Universidad de las Ciencias Informáticas
Facultad 3
CEGEL

Título:

Sistema Informático para la Evaluación de los Especialistas de Seguridad del Control de Acceso de los Sistemas de Información del Centro de Gobierno Electrónico.

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

Autor:
Arian González Au

Tutores:
MsC. Maribel Silva Muñoz
Ing. Juniel Tamayo Hernández

Ciudad de la Habana, Cuba 2015

La vida se encuentra plagada de retos, y uno de ellos es la universidad. Tras vernos dentro de ella, nos damos cuenta que más allá de ser un reto, es una base no solo para nuestro entendimiento del campo en el que estamos inmersos, sino para lo que concierne a la vida y nuestro futuro.

DECLARACIÓN DE AUTORÍA

Declaro por este medio que yo _____, con Carnet de Identidad _____ soy el autor principal del trabajo final Sistema Informático para la Evaluación de los Especialistas de Seguridad del Control de Acceso de los Sistemas de Información del Centro de Gobierno Electrónico y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo. Y para que así conste, firmo la presente declaración jurada de autoría en La Habana a los __ días del mes de _____ del año _____.

_____ _____

Arian González Au MsC. Maribel Silva Muñoz

_____

Ing. Juniel Tamayo Hernández

DEDICATORIA

Este trabajo está dedicado a personas muy importantes, que siempre han estado a mi lado apoyándome y dando lo mejor de sí para que siga progresando personal y profesionalmente.

Principalmente a mis padres por haberme forjado como la persona que soy y por motivarme constantemente para alcanzar mis anhelos, muchos de mis logros se los debo a ustedes entre los que se incluye este.

A mi madre por regalarme el amor y el significado de lo que es la verdadera familia.

A mi padre por siempre ofrecernos lo mejor de él. Por trabajar duro y sin importar el cansancio siempre tener una sonrisa que regalarnos.

A mi hermana por permitirme ser su amigo y su ejemplo.

A mis Abuelos por regalarme todo su amor y a mi familia de manera general.

.

AGRADECIMIENTOS

A Dios por darme la oportunidad de vivir y por estar conmigo en cada paso que doy, por fortalecer mi corazón e iluminar mi mente. Gracias por darme una familia maravillosa y haber puesto en mi camino a aquellas personas que han sido mi soporte y compañía durante todo el periodo de estudio y el transcurso de mi vida.

A mis padres María y Lázaro, por ser el pilar fundamental en todo lo que soy, en toda mi educación, tanto académica, como en la vida, por su apoyo incondicional mantenido a través de los años. Por ser mi ejemplo de fuerza y superación.

A mi hermana que la amo y que vea en mí un ejemplo a seguir.

A Arianna por quererme siempre, por permitirme conocerla y regalarme su corazón.

A mis tutores Maribel y Juniel, por todo el apoyo y la paciencia desinteresada que siempre me brindaron.

A Rosalina por ser una excelente profesora y sobre todo ser mi amiga incondicional.

A Anie que siempre estuvo disponible y me ayudo a prepararme para las pruebas de IA.

En la vida hay personas que te marcan con su huella, personas que te iluminan el camino, hasta que sin darte cuenta se convierten en tu motor impulsor, que a pesar de la distancia nunca se detiene. A mis amigos que ya son parte de mi familia Yamisleidys, Mairena, Carlos, Anabel, Diana y Ebrik.
También están aquellos amigos de Fiestas y Estudios que la vida me ha puesto en el camino y me ha permitido conocer, pero sobre todo esos amigos que siempre han estado en las buenas y malas.
A todas esas personas Muchas Gracias!

RESUMEN

La información se ha convertido en un activo de interés para las organizaciones. Los Sistemas de Información se encuentran expuestos a múltiples amenazas y vulnerabilidades provocadas por accesos no autorizados a la información que gestionan. La evaluación de la seguridad del Control de Acceso de la información de los Sistemas de Informaciones es un objetivo esencial de los directivos de las organizaciones. Los especialistas que tienen la responsabilidad de llevar a cabo este proceso deben tener las competencias adecuadas. En el Centro de Gobierno Electrónico la evaluación de la seguridad es realizada por alguno de los especialistas que desarrollan los diferentes proyectos productivos. No existe un proceso de selección del personal acorde a las competencias que poseen en el área de la seguridad del Control de Acceso. Este trabajo presenta un sistema informático para la evaluación de las competencias de los especialistas de la seguridad del Control de Acceso de los Sistemas de Información que se desarrollan en centros productivos de la Universidad de las Ciencias Informáticas. Está basado en un conjunto de criterios y un algoritmo de evaluación. Contribuye a la fiabilidad del proceso de evaluación de la seguridad del Control de Acceso.

PALABRAS CLAVE

Competencia, Control de Acceso, Especialista, Evaluación, Método, Seguridad

ÍNDICE

Introducción....1

CAPÍTULO 1: Fundamento Teórico....6

1.1- Normas o estándares para la evaluación de especialistas de la seguridad del Control de Acceso de los Sistemas de Información. 6

1.1.1- ISO/IEC 27002....6

1.1.2- ISO/IEC 15504....6

1.1.3- ISO/IEC 15408 Criterio Común 6

1.1.4- ISO 19011....7

1.1.5- SOAR....7

1.1.6- FIPS 199 y 200 7

1.1.7- Resolución 127/2007....7

1.1.8- Resolución 60/11 7

1.1.9- Decreto Ley 281....7

1.1.10- Ley 199 del MININT....7

1.2- Tecnologías existentes para la evaluación de especialistas de seguridad del Control de Acceso de los Sistemas de Información. 8

1.2.1- Objetivos de Control para la Información y la Tecnología relacionada (COBIT 8

1.2.2- Metodología de análisis y gestión de riesgos de los SI de las administraciones públicas ...8

1.2.3- Biblioteca de Infraestructura de TI 8

1.2.4- Publicaciones 800-12 y 800-53 del NIST 8

1.2.5- Proyecto de los 10 Mejores Elementos....8

1.2.6- Las guías de OWASP....9

1.2.7- Norma de Buenas Prácticas para la Seguridad de la Información 9

1.3- Sistemas para la evaluación de competencias de los especialistas de la seguridad del Control de Acceso. 9

1.3.1- Sistema de Puntuación de Vulnerabilidades Comunes....9

1.3.2- Técnica Ajustada de Puntuación de Riesgos de Negocio. 9
1.3.3- Suite GESPRO 10
1.3.4- Sistema Informático para la Evaluación de la seguridad del Control de Acceso 10
1.4- Tendencias actuales para el desarrollo de software. 10
1.4.1- Metodologías de desarrollo. 10
1.4.2- Herramientas de Ingeniería de Software Asistida por Computadora. 12
1.4.3- Lenguajes de modelado. 13
1.4.4- Lenguajes de programación. 14
1.4.5- Marcos de trabajo para Java. 15
1.4.6- Entornos de Desarrollo Integrado. 16
1.4.7- Sistemas de Gestores de Bases de Datos (SGBD). 17
1.4.8- Tipo de Arquitectura. 18
1.4.9- Métricas de desarrollo de software. 19
1.4.10- Pruebas. 19
Conclusiones del capítulo: 21
CAPÍTULO 2: Solución Informática. 22
2.1- SEESCA. 22
2.2- Componentes de la solución informática. 22
2.2.1- Criterios utilizados en el desarrollo de la aplicación. 22
2.2.2- Algoritmo Lógica de Puntuación de Preferencias adaptado. 26
2.2.3- Requisitos del Sistema. 27
2.2.4- Historias de Usuario. 29
2.2.5- Tarjetas Clase-Responsabilidad-Colaboración. 33
2.2.6- Arquitectura del sistema. 34
2.2.7- Patrones de Diseño. 35
2.2.8- Modelo de Datos. 40
Conclusiones del capítulo: 41

CAPITULO 3: Análisis de los resultados. 42
3.1- Resultados de las Métricas de Desarrollo. 42
3.1.1- Tamaño Operacional de las Clases. 42
3.1.2- Relación entre Clases. 45
3.2- Resultados de la Pruebas. 47
3.2.1- Complejidad Ciclomática. 47
3.2.2- Prueba del Camino Básico. 49
3.2.3- Pruebas de Caja Negra. 51
3.2.4- Resultados de las Pruebas de Aceptación. 52
Conclusiones del capítulo: 53
Conclusiones Generales: 54
REFERENCIAS BIBLIOGRÁFICAS: 56

Índice de Tablas:

Tabla 1: Relación entre criterios, las normas y estándares para la evaluación de las competencias de los especialistas de seguridad. 23
Tabla 2: Relación entre criterios, las normas y estándares para la evaluación de las competencias de los especialistas de seguridad. 24
Tabla 3: Relación entre criterios, las normas y estándares para la evaluación de las competencias de los especialistas de seguridad. 25
Tabla 4. HU Gestionar Criterio. 29
Tabla 5. HU Gestionar Especialista. 30
Tabla 6. HU Evaluar Especialista. 31
Tabla 7. CRC Gestionar Criterio. 33
Tabla 8. CRC Gestionar Especialista. 33
Tabla 9: CRC Evaluar Especialista. 34
Tabla 10. CRC gnEspecialista. 34
Tabla 11. Resultados obtenidos en la aplicación de la métrica TOC en las clases del paquete *GestoresAD*. 43
Tabla 12. Resultados obtenidos en la aplicación de la métrica RC en las clases del paquete *GestoresAD*. 45
Tabla 16. Iteraciones de los casos de pruebas de Caja Negra. 51
Tabla 17. Descripción de las variables del caso de prueba Adicionar Especialista. 51
Tabla 14. Caso de prueba para la HU Evaluar Especialista. 52
Tabla 15. Iteraciones de pruebas funcionales. 53
Tabla 18. HU Adicionar Criterio. 63
Tabla 19. HU Adicionar Especialista. 63
Tabla 20. HU Modificar Criterio. 64
Tabla 21. HU Modificar Especialista. 64
Tabla 22. HU Eliminar Criterio. 65
Tabla 24. HU Eliminar Especialista. 65

Índice de Figuras:

Figura 1: Estructura del algoritmo LSP Adaptado (Bayucan, 1996). 27
Figura 2. Interfaz Gestionar Criterio. 30
Figura 3. Interfaz Gestionar Especialista. 31
Figura 4. Interfaz Evaluar Especialista. 32
Figura 5: Arquitectura de SEESCA. 35
Figura 6: Ejemplo de uso del Patrón Creador. 36
Figura 7: Ejemplo de uso del Patrón Creador. 37
Figura 8. Ejemplo de uso del Patrón Experto. 37
Figura 9. Ejemplo de uso en el sistema del patrón Controlador. 38
Figura 10. Ejemplo de uso en el sistema del patrón Alta Cohesión. 38
Figura 11. Ejemplo de uso en el sistema del patrón Bajo Acoplamiento. 39
Figura 12. Ejemplo de uso en el sistema del patrón Instancia Única. 39
Figura 13. Modelo de Datos de SEESCA. 40
Figura 14. Atributo de calidad que representa la responsabilidad después de aplicada la métrica TOC. 44
Figura 15. Atributo de calidad que representa la complejidad después de aplicada la métrica TOC. . 44
Figura 16. Atributo de calidad que representa la reutilización después de aplicada la métrica TOC. . 44
Figura 17. Atributo de calidad que representa el acoplamiento después de aplicada la métrica RC. . 46
Figura 18. Atributo de calidad que representa la complejidad de mantenimiento después de aplicada la métrica RC. 46
Figura 19. Atributo de calidad que representa la cantidad de pruebas después de aplicada la métrica RC. 46
Figura 20. Atributo de calidad que representa la reutilización después de aplicada la métrica RC. ... 47
Figura 21. Ejemplo del código utilizado para calcular la complejidad ciclomática. 48
Figura 22. Grafo que muestra el camino básico de la funcionalidad Adicionar Especialista. 50
Figura 23. Descripción del caso de prueba Adicionar Especialista. 52
Figura 24. Interfaz Reportes de Evaluaciones. 66
Figura 25. Interfaz de Contacto. 67

Introducción

En la actualidad la información se ha convertido en un activo significativo para el éxito y la persistencia en el mercado de cualquier organización. Los procesos empresariales en las organizaciones se hacen cada vez más dependientes de la tecnología para llevar a cabo sus objetivos y el desarrollo de la información. Esto conlleva a que los Sistemas de Información (SI) se encuentren constantemente expuestos a múltiples amenazas y ambientes agresivos.
Según (SAMUELSON, 1977) un SI es la combinación de recursos humanos y materiales que resultan de las operaciones de almacenar, recuperar y usar datos con el propósito de una gestión eficiente en las operaciones de las organizaciones los cuales se encargan de la administración de procesos y están asociados a un único dominio de seguridad donde la gestión de las identidades es responsabilidad de la propia organización (SENN, 2001). Entre las diferentes funciones que realizan se encuentra la recopilación de datos y el apoyo a la toma de decisiones. La seguridad de la información de los SI es un objetivo primordial para los directivos de las organizaciones ya que permite un mejor control de los datos.

La evaluación de la seguridad es un elemento importante que permite detectar y corregir riesgos de acceso no autorizado a la información. Esta se obtiene implementando un adecuado conjunto de controles que incluye políticas, procesos, procedimientos, estructuras organizacionales, funciones de software y hardware con lo que se contribuye al flujo de fondos, la ventaja competitiva, la productividad y a hacer cumplir las leyes (Cordero Calderón, 2008).

El Control de Acceso es un mecanismo que garantiza que solo tengan acceso a un SI o a la información que éste contiene, aquellas personas o usuarios debidamente autorizados para ello. Se implementa utilizando técnicas de software y de hardware que por lo general incluyen: identificación y autenticación de usuarios; limitación de acceso, monitorización de las acciones de los usuarios y un sistema de auditoría (Resolución No. 127 /2007, 2007). Protege a los SI y por extensión a la información que estos contienen, evitando el acceso no autorizado a la misma.

La Universidad de las Ciencias Informáticas (UCI) se encuentra estructurada en diferentes centros productivos entre los que se destaca el Centro de Gobierno Electrónico (CEGEL). En este centro se desarrollan SI que gestionan información sensible y confidencial del gobierno cubano. Esto conlleva a que sea necesario contribuir a la seguridad de la información de los SI. Los SI que se desarrollan en

el CEGEL son entre otros: Sistema de Gestión Fiscal SIGEF II, Sistema de Información de Tribunales, Cámara de Comercio de la República de Cuba.

La evaluación de los SI que se desarrollan en el CEGEL se realiza mediante la contratación del servicio del Centro Nacional de Calidad de Software (Calisoft). Según encuestas realizadas (Muñoz, 2014) este proceso involucra el esfuerzo de un grupo de aproximadamente seis especialistas que aplican controles de seguridad extraídos de normas y estándares para la evaluación de la seguridad. Los especialistas que llevan a cabo el proceso de evaluación en la entidad son especialistas en informática, los cuales han ido adquiriendo conocimientos de manera empírica en el área de la evaluación de la seguridad del Control de Acceso.

Actualmente no existe un proceso para la selección de los especialistas que evalúan los proyectos productivos desarrollados en el CEGEL que contribuya a la evaluación de la seguridad del Control de Acceso de los SI.

A continuación se especifican un conjunto de limitaciones asociadas al proceso de selección de los especialistas de seguridad que atentan contra el desarrollo exitoso del proceso de evaluación de la seguridad del Control de Acceso de los SI.

- No existe un método para medir las competencias de los especialistas.
- Es insuficiente el desarrollo de políticas claras y mejores prácticas para la evaluación de las competencias de los especialistas.
- No existen objetivos de control relacionados con la evaluación de las competencias de los especialistas.
- No se lleva a cabo la gestión dinámica de normas, estándares, guías, recomendaciones, resoluciones y decretos.
- No existe un lenguaje común para comunicar las metas, objetivos y resultados de la evaluación de las competencias de los especialistas.

El Modelo para la Evaluación de la Seguridad del Control de Acceso de los SI (MECA) provee solución a un conjunto de escenarios no cubiertos por los modelos existentes en la actualidad. La interrelación de los componentes del modelo permite evaluar eficientemente los procesos de Identificación y Autenticación, Autorización y Auditoría. Este modelo fue probado satisfactoriamente

en cinco de los proyectos del CEGEL, su continua aplicación ya depende totalmente de la organización en cuestión (Muñoz, 2014).

El MECA define 10 indicadores para la evaluación. Estos indicadores sintetizan los controles de las principales directrices en el área de la evaluación de la seguridad del Control de Acceso de aplicación internacional. Se definen 27 criterios medibles que describen las diferentes soluciones de Control de Acceso y contribuyen a la evaluación de los procesos de Identificación y Autenticación, Autorización y Auditoría (Muñoz, 2014). El proceso de evaluación se realiza mediante la adaptación del algoritmo LSP a las condiciones del modelo permitiendo conocer el estado (Bajo, Medio o Alto) de cada uno de los procesos del Control de Acceso y conocer sobre los criterios que incidieron en los resultados obtenidos. La entrada del modelo es la información proveniente de los SI de centros productivos de la UCI que utilizan un mecanismo de Control de Acceso de la información. Las salidas del modelo se reflejan en el estado de los procesos de Identificación y Autenticación, Autorización y Auditoría de los SI y en las recomendaciones derivadas del análisis de los resultados de la evaluación.

En la segunda etapa del modelo tiene lugar la selección del especialista de seguridad que evalúa el mecanismo de Control de Acceso del SI que se evalúa. Es conveniente que este especialista cumpla con ciertas competencias según lo que establecen las principales directrices en el área de la seguridad del Control de Acceso de los SI, para que la evaluación de la seguridad pueda ser exitosa en la medida en que se obtengan resultados.

El proceso de selección del responsable de evaluar la seguridad del Control de Acceso puede ser complejo. La eficiencia del mismo está condicionada por la medición adecuada de las competencias que posee el especialista correspondiente. El proceso de evaluación según MECA no incluye la selección de un especialista de seguridad que posea las competencias necesarias según normas y estándares en el área de la seguridad del Control de Acceso. Esto provoca que la evaluación obtenida según MECA no sea fiable en lo que se refiere al conocimiento que posee el evaluador.

