res = "";
        Enumeration e;
```

```
            e = ht.keys();
            while (e.hasMoreElements()) {
                    String key, val;
                    key = (String) e.nextElement();
                    val = (String) ht.get(key);
                    // <input name="accion" type="hidden"
                    // value="execBajaValor">
                    res = res + "<INPUT TYPE=\"HIDDEN\" NAME=\"" +
                            key + "\" VALUE=\"" + val + "\">\n";
            }
            return res;
    }

    /**
     * Funcion de ayuda de HTML Crea un INPUT tag por default
     * @param name - nombre del INPUT tag
     * @return
     */
    public static String createHTMLInputTag(String name) {
            return createHTMLInputTag(name, null, 40);
    }

    /**
     * Funcion de ayuda de HTML Crea un INPUT tag, especificando el nombre y
     * tamaño
     * @param name
     * @param size
     * @return
     */
    public static String createHTMLInputTag(String name, int size) {
            return createHTMLInputTag(name, null, size);
    }

    /**
     * Función de ayuda de HTML Crea un INPUT tag, especificando el nombre y
     * valor
     * @param name
     * @param value
     * @return
     */
    public static String createHTMLInputTag(String name, String value) {
            return createHTMLInputTag(name, value, 40);
    }

    /**
     * Función de ayuda de HTML Crea un INPUT tag, especificando el nombre,
     * valor y tamaño
     *
     * @param name
     * @param value
     * @param size
     * @return
     */
    public static String createHTMLInputTag(String name, String value,
                    int size) {
            String res;
            res = "<INPUT NAME=\"" + name + "\" TYPE=\"TEXT\" ";
            if (!(value == null || value.equalsIgnoreCase(""))) {
                    res = res + "VALUE=\"" + value + "\"";
            }
            if (size > 0) {
                    res = res + "SIZE=\"" + size + "\"";
            }
            res = res + ">";
            return res;
    }

    /**
     * Funcion de ayuda de HTML Crea un SELECT tag, especificando el nombre
```



```
     * y el query de selección
     * @param name  - nombre del select
     * @param db    - conexion con la base de datos
     * @param query - query a ejecutar.
     * @return
     */
    public static String createHTMLSelectOptionTag(String name, DB db,
            String query) {
            return Utils.createHTMLSelectOptionTag(name, db, query, null);
    }

    /**
     * Función de ayuda de HTML Crea un SELECT tag, especificando el nombre,
     * el query de seleccion y el elemento a estar selecionado
     * @param name  -  nombre del SELECT
     * @param db    -  conexion con la base de datos
     * @param query -  query a ejecutar Es forzososo que el query
     *                 regrese solo dos  campos, donde el primero es la
     *                 llave (de tipo numerico) y el segundo es el valor
     *                 a desplegar (de tipo string)
     * @param selected - elemento a marcar como seleccionado.
     * @return TOREVIEW
     */
    public static String createHTMLSelectOptionTag(String name, DB db,
            String query, String selected) {
            ResultSet rs;
            StringBuffer res;
            String key = "", val = "";
            Statement stmt;
            res = new StringBuffer();
            stmt = db.getStmt();
            res.append("<SELECT NAME=\"");
            res.append(name);
            res.append("\">\n");
            try {
                    rs = stmt.executeQuery(query);
                    ResultSetMetaData rsmt = rs.getMetaData();
                    int keyType;
                    while (rs.next()) {
                            // Es forzososo que el query regrese solo
                            // dos campos, donde el
                            // primero es la llave (de tipo numerico)
                            // y el segundo es el valor a desplegar (de
                            // tipo string)
                            keyType = rsmt.getColumnType(1);
                            switch (keyType) {
                                    // case FieldInfo.INTEGER_TYPE:
                                    case FieldInfo.BIGINT_TYPE:
                                    case FieldInfo.COUNTER_TYPE:
                                            key = rs.getInt(1) + "";
                                            break;
                                    case FieldInfo.DOUBLE_TYPE:
                                            key = rs.getDouble(1) + "";
                                            break;
                                    // case FieldInfo.LONGCHAR_TYPE :
                                    case FieldInfo.TEXT_TYPE:
                                    case FieldInfo.VARCHAR_TYPE:
                                            key = rs.getString(1) + "";
                                            break;
                            }
                            keyType = rsmt.getColumnType(2);
                            switch (keyType) {
                                    // case FieldInfo.INTEGER_TYPE:
                                    case FieldInfo.BIGINT_TYPE:
                                    case FieldInfo.COUNTER_TYPE:
                                            val = rs.getInt(2) + "";
                                            break;
                                    case FieldInfo.DOUBLE_TYPE:
                                            val = rs.getDouble(2) + "";
```

```
                                        break;
                                // case FieldInfo.LONGCHAR_TYPE :
                                case FieldInfo.TEXT_TYPE:
                                case FieldInfo.VARCHAR_TYPE:
                                        val = rs.getString(2);
                                        break;
                        }
                        res.append("\t\t\t\t<OPTION VALUE=\"");
                        res.append(key);
                        res.append("\"");
                        if (selected != null) {
                                if (selected.equalsIgnoreCase(key)) {
                                        res.append(" selected");
                                }
                        }
                        res.append(">");
                        res.append(val);
                        res.append("</OPTION>\n");
                }
                res.append("\t\t\t</SELECT>");
        } catch (SQLException e) {
                System.err.println("Query -" + query);
                System.err.println("SQLException: " + e.getMessage());
                e.printStackTrace();
        }
        return res.toString();
}

/**
 * Función de ayuda de HTML Crea un tag de TEXT AREA, especificando el
 * nombre
 * @param name -  nombre del TEXT AREA
 * @return
 */
public static String createHTMLTextAreaTag(String name) {
        return createHTMLTextAreaTag(name, null, 40, 6);
}

/**
 * Función de ayuda de HTML Crea un tag de TEXT AREA, especificando el
 * nombre, alto y ancho.
 * @param name -  nombre del TEXT AREA
 * @param cols -  Ancho
 * @param rows -  Alto
 * @return
 */
public static String createHTMLTextAreaTag(String name, int cols,
                int rows) {
        return createHTMLTextAreaTag(name, null, cols, rows);
}

/**
 * Función de ayuda de HTML Crea un tag de TEXT AREA, especificando el
 * nombre y valor
 * @param name  - nombre del TEXT AREA
 * @param value - valor
 * @return
 */
public static String createHTMLTextAreaTag(String name, String value) {
        return createHTMLTextAreaTag(name, value, 40, 6);
}

/**
 * Función de ayuda de HTML Crea un tag de TEXT AREA, especificando el
 * nombre, valor, alto y ancho.
 * @param name  -  nombre del TEXT AREA
 * @param value -  valor
 * @param cols  -  ancho
 * @param rows  -  alto
```



```
 * @return
 */
public static String createHTMLTextAreaTag(String name, String value,
        int cols, int rows) {
        String res;
        res = "<TEXTAREA NAME=\"" + name + "\"";
        if (cols > 0 && rows > 0) {
                res = res + "COLS=\"" + cols + "\" "
                        +"ROWS=\"" + rows + "\"";
        }
        res = res + ">";
        if (!(value == null || value.equalsIgnoreCase(""))) {
                res = res + value + " ";
        }
        res = res + "</TEXTAREA> ";
        return res;
}

/**
 * Función de ayuda HTML. Crea un menu de selecion de RADIO BOTTON
 * @param name -  nombre del radio boton
 * @param ht   -  relacion nombre- valor
 * @return
 */
public static String createOptionButtonMenu(String name, Hashtable ht) {
        StringBuffer res = new StringBuffer();
        Enumeration e;
        e = ht.keys();
        while (e.hasMoreElements()) {
                String key, val;
                key = (String) e.nextElement();
                val = (String) ht.get(key);
                // <input name="accion" type="radio"
                // value="(*SubMenuAction*)"  checked>
                res.append("\t\t\t<INPUT NAME=\"" + name
                        + "\" TYPE=\"RADIO\" VALUE=\"" + val + "\" >");
                res.append(key);
                res.append("<BR>\n ");
        }
        return res.toString();
}