Considerando la problemática anteriormente descrita se asumió como **problema de la investigación**: ¿Cómo evaluar las competencias de los especialistas para determinar el responsable de la seguridad del Control de Acceso de los Sistemas de Información? Luego de un análisis del problema antes planteado se tuvo como **objeto de estudio**: La Seguridad del Control de Acceso de los Sistemas de Información. El **campo de acción** en que se enmarcó la investigación es: el Proceso de desarrollo de software para la evaluación de las competencias de los especialistas de la seguridad del Control de Acceso de los Sistemas de Información del CEGEL. Para dar solución al problema de

la investigación se definió como **objetivo general**: Desarrollar un Sistema Informático para la evaluación de las competencias de los especialistas de la seguridad del Control de Acceso de los Sistemas de Información del CEGEL.

Para cumplir el objetivo general fue necesario definir los siguientes **objetivos específicos**:

- Definir el marco teórico conceptual de la investigación, relacionado con los sistemas más utilizados para la evaluación de las competencias de los especialistas de la seguridad del Control de Acceso de los Sistemas de Información.
- Realizar el análisis y diseño del sistema informático para la evaluación de las competencias de los especialistas de seguridad del Control de Acceso de los Sistemas de Información mediante la metodología de desarrollo seleccionada.
- Implementar el diseño propuesto para la evaluación de las competencias de los especialistas de seguridad del Control de Acceso de los Sistemas de Información mediante las herramientas y tecnologías seleccionadas para la obtención del Sistema Informático.
- Validar el sistema informático propuesto mediante pruebas de software empleando los métodos caja negra, caja blanca y mediante la aplicación de las pruebas de usuario para la aceptación del producto.

Se define como **idea a defender**: El desarrollo de un Sistema Informático para la evaluación de las competencias de los especialistas de la seguridad del Control de Acceso de los Sistemas de Información, permitirá determinar el responsable de la seguridad del Control de Acceso de los Sistemas de Información.

Para realizar la investigación se utilizaron diferentes métodos científicos. Entre los métodos teóricos utilizados está el Analítico Sintético mediante el cual se realizó un análisis de los documentos, publicaciones y en general toda la información relacionada con las competencias que debe tener un especialista para evaluar la seguridad del Control de Acceso de un SI. El método teórico Histórico Lógico se utilizó para el estudio del estado del arte, permitiendo evaluar el desarrollo de los marcos de trabajo existentes, la evolución de los sistemas y herramientas de desarrollo que se utilizaran en el desarrollo de la solución propuesta. Entre los métodos empíricos se utilizó la Entrevista, la cual permitió obtener información valiosa sobre las necesidades del cliente identificando los requisitos para el desarrollo de software y la forma actual en la que se realiza este proceso de evaluación.

El trabajo se encuentra estructurado en 3 capítulos. En el Capítulo 1 se presentan las normas y estándares que constituyen las directrices en el área de evaluación de las competencias de los

especialistas de la seguridad del Control de Acceso, además las soluciones que existen actualmente para llevar a cabo este proceso. Se realiza el análisis de la metodología de desarrollo de software a usar, así como el lenguaje de programación y las herramientas apropiadas para darle solución al problema a resolver. En el Capítulo 2 se presentan los componentes del sistema informático y las tecnologías utilizadas para el desarrollo del mismo. Finalmente en el Capítulo 3 se presentan las pruebas hechas al software.

CAPÍTULO 1: Fundamento Teórico.

En el presente capítulo se presentan aquellos conceptos, valoraciones y teorías necesarias para la comprensión del problema de investigación antes descrito, realizando principalmente un análisis detallado y crítico de las diferentes normas, estándares y resoluciones existentes en el mundo referente a la evaluación de las competencias de los especialistas en el área de la seguridad del Control de Acceso. Se realiza un estudio de los sistemas informáticos para la evaluación de procesos de manera general y de la seguridad del Control de Acceso en particular. Se hizo un análisis de las metodologías de desarrollo, lenguajes de modelado, herramientas de modelado, marcos de trabajo, entornos de desarrollo y sistemas para la gestión de las bases de datos. Además de analizar las soluciones existentes para la evaluación de las competencias de un especialista de la seguridad del Control de Acceso con el fin de valorar la pertinencia de cada una en el área en la que se investiga.

1.1- Normas o estándares para la evaluación de especialistas de la seguridad del Control de Acceso de los Sistemas de Información.

En la actualidad existen diferentes normas, estándares, resoluciones, decretos y tecnologías tanto en el ámbito nacional como internacional que son utilizadas para dirigir la fiabilidad de la evaluación del Control de Acceso. Estas directrices permitieron establecer un conjunto de competencias con las cuales debe cumplir el especialista encargado de evaluar la seguridad del Control de Acceso. A continuación se presentan las principales directrices área de la seguridad del Control de Acceso que apoyan el proceso de evaluación.

1.1.1- ISO/IEC 27002: establece un conjunto de controles para la evaluación de la seguridad estableciendo los lineamientos y los principios generales de forma que se pueda iniciar, implementar, mantener y mejorar la gestión de la seguridad de la información en una organización (STANDARD, 2005).

1.1.2- ISO/IEC 15504: conocido como Determinación de la Capacidad de Mejora del Proceso de Software es un modelo para la mejora y evaluación de los procesos de desarrollo, además del mantenimiento de SI y productos de software (Jalote, 2000).

1.1.3- ISO/IEC 15408 Criterio Común (CC): posibilita las evaluaciones de productos y sistemas de seguridad en un esfuerzo para definir una evaluación de la seguridad de TI. Constituye una guía útil para la gestión de productos y sistemas de TI con funciones de seguridad (Herrmann, 2001).

1.1.4- ISO 19011: ofrece guías sobre principios de auditorías y sobre evaluación de competencias de individuos involucrados en el proceso de auditorías. Incluye a las personas para administrar el programa de auditorías, los auditores y los equipos de auditores (ISO-19011, 2011).

1.1.5- SOAR (Estado del informe de arte): describe las actividades actuales y emergentes, enfoques, tecnologías y entidades que contribuyen directamente a la disciplina de la seguridad del software, es decir, el arte y la ciencia de la producción de software seguro. Realiza una evaluación, análisis de riesgos y amenazas que son fuentes clave de requisitos que afectan a la seguridad del software (IATAC, 2007).

1.1.6- FIPS 199 y 200: FIPS 199 define tres niveles de potencial impacto en las organizaciones al existir una violación de seguridad. Estas definiciones deben tener lugar en el contexto de cada organización (FIPS-PUB-199, February 2004.). FIPS 200 señala la necesidad de desarrollar, documentar e implementar un programa para toda empresa proporcionando seguridad a los SI. Especifica cuáles son los requisitos mínimos de seguridad para los SI (FIPS-PUB-200, 2006).

1.1.7- Resolución 127/2007: norma cubana que tiene por objeto establecer los requerimientos que rigen la seguridad de las tecnologías de la información y garantizar un respaldo legal que responda a las condiciones y necesidades del proceso de informatización del país (Resolución No. 127 /2007, 2007).

1.1.8- Resolución 60/11: establece normas y principios básicos de obligada observancia para la Contraloría General de la República y los sujetos a las acciones de auditoría, supervisión y control de este Órgano. Constituye un modelo estándar del Sistema de Control Interno (ResoluciónNo.60, 2012).

1.1.9- Decreto Ley 281: constituye un conjunto de disposiciones legales que establecen los principios para la integración del Sistema de Información del Gobierno determinado por su organización y funcionamiento. Evalúa propuestas de políticas generales sobre la dirección del Sistema de Información del Gobierno, contribuye a garantizar su coherencia y adecuado funcionamiento (Decreto-Ley, 2011).

1.1.10- Ley 199 del MININT: establece y regula las normas, procedimientos y responsabilidades para el empleo de la Criptografía y el Servicio Cifrado. Los Servicios de Cifrado deben aplicarse y cumplirse en los órganos, organismos, entidades y sus dependencias o por cualquier otra persona jurídica radicada en el territorio nacional y las personas naturales residentes en el país (Decreto-Ley-No.199, 1999).

1.2- Tecnologías existentes para la evaluación de especialistas de seguridad del Control de Acceso de los Sistemas de Información.

El objetivo de las tecnologías para la evaluación de las competencias de los especialistas de la seguridad del Control de Acceso, es brindar un conjunto de elementos relacionados con la seguridad del Control de Acceso de los SI. Estas posibilitan realizar una adecuada evaluación, teniendo en cuenta un conjunto de competencias que debe tener un el especialista que desarrolle esta tarea. A continuación se presentan estas tecnologías que proporcionan buenas prácticas para un adecuado desarrollo del Control de Acceso:

1.2.1- Objetivos de Control para la Información y la Tecnología relacionada (COBIT): marco de trabajo que proporciona buenas prácticas combinando exitosamente el control y la Tecnología de Información (TI). Tiene como misión: investigar, desarrollar, publicar y promover un conjunto internacional y actualizado de objetivos de control para las TI que sea de uso cotidiano para gerentes y auditores (Domínguez, 2004).

1.2.2- Metodología de análisis y gestión de riesgos de los SI de las administraciones públicas (MAGERIT): se encarga de alertar a los responsables de los SI de la existencia de riesgos y que éstos sean controlados antes de que se materialicen, ofrecer un método sistemático para el análisis de dichos riesgos, ayudando a descubrirlos y manteniéndolos bajo control (MAGERIT, 2006).

1.2.3- Biblioteca de Infraestructura de TI (ITIL): no es solamente un marco de referencia, sino una metodología y filosofía enfocada en el manejo de la calidad total (OGC, 2005). Permite cumplir con los atributos de disponibilidad, confidencialidad, integridad, autenticidad y no repudio de la información (Garzaro, 2007).

1.2.4- Publicaciones 800-12 y 800-53 del NIST: proponen un conjunto de controles de seguridad con el fin de que sean implementados en una organización para brindar protección a la información y a los SI de tipo federal (PB800-53, 2009). 800-12, ilustra los beneficios de los controles de seguridad, las mayores técnicas o enfoques para cada control. Además proporciona asesoramiento y orientación para aquellos que tienen responsabilidades en seguridad informática (NIST-PUB-12).

1.2.5- Proyecto de los 10 Mejores Elementos: integra parte del Proyecto de Seguridad para Aplicaciones Web de Código Abierto (OWASP por sus siglas en inglés). Contribuye a lograr la

seguridad de aplicaciones y gestionar el riesgo que las aplicaciones de software crean en su organización (OWASP, 2010).

1.2.6- Las guías de OWASP: permiten verificar la seguridad que el sistema ejecuta. Mantienen el enfoque de un problema general, sin proporcionar suficiente información para encontrar, diagnosticar y resolver los problemas de seguridad (OWASP, 2008).

1.2.7- Norma de Buenas Prácticas para la Seguridad de la Información: contiene una amplia gama de características que cubre todo el espectro de los arreglos que deben hacerse para mantener los riesgos de negocio asociados con los SI. Propone elementos para el desarrollo de la arquitectura de seguridad en toda la organización (2007).

1.3- Sistemas para la evaluación de competencias de los especialistas de la seguridad del Control de Acceso.

En la actualidad existen diferentes soluciones para la evaluación en disimiles áreas, por lo que se realizará un estudio de dichas soluciones con el objetivo de conocerlas a profundidad. Se hace necesario determinar sus ventajas, desventajas y si brindan solución al problema antes planteado.
A continuación se presentan las definiciones y características de diferentes sistemas internacionales utilizados para la evaluación.

1.3.1- Sistema de Puntuación de Vulnerabilidades Comunes.

CVSS por sus siglas en ingles es un estándar abierto que se aplica universalmente a cualquier vulnerabilidad informática y permite obtener una evaluación del riesgo asociado (NIST, 2007). A pesar de que proporciona a los usuarios una representación clara e intuitiva de una vulnerabilidad y una taxonomía común para la descripción sus métricas base consideran mayormente la Autenticación como proceso del Control de Acceso, no siendo así los procesos de Autorización y Auditoría.

1.3.2- Técnica Ajustada de Puntuación de Riesgos de Negocio.

BAR (por sus siglas en inglés) esta técnica permite la clasificación de defectos de seguridad de acuerdo a su tipo de vulnerabilidad, el grado de riesgo y posible impacto (BARABANOV, 2011), (JAQUITH, 2002). A pesar de que permite clasificar defectos de seguridad por su respectivo tipo de seguridad, el método mide impacto y riesgo en el negocio. No considera indicadores o criterios asociados a los procesos de Identificación y Autenticación, Autorización y Auditoría.

A continuación se presentan las definiciones y características de diferentes sistemas nacionales utilizados para la evaluación.

1.3.3- Suite GESPRO (Gestor de Proyectos).

La Suite Gestor de Proyectos es un sistema informático para la dirección integrada de proyectos. Permite el control de la ejecución de proyectos basado en indicadores relacionados con la gestión del tiempo, costos, calidad, logística y el rendimiento de los recursos humanos. Permite la evaluación de la ejecución del proyecto en un corte haciendo uso de los indicadores propuestos permitiendo mejorar la capacidad de ayuda a la toma de decisiones (García, 2012). Este sistema no utiliza criterios o indicadores medibles para la evaluación de la seguridad del Control de Acceso, ni utiliza algoritmos de evaluación.

1.3.4- Sistema Informático para la Evaluación de la seguridad del Control de Acceso.

SIECA garantiza la fiabilidad de la evaluación de la seguridad del Control de Acceso en los SI sobre la base del uso de tecnologías libres, multiplataforma, con alto grado de seguridad y adaptabilidad. Para su desarrollo se tuvieron en cuenta un conjunto de criterios e indicadores de evaluación. SIECA posee una base de datos que debe ser instalada en las estaciones de trabajo donde se realice la evaluación lo cual limita su portabilidad. De ahí que ello provoque recursos y tiempos adicionales al proceso de evaluación de la seguridad del Control de Acceso (Muñoz, 2014).

1.4- Tendencias actuales para el desarrollo de software.

El SEESCA es un sistema informático que contribuye a la selección efectiva de los especialistas que evalúan la seguridad del Control de Acceso de los SI de los centros productivos de la UCI. Permite añadir nuevos criterios de evaluación para seguidamente poder insertar un nuevo especialista al que se le quiera realizar una evaluación. Este sistema permite la evaluación correspondiente al especialista y almacenar los un conjunto de reportes de dichas evaluaciones. Es un sistema que contiene requisitos poco complejos, por lo que necesita del esfuerzo de un solo desarrollador ya que es un sistema pequeño. EL SEESCA no precisa ser distribuido, ya que será utilizado como herramienta de trabajo para los directivos de las organizaciones.

1.4.1- Metodologías de desarrollo.

Las metodologías de desarrollo de software son un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información (CMS, 2008), son las encargadas de definir quién hace qué, cuándo y cómo lo tiene que hacer. Existen dos importantes

tendencias las cuales agrupan las diferentes tecnologías. Las pesadas que son utilizadas para desarrollar proyectos de gran envergadura que exigen una cantidad de tiempo y un personal para el desarrollo. Por otra parte las metodologías ágiles son utilizadas para proyectos menos formales, donde existen pocos desarrolladores y el tiempo es una limitante.

- **Crystal Clear.**

Crystal Clear desarrollada por el investigador de IBM el Dr. Alistair Cockburn, es una metodología centrada en el factor humano. Se centra en tres propiedades claves: efectuar entregas frecuentes, que consiste en liberar código ejecutable y validado a usuarios reales cada pocas semanas o meses; realizar una mejora reflexiva donde todo el equipo unido en un poco tiempo determinara qué está y qué no está funcionado. Otras propiedades importantes que pueden ser añadidas para incrementar la seguridad del proyecto son: crear confianza personal entre los miembros del equipo, mantener el foco en la tarea, tener acceso fácil a usuarios expertos y trabajar en un ambiente técnico con pruebas automáticas, administración de configuración e integración frecuente (Abreu, 2008).

- **SCRUM**

Fue desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Scrum es una metodología que no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto. Scrum es una metodología ágil, y como tal:

- Es un modo de desarrollo de carácter adaptable más que predictivo.
- Orientado a las personas más que a los procesos.
- Emplea la estructura de desarrollo ágil: incremental basada en iteraciones y revisiones (Palacio, 2006).
- El equipo de desarrollo, se dirige hacia un único objetivo: construir un producto de calidad, por lo que la gestión de proyecto es la encargada de definir las características del producto, así como de eliminar cualquier obstáculo que impida el cumplimiento del objetivo final del equipo (Figueroa, y otros, 2010).

- **Programación Extrema.**

La Programación Extrema (XP) es una metodología ágil de desarrollo de software ideada principalmente por Kent Beck en 1999. Consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo al usuario final. El equipo de desarrollo sigue estrictamente el orden de prioridad de las tareas definidas por el cliente (Letelier, y otros, 2008). El

ciclo de vida de XP consiste básicamente de seis fases: Exploración, Planificación, Iteraciones en la edición, Producción, Mantenimiento y Muerte (Acosta, 2010). Sus principales características son:

- Diseñada principalmente para entornos dinámicos.
- Pensada para equipos pequeños (pocos programadores)
- Orientada hacia la codificación.
- Hace énfasis en la comunicación informal, verbal con el cliente.
- Utiliza poco tiempo de desarrollo.

El análisis de las metodologías de desarrollo estudiadas, permitió la selección de la metodología XP debido a que se adapta a las necesidades del desarrollo con relación a tiempo, entorno, cantidad de desarrolladores e interacción estrecha del cliente con el desarrollador. Está diseñada para adaptarse de forma inmediata a los cambios y trabajar con bajos costos asociados, en cualquier etapa del ciclo de vida del software. Esta metodología presenta un modelo iterativo que permite que una fase completada pueda realizarse nuevamente con el fin de mejorarla.

1.4.2- Herramientas de Ingeniería de Software Asistida por Computadora.

Las herramientas de Ingeniería de Software Asistida por Computadora (CASE por sus siglas en inglés) son aplicaciones cuyo destino fundamental es aumentar la productividad en el desarrollo de software reduciendo la cantidad de tiempo y dinero utilizadas en su creación. A continuación se presentan las mismas:

- Rational Rose.

La herramienta CASE Rational Rose en su versión 7.0 está diseñada para la modelación visual mediante el UML para el análisis y diseño de sistemas orientados a objetos. A través de esta se puede generar código para Java y otros lenguajes de programación. Se utiliza para modelar los sistemas informáticos antes de producirlos y abarca todo el ciclo de vida del sistema (Mestras Pavón, 2007). Proporciona de acuerdo con el lenguaje UML, construcciones basadas en un modelo para la automatización del desarrollo, incluida la ejecución de modelos y la generación de código totalmente ejecutable (IBM, 2013).