/**
 * Función de ayuda de HTML Crea renglones con RADIO BUTTON
 * @param name - nombre de la lista RADIO BUTTON
 * @param ht   - relación nombre-valor
 * @return
 */
public static String createOptionButtonMenuRows(String name,
                Hashtable ht) {
        StringBuffer res = new StringBuffer();
        Enumeration e;
        e = ht.keys();
        while (e.hasMoreElements()) {
                String key, val;
                key = (String) e.nextElement();
                val = (String) ht.get(key);
                // <input name="accion" type="radio"
                // value="(*SubMenuAction*)" checked>
                res.append("\t<TR>\n");
                res.append("\t\t<TD>");
                res.append("<INPUT NAME=\"" + name
                        + "\" TYPE=\"RADIO\" VALUE=\""+ val + "\">");
                res.append(key);
                res.append("</TD>\n ");
                res.append("</TR>\n");
        }
        return res.toString();
}
```

```
    /**
     * Crea un registro de Radio Botton en formato HTML
     * @param key - nombre del RADIO BOTTON
     * @param val - valor
     * @return
     */
    public static String createRadioBottonField(String key, String val) {
            String res = "<input type=\"radio\" name=\""
                    + key + "\" value=\"" + val + "\" checked >";
            return res;
    }

    /**
     * Función de ayuda HTML Crea un campo de Radio Bottons con sus posibles
     * valores
     * @param fields - Especificaciones de los campos
     * @param db     - conexion con la base de datos
     * @param query  - query a ejecutar.
     * @return TOREVIEW
     */
    public static String createRadioBottonRows(Vector fields, DB db,
            String query) {
            FieldInfo fieldInfo;
            StringBuffer res = new StringBuffer();
            ResultSet rs;
            Statement stmt;
            stmt = db.getStmt();
            try {
                    rs = stmt.executeQuery(query);
                    res.append("\n\t\t<TR>\n");
                    for (int j = 0;
                            j < rs.getMetaData().getColumnCount(); j++) {
                            fieldInfo = (FieldInfo) fields.get(j);
                            res.append("\t\t\t<TD>");
                            res.append(fieldInfo.getTitle());
                            res.append("</TD>\n");
                    }
                    res.append("\t\t</TR>");
                    while (rs.next()) {
                    //El primer campo debe ser el identificador del registro
                    // <input type="radio" name="criterio" value="1">
                            fieldInfo = (FieldInfo) fields.firstElement();
                            res.append("\n\t\t<TR>\n");
                            res.append("\t\t\t<TD>");
                            res.append("<INPUT TYPE=\"RADIO\" NAME=\""
                                    + fieldInfo.getName() + "\"");
                            res.append(" VALUE=\"");
                            switch (fieldInfo.getIdType()) {
                                    case FieldInfo.VARCHAR_TYPE:
                                    case FieldInfo.TEXT_TYPE:
                                    // case FieldInfo.LONGCHAR_TYPE:
                                            res.append(rs.getString(1));
                                            break;
                                    case FieldInfo.COUNTER_TYPE:
                                    case FieldInfo.BIGINT_TYPE:
                                    // case FieldInfo.INTEGER_TYPE:
                                            res.append(rs.getInt(1));
                                            break;
                                    case FieldInfo.DOUBLE_TYPE:
                                            res.append(rs.getDouble(1));
                            }
                            res.append("\"></TD>\n");

                            for (int i = 1;
                                    i < rs.getMetaData().getColumnCount();
                                    i++) {
                                    int j = i + 1;
                                    fieldInfo = (FieldInfo) fields.get(i);
                                    res.append("\t\t\t<TD>");
                                    switch (fieldInfo.getIdType()) {
                                            case FieldInfo.VARCHAR_TYPE:
                                            case FieldInfo.TEXT_TYPE:
                                            // case FieldInfo.LONGCHAR_TYPE:
                                                    res.append(
                                                       rs.getString(j));
                                                    break;
                                            case FieldInfo.COUNTER_TYPE:
                                            case FieldInfo.BIGINT_TYPE:
                                            // case FieldInfo.INTEGER_TYPE:
                                                  res.append(rs.getInt(j));
                                                    break;
                                            case FieldInfo.DOUBLE_TYPE:
                                                res.append(rs.getDouble(j));
                                    }
                                    res.append("</TD>\n");
                            }
                            res.append("\t\t</TR>");
                    }
            } catch (SQLException e) {
                    System.err.println("Query =" + query);
                    System.err.println("SQLException: " + e.getMessage());
                    e.printStackTrace();
            }
            return res.toString();
    }

    /**
     * Crea un query para buscar un campo de una tabla.
     * @param tabla
     * @param campo
     * @param valor
     */
    public static String createSelectQuery(String tabla, String campo,
            String donde) {
            String res;
            res = "SELECT " + campo + " FROM " + tabla
                    + " WHERE " + donde + ";";
            return res;
    }

    /**
     * Crea un query para actualizar datos a una tabla.
     * @param tabla
     * @param data - relación llave -valor
     * @param donde
     */
    public static String createUpdateQuery(String tabla,
                    SortedHashtable data, String donde) {
            String res;
            String key = "", val[];
            int i = 0;
            Enumeration keys = data.keys();
            res = "UPDATE " + tabla + " SET ";
            while (keys.hasMoreElements()) {
                    key = (String) keys.nextElement();
                    val = (String[]) data.get(key);
                    res = res + key + "=" + val[0];
                    if (i != (data.size() - 1)) {
                            res = res + ",";
                    }
                    res = res + " ";
                    i++;
            }
            res = res + " WHERE " + donde + ";";
            return res;
    }
```

```
        /**
         * Crea un query para actualizar datos a una tabla.
         * @param tabla
         * @param campos
         * @param valores
         * @param donde
         */
        public static String createUpdateQuery(String tabla, String[] campos,
                String[] valores, String donde) {
                String res;
                res = "UPDATE " + tabla + " SET ";
                for (int i = 0; i < campos.length; i++) {
                        res = res + campos[i] + "=" + valores[i];
                        if (i != (campos.length - 1)) {
                                res = res + ",";
                        }
                        res = res + " ";
                }
                res = res + " WHERE " + donde + ";";
                return res;
        }

        /**
         * Entrecomilla un String con comillas dobles (")
         * @param in - String de entrada.
         * @return
         */
        public static String doubleQuote(String in) {
                String res;
                res = "\"" + in + "\"";
                return res;
        }

        /**
         * Crea un campos (TD) de una tabla en formato HTML
         * @param in[] -  Datos a insertar en campos.
         * @return
         */
        public static String getTableRow(String[] in) {
                String res = "";
                for (int i = 0; i < in.length; i++) {
                        res += "<td>" + in[i] + "</td>\n";
                }
                return res;
        }

        /**
         * Función de ayuda SortedHashTable. Obtiene la index-esima llave de un
         * SortedHashTable.
         * @param ht    -  HashTable
         * @param index -  índice
         * @return
         */
        public static String hashTableGetKeyAt(SortedHashtable ht, int index) {
                int j = 0;
                Enumeration e;
                e = ht.keys();
                while (e.hasMoreElements()) {
                        String key;
                        key = (String) e.nextElement();
                        if (j == index) {
                                return key;
                        }
                        j++;
                }
                return null;
        }

        /**
```



```
         * Función de ayuda Sorted SortedHashTable.
         * Obtiene el index-esimo elemento de un SortedHashTable
         * @param ht
         * @param index
         * @return
         */
        public static String hashTableGetValueAt(SortedHashtable ht,int index) {
                int j = 0;
                Enumeration e;
                e = ht.keys();
                while (e.hasMoreElements()) {
                        String key, val[];
                        key = (String) e.nextElement();
                        val = (String[]) ht.get(key);
                        if (j == index) {
                                return val[0];
                        }
                        j++;
                }
                return null;
        }

        /**
         * Funcion de ayuda HashTable. Convierte un hash table a string
         * @param ht
         * @return
         */
        public static String hashTableToString(Hashtable ht) {
                String res = "";
                Enumeration e;
                e = ht.keys();

                while (e.hasMoreElements()) {
                        String key, val;
                        key = (String) e.nextElement();
                        val = (String) ht.get(key);
                        res = res + key + " " + val + "\n";
                }
                return res;
        }