- Visual Paradigm.

Es una herramienta CASE que propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Esta herramienta permite aumentar la calidad del

software, a través de la mejora de la productividad en el desarrollo y mantenimiento del software. Aumenta el conocimiento informático de una empresa ayudando así a la búsqueda de soluciones para los requisitos (Pressman, 2002).

De acuerdo al análisis antes realizado y a las características previamente mencionadas se escoge la herramienta CASE Visual Paradigm en su versión 8.0 ya que brinda soporte para el lenguaje de programación Java, permite el modelado de bases de datos y su posterior exportación a código SQL, es capaz de generar toda la documentación necesaria. Además se puede emplear en cualquier fase del desarrollo del software, por lo que contribuirá a una mayor flexibilidad y agilidad para la adaptación al cambio en los requerimientos.

1.4.3- Lenguajes de modelado.

El lenguaje de modelado es la notación (principalmente gráfica) para expresar un diseño. El proceso indica los pasos que se deben seguir para llegar a un diseño (Cornejo, 2008) en el que se utilizan un conjunto de símbolos y reglas, utilizados en la actualidad para simplificar el análisis del diseño de aplicaciones informáticas. La utilización de estos modelos permite a los ingenieros visualizar y representar el sistema que se va a construir de una manera sencilla y fácilmente comprensible, por lo que pueden utilizarse como medio de comunicación con el cliente. A continuación se especifican los lenguajes de modelados que constituyen tendencias actuales en el desarrollo de software.

- Notación de Modelado de Procesos de Negocio.

La Notación de Modelado de Procesos de Negocio (BPMN) fue publicado en el año 2004 en su versión 1.0 por su desarrollador Business Process Management Initiative (BPMI). El principal objetivo es brindar la capacidad de entender sus procedimientos internos de negocio en una notación gráfica que facilita el entendimiento de la utilidad de las transacciones de colaboración y negocios entre las organizaciones (Object Management Group, 2014).

Se utiliza como lenguaje unificador para cerrar la brecha existente entre el diseño de todos los procesos de negocios que realizan las empresas y su posterior implementación por parte de los desarrolladores.

- Lenguaje Unificado de Modelado.

El Lenguaje Unificado de Modelado (UML) es uno de los más conocidos y usados en la actualidad, fue creado por Ivar Jacobson, James Rumbaugh y Grady Booch. Este lenguaje soporta un amplio conjunto de elementos de notaciones gráficas. Describe la notación para clases, componentes,

nodos, actividades, flujos de trabajo, casos de uso, objetos, estados y como modelar la relación entre todos estos elementos. Con este lenguaje los ingenieros de software y las organizaciones pueden construir modelos rigurosos, que soporten el ciclo de vida de desarrollo del software completo (Sparks, 2006).

El UML fue seleccionado para el modelado del sistema ya que se integra correctamente con la Programación Orientada a Objetos y representa eficientemente las bases de datos que se necesitan. Permite detallar profundamente los artefactos del sistema cumpliendo con las necesidades para la creación de la solución informática.

1.4.4- Lenguajes de programación.

Un lenguaje de programación es un conjunto de reglas que especifican cuales secuencias de símbolos constituyen un programa, y lo que la ejecución del programa describe (Ben-Ari, 1996). A continuación se especifican los lenguajes de programación de alto nivel más usados en la actualidad en el desarrollo de soluciones informáticas.

- C++

C++ es un lenguaje de programación multiparadigma diseñado a mediados de los años 80 del siglo pasado por Bjarne Stroustrup con la intención de extender el uso del lenguaje C. Soporta varios estilos de programación, por ejemplo: procedural, orientado a objetos; intenta ser tan eficiente y portable como lo es el lenguaje C. Además, permite la sobrecarga de los operadores, la inclusión de directivas del preprocesador y los tipos genéricos, entre otras funcionalidades. Soporta plenamente el polimorfismo y la herencia, destacándose la existencia de la herencia múltiple (Pressman, 2002).

- C #

C Sharp (C#) es un lenguaje de programación orientado a objetos, creado en 1999 por Microsoft. Este lenguaje se diseñó para ser robusto y a la vez sencillo, su uso está extendido y sus principales aplicaciones sobresalen en los entornos de escritorio y web. Entre sus principales características y ventajas destacan que es: completamente orientado a objetos lo que facilita el uso con este tipo de paradigma de programación, soporta la encapsulación, herencia y polimorfismo, incluye mecanismos de control de acceso, gestión dinámica de la memoria del sistema para evitar el desbordamiento de la misma, posee normas de sintaxis que evitan la conversión entre tipos de datos que no son compatibles y un alto grado de manejo de excepciones.

- Java

Java fue desarrollado por Sun Microsystems a principios de los años 90, es un Lenguaje de Programación Orientado a Objetos. Las principales características que lo distinguen son:

- La robustez, ya que contiene varias funciones integradas que la garantizan, por ejemplo fomenta el uso de interfaces.
- La seguridad, ya que tiene integradas funciones de seguridad como son: el verificador de código de bytes que proporciona el primer nivel de defensas.
- Es interpretado, por lo que no son lenguajes dependientes a ningún entorno de ejecución o sistema operativo.
- Orientado a objetos ya que Java cuenta con bibliotecas de clases previamente diseñadas que se van enriqueciendo de una a otra versión y constituyen la base sobre la cual el programador trabaja (Fernández, 2010).

Para la implementación de la aplicación se ha seleccionado el lenguaje de programación Java ya que este posee una sintaxis correctamente estructurada, alto grado de robustez. Es un sistema dinámico debido a la evolución constante de sus bibliotecas, sin embargo eso no implica que los programas anteriores dejen de funcionar. Además posee un entorno de ejecución reducido y altamente optimizado, utilizado en aplicaciones de escritorio, por lo que posee una gran portabilidad.

1.4.5- Marcos de trabajo para Java.

Un marco de trabajo es un diseño abstracto orientado a objetos para un determinado tipo de aplicación, es un patrón arquitectónico que proporciona una plantilla extensible para un tipo específico de aplicaciones. Cada paquete está compuesto por componentes y estos a su vez definen una interfaz de comportamiento con el objetivo de extender sus funcionalidades y adaptarse a posibles escenarios (Martínez, 2015).

- **Swing.**

Swing es una biblioteca gráfica para Java. Incluye *widgets* para la interfaz gráfica de usuario tales como cajas de texto, botones, desplegables y tablas. Desde el punto de vista arquitectónico es un marco de trabajo Modelo Vista Controlador para desarrollar interfaces gráficas para Java con independencia de la plataforma (Llorente, 2011).

Swing se integra con la Máquina Virtual de Java desde su versión 6.0, es capaz de ejecutarse correctamente en cualquier sistema operativo gracias a que utiliza componentes ligeros y reutilizables que no dependen del entorno de desarrollo de los sistemas, sino que utiliza los propios.

Todo esto contribuye a que el desarrollo de sus componentes sea más activo y permite representar varios estilos de apariencia en dependencia del sistema operativo en el que se ejecute.

- **API de Persistencia de Java.**

La API de persistencia de java (JPA, por sus siglas en inglés) está fundamentalmente desarrollada para la plataforma Java EE, por lo que es utilizada principalmente para manejar información de bases de datos relacionales en aplicaciones de escritorio. Sus principales ventajas son que agilizan el desarrollo orientado a objetos, implementa funciones seguras contra posibles ataques, permite la persistencia de datos de una manera ágil y rápida, es compatible con buenas prácticas de diseño y permite un enfoque orientado a objetos al interactuar con una base de datos, siguiendo un patrón de mapeo de objetos (Keith, 2009).

1.4.6- Entornos de Desarrollo Integrado.

Son programas informáticos integrados por un conjunto de utilidades que posibilitan la creación de aplicaciones informáticas para equipos de cómputo, servidores, dispositivos móviles, etc. Son capaces de trabajar con varios lenguajes de programación. Generalmente están compuestos por un editor de código, un compilador, un depurador y un constructor de interfaz gráfica usuario. A continuación se especifican los diferentes entornos de desarrollo utilizados para la creación de aplicaciones informáticas.

- **Visual Studio**

Visual Studio 2010 es un Entorno de Desarrollo Integrado (IDE) desarrollado específicamente para el sistema operativo *Windows*, fue lanzado por primera vez en el año 1998. Utiliza lenguajes como Visual C++, Visual C# y otros. Se caracteriza por poseer licencia propietaria y con soporte exclusivo de Microsoft. Posibilita la creación de aplicaciones para estaciones de trabajo. Provee un conjunto de herramientas integradas que permiten un flujo de trabajo completo. Su interfaz gráfica es altamente personalizable, además reduce significativamente la complejidad de acceso a bases de datos (MSDN, 2008).

- **NetBeans en su versión 8.0.1**

Es un IDE de código abierto y una plataforma de aplicaciones que permiten a los desarrolladores la creación rápida de diferentes tipos de aplicaciones utilizando la plataforma Java. Permite escribir, compilar, hacer depuración de código, ensamblar y desplegar aplicaciones, aunque está escrito en Java, brinda soporte para toda clase de lenguajes de programación (Netbeans, 2014). Entre sus

principales características se encuentran: que es un software libre y multiplataforma que permite la gestión de aplicaciones de escritorio. Posee completa compatibilidad con sistemas Windows y Linux, integración total con herramientas CASE como Visual Paradigm y está enfocado al trabajo con herramientas para la creación de interfaces gráficas de usuario como Swing.

Para el desarrollo de la aplicación informática fue seleccionado el NetBeans en su versión 8.0.1. Se eligió por ser un IDE libre, multiplataforma, con soporte para el lenguaje de programación Java. Esta altamente integrado con herramientas para la creación de interfaces gráficas de usuario Swing. Es sencillo, amigable y tiene completa vinculación con herramientas para el control de versiones y modelado UML.

1.4.7- Sistemas de Gestores de Bases de Datos (SGBD).

Los Sistemas de Gestión de Bases de Datos (Database Management System) son software dedicados a servir de interfaz entre las bases de datos y las aplicaciones que la utilizan (Alvear Rodriguez, 2005). El objetivo primordial de un SGBD es proporcionar un entorno que sea a la vez conveniente y eficiente para ser utilizado al extraer y almacenar información de la bases de datos (Leivas, 2009). A continuación se especifican el SGBD que garantizan la portabilidad de la Base de Datos.

- MySQL.

MySql opera a través de un intérprete de comandos. Los comandos de MySQL siguen el estándar de SQL (Structured Query Language), que es un lenguaje normalizado para operar con Bases de Datos Relacionales. Es desarrollado sobre la filosofía de software libre (Oracle, 2010). Se puede obtener bajo licencia GNU si el proyecto es compatible con esta licencia de lo contrario hay que comprarlo, para agilizar las búsquedas utilizan las tablas organizadas en b-trees, optimizando así dichas búsquedas. Ofrece seguridad mediante su sistema de contraseñas y privilegios seguros, permite el uso de memoria compartida entre clientes y servidores.

- PostgreSQL.

Es un sistema para gestionar bases de datos relacionales, orientado a objetos. Su inicio se remonta al año 1982 en el proyecto “Ingres” en la Universidad de Berkeley liderado por Michael Stonebraker (PostgreSQL Core Team, 2013). Es un sistema objeto-relacional ya que incluye herencia, tipos de datos y funciones. Posibilita la gestión de múltiples procesos sobre las tablas, es decir, un usuario puede estar escribiendo y el otro puede leer sin problemas de bloqueo, es capaz de utilizar tipos de

datos creados por los propios usuarios implementa disparadores (Triggers del inglés) que a su vez posibilitan la creación de vistas, herencia de tablas e integridad transaccional, es un sistema multiusuario y multiplataforma, soporta el uso de lenguajes como C, C++, Java, etc.

- Sqlite.

Es una librería compacta y autocontenida de código abierto. Está distribuida bajo dominio público que implementa un gestor de bases de datos Lenguaje de Consulta Estructurado (SQL), no necesita de la instalación ni la configuración aparte de la propia aplicación. Sqlite es ideal para trabajar con volúmenes medianos o pequeños de información, de manera ágil y eficiente. Con SQLite, el proceso que quiere acceder a la base de datos, lee y escribe directamente en disco sin que haya servicio intermediario. De esta manera se garantiza que se puede hacer una aplicación totalmente autónoma y portable. Requiere muy poco soporte de librerías externas o del sistema operativo (Montiel, enero del 2008).

Se selecciona a Sqlite ya que provee soluciones a las necesidades existentes. Es un sistema de base de datos de código abierto. Soporta gran parte del estándar SQL. Es ideal para el trabajo con medianos o pequeños volúmenes de información, lo cual lo hace ágil. Es multiplataforma y fácil de usar. Además de garantizar la portabilidad de la base de datos siendo esta una principal característica que debe poseer la aplicación.

1.4.8- Tipo de Arquitectura.

Es una estructura de organización esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. También tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño de tal sistema.

La arquitectura en tres capas define cómo organizar el modelo de diseño en capas, que pueden estar físicamente distribuidas, lo cual conlleva a que los componentes de una capa sólo pueden hacer referencia a componentes en capas inmediatamente inferiores. Esta arquitectura es importante porque simplifica la comprensión y la organización del desarrollo de sistemas complejos, reduciendo las dependencias entre las capas. (Informática, 2010).

Brinda la posibilidad de que el desarrollo se pueda realizar en varios niveles para que en el caso de que ocurra algún cambio, sólo se modifique la capa requerida sin tener que modificar todo el código.

1.4.9- Métricas de desarrollo de software.

Las mediciones que se realizan al software, pueden ser aplicadas al proceso de desarrollo con el objetivo de mejorarlo. Se utilizan para la estimación, el control de la calidad, la evaluación de la productividad y el control de proyectos. Estas métricas contribuyen a evaluar la calidad de los resultados de trabajos técnicos y en la toma de decisiones tácticas a medida que el proyecto evoluciona (Ortiz, 2007).

- Tamaño Operacional de Clase.

El Tamaño Operacional de Clase (TOC) expresa el número de métodos asignados a una clase. El impacto que tiene su aplicación en los atributos de calidad mencionados se evidencia mayormente en responsabilidad, complejidad y reutilización. La relación está expresada porque el aumento de la cantidad de métodos de la clase es directamente proporcional a la responsabilidad y complejidad de la misma clase, lo que supone un aumento de dicha clase. De igual manera, se traduce en una reducción en la reutilización que pueda tener la misma (Pressman, 2002).

El aumento del TOC provoca consecuencias a la clase como: aumento de la responsabilidad asignada, de la complejidad de implementación y la disminución del grado de reutilización debido a que la clase posee un alto grado de especialización de sus componentes.

- Métrica Relación entre Clases.

Las relaciones entre clases, está dada por el número de relaciones de uso de una clase con otras. Se encarga de medir la calidad de acuerdo a los siguientes atributos (Lorenz, 1994).

Acoplamiento: Consiste en las conexiones físicas entre los elementos del diseño orientado a objeto, representan el acoplamiento dentro de un sistema orientado a objeto.

Complejidad del mantenimiento: Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software.

Cantidad de Pruebas: Consiste en el número o grado de esfuerzo para realizar las pruebas de calidad del producto diseñado.

Reutilización: Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.

1.4.10- Pruebas.

Durante el proceso de desarrollo de software puede que se cometan irregularidades por parte de los desarrolladores. Estas irregularidades pueden ocurrir desde el mismo comienzo de la planificación

por lo que es necesario definir una serie de pruebas que sean capaces de medir, comprobar o evaluar el estado del sistema en desarrollo, debido a que dicho estado es un elemento crítico para determinar su calidad. Estas pruebas evitan efectos colaterales no deseados a la hora de realizar modificaciones (Noble, y otros, 2004).

- Técnica del Camino Básico.

La técnica del camino básico es utilizada en la prueba de Caja Blanca, permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecute por lo menos una vez cada sentencia del programa (Pressman, 2002).

- Métrica Complejidad Ciclomática.

La complejidad ciclomática es una métrica del software utilizada en las pruebas de Caja Blanca, y proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto del método de prueba del camino básico, el valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez (Pressman, 2002).

La complejidad ciclomática puede calcularse de tres maneras diferentes:

1. El número de regiones del grafo de flujo coincide con la complejidad ciclomática.
2. La complejidad ciclomática, V (G), de un grafo de flujo G se define como V (G)=A-N+2, donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.
3. La complejidad ciclomática, V (G), de un grafo de flujo G también se define como V (G) P+ 1, donde P es el número de nodos predicado contenidos en el grafo de flujo G.

- Pruebas de Aceptación.

Son diseñadas a partir de cada iteración y basadas en las HU, en ella es el cliente el encargado de comprobar que ha sido correctamente implementada. Estas pruebas son reconocidas por su similitud con las llamadas pruebas de caja negra, pues como los clientes son los principales responsables de la inspección, también son los autorizados de diseñar la prioridad o el orden de resolución en caso de que falle alguna funcionalidad (Escribano, 2002).

- Prueba de Caja Negra.

La prueba de caja negra se centra en los requisitos funcionales del sistema. La prueba de caja negra intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la Caja Blanca (Pressman, 2002). Mediante estas pruebas es posible encontrar:

1. Funciones incorrectas o ausentes.
2. Errores de interfaz.
3. Errores en estructuras de datos o en accesos a las Bases de Datos externas.
4. Errores de rendimiento.
5. Errores de inicialización y terminación.

Para la validación de la solución informática propuesta, se realizarán todas las pruebas y métricas anteriormente expuestas, dado que son un mecanismo crítico para la garantía de la calidad del software y representa una revisión final, de los requisitos, del diseño y de la codificación.

Conclusiones del capítulo:

Después de realizado un estudio del marco teórico de la investigación, se pudo arribar a las siguientes conclusiones:

- El estudio de las principales directrices (normas, estándares, resoluciones, decretos y tecnologías) permitió realizar un análisis y síntesis de los principales controles establecidos en el área de la seguridad del control de acceso.
- En la actualidad existen modelos para la evaluación, pero no satisfacen las necesidades de la problemática existente. Para esto es necesario la elaboración de un nuevo modelo teniendo en cuenta criterios de evaluación.
- Las tecnologías seleccionados para el desarrollo del sistema fueron: XP como metodología de desarrollo, UML para el modelado, VisualParadigm como herramienta CASE, Java como lenguaje de programación, NetBeans para el desarrollo de la aplicación y Sqlite gestor de bases de datos garantizando la portabilidad de los mismos.