        /**
         * Entrecomilla un String con comillas sencillas (')
         * @param in -String de entrada.
         * @return
         */
        public static String singleQuote(String in) {
                String res;
                res = "'" + in + "'";
                return res;
        }

        /**
         * Función de ayuda HTML crea el tag de inicio FORM, es decir, crea el
         * siguiente, invocando un script generico en javaScript que verifica la
         * forma.
         * <FORM NAME="formName" METHOD="method" ACTION="action"
         * onSubmit= "return  funtionName();" >
         * @param formName
         * @param functionName
         * @param method
         * @param action
         * @param javaScriptProperties
         * @return
         */
        public static String createHTMLFormTag(String formName,
                String functionName, String method, String action,
                SortedHashtable javaScriptProperties) {
                StringBuffer res = new StringBuffer();
```

```
                String sname = "NAME=\"" + formName + "\" ";
                String saction = "ACTION=\"" + action + "\" ";
                String smethod = "METHOD=\"";
                StringBuffer varlist = new StringBuffer();
                String ret = "return " + functionName + "(this); \n";
                if (!javaScriptProperties.isEmpty()) {
                        Enumeration aux = javaScriptProperties.keys();
                        while (aux.hasMoreElements()) {
                                String key, val;
                                key = (String) aux.nextElement();
                                val = (String) javaScriptProperties.get(key);
                                varlist.append("\t" + key + "=" + val + ";\n");
                        }
                }
                if (isNullString(method)) {
                        // Este sería el metodo por default en la forma.
                        smethod = smethod + "GET\" ";
                } else {
                        // /Deberia mandar una excepcion si
                        // ponen un metodo invalido
                        smethod = smethod + method + "\" ";
                }
                res.append("<FORM ");
                res.append(sname);
                res.append(smethod);
                res.append(saction);
                if (!isNullString(functionName)) {
                        res.append("onSubmit=");
                        res.append("\"\n");
                        res.append(varlist);
                        res.append(ret);
                        res.append("\"");
                }
                res.append(" >\n");
                return res.toString();
        }

        /**
         * Función de ayuda HTML crea el tag de inicio FORM, es decir, crea el
         * siguiente String
         * <FORM NAME="formName" METHOD="GET" ACTION="action">
         * @param formName
         * @param action
         * @return
         */
        public static String createHTMLFormTag(String formName, String action) {
                return createHTMLFormTag(formName, null, null, action,
                        new SortedHashtable());
        }

        /**
         * Función de ayuda HTML crea el tag de inicio FORM, es decir, crea el
         * siguiente String
         * <FORM NAME="formName" METHOD="GET" ACTION="action" onSubmit= "return
         * funtionName();" >
         * @param formName
         * @param action
         * @param functionName
         * @return
         */
        public static String createHTMLFormTag(String formName, String action,
                String functionName) {
                return createHTMLFormTag(formName, functionName, null, action,
                        new SortedHashtable());
        }

        /**
         * Función de ayuda HTML crea el tag de inicio FORM
         * @param formName
```



```
         * @param action
         * @param functionName
         * @param ht
         * @return
         */
        public static String createHTMLFormTag(String formName, String action,
                String functionName, SortedHashtable ht) {
                return createHTMLFormTag(formName, functionName, null, action, h
t);
        }

        /**
         * Función de ayuda HTML crea el tag de inicio FORM
         * @param formName
         * @param functionName
         * @param method
         * @param action
         * @return
         */
        public static String createHTMLFormTag(String formName,
                String functionName, String method, String action) {
                return createHTMLFormTag(formName, functionName, method, action,
                        new SortedHashtable());
        }

        /**
         * Verifica si un String es nulo (null) o es un string vacio ("")
         * @param s
         * @return
         */
        public static boolean isNullString(String s) {
                if (s == null)
                        return true;
                if (s.equalsIgnoreCase(""))
                        return true;
                return false;
        }

        /**
         * Crea un OPTION tag con las opciones de verdadero/falso
         * @param name
         * @return
         */
        public static String createTrueFalseOption(String name) {
                return createTrueFalseOption(name, "0");
        }

        /**
         * Crea un OPTION Tag con las opciones de verdadero/falso indicando la
         * opción seleccionada.
         *
         * @param name
         * @param selected
         * @return
         */
        public static String createTrueFalseOption(String name, String selected)
{
                StringBuffer res = new StringBuffer();
                res.append("\t<SELECT NAME=\"");
                res.append(name);
                res.append("\">\n");
                res.append("\t\t<OPTION VALUE=\"1\">Verdadero </OPTION>\n");
                if (selected != null) {
                        if (selected.equalsIgnoreCase("0")) {
                                res.append("\t\t<OPTION VALUE=\"0\" >"
                                                + "Falso </OPTION>\n");
                        } else {
                                res.append("\t\t<OPTION VALUE=\"0\" SELECTED >"
                                                + "Falso </OPTION>\n");
```

```
                }
            } else {
                res.append("\t\t<OPTION VALUE=\"0\">Falso</OPTION>\n");
            }
            res.append("\t</SELECT>\n");
            return res.toString();
        }

        /**
         * Conviete una direccion IP en formato A.B.C.D a su entero de 32 bits
         * equivalente
         */
        public static int strIp2Int(String strIP) {
            int res = 0;
            String myStrIP[] = strIP.split("\.");
            int myIntIP[] = new int[4];
            for (int i = 0; i < myStrIP.length; i++) {
                myIntIP[i] = Integer.parseInt(myStrIP[i]);
            }
            res = myIntIP[0] * 256 * 256 * 256 + myIntIP[1] * 256 * 256
                    + myIntIP[2] * 256 + myIntIP[3];
            return res;
        }

        /**
         * Obtiene el valor evaluado de un queryAutomatico de un equipo en
         * especifico.
         * @param factor
         * @param db
         * @param ip
         * @return
         */
        public static double getValueFromQueryAutomatico(Factor factor, DB db,
                String ip) {
            Statement stmt = db.getStmt();
            ResultSet rs, rsAutomatico;
            int idConexionBD = 0;
            double res = 0;
            String query, queryAutomatico = null;
            query = "SELECT * FROM QueryAutomatico WHERE idQueryAutomatico="
                    + factor.idQueryTipoAutomatico;
            try {
                rs = stmt.executeQuery(query);
                if (rs.next()) {
                    String cond = rs.getString("cond");
                    queryAutomatico = rs.getString("query");
                    idConexionBD = rs.getInt("idConexionBD");
                    // El campo ip es de tipo string
                    if ((rs.getInt("stringIP")) == 1) {
                        if (cond.equalsIgnoreCase("")
                                || cond == null) {
                            queryAutomatico = queryAutomatico
                                    + " WHERE "
                                    + rs.getString("campoIP")
                                    + "=" + "'" + ip
                                    + "'";
                        } else {
                            queryAutomatico = queryAutomatico
                                    + " WHERE "
                                    + rs.getString("cond")
                                    + " AND "
                                    + rs.getString("campoIP")
                                    + "=" + "'" + ip
                                    + "'";
                        }
                    } else {
                        int iip = strIp2Int(ip);
                        queryAutomatico = queryAutomatico
                                + " HAVING "
```



```
                                    + rs.getString("campoIP") + "="
                                    + iip;
                        }
                    } else {
                        queryAutomatico = null;
                        return 0;
                    }
                } catch (SQLException e) {
                    System.err.println("Query =" + query);
                    System.err.println("SQLException: " + e.getMessage());
                    e.printStackTrace();
                }
                if (queryAutomatico != null) {
                    try {
                        Connection conaux =
                                db.newConectionFromdb(idConexionBD);
                        Statement stmtaux = conaux.createStatement();
                        rsAutomatico =
                                stmtaux.executeQuery(queryAutomatico);
                        if (rsAutomatico.next()) {
                            res = rsAutomatico.getInt(1);
                        } else {
                            res = 0;
                        }
                        stmtaux.close();
                        conaux.close();
                    } catch (SQLException e) {
                        System.err.println("QueryAutomatico ="
                                + queryAutomatico);
                        System.err.println("SQLException: "
                                + e.getMessage());
                        e.printStackTrace();
                    } catch (BDException e) {
                        System.err.println("QueryAutomatico ="
                                + queryAutomatico);
                        System.err.println("BDException: "
                                + e.getMessage());
                        e.printStackTrace();
                    }
                }
                return res;
            }