CAPÍTULO 2: Solución Informática.

En este capítulo se presentan los elementos fundamentales de SEESCA. Se realiza un análisis de las etapas de diseño e implementación de la solución presentando los artefactos generados por la metodología seleccionada, conjuntamente con los requisitos funcionales y no funcionales que fueron identificados. Se realiza un estudio de la arquitectura empleada para la construcción del sistema y los Patrones Generales de Software para Asignación de Responsabilidades (GRASP por sus siglas en inglés), así como la presentación y explicación de las Historias de Usuario (HU) y las Tarjetas Clase, Responsabilidad y Colaboración (CRC).

2.1- SEESCA.

El Sistema Informático para la Evaluación de los Especialistas de Seguridad del Control de Acceso de los Sistemas de Información (SEESCA) contribuye a la selección efectiva de los especialistas que evalúan la seguridad del Control de Acceso de los SI. Esto garantiza una mejor fiabilidad a la hora de realizar la evaluación de la seguridad del Control de Acceso en los SI sobre la base del uso de tecnologías libres, multiplataforma, con alto grado de seguridad y adaptabilidad.

2.2- Componentes de la solución informática.

Los componentes de una solución informática son utilizados con el propósito de dar solución a necesidades específicas del negocio y ayudan a mejorar la capacidad de uso, de administración o la seguridad de la infraestructura de la solución.

2.2.1- Criterios utilizados en el desarrollo de la aplicación.

Para el desarrollo de SEESCA fue necesario el estudio previo de las directrices en el área de la evaluación de la seguridad del Control de Acceso (normas, estándares, resoluciones y soluciones). Se pudo constatar un conjunto de diez criterios que representan la agrupación y síntesis de los controles para la selección de las competencias de los especialistas en el área de la seguridad del Control de Acceso.

A continuación se describen los criterios obtenidos en la investigación:

- **C1-** Caracterizar los controles internos propuestos por marcos de trabajo de aplicación internacional. Ej: COSO, MDE.

- **C2-** Caracterizar los algoritmos de cifrado simétrico y asimétrico. Ej: simétricos: (DES-TDES, RC2/RC4/RC5/RC6, AES, IDEA, SEAL, Blowfish, Serpent), asimétricos: (Diffie-Hellman, RSA, DSA, ElGamal, Rabin).
- **C3-** Caracterizar estándares para la federación de identidades entre dominios. Ej: Shibboleth, Liberty Alliance.
- **C4-** Caracterizar los estándares para la representación de la información de autorización. Ej: SPKI/SDSI, SAML, TACACS+
- **C5-** Caracterizar los protocolos para la gestión de autorización de la red. Ej: RADIUS, DIAMETER, WEP, WPA.
- **C6-** Caracterizar los estándares de Control de Acceso de la red. Ej: 802.1X, PANA, EAP.
- **C7-** Caracterizar los estándares para el intercambio de información. Ej: Kerberos, OpenPGP, EDIFACT, AC X.509 (PKI X.509, S/MIME, IPSec, TLS, WAP, SOAP, WSDL, UDDI, WS Security, X.500, LDAP, XACML, XKMS).
- **C8-** Caracterizar soluciones propias de la organización para la gestión de la Identificación y Autenticación.
- **C9-** Caracterizar soluciones propias de la organización para la gestión de la Autorización.
- **C10-** Caracterizar soluciones propias de la organización para la gestión de la Auditoría.

Estos criterios fueron agrupados en los diferentes procesos que tiene el Control de Acceso. A continuación se representa la síntesis y agrupación de las principales directrices en el área de la seguridad del Control de Acceso, la relación existente con los criterios de evaluación y los apéndices o epígrafes en los cuales se hace referencia:

Tabla 1: Relación entre criterios, las normas y estándares para la evaluación de las competencias de los especialistas de seguridad.

Autenticación	Normas, Estándares y Soluciones existentes en la actualidad											
Criterios	Guía OWASP	MAGERIT	COBIT - Focus-V1-2014	Resolución 127/2007	Resolución No. 60/11	ISO 27002	ISO 19011	NIST PE 800-12	NIST PE 800-53	Norma de Buenas Prácticas para la SI	SOAR	FIPS 199
C2	4.4.1	2.1, 2.3	4	Art47, 69	Art78	12.3	A7	8.4.1.1, 10.1.2, 10.2.1, 10.2.2,	CNSSI	SM4.1.3	5.2.1.2	Controles

								10.2.3, 10.2.4, 11.2.4, 13.4, 14.4, 14.5.2, 14.5.3	4009			de seguridad
C3	4.5	2.1	7, 8, 9, 10	Art48, 62, 69	Art 78	10.8, 10.6	A7	16.1.1, 16.1.2, 16.2.1, 16.2.2, 16.4.1, 16.4.2, 16.4.3, 17.3.1.3, 17.3.1.5, 17.3.2, 17.3.2.1, 17.3.2.3, 18.1.1, 18.1.2, 18.1.3, 18.2.1, 18.2.2, 18.2.2.1, 18.2.2.2, 19.1	AC 5, 6, 16, 21, 22	SM5.5.4, SM4.1.3, UE3.4.7, SM6.6.5, CB5.3.2, NW4.4.3, SD3.5.3	5.2.1.2	Controles de seguridad
C7	4.7, 4.8	2.1, 2.3, 2.4, 2.8	1, 10	Art5, Art6, Art8, Art10, Art21, Art44, Art47, Art48, Art53, Art62, Art69	Art78	10.6, 10.7	A7	17.3.2	AC (2.14 .16)	SM4.1.3	5.2.1.2	Controles de seguridad
C8	4.5, 4.6.1	2.1	1, 2, 4, 7, 8, 9, 10	Art76, Art86	Art12 incisoC, Art12 incisoE, Art16, Art78, Art90	5.1, 10.5, 10.6, 10.7, 10.8, 11.2, 11.3, 11.4, 11.6, 12.3, 15.2,	5.3.1, 5.3.2, 7.2.3.3, A7,	3.1	CNSSI 4009	SM4.1.3 UE3.4.7 SM6.6.5 CB5.3.2 NW4.4.3 SD3.5.3	5.2.1.2	Controles de seguridad

Tabla 2: Relación entre criterios, las normas y estándares para la evaluación de las competencias de los especialistas de seguridad.

Autorización	Normas, Estándares y Soluciones existentes en la actualidad

Criterios	Guía OWASP	MAGERIT	COBIT - Focus-V1-2014	Resolución 127/2007	Resolución No. 60/11	ISO 27002	ISO 19011	NIST PE 800-12	NIST PE 800-53	Norma de Buenas Prácticas para la SI	SOAR	FIPS
C4	4.4.1, 4.5, 4.6.1, 4.6.2, 4.6.3	2.1	10, 12	Art47, 48, 62	Inciso C del Art12, Art78	5.1, 11.2, 11.6	5.3.2, A7	10.2.1, 10.2.3	CNSSI 4009	NW 4.4.3	5.2.1.2	Controles de seguridad
C5	4.4.1, 4.5, 4.6.1, 4.6.2, 4.6.3	2.3	1, 4, 10	Art62, 69	Art78	10.6, 10.8, 11.4, 11.6	A7	17.3.2	AC1,3, 4,5,10,11,12,14, 15,22	NW 4.4.3	5.2.1.2	Controles de seguridad
C6	4.4.1, 4.5, 4.6.1, 4.6.2, 4.6.3	2.3	1, 4, 10	Art62, 69	Art78	10.6, 10.8, 11.4, 11.6	A7	17.3.2	AC-17, AC-18, AC-19, AC-20, AC-21, AC-22, AC-18	NW 4.4.3	5.2.1.2	Controles de seguridad
C9	4.4.1	2.1	1, 2, 4, 7, 8, 9, 10	Art2, 3, 76, 86	Art12 incisoC, Art12 incisoE, Art16, Art78, Art90	5.1, 10.5, 10.6, 10.7, 10.8, 11.2, 11.3, 11.4, 11.6, 12.3, 15.2,	5.3.1, 5.3.2, 7.2.3.3, A7,	3.1, 3.2	CNSSI 4009	SM4.1.3	5.2.1.2	Controles de seguridad

Tabla 3: Relación entre criterios, las normas y estándares para la evaluación de las competencias de los especialistas de seguridad.

Auditoria	Normas, Estándares y Soluciones existentes en la actualidad											
Criterios	Guía OWASP	MAGERIT	COBIT - Focus-V1-2014	Resolución 127/2007	Resolución No. 60/11	ISO 27002	ISO 19011	NIST PE 800-12	NIST PE 800-53	Norma de Buenas Prácticas para la SI	SOAR	FIPS

C1	4.5, 4.6.2, 4.6.3	2.1, 2.3, 2.4,		Art2, 3, 8, 21	Art1, 2, 3, 6	5.1		3.1, 3.2, 3.3, 3.4, 3.5	AC 7,8, 9,13	SM4.1.3 SM6.6.5 CB5.3.2 NW4.4.3	5.2.1.2	Controles de seguridad
C10	4.6.2		1, 2, 4, 7, 8, 9, 10	Art76, Art86	Art12 incisoC, Art12 incisoE, Art16, Art78, Art90	5.1, 10.5, 10.6, 10.7, 10.8, 11.2, 11.3, 11.4, 11.6, 12.3, 15.2,	5.3.1, 5.3.2, 7.2.3.3, A7,	10.2.2	CNSSI 4009		5.2.1.2	Controles de seguridad

2.2.2- Algoritmo Lógica de Puntuación de Preferencias adaptado.

SEESCA presenta una evaluación cuantitativa y está basada en las evaluaciones mediante criterios de evaluación. A continuación se presenta la adaptación del algoritmo Lógica de Puntuación de Preferencias (LSP) adaptado para la evaluación de criterios. El proceso de evaluación LSP permite evaluar cuáles son los requerimientos que deben cumplir los sistemas, los principales atributos a evaluar y los rangos de valores que éstos pueden tomar. Estos atributos son llamados Variables de Performance, cada una de ellas es transformada en una Preferencia Elemental al aplicarle el correspondiente Criterio Elemental.

En el algoritmo LSP una Variable de Performance mediante la aplicación de un Criterio Elemental es transformada en valores que pertenecen al intervalo [0, 1] (Preferencia Elemental). Estos valores representan el grado de satisfacción de los requerimientos, 0 cuando no satisface el requerimiento, 1 cuando los satisface plenamente. Las Preferencias Elementales constituyen los parámetros de la Función de Criterio del LSP. Esta función retorna un único indicador global, que es el grado de satisfacción de los requerimientos del sistema en general (Bayucan, 1996).

En la figura 1 se muestra el algoritmo LSP adaptado para la evaluación de un sistema.

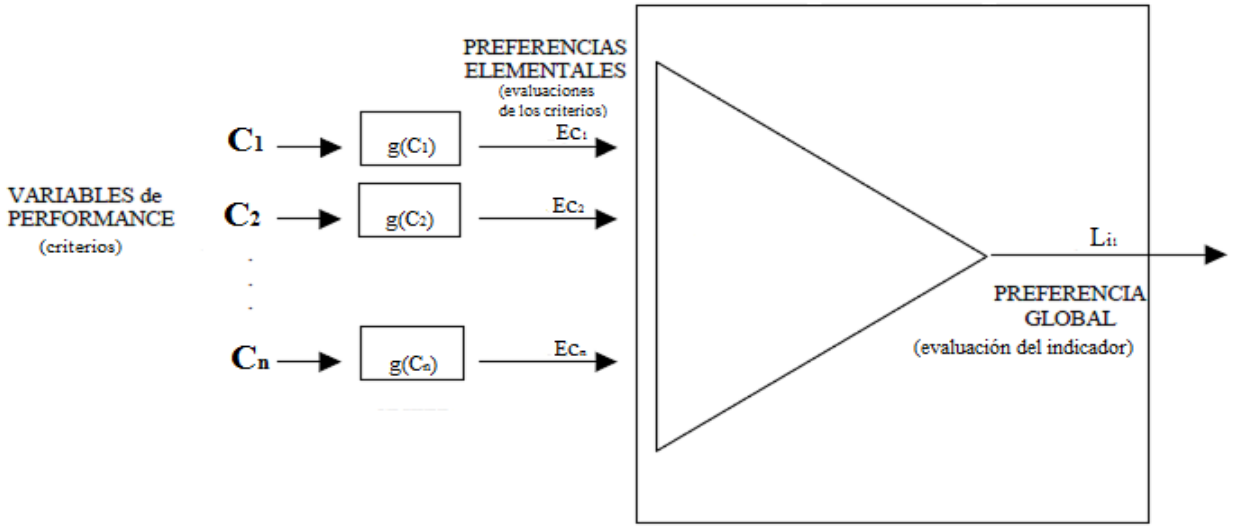

Figura 1: Estructura del algoritmo LSP Adaptado (Bayucan, 1996).

Las funciones $g(Cn)$ determinan la parte que representa la cantidad de criterios con que cumple el especialista que se evalúa, de la sumatoria de todos los criterios que deben ser cumplidos por el propio especialista (10 criterios). Las preferencias $E(Cn)$ son las criticidades de los criterios de evaluación y Li es la preferencia global o evaluación del especialista.

2.2.3- Requisitos del Sistema.

Un requisito puede definirse como un elemento necesario dentro de un sistema, que puede representar una condición o capacidad, para satisfacer un contrato, estándar, especificación u otro documento formal.

Los requisitos son divididos en dos categorías: Funcionales y No Funcionales. Una buena definición de requisitos no funcionales es tan importante como los requisitos funcionales; estos deben de ser apropiadamente definidos, analizados y trazados (Díaz, 2010).

- Requisitos Funcionales.

Los requisitos funcionales son las características requeridas del sistema que expresa una capacidad de acción del mismo, una funcionalidad; generalmente expresada en una declaración en forma verbal (Synergix, 2013).

No alteran la funcionalidad del producto, por lo que se mantienen invariables sin importar con que propiedades o cualidades se relacionen (Díaz, 2010). Son todas las condiciones y capacidades que debe cumplir el software.

Los requisitos funcionales de SEESCA son:

- Registrar Criterios.
- Modificar Criterios.
- Eliminar Criterios.
- Registrar Especialista.
- Modificar Especialistas.
- Eliminar Especialistas.
- Evaluar Especialista.
- Mostrar listado con la información de los Criterios registrados.
- Mostrar listado con la información de los Especialistas registrados.
- Mostrar listado de Criterios para la evaluación de los Especialistas.
- Guardar el resultado de la evaluación.
- Mostrar un reporte de la evaluación realizada.
- Mostrar un listado de reportes de las evaluaciones realizadas.
- Eliminar Reportes de Evaluación.

- Requisitos No Funcionales.

Lo requisitos no funcionales son las características requeridas del sistema, del proceso de desarrollo, del servicio prestado o de cualquier otro aspecto del desarrollo, que señala una restricción del mismo (Synergix, 2013).

Según las definiciones del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) los requisitos no funcionales se dividen en las siguientes categorías:

- **Requisitos de Software de SEESCA:**

SEESCA necesita para su correcto funcionamiento:

1. Máquina Virtual de Java versión 8 o posterior para el sistema operativo que utilice.
2. Un tiempo de entrenamiento de un máximo de 5 días para administradores del sistema.

- **Requisitos de Hardware de SEESCA:**

SEESCA para su ejecución necesita una computadora personal (PC) con los siguientes requisitos como mínimo:

1. Procesador: 1.5 GHz.
2. RAM: 512 Mb.
3. Disco Duro: 40 Gb.

- **Requisitos de diseño y de implementación:** especifica o restringe la codificación o construcción de un sistema; son restricciones que han sido ordenadas y deben ser cumplidas estrictamente.

El sistema SEESCA debe ser capaz de ser ejecutado en equipos sin importar el sistema operativo instalado, además con los patrones de diseño más utilizados a nivel mundial. Debe garantizar la portabilidad de la base de datos ya que es un sistema que será asignado a los administradores de cada organización de forma independiente.

- **Requisitos de apariencia o interfaz externa:** describe la apariencia del producto. Es importante destacar que no se trata del diseño de la interfaz en detalle sino que especifican cómo se pretende que sea la interfaz externa del producto.

SEESCA debe ser de fácil uso con una interfaz amigable para sus usuarios, de modo que no cause ninguna molestia al cliente. Estará organizada de forma tal que no será ninguna dificultad para cualquier tipo de usuario utilizar fácilmente la aplicación.

- **Requisitos de Seguridad:** No necesitara implementar requisitos de seguridad ya que no será un sistema distribuido y por ende no tendrá accesos por más de un administrador a la vez.

2.2.4- Historias de Usuario.

Las HU se aplican a marcos de trabajo ágiles, tales como Scrum o en este caso XP. Son utilizadas para iniciar conversaciones entre desarrolladores de software y usuarios del área de negocio, las cuales servirán para identificar los requerimientos del negocio, requerimientos técnicos y para encontrar los supuestos (premisas) no visibles en una primera aproximación. Su propósito, no es describir en detalle la funcionalidad. Para estas estimaciones, se le asignan unidades de medida a las historias que representen magnitud y no días reales de duración, tales como: puntos de historia, días ideales u otra unidad de medida (Kaczor, 2011).

Las historias de usuario (HU) que se identificaron durante la etapa de análisis de la solución informática son: Gestionar Criterio, Gestionar Especialista y Evaluar Especialista.

Tabla 4. HU Gestionar Criterio.

HU Gestionar Criterio	
Número: 1	**Nombre HU**: Gestionar Criterio
Modificación de Historia de Usuario Número:	

Referencia: No	
Programador: Arian González Au	**Iteración Asignada**: 1
Prioridad en Negocio: Alta **(Alta/Media/Baja)**	**Puntos Estimados(días)**: 10
Riesgo en Desarrollo: Alto	**Puntos Reales(días)**: 18
Descripción: La aplicación muestra una interfaz para registrar un nuevo criterio (Interfaz para Adicionar), también se puede modificar un criterio (Interfaz para Modificar) y eliminar (Interfaz para Eliminar) el indicador utilizado para esto es el identificador y los atributos del propio criterio.	
Observaciones:	

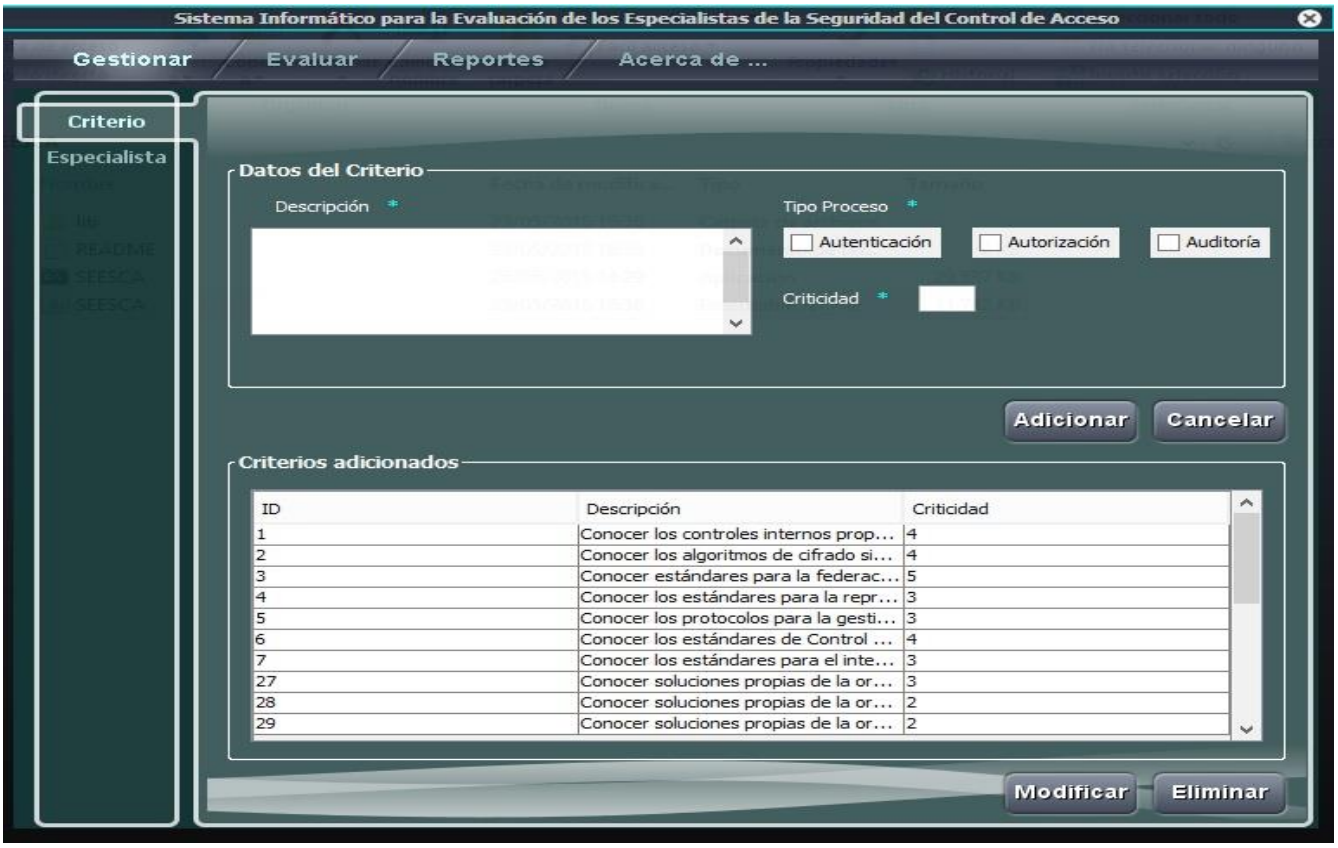

Figura 2. Interfaz Gestionar Criterio.

Tabla 5. HU Gestionar Especialista.

HU Gestionar Especialista	
Número: 2	**Nombre HU**: Gestionar Especialista

Modificación de Historia de Usuario Número:	
Referencia: No	
Programador: Arian González Au	**Iteración Asignada**: 1
Prioridad en Negocio: Alta **(Alta/Media/Baja)**	**Puntos Estimados(días)**: 15
Riesgo en Desarrollo: Alto	**Puntos Reales(días)**: 20
Descripción: La aplicación muestra una interfaz para registrar un nuevo especialista (Interfaz para Adicionar), también se puede modificar un especialista (Interfaz para Modificar) y eliminar (Interfaz para Eliminar) el indicador utilizado para esto es el identificador y los atributos del propio especialista.	
Observaciones:	

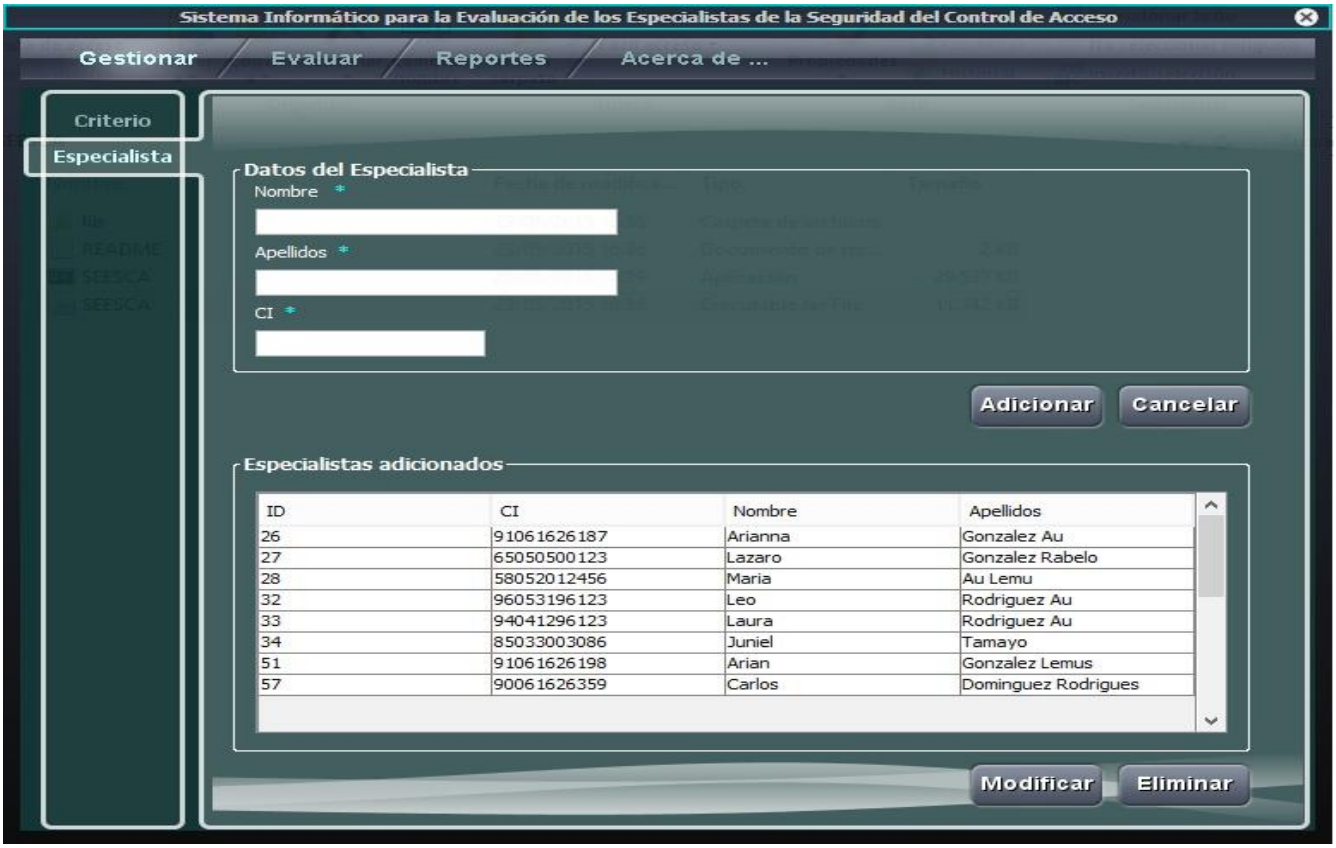

Figura 3. Interfaz Gestionar Especialista.

Tabla 6. HU Evaluar Especialista.

HU Evaluar Especialista

Número: 3	**Nombre HU**: Evaluar Especialista
Modificación de Historia de Usuario Número:	
Referencia: No	
Programador: Arian González Au	**Iteración Asignada**: 1
Prioridad en Negocio: Alta **(Alta/Media/Baja)**	**Puntos Estimados(días)**: 18
Riesgo en Desarrollo: Alto	**Puntos Reales(días)**: 23
Descripción: La aplicación muestra una interfaz para evaluar un especialista (Interfaz para Evaluar), Aquí se selecciona un especialista y se le asignan un conjunto de criterios con los que cumple, representado una parte del total de criterios existentes. Finalmente mostrándole al usuario el nivel de conocimiento que tiene dicho especialista, permitiendo ubicar en una escala de intervalos el resultado obtenido para finalmente evaluar cualitativamente las competencias del especialista en Bajo, Medio o Alto.	
Observaciones:	

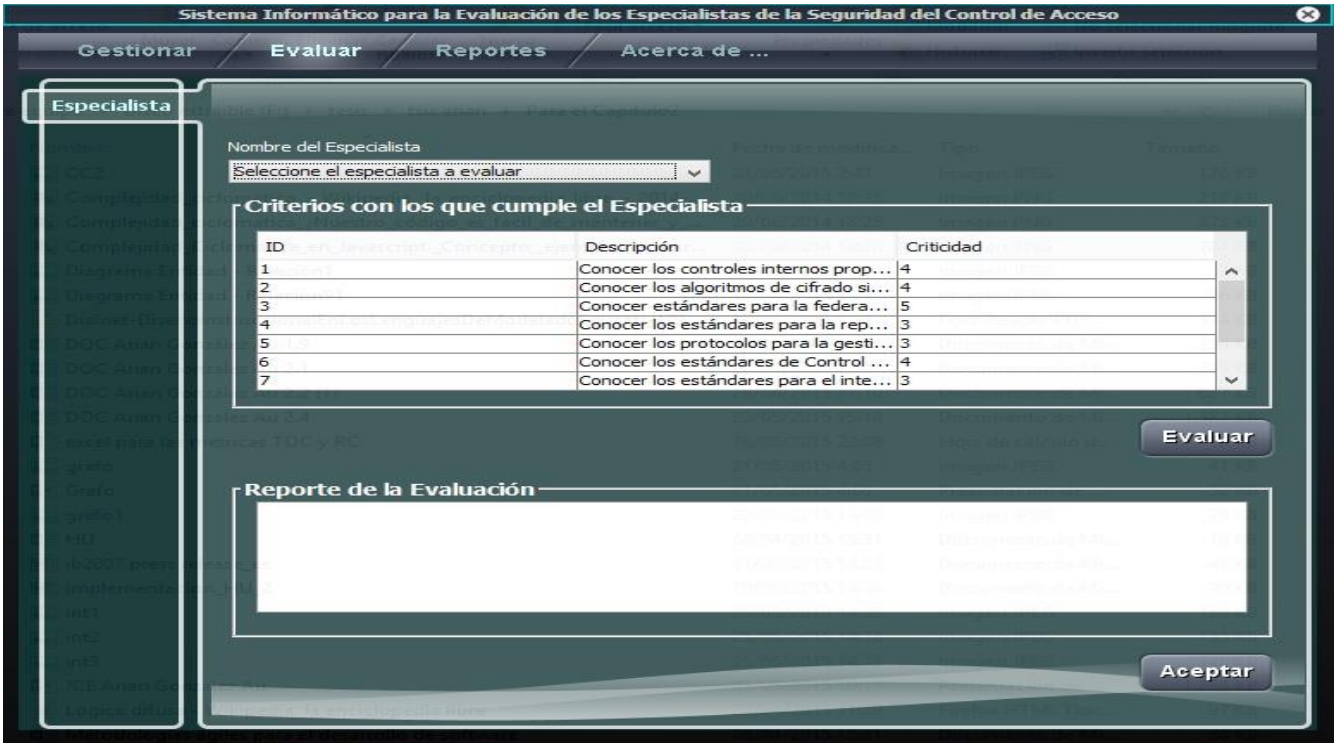

Figura 4. Interfaz Evaluar Especialista.

Las descripciones de las HU presentadas anteriormente demuestran que la solución informática propuesta está acorde a las necesidades del cliente, contribuyendo a la solución de los requerimientos del sistema. Además posee un alto nivel de prioridad en el negocio y un riesgo de desarrollo alto debido a que, por la complejidad de su implementación puedan surgir errores que lleven a la inoperatividad del sistema.

2.2.5- Tarjetas Clase-Responsabilidad-Colaboración.

Las tarjetas CRC fueron presentadas por Beck y Cunningham en 1989 para la enseñanza de la Programación Orientada a Objetos (OOP) (Casas, 2009) permitiendo al programador centrarse en el desarrollo a través de la representación de las clases. Son una de las principales piezas de diseño que propone la metodología XP (Jiménez, 2011).

Estas tarjetas constan de tres secciones donde se recogen el nombre de la clase, las funcionalidades o responsabilidades y los colaboradores con otras clases. Una clase es cualquier persona, cosa, evento, concepto, pantalla o reporte. Las responsabilidades de una clase son las características y las acciones que realizan, sus atributos y métodos. Los colaboradores son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades (Jiménez, 2011).

A continuación se describen las tarjetas CRC identificadas:

Tabla 7. CRC Gestionar Criterio.

CRC Gestionar Criterio	
Responsabilidad	**Colaboradores**
Registrar un nuevo criterio.	*gADCriterio*
Asociar un proceso al criterio.	*gADCriterio*
Eliminar un criterio determinado.	*gADCriterio*
Modificar un criterio dado su identificador.	*gADCriterio*
Mostrar listado de criterios.	*gADCriterio*

Tabla 8. CRC Gestionar Especialista.

CRC Gestionar Especialista	
Responsabilidad	**Colaboradores**
Registrar un nuevo especialista.	*gADEspecialista*
Modificar un especialista dado su identificador.	*gADEspecialista*

Eliminar un especialista.	*gADEspecialista*
Mostrar listado de los especialistas.	*gADEspecialista*

Tabla 9: CRC Evaluar Especialista.

CRC Evaluar Especialista	
Responsabilidad	**Colaboradores**
Seleccionar el especialista a evaluar.	*gADEspecialista*
Seleccionar los criterios con los que cumple.	*gADCriterio*
Mostrar el reporte de evaluación.	*gADEspecialista, gADCriterio, gADEspecialistanCriterio*
Guardar la evaluación realizada.	*gADevaluacion*

Tabla 10. CRC gnEspecialista.

CRC gnEspecialista	
Responsabilidades	**Colaboradores**
crearEspecialista (nEspecialista e)	*Especialista*
crearEspecialista (Especialista e)	*nEspecialista*

2.2.6- Arquitectura del sistema.

En la implementación del SEESCA se utilizó una arquitectura basada en tres capas. Como se puede observar en la figura 5 las capas son Presentación, Lógica de Negocio y Acceso a Datos. Esta arquitectura permite la separación de la lógica de negocios de la lógica de clases responsables del acceso a los datos. Tiene como ventaja principal que el desarrollo se puede realizar en varios niveles para que en caso de que ocurra algún cambio, sólo se modifique la capa requerida sin tener que revisar todo el código.

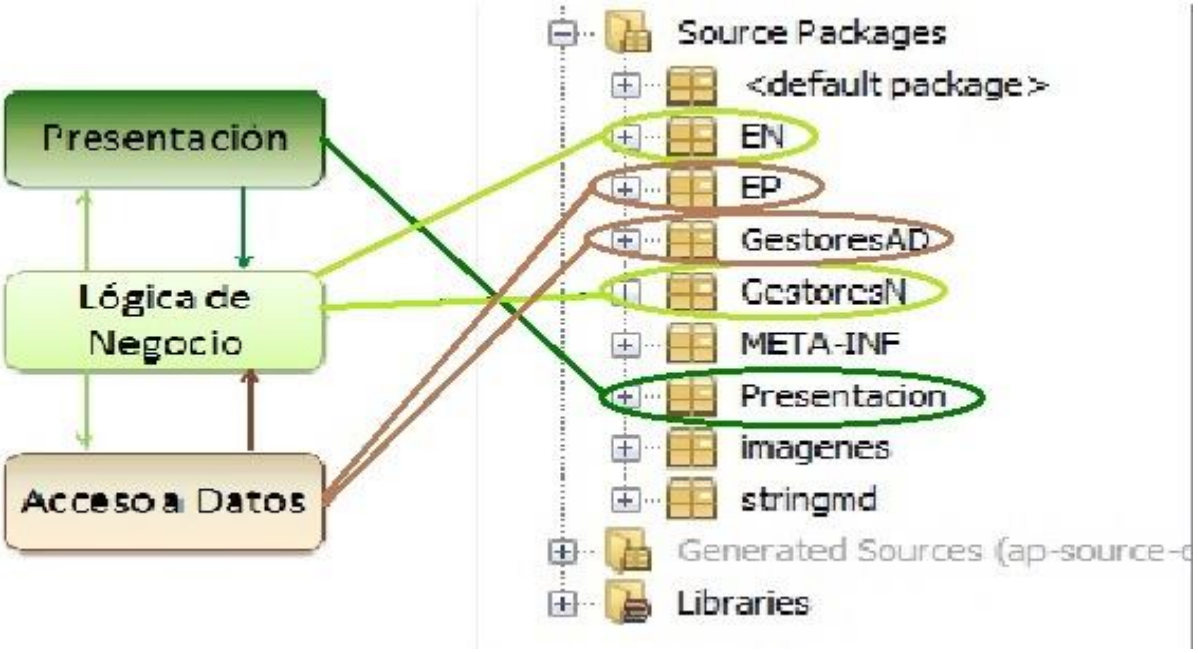

Figura 5: Arquitectura de SEESCA.