            /**
             * Obtiene el valor Maximo de un queryAutomatico de todos los equipos
             * @param factor
             * @param db
             * @return
             */
            public static double getMaxValueFromQueryAutomatico(Factor factor,
                    DB db) {
                Statement stmt = db.getStmt();
                ResultSet rs, rsAutomatico;
                int idConexionBD = 0;
                double res = 0;
                String query, maxQuery = null;

                query = "SELECT * FROM QueryAutomatico WHERE idQueryAutomatico="
                        + factor.idQueryTipoAutomatico;
                try {
                    rs = stmt.executeQuery(query);
                    if (rs.next()) {
                        String cond = rs.getString("cond");
                        idConexionBD = rs.getInt("idConexionBD");
                        maxQuery = rs.getString("maxQuery");
                        if ((rs.getInt("stringIP")) == 1) {
                            if (cond.equalsIgnoreCase("") ||
                                    cond == null) {
                                maxQuery = maxQuery+"";
```

```
                                } else {
                                        maxQuery = maxQuery
                                          + " WHERE "
                                          + rs.getString("cond");
                                }
                        }
                } else {
                        maxQuery = null;
                        return 0;
                }
        } catch (SQLException e) {
                System.err.println("MaxQuery =" + maxQuery);
                System.err.println("SQLException: " + e.getMessage());
                e.printStackTrace();
        }
        if (maxQuery != null) {
                try {
                        Connection conaux =
                                db.newConectionFromdb(idConexionBD);
                        Statement stmtaux = conaux.createStatement();
                        ResultSetMetaData rsmd;
                        rsAutomatico = stmtaux.executeQuery(maxQuery);
                        rsmd = rsAutomatico.getMetaData();
                        if (rsAutomatico.next()) {
                                switch (rsmd.getColumnType(1)) {
                                        case FieldInfo.BIGINT_TYPE:
                                        // case FieldInfo.COUNTER_TYPE:
                                        case FieldInfo.INTEGER_TYPE:
                                           res = rsAutomatico.getInt(1);
                                           break;
                                        case FieldInfo.DOUBLE_TYPE:
                                          res=rsAutomatico.getDouble(1);
                                          break;
                                }
                        } else {
                                res = 0;
                        }
                        stmtaux.close();
                        conaux.close();
                } catch (SQLException e) {
                        System.err.println("maxQuery =" + maxQuery);
                        System.err.println("SQLException: "
                                        + e.getMessage());
                        e.printStackTrace();
                } catch (BDException e) {
                        System.err.println("maxQuery =" + maxQuery);
                        System.err.println("BDException: "
                                        + e.getMessage());
                        e.printStackTrace();
                }
        }
        return res;
}

/**
 * Obtiene el valor evaluado de una Función Numérica de
 * un equipo en especifico.
 * @param factor
 * @param db
 * @param ip
 * @return
 */
public static double getEvaluatedValueFromFunctionFactor(Factor factor,
        DB db, String ip) {
        Statement stmt = db.getStmt();
        ResultSet rs, rsEvaluacion;
        String query, funcionFactor = "", queryEvaluacion, var = "";
        int valor = 0;
        query = "SELECT * FROM FuncionFactor WHERE idFuncion="
```



```
                        + factor.idQueryTipoAutomatico;
                try {
                        rs = stmt.executeQuery(query);
                        if (rs.next()) {
                                funcionFactor = rs.getString("funcion");
                                var = rs.getString("variable");
                        }
                } catch (SQLException e) {
                        System.err.println("Query =" + query);
                        System.err.println("SQLException: " + e.getMessage());
                        e.printStackTrace();
                }
                queryEvaluacion = "SELECT * FROM Evaluacion WHERE "+ "idFactor="
                        + factor.idFactor + " AND ip='" + ip + "'";
                try {
                        rsEvaluacion = stmt.executeQuery(queryEvaluacion);
                        if (rsEvaluacion.next()) {
                                valor = rsEvaluacion.getInt("valor");
                        }
                } catch (SQLException e) {
                        System.err.println("QueryEvaluacion ="
                                        + queryEvaluacion);
                        System.err.println("SQLException: " + e.getMessage());
                        e.printStackTrace();
                }