- **Capa de Presentación:** permite presentar el sistema al usuario, además de mostrar la información al usuario y recibir la información preveniente del mismo. Esta capa se comunica únicamente con la capa lógica de negocio.
- **Capa de Lógica de Negocio:** se ubican las entidades y los gestores del negocio. En esta capa se establecen todas las reglas de negocio que deben cumplirse. Se comunica con la Capa de Presentación, para recibir las solicitudes del usuario y presentar los resultados obtenidos, y con la capa de Acceso a Datos, para enviar datos que necesitan persistirse en la base de datos o recibirlos de la misma.
- **Capa de Acceso a Datos:** está conformada por las entidades y los gestores del acceso a datos, además de realizar todo el almacenamiento de la información. Recibe solicitudes de almacenamiento u obtención de información desde la capa de lógica de negocio.

2.2.7- Patrones de Diseño.

Un patrón de diseño es una solución a un problema de diseño no trivial que es efectiva y reusable. El uso de los patrones de diseño beneficia en gran medida el desarrollo y la calidad del software, pues evita la reiteración en la búsqueda de soluciones a problemas ya conocidos y resueltos anteriormente, además formalizar un vocabulario común entre diseñadores y estandarizar el modo en que se realiza el diseño (Hernández, 2013).

- Patrones Generales de Software para Asignación de Responsabilidades.

Los Patrones Generales de Software para Asignación de Responsabilidades (GRASP, por sus siglas en inglés) describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable (Larman, 2009).

- **Patrón Creador.**

Las clases gestoras de la capa de negocio incluyen las acciones que pueden ser realizadas en el sistema, entre ellas las relacionadas con la creación de los objetos necesarios para acceder a las funcionalidades de las clases persistentes que gestionan el acceso a la base de datos.

Ejemplo: *gnEspecialista.java; gADEspecialista.java.*

```
public void SalvarEspecialista(nEspecialista c) {
    em.getTransaction().begin();
    Especialista esp = gnEspecialista.crearEspecialista(c);
    em.persist(esp);
    em.getTransaction().commit();

}

public void SalvarEspecialista(Especialista c) {
    em.getTransaction().begin();
    em.persist(c);
    em.flush();
    em.getTransaction().commit();

}
```

Figura 6: Ejemplo de uso del Patrón Creador.

```
public class gnEspecialista {

    public static final Especialista crearEspecialista( nEspecialista e){
        return new Especialista(e.getId(), e.getCi(), e.getNombre(), e.getApellidos());
    }

    public static final nEspecialista crearEspecialista(Especialista e){
        return new nEspecialista(e.getIdEspecialista(), e.getCi(), e.getNombre(), e.getApellidos());
    }
```

Figura 7: Ejemplo de uso del Patrón Creador.

- **Patrón Experto.**

El mapeo[1] de objetos relacionales de la capa de acceso a datos de la Interfaz de Aplicación de Programación para la Persistencia Java (JPA por sus siglas en inglés) encapsula la lógica de los datos. JPA genera clases expertas en manejar la información pertinente a la entidad que representan. Ejemplo: *gADEspecialista.java.*

```
 */
public class gADEspecialista {

    private EntityManagerFactory emf;
    private EntityManager em;

    public gADEspecialista() {
        this.emf = Persistence.createEntityManagerFactory("JavaApplication1PU");
        this.em = emf.createEntityManager();
    }
```

Figura 8. Ejemplo de uso del Patrón Experto.

[1] El mapeo objeto-relacional (más conocido por su nombre en inglés, Object-Relational mapping, o sus siglas O/RM, ORM, y O/R mapping) es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, utilizando un motor de persistencia.

- **Patrón Controlador.**

Las clases gestoras de SEESCA presentes en las capas de acceso a datos y de las reglas del negocio se encargan de la gestión de las entidades. Ejemplo: *gADEvaluacion.java*.

```
 */
public class gADEvaluacion {

    private EntityManagerFactory emf;
    private EntityManager em;

    public gADEvaluacion() {
        this.emf = Persistence.createEntityManagerFactory("JavaApplication1PU");
        this.em = emf.createEntityManager();
    }

    public void SalvarEvaluacion(nEvaluacion c) {
        em.getTransaction().begin();
        Evaluacion eva = gnEvaluacion.crearEvaluacion(c);
        em.persist(eva);
        em.getTransaction().commit();

    }

    public void SalvarEvaluacion(Evaluacion c) {
        em.getTransaction().begin();
        em.persist(c);
        em.flush();
        em.getTransaction().commit();

    }
```

Figura 9. Ejemplo de uso en el sistema del patrón Controlador.

- **Patrón Alta Cohesión.**

La información que controlan las clases debe ser coherente y relacionada con la propia clase. La utilización de la arquitectura en capas permite la aplicación de este patrón. Las clases de la capa de presentación permiten la creación de objetos y de operaciones sobre los mismos, permitiendo que exista flexibilidad a cambios. Ejemplo: *NewFrmPrincipal.java*

```
    gADEspecialista gtrsis = new gADEspecialista();
    DefaultComboBoxModel modelo1 = new DefaultComboBoxModel();
    modelo1.addElement("Seleccione el especialista a evaluar");
    for (int i = 0; i < gtrsis.cargarEspecialista().size(); i++) {
        modelo1.addElement(gtrsis.cargarEspecialista().get(i).getNombre()
                + " " + gtrsis.cargarEspecialista().get(i).getApellidos());
    }
    jComboBoxES2.setModel(modelo1);
}
```

Figura 10. Ejemplo de uso en el sistema del patrón Alta Cohesión.

- **Patrón Bajo Acoplamiento.**

Este patrón se puede garantizar en la aplicación basándose en la propia arquitectura del sistema, lo que permite que las dependencias entre las clases sean muy poca, ya que solamente las clases de una capa se pueden comunicar con las de la capa inmediatamente adyacente.

```
public class gnEspecialistanCriterio {

    public static final Especialista_nCriterio crearEspecialistanCriterio(EspecialistanCriterio n) {
        Criterio cri = new gnCriterio().crearCriterio(n.getNCriterioidcriterio());
        nEspecialista esp = new gnEspecialista().crearEspecialista(n.getEspecialistaidespecialista());

        return new Especialista_nCriterio( esp.getId(), cri.getIdcriterio());
    }
```

Figura 11. Ejemplo de uso en el sistema del patrón Bajo Acoplamiento.

- Patrones del Grupo de Cuatro.

Los Patrones del Grupo de Cuatro (GoF, por sus siglas en inglés) resuelven problemas específicos de diseño, y vuelven al diseño orientado a objetos más flexible, elegante y extremadamente reutilizable. Ayudan a los diseñadores a reutilizar diseños exitosos basando nuevos diseños en experiencia previa (Pozo, 2012).

- **Patrón Instancia Única.**

Instancia Única es un patrón GoF utilizado en las clases gestoras de la capa de lógica de negocio, mediante el cual se define un mecanismo para acceder a los objetos de forma global. Ejemplo: la funcionalidad *crearEspecialista* para la creación de objetos de tipo Especialista puede ser accedido desde cualquier lugar del sistema.

```
public class gnEspecialista {

    public static final Especialista crearEspecialista( nEspecialista e){
        return new Especialista(e.getId(), e.getCi(), e.getNombre(), e.getApellidos());
    }

    public static final nEspecialista crearEspecialista(Especialista e){
        return new nEspecialista(e.getIdEspecialista(), e.getCi(), e.getNombre(), e.getApellidos());
    }
```

Figura 12. Ejemplo de uso en el sistema del patrón Instancia Única.

2.2.8- Modelo de Datos.

SEESCA hace uso de un modelo de datos que está compuesto de un conjunto de tablas que contribuyen a la correcta gestión de criterios y especialistas como se muestra en la Figura 13, debido a las necesidades del negocio y al propio diseño de la base de datos. No hay redundancia de datos, atributos nulos, ni dependencias transitivas, las llaves primarias son únicas, los atributos de cada tabla dependen totalmente de su llave primaria. Las tablas *nCriterio* y *Especialista* almacenan el registro de los criterios y especialistas que se sean adicionados. Al evaluar un nuevo especialista en la tabla *Especialista_nCriterio* se almacenara la relación entre el id del especialista con los id de los criterios con los que cumple. La tabla *Evaluación* registra la información de la evaluación de un especialista una vez que haya sido evaluado.

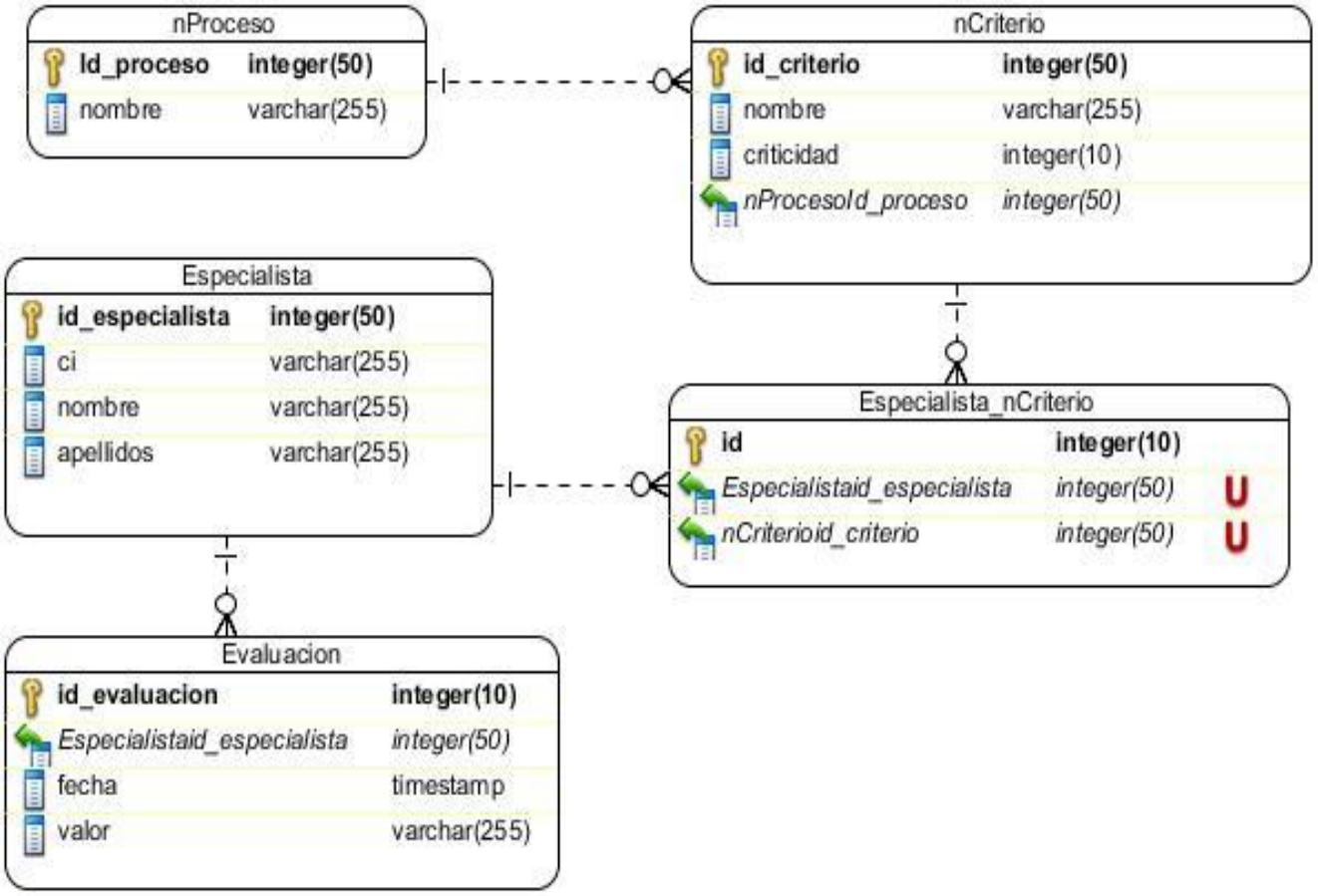

Figura 13. Modelo de Datos de SEESCA.

Conclusiones del capítulo:

Después de desarrollada la solución informática, se pudo arribar a las siguientes conclusiones:

- El estudio de las principales directrices (normas, estándares, resoluciones, decretos y tecnologías) en el área de la seguridad del control de acceso, permitió la obtención de un conjunto de diez criterios que posibilitarán la selección efectiva del especialista que se encargara de evaluar la seguridad del control de acceso de los sistemas de información.
- La correcta definición de los Requisitos Funcionales y No Funcionales permitieron el desarrollo de un sistema que da cumplimiento a las necesidades del negocio a partir de la realización de un diseño funcional.
- La utilización de las HU y las tarjetas CRC permitieron realizar un adecuado diseño e implementación del sistema basado en los requisitos funcionales.
- La Arquitectura N-capas utilizada para el sistema permitió la separación de la lógica de negocios de la lógica de clases responsables del acceso a los datos, permitiendo que en caso de ocurrir algún cambio, sólo se modifique la capa requerida sin tener que revisar todo el código.
- Los patrones de diseño utilizados posibilitaron la correcta asignación de responsabilidades desde el punto de vista del diseño de la solución, contribuyen a las buenas prácticas, a diseños exitosos, a la reutilización de código entre otros.
- El modelo de datos utilizado por SEESCA cumple con las necesidades del sistema, ya que permite una correcta gestión de criterios especialistas y evaluaciones, logrando evitar la redundancia de datos, los atributos nulos, las llaves primarias son únicas y los atributos de cada tabla dependen totalmente de su llave primaria.

CAPITULO 3: Análisis de los resultados.

En el presente capítulo se muestran los resultados obtenidos a través de las métricas y pruebas realizadas al sistema. Se aplicó la métrica de Tamaño Operacional de Clase, la métrica Relación entre Clases, las Pruebas del Camino Básico y Complejidad Ciclomática, además de las Pruebas de Aceptación y las de Caja Negra. Todas estas pruebas se realizaron sobre las funcionalidades de SEESCA, obteniéndose y mostrándose de esta forma el análisis de los resultados correspondientes a las pruebas realizadas.

3.1- Resultados de las Métricas.

Se aplicaron diferentes métricas durante el desarrollo del software con el objetivo de evaluar la calidad de los resultados de los trabajos técnicos. Las clases a las que se le aplicaron las diferentes métricas fueron las del paquete *GestoresAD*.

3.1.1- Especificidad de los requisitos.

Esta métrica está basada en la consistencia en la interpretación de los revisores de cada requisito. Es utilizada para la validación de los requisitos y se calcula como:

$$Q_1=\frac{N_{ui}}{N_r}$$

Donde:

Q_1= Especificidad de los requisitos.

N_{ui}= Número de requisitos que todos los revisores interpretaron de la misma manera.

N_r= Total de requisitos definidos.

Mientras más cercano esté a 1 mayor será la consistencia de la interpretación de los revisores y menor será la ambigüedad en los requisitos.

Aplicando la métrica se obtuvo el siguiente resultado, Q_1=14/14=1. Como el valor es 1 se puede interpretar que los requisitos no están ambiguos.

3.1.2- Grado de validación de los requisitos.

Esta métrica mide la corrección de la definición de los requisitos. Es utilizada para la validación de los requisitos y se calcula como:

$$Q_3=N_c/(N_c+N_{nv})$$

Donde:

Q_3= Grado de validación de los requisitos.

N_c= Total de requisitos validados correctamente.

N_{nv}= Total de requisitos no validados.

Mientras más cercano esté en resultado a 1 mayor será la corrección de los requisitos.

Aplicando la métrica se obtuvo el siguiente resultado, Q_3=14/(14+0)= 1. Como el resultado es 1 se puede interpretar que la definición de los requisitos es correcta.

3.1.3- Tamaño Operacional de las Clases.

Las clases a las que se aplicó la métrica Tamaño Operacional de Clase fueron las del paquete *GestoresAD* como se muestra se muestra en la Tabla 9. El paquete *GestoresAD* tiene un total de 5 clases y una cantidad de procedimientos total igual a 21 por lo que según los cálculos definidos en la métrica, el promedio de procedimientos es igual a 4.2.

Para las clases del paquete *GestoresAD* según los umbrales definidos por los atributos de calidad: responsabilidad y complejidad de la implementación se considera categoría Baja a valores inferiores a 4.2, categoría Media a resultados que se encuentran en el intervalo de 4,2 a 8,4 y categoría Alta en otro caso. Para el atributo reutilización se considera categoría Baja a valores superiores a 8.4, categoría Media a resultados que se encuentran en el intervalo de 4,2 a 8,4 y categoría Alta en otro caso. La tabla muestra los resultados obtenidos en la evaluación de estos atributos en cada clase del paquete *GestoresAD*.

Tabla 11. Resultados obtenidos en la aplicación de la métrica TOC en las clases del paquete *GestoresAD*.

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
gADProceso	1	Baja	Baja	Alta
gADCriterio	6	Media	Media	Media
gADEspecialista	7	Media	Media	Media
gADEvaluacion	4	Baja	Baja	Alta
gADEspecialistanCriterio	3	Baja	Baja	Alta

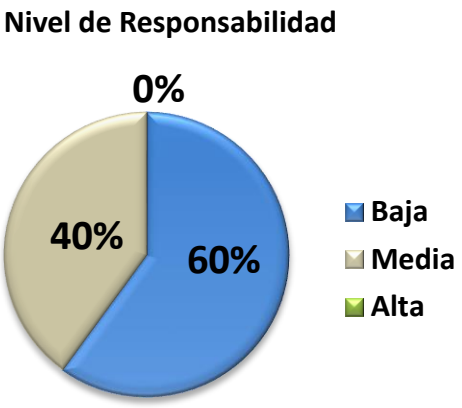

Figura 14. Atributo de calidad que representa la responsabilidad después de aplicada la métrica TOC.

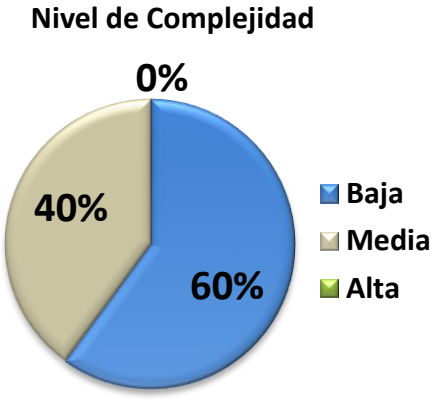

Figura 15. Atributo de calidad que representa la complejidad después de aplicada la métrica TOC.

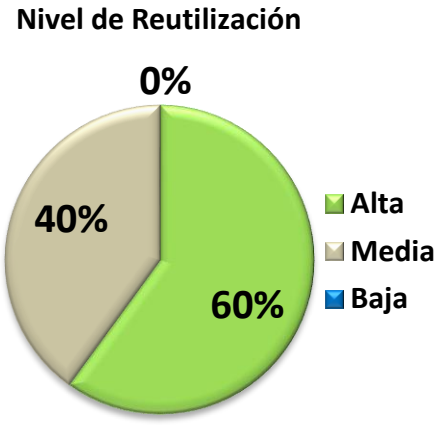

Figura 16. Atributo de calidad que representa la reutilización después de aplicada la métrica TOC.

Analizando la información contenida en las Figura 14, Figura 15 y Figura 16 se puede demostrar que el conjunto de clases contenidas dentro del paquete *GestoresAD* poseen un grado de responsabilidad baja equivalente al 60%, complejidad baja al 60% y reutilización alta al 60% respectivamente. Lo anteriormente planteado demuestra que SEESCA posee un nivel de tolerancia a los cambios y reutilización alto, atendiendo a los parámetros establecidos por la propia métrica TOC, cumpliendo así con una de las metas trazadas durante la fase de desarrollo del sistema.

3.1.4- Relación entre Clases.

Las clases a las que se aplicó la métrica Relación entre Clases fueron las del paquete *GestoresAD* como se muestra se muestra en la Tabla 9. El paquete *GestoresAD* tiene un total de 5 clases y cantidad de relaciones de uso igual a 8 por lo que según los cálculos definidos en la métrica, el promedio de procedimientos es igual a 1.6.

Para las clases del paquete *GestoresAD* según los umbrales definidos por el atributo de calidad acoplamiento se considera categoría Baja a valores igual a 1, Media igual a 2 y Alta a valores mayores que 2. Para los atributos complejidad de mantenimiento y cantidad de pruebas, se considera categoría Baja a valores inferiores a 1.6, categoría Media a valores en el intervalo de 1.6 a 3.2 y categoría Alta en otro caso. Para el atributo reutilización se considera categoría Baja a valores superiores a 3.2, categoría Media a valores en el intervalo de 1.6 a 3.2 y categoría Alta en otro caso. La tabla muestra los resultados obtenidos en la evaluación de estos atributos en cada clase del paquete *GestoresAD*.

Tabla 12. Resultados obtenidos en la aplicación de la métrica RC en las clases del paquete *GestoresAD*.

Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mant.	Reutilización	Cantidad de Pruebas
gADProceso	1	Bajo	Baja	Alta	Baja
gADCriterio	2	Medio	Media	Media	Media
gADEspecialista	2	Medio	Media	Media	Media
gADEvaluacion	1	Bajo	Baja	Alta	Baja
gADEspecialistanCriterio	2	Medio	Media	Media	Media

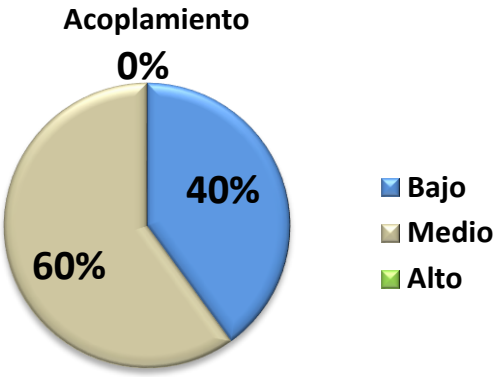

Figura 17. Atributo de calidad que representa el acoplamiento después de aplicada la métrica RC.

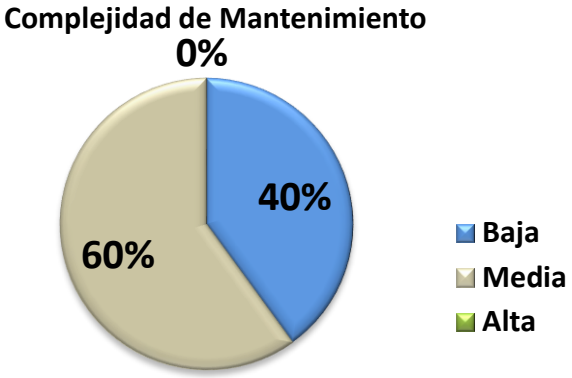

Figura 18. Atributo de calidad que representa la complejidad de mantenimiento después de aplicada la métrica RC.

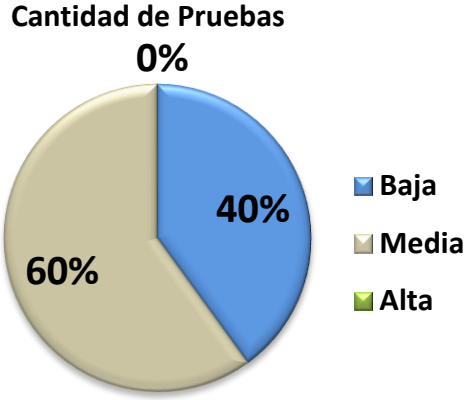

Figura 19. Atributo de calidad que representa la cantidad de pruebas después de aplicada la métrica RC.

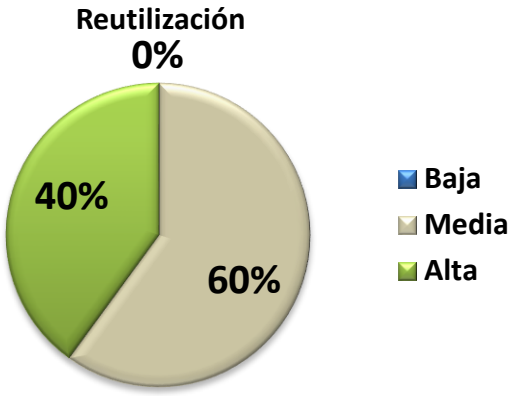

Figura 20. Atributo de calidad que representa la reutilización después de aplicada la métrica RC.

Los resultados obtenidos mediante la evaluación de la métrica RC demostraron que el diseño propuesto se encuentra dentro de los niveles de calidad requeridos. Los atributos de calidad fueron evaluados satisfactoriamente confirmando que las clases están diseñadas correctamente ya que presentan un acoplamiento entre medio y bajo, su complejidad de mantenimiento esta entre media y baja, la cantidad de pruebas que poseen se encuentra entre media y baja, por último el nivel de reutilización se encuentra entre medio y alto, la que implica que se necesitará menos esfuerzo a la hora de realizar pruebas unitarias a estas clases y presentan un mayor nivel de reutilización.

3.2- Resultados de la Pruebas.

A continuación se presentan los resultados obtenidos después de aplicada las diferentes pruebas realizadas al software con el objetivo de validar el diseño, demostrar que la aplicación no contenía errores.

3.2.1- Complejidad Ciclomática.

La Complejidad Ciclomática se realiza a todos los métodos o algoritmos de un sistema informático. A continuación se presenta la prueba realizada al método para *AdicionarEspecialista* ya que posee importancia en el contexto de la investigación, la cual posibilita que se realice una correcta evaluación de un especialista.

```
private void jButtonAS1ActionPerformed(java.awt.event.ActionEvent evt) {
    Especialista esp = new Especialista();
    gADEspecialista gesp = new gADEspecialista();
    gADEspecialista gespe = new gADEspecialista();
    if (sololetrasComiensaMayus(jTextField2.getText()) == false && sololetrasComiensaMayus(jTextField3.getText()) == true) {
        JOptionPane.showMessageDialog(panelCurves2, "El campo Nombre debe comenzar con mayúscula y no puede estar vacío, "
                + "ni contener caracteres especiales.", "Error", 0);
    }
    if (sololetrasComiensaMayus(jTextField2.getText()) == true && sololetrasComiensaMayus(jTextField3.getText()) == false) {
        JOptionPane.showMessageDialog(panelCurves2, "El campo Apellidos debe comenzar con mayúscula y no puede estar vacío, "
                + "ni contener caracteres especiales.", "Error", 0);
    }
    if (sololetrasComiensaMayus(jTextField2.getText()) == false && sololetrasComiensaMayus(jTextField3.getText()) == false) {
        JOptionPane.showMessageDialog(panelCurves2, "Los campos Nombre y Apellidos deben comenzar con mayúscula y no pueden "
                + "estar vacíos, ni contener caracteres especiales.", "Error", 0);
    }
    if (sololetrasComiensaMayus(jTextField2.getText()) && sololetrasComiensaMayus(jTextField3.getText())) {
        if (formatoCI(jTextField1.getText()) == false) {
            JOptionPane.showMessageDialog(panelCurves2, "El CI no debe estar vacío y debe tener el formato correcto.", "Alerta", 1);
        }
    else {
            esp = gesp.buscarEspecialistaPorCI(jTextField1.getText());
            if (esp != null && (esp.getCi().equals(jTextField1.getText()))) {
                JOptionPane.showMessageDialog(this, "Ya existe un especialista con el mismo CI.");
            } else {
                nEspecialista especialista = new nEspecialista(jTextField1.getText(), jTextField2.getText(), jTextField3.getText());
                gespe.SalvarEspecialista(especialista);
                jTextField1.setText("");
                jTextField2.setText("");
                jTextField3.setText("");
                jLabel30.setText("Especialista adicionado con éxito");
                jLabel30.setForeground(Color.yellow);
                ActualizarDatosAddEspecialista();
                CargarEspecialista();
            }
        }
    }
```

Figura 21. Ejemplo del código utilizado para calcular la complejidad ciclomática.

Para el cálculo de la complejidad temporal existe tres fórmulas diferentes a continuación se presentan 2 de ellas. La primera se utiliza en programas que realizan los cálculos de forma automatizada y la formula 3 es utilizada para cuando se calcula de forma manual.

Fórmula 1:

V (G) = (A - N) + 2

Donde “A” es la cantidad de Aristas y “N” la cantidad de Nodos.

V (G)=22-17+2=7

Fórmula 2:

V (G)=P+ 1, donde P es el número de nodos predicado contenidos en el grafo de flujo G.

V (G)=6+ 1=7

Fórmula 3:

V (G) = R

Donde “R” representa la cantidad de regiones en el grafo.

V (G) = 7

Como se puede observar después de aplicadas las formulas 1, 2 y 3 la funcionalidad *AdicionarEspecialista* posee una complejidad ciclomática igual a 7, lo cual demuestra que las tres fórmulas son efectivas.

3.2.2- Prueba del Camino Básico.

Una vez calculada la complejidad ciclomática se define como límite superior 7 pruebas a realizar para que todo el código se encuentre completamente probado. La funcionalidad Adicionar Especialista posee poco riesgo debido a que el resultado arrojado por la métrica pertenece al intervalo entre 1-10, siendo igual a la cantidad caminos básicos.

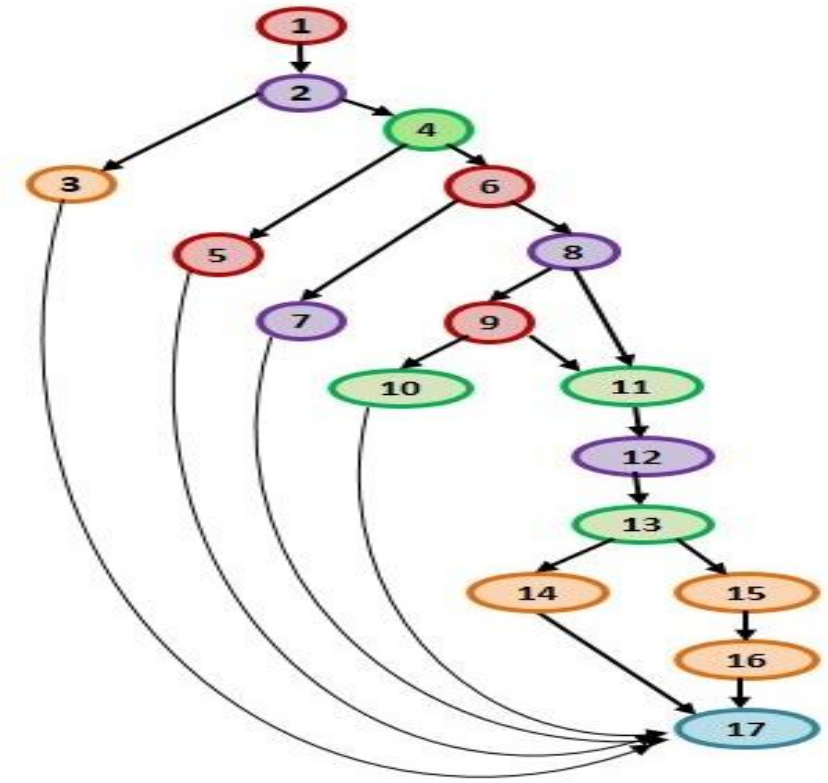

Figura 22. Grafo que muestra el camino básico de la funcionalidad Adicionar Especialista.

El total de caminos independientes establecidos fue de 7 y a continuación se muestran:

El camino 1 inicia con el primer bloque de instrucciones, seguido una condicional y posteriormente finaliza lanzando una excepción.

- **Camino 1:** 1-2-3-17

El camino 2 inicia con el primer bloque de instrucciones, seguido dos condicionales y posteriormente finaliza lanzando una excepción.

- **Camino 2:** 1-2-4-5-17

El camino 3 inicia con el primer bloque de instrucciones, seguido tres condicionales y posteriormente finaliza lanzando una excepción.

- **Camino 3:** 1-2-4-6-7-17

El camino 4 inicia con el primer bloque de instrucciones, seguido 5 condicionales y posteriormente finaliza lanzando una excepción.

- **Camino 4:** 1-2-4-6-8-9-10-17

El camino 5 inicia con el primer bloque de instrucciones, seguido 6 condicionales y posteriormente finaliza insertando el especialista correctamente en el sistema.

- **Camino 5:** 1-2-4-6-8-9-11-12-13-15-16-17

El camino 6 inicia con el primer bloque de instrucciones, seguido 5 condicionales y posteriormente finaliza lanzando una excepción.

- **Camino 6:** 1-2-4-6-8-11-12-13-14-17

El camino 7 inicia con el primer bloque de instrucciones, seguido 5 condicionales y posteriormente finaliza insertando el especialista correctamente en el sistema.

- **Camino 7:** 1-2-4-6-8-11-12-13-15-16-17

3.2.3- Pruebas de Caja Negra.

Para la realización de esta prueba se utilizó la técnica de partición de equivalencia garantizando efectividad al examinar los valores válidos e inválidos de las entradas existentes en el software. Se realizaron un total de 8 casos de pruebas, a los cuales se le realizaron 3 iteraciones. Finalmente se realizó la liberación de la aplicación por el grupo de Calidad del CEGEL, ya que a la tercera iteración la aplicación no contenía errores. A continuación se muestra un ejemplo del caso de prueba para *AdicionarEspecialista* y una tabla con la información de los resultados obtenidos en las pruebas en cada iteración:

Tabla 13. Iteraciones de los casos de pruebas de Caja Negra.

Iteración	Total de No Conformidades
1	30
2	9
3	0

Tabla 14. Descripción de las variables del caso de prueba Adicionar Especialista.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Nombre	campo de texto	No	Longitud: Máximo 250 caracteres. Formato: Comienza con mayúscula.
2	Apellidos	campo de texto	No	Longitud: Máximo 250 caracteres. Formato: Comienza con mayúscula.
3	CI	numero	No	El número de CI debe tener la longitud de 11 dígitos.

Escenario 1 (Datos generales)	Descripción	Nombre	Apellidos	CI	Respuesta del sistema	Flujo central
EC 1.1 Escenario válido.	Permite adicionar un	V Lorena	V Gonzalez Gonzalez	V 91061626196	Se registran los datos generales correctamente.	Seleccionar en el menú la opción Gestionar. Posteriormente en el submenú que aparece se debe seleccionar la pestaña Especialista y se comienzan a llenar los campos para
EC 1.2 Escenario inválido.	No permite adicionar un especialista seleccionado.	I 12	I 12	I Arian	El sistema muestra un mensaje indicando que hay campos que son obligatorios.	
		I vacio	I vacio	I vacio		
EC 1.3 Escenario incompleto.	No permite adicionar un especialista satisfactoriamente debido a que existen campos obligatorios vacíos.	I vacio	I vacio	V 91061626196	El sistema muestra un mensaje de que existen campos vacios, que son obligatorios para poder realizar correctamente la operación.	
		I vacio	V Gonzalez Gonzalez	V 91061626196		
		I vacio	V Gonzalez Gonzalez	I vacio		
		V Lorena	I vacio	I vacio		
		V Lorena	V Gonzalez Gonzalez	I vacio		
		V Lorena	I vacio	V 91061626196		
		V Lorena	I 12	I Arian		
		V Lorena	V Gonzalez Gonzalez	I Arian		
		V lorena	I 12	V 91061626187		
		I 12	V Gonzalez Gonzalez	V 91061626187		
		I 12	I 12	V 91061626187		
		I 12	V Gonzalez Gonzalez	I Arian		

Figura 23. Descripción del caso de prueba Adicionar Especialista.

3.2.4- Resultados de las Pruebas de Aceptación.

El caso de prueba diseñado para la HU Evaluar Especialista incluye un conjunto de funcionalidades críticas para el desarrollo y despliegue exitoso de SEESCA. La tabla 15 muestra el resultado satisfactorio obtenido en la prueba de aceptación Evaluar Especialista.

Tabla 15. Caso de prueba para la HU Evaluar Especialista.

Caso de prueba de aceptación	
Código: HU4_P1	**Historia de Usuario**: 4
Nombre: Evaluar Especialista	

Descripción: Prueba para la funcionalidad de evaluar el especialista que será seleccionado para evaluar la seguridad del Control de Acceso.
Condiciones de Ejecución: El usuario debe seleccionar un especialista a evaluar. El usuario debe seleccionar aquellos criterios con los que cumple el especialista que se evalúa.
Resultados esperados: El sistema guarda la configuración seleccionada y muestra un mensaje informando el resultado de la evaluación realizada con éxito.
Evaluación de la prueba: Prueba satisfactoria.