                String expresion = funcionFactor.replaceAll(var, valor + "");
                double res1 = 0;
                expeval.UserExp ue = new UserExp();
                ue.setExp(expresion);
                ExpEval ee = new ExpEval();
                try {
                        res1 = ee.evaluate(ue);
                } catch (EvaluateError e1) {
                        System.err.println("Expresion" + expresion);
                        e1.printStackTrace();
                }
                return res1;
        }
}

/*
 * TODOs Partir esta clase en varias secciones - db, html , strigs, etc.
 */
```

```
package itesm.seguridad;
/*
 * Created on Apr 22, 2004
 *  @author      Ing. M. Damián Guerra Farías.
 * ITESM-CEM
 * Tesis:
 *      Sistema para la Definición y Evaluación de Riesgos Informaticos(SDERI)
 */
import java.util.*;
import javax.servlet.http.*;
import itesm.utilsdsc.*;

/**
 * Utilerias para web.
 */

public class WebUtils {
        /**
         * Le pone formato a los datos para ser introduciodos en un query
         * @param query
         * @param data
         * @return
         */
        public static SortedHashtable formatDataforDB(String query,
                SortedHashtable data) {
                String key;
                String vals[], aux[];
                SortedHashtable res = new SortedHashtable();
                DbWeb dbweb = new DbWeb();
                dbweb.setQuery(query);
                Object keyArray[]= (data.keySet()).toArray();
                for (int i=0; i< data.size(); i++) {
                        key = ((String) keyArray[i]).toString();
                        vals = (String[]) data.get(key);
                        switch (dbweb.getTypeOf(key)) {
                                case DbWeb.NUMERIC_TYPE:
                                        res.put(key, vals);
                                        break;
                                case DbWeb.STRING_TYPE:
                                        aux = new String[vals.length];
                                        for (int j = 0; j < vals.length; j++) {
                                            aux[j] = Utils.singleQuote(vals[j]);
                                        }
                                        res.put(key, aux);
                                        break;
                        }
                }
                return res;
        }

        /**
         * Obtiene el nombre de los campos captados por una forma
         * @param req
         * @return
         */
        @SuppressWarnings("unchecked")
        public static String[] getFields(HttpServletRequest req) {
                SortedHashtable aux = new SortedHashtable();
                Map map = req.getParameterMap();
                aux.putAll(map);
                Enumeration e;
                e=aux.keys();
                String res[] = new String[aux.size()];
                int i = 0;
                while (e.hasMoreElements()) {
                        res[i] = (String) e.nextElement();
                        i++;
                }
                return res;
```



```
        }

        /**
         * Obtiene los campos y valores capturados en una forma
         * @param req
         * @return
         */
        @SuppressWarnings("unchecked")
        public static SortedHashtable getFieldsAndValues(
                        HttpServletRequest req) {
                SortedHashtable res = new SortedHashtable();
                try {
                        req.setCharacterEncoding("ISO8859_1");
                }
                catch (java.io.UnsupportedEncodingException e) {
                        e.printStackTrace();
                }
                Map map = req.getParameterMap();
                res.putAll(map);
                return res;
        }

        /**
         * Obtiene los valores de una forma
         * @param req
         * @return
         */
        @SuppressWarnings("unchecked")
        public static String[] getValues(HttpServletRequest req) {
                String key, val;
                SortedHashtable aux = new SortedHashtable();
                Map map = req.getParameterMap();
                aux.putAll(map);
                Enumeration e;
                e=aux.keys();
                String res[] = new String[aux.size()];
                for (int i = 0; i < aux.size(); i++) {
                        res[i] = new String();
                }
                int i = 0;
                while (e.hasMoreElements()) {
                        key = (String) e.nextElement();
                        val = (String) aux.get(key);
                        res[i] = val;
                        i++;
                }
                return res;
        }
}
```