Después de aplicar el caso de prueba correspondiente al proceso Evaluar Especialista, se pudo comprobar que dicha funcionalidad después de realizar los pasos necesarios para su ejecución esta funcionaba correctamente según se necesita.

Fueron diseñadas 7 pruebas funcionales. La realización de las pruebas tuvo lugar en un total de 3 iteraciones y los resultados obtenidos se muestran en la siguiente tabla.

Tabla 16. Iteraciones de pruebas funcionales.

Iteración	Resultado Satisfactorio	Resultado Insatisfactorio	Total
1	4	3	7
2	6	1	7
3	7	0	7

Conclusiones del capítulo:

Después de analizado los resultados de las diferentes métricas para el desarrollo y las pruebas realizadas al sistema, se arrojaron las siguientes conclusiones:

- SEESCA poseen un grado de responsabilidad baja equivalente al 60%, complejidad baja al 60% y reutilización alta al 60% respectivamente. Posibilitando que posea un nivel de tolerancia a los cambios y reutilización alto.
- El diseño propuesto se encuentra dentro de los niveles de calidad requeridos. Ya que presenta un porciento entre bajo y medio en su acoplamiento, su complejidad de mantenimiento y su cantidad de pruebas. El nivel de reutilización se encuentra entre medio y alto lo que implicando un mayor nivel de reutilización.
- Las Pruebas Funcionales demostraron que la aplicación se ejecuta de acorde a las necesidades, obteniéndose resultados satisfactorios a los casos de prueba utilizados.

Conclusiones Generales:

- SEESCA constituye una herramienta para la evaluación de las competencias de los especialistas de la seguridad del Control de Acceso de los SI del CEGEL.

- El uso de la metodología de desarrollo XP, el lenguaje de modelado UML, la herramienta de modelado Visual Paradigm, el gestor de base de datos SQlite, el lenguaje de programación JAVA y el entorno de desarrollo integrado Netbeans, contribuyó a la correcta implementación del sistema SEESCA.

- El análisis de estándares, métricas, recomendaciones, resoluciones, controles y decretos permitió la síntesis y agrupación de 10 criterios de evaluación para la selección efectiva de los especialistas de la seguridad del Control de Acceso de los SI.

- El empleo del algoritmo LSP para la evaluación, la aplicación de los patrones GRASP y GOF para el diseño y el empleo de buenas prácticas, permitió la obtención del SEESCA.

- El empleo de las métricas para el diseño conjuntamente con las pruebas realizadas al sistema, permitieron una correcta validación de la aplicación y los resultados arrojados fueron satisfactorios.

Recomendaciones:

Una vez concluida la presente investigación, se recomienda:

- Añadir nuevos algoritmos de evaluación, permitiendo dar solución a problemas similares en las organizaciones.
- Mantener un invariable estudio de las principales directrices en el área de la seguridad del Control de Acceso, ya que la ciencia y la tecnología están en constante evolución.
- Constatar el uso del SEESCA y que sea extendido al resto de los centros de la facultad y la universidad en general, contribuyendo a la fiabilidad de la seguridad del control de acceso.

BIBLIOGRAFÍA REFERENCIADA:

1. **Abreu, R. B. 2008.** Investigación de la metodología ágil Crystal Clear y su aplicación a un caso de estudio. Instituto Superior Politécnico José Antonio Echeverría. La Habana : s.n., 2008. Tesis de Grado.
2. **Acosta, César. 2010.** Extreme Programing (XP). [En línea] 2010. [Citado el: 06 de diciembre de 2013.] http://www.slideshare.net/cesar_acosta/programacin-extrema-extream-programming-xp..
3. **Alvear Rodriguez, Tatiana y Ronda Ceballos, Carlos. 2005.** Sistemas de Información para el Control de Gestión. Un apoyo a la gestión empresarial. Santiago de Chile : Universidad de Chile, 2005.
4. **BARABANOV, R., KOWALSKI, S. AND YNGSTRÖM, L. 2011.** Information Security Metrics State of the Art. DSV Report series. 2011. vol. 11-007.
5. **Bayucan, A.R. 1996.** Quantitative Evaluation of Windowed. San Francisco : State University,, 1996. Environments M.S. Thesis, .
6. **Ben-Ari, M. 1996.** Understanding Programming Languages. Chichester : John Wiley & Sons, 1996.
7. **Casas, Sandra I. 2009.** Aspectos tempranos, Un enfoque basado en tarjetas CRC. Universidad Nacional de la Patagonia . Santa Cruz, Argentina : s.n., 2009.
8. **CMS. 2008.** Content Management System. [En línea] marzo de 2008. [Citado el: 7 de noviembre de 2013.] http://www.cms.hhs.gov/SystemLifecycleFramework/Downloads/SelectingDevelopmentApproach.pdf.
9. **Cordero Calderón, Marcia Iveth. 2008.** Auditoria de Riesgos Informáticos del Departamento de Sistemas de Teleamazonas usando COBIT. Tesis. Quito : s.n., Abril de 2008.
10. **Cornejo, José Enrique González. 2008.** Dociris. [En línea] 1 de 2008. [Citado el: 10 de 2 de 2015.] http://www.docirs.com/uml.htm.
11. **Decreto-Ley. 2011.** Consejo de Estado. Habana : s.n., 2011. Decreto Ley. 281.
12. **Decreto-Ley-No.199. 1999.** Seguridad y Protección de la Información Oficial. s.l. : MINISTRO DEL INTERIOR, 1999.
13. **Díaz, Martínez. 2010.** Módulo Medios de Diagnóstico del Subsistema Web del Sistema Integral para la Atención Primaria. La Habana : s.n., 2010.

14. **Domínguez, Jimmy Arturo Brito. 2004.** Análisis y aprovechamiento de los sistemas de información para una eficiente auditoría y control de gestión. Guayaquil : s.n., 2004.
15. **Escribano, G. F. 2002.** Introducción a Extreme Programming. 2002.
16. **Fernández. 2010.** Sitio Oficial JAVA. [En línea] 2010. [Citado el: 06 de diciembre de 2013.] http://java.sun.com/people/jag/OriginalJavaWhitepaper.pdf.
17. **Figueroa, Roberth G, S., C. J. y Cabrera, A. A. 2010.** Metodologías tradicionales vs. Metodologías ágiles. 2010.
18. **FIPS-PUB-199. February 2004..** Standards for Security Categorization of Federal Information and Information Systems. February 2004.
19. **FIPS-PUB-200. 2006.** Minimum Security Requirements for Federal Information and Information Systems. 2006.
20. **García, Ing. José Alejandro Lugo. 2012.** MODELO PARA EL CONTROL DE LA EJECUCIÓN DE PROYECTOS BASADO EN INDICADORES Y LÓGICA BORROSA. La Habana : s.n., 2012. título de Máster en Gestión de Proyectos Informáticos.
21. **Garzaro, Manuel Guillén. 2007.** Information Technology Infrastructure Library (ITIL). MCSM. 2007.
22. **Hernández, Suany Leyva. 2013.** Sistema informático para la conformación automática de tribunales de tesis en la Facultad 2. La Habana : s.n., 2013.
23. **Herrmann, Stephen. 2001.** Application of Common Criteria to Telecomm . 2001. 2.
24. **IATAC. 2007.** Software Security Assurance. 2007.
25. **IBM. 2013.** Rational Rose. [En línea] IBM, 2013. [Citado el: 06 de noviembre de 2013.] http://www-03.ibm.com/software/products/es/rosetechdev/.
26. **Informática. 2010.** Revista de Ingeniería de Sistemas e Informática. 2010.
27. **ISO-19011. 2011.** Guías y Lineamientos para auditorias de sistemas de administración. 2011.
28. **Jalote, Pankaj. 2000.** CMM in Practice, SEI Series in Software Engineering. 2000.
29. **JAQUITH, A. 2002.** The Security of Applications: Not All Are Created Equal. In. New York. 2002. pág. p. 12.
30. **Jiménez, Alain López. 2011.** Desarrollo de un componente para modelación y visualización en tres dimensiones de objetos geológicos utilizando el framework VTK. La Habana : s.n., 2011.
31. **Kaczor, Kristian, 2011**. 5 Common Mistakes We Make Writing User Stories, 2011.
32. **Keith, Mike. 2009.** ProJPA2, Mastering the Java™ Persistence API. 2009.

33. **Larman, Craig . 2009.** UML y Patrones. 2009.
34. **Leivas, Daniel Hugo Enriquez. 2009.** Bases de Datos IBM AS / 400. [En línea] 2009. [Citado el: 06 de diciembre de 2013.] http://www.recursos-as400.com.
35. **Llorente, Omar Macias. 2011.** Componente visual para la representación de árboles genealógicos. La Habana : s.n., 2011.
36. **Lorenz, Mark and Kidd, jeff. 1994.** Object-oriented software metrics: a practical guide. . s.l. : Prentice-Hall, 1994. ISBN 013179292X.
37. **MAGERIT. 2006.** Ministerio de Administraciones Públicas. Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información. Madrid : s.n., 2006. V2.
38. **Martínez, Ramón Alexander Anglada. 2015.** Revista Cubana de Ciencias Informáticas. Revista Cubana de Ciencias Informáticas. [En línea] 2015. [Citado el: 5 de 2 de 2015.] http://scielo.sld.cu/scielo.php?pid=S2227-18992013000200006&script=sci_arttext. vol.7 no.2.
39. **Mestras Pavón, Juan. 2007.** [En línea] 2007. [Citado el: 25 de octubre de 2013.] http://www.fdi.ucm.es/profesor/jpavon/is2/Lab01RationalRose.pdf.
40. **Montiel, Daniel Ponsoda. enero del 2008.** Segundo curso de Administración de sistemas informáticos. enero del 2008. Primera versión.
41. **MSDN. 2008.** MSDN Visual Studio 2008 Development System. 2008.
42. **Muñoz, Maribel Silva. 2014.** Modelo para la Evaluación de la Seguridad del Control de Acceso de los Sistemas de Información de centros productivos de la UCI. Dpto Programación de la FICI, UCI. Habana : s.n., 2014. Tesis de Maestría.
43. **Netbeans. 2014.** Netbeans Comunity. [En línea] 2014. [Citado el: 14 de marzo de 2014.] www.netbeans.org/index_es.html.
44. **NIST. 2007.** NVD Common Vulnerability Scoring System Support. NIST National Institute of Standards and Technology . 2007.
45. **NIST-PUB-12.** An Introduction to Computer Security.
46. **Noble, Angela Martin y de Robert Biddle, James. 2004.** The XP Customer Role in Practice: Three Studies Agile. 2004.
47. **Object Management Group. 2014.** Object Management Group Business Process Model and Notation. [En línea] Object Management Group, Inc., 2014. [Citado el: 14 de marzo de 2014.] http://www.bpmn.org/.
48. **OGC. 2005.** ITIL Managing IT Services. TSO. Office of Government Commerce. London : s.n., 2005.

49. **Oracle. 2010.** Oracle Identity Management. [En línea] 2010. [Citado el: 25 de noviembre de 2013.] http://www.oracle.com.
50. **Ortiz, Yudisney Vázquez. 2007.** Análisis y diseño del sistema de información y gestión para la calidad del Software Educativo. La Habana : s.n., 2007.
51. **OWASP. 2008.** GUÍA DE PRUEBAS OWASP. 2008. V3.0.
52. **Palacio, Juan**. **2006.** El modelo Scrum. http://www.navegapolis.net.
53. **PB800-53, NIST. 2009.** Recommended Security Controlsf for Federal Information Systems and Organizations, . 2009.
54. **PostgreSQL Core Team. 2013.** PostgreSQL. [En línea] 2013. [Citado el: 25 de octubre de 2013.] http://www.postgresql.org/.
55. **Pozo, Boizán del. 2012.** Plugin de la herramienta Visual Paradigm para la evaluación del diseño Orientado a Objeto. La Habana : s.n., 2012.
56. **Pressman, Roger. 2002.** Ingeniería de Software “Un enfoque práctico” 5ta ed. s.l. : Graw Hill, 2002.
57. **Resolución No. 127 /2007**. Ministerio de la Informática y las Comunicaciones. 2007**.** La Habana : s.n., 2007.
58. **ResoluciónNo.60. 2012.** Normas del Sistema de Control Interno. Guia de Autocontrol General. 2012.
59. **SAMUELSON. 1977.** Information Systems and networks. Amsterdam: North Holland : s.n., 1977. pág. 3.
60. **SENN, J.A. 2001.** Análisis y Deseño de Sistemas de Información . s.l. : USA: McGraw-Hill., 2001.
61. **Sparks, Geoffrey. 2006.** Craftware. [En línea] 2006. [Citado el: 25 de 10 de 2013.] http://www.craftware.net/es/products/ea/modelo_de_proceso_de_negocio.pdf..
62. **STANDARD, I. AND TECHNIQUES. 2005.** Code of practice for information security management. International Standard. 2005.
63. **Synergix. 2013.** Tipos de requisitos: Funcional vs. No Funcional. [En línea] Synergix, 2013. [Citado el: 05 de marzo de 2014.] http://synergix.wordpress.com/2008/07/07/requisito-funcional-y-no-funcional/.

BIBLIOGRAFÍA CONSULTADA:

1. **Alarcón, Vicenç Fernández. 2006.** Desarrollo de sistemas de información. Una metodología basada en el modelado. Catalunya : Universitat Politècnica de Catalunya, SL, 2006. ISBN: 84-8301-862-4.
2. **Altova. 2010.** Altova UModel. [En línea] 2010. [Citado el: 25 de octubre de 2013.] http://www.altova.com/umodel.html.
3. **Alvarez Zurita, Flor María y Pamela, García Guzmán Anabel. 2007.** Implementación de un sistema de gestión de seguridad de la información basado en la norma ISO 27001, para la intranet de la Corporación Metropolitana de Salud. Quito : s.n., 2007.
4. **ARRIETA, Sebastián**. 2002. Prácticas Extremas en ORTsf. Universidad ORT Uruguay.
5. **Asensio, Rafael Menéndez-Barzanallana. 2014.** Ingeniería del software. Principales herramientas CASE del mercado y su uso. [En línea] Departamento Informática y Sistemas de la Universidad de Murcia, 2014. [Citado el: 14 de marzo de 2014.] http://www.um.es/docencia/barzana/IAGP/Enlaces/CASE_principales.html.
6. **AUDISEC. 2010.** Guía de implantación de un sisitema de gestión de seguridad de la información UNE – ISO/IEC 27001:2007. 2010.
7. **Baryolo, Oiner Gómez. 2012.** CAEM: Modelo de control de acceso para sistemas de información en entornos multidominios. La Habana : s.n., 2012.
8. **Boizán del Pozo, Yurnelis y Galano González, Alexis. 2012.** Plugin de la herramienta Visual Paradigm para la evaluación del diseño Orientado a Objeto. La Habana : s.n., 2012.
9. **Centro Nacional de Inteligencia - Centro Nacional Criptológico. 2009.** Las amenazas y vulnerabilidades sobre los Sistemas de Información. [En línea] 2009. [Citado el: 25 de 11 de 2013.] http://www.ati.es/IMG/pdf/vulnerabilidades_press.pdf.
10. **Cid Escalona, Lilian y Sarduy Sánchez, Ismel. 2009.** Análisis y diseño del subsistema Multimoneda del ERP-Cuba. La Habana : s.n., 2009.
11. **Ciolli, María Elena. 2007.** Testing de migración de aplicaciones distribuidas a entornos Web. 2007.
12. **CIS**. (Mayo de 2009). **The CIS Security Metrics**. v1.0.0
13. **COBIT Focus**. (Enero de 2014). COBIT Focus, 1, 2.
14. **COBIT. (s.f.)**. Un Marco de Negocio para el Gobierno y la Gestión de las TI de la Empresa, 5, 13.

15. **Diccionario de la Real Academia.** (s.f.).
16. **Dirección de Información. 2012.** Repositorio Institucional. [En línea] 2012. [Citado el: 06 de diciembre de 2013.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_04358_11/1/TD_04358_11.pdf.
17. **Downs, D.** Issues in Discretionary Access Control. Oakland, USA : IEEE Computer Society.
18. **Duharte, F. R. 2008.** Pruebas Unitarias de Software en la Plataforma J2EE. 2008.
19. **Ecured.** (2014). Obtenido de "http://www.ecured.cuSeguridad Informática. Capítulo 4 Control de acceso"
20. **—. 2013.** Venezuela corta Internet en todo el país por un ataque informático contra Maduro. [En línea] EDICIONES EL PAIS, S.L., 15 de abril de 2013. [Citado el: 10 de diciembre de 2013.]
21. **Esteban, Juan Jesús Muñoz. 2004.** Metodología para la incorporación de medidas de seguridad en sistemas de información de gran implantación. Madrid : s.n., 2004.
22. **Revista de Ingeniería de Sistemas e Informática. 2010** Julio - Diciembre vol. 7, N.º 2, 2010
23. **Internet Engineering Task Force. 2013.** Protocolo LDAP. [En línea] 2013. [Citado el: 25 de octubre de 2013.] http://www.ldap-es.org/contenido/04/12/1.-%C2%BFque-esldap%.
24. **ISO.** ISO 27000. [En línea] [Citado el: 21 de octubre de 2013.] WWW.ISO27000.ES.
25. **—. 2005.** ISO/IEC 17799. 2005.
26. ISO/IEC 17799. **International Organization for Standardization. 2005.** 2005.
27. **ISO/IEC FCD 25040. 2009.** Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) – Evaluation reference model and guide. 2009.
28. **Letelier, Patricio y Penadés, Carmen. 2008.** Métodologías ágiles para el desarrollo de software:eXtreme Programming (XP). 2008.
29. **Lobos, Maria Elena. 2013.** Aprende a Programar. [En línea] 2013. [Citado el: 06 de diciembre de 2013.] http://w.ww.emagister.com/curso-aprende-programar/concepto-lenguaje-programacion.
30. **Julián Marcelo Cocho. MAGERIT.** Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información. Guía para responsables del dominio protegible. V1
31. **Mendez, Zoilibeth Moreno. 2011.** Importancia y Prevención de La Seguridad de la Información . [En línea] 19 de junio de 2011. [Citado el: 17 de octubre de 2013.] http://controltecnologico.com/2011/06/importancia-y-prevencion-de-la.html.

32. **Microsoft. 2008.** Control de acceso. [En línea] febrero de 2008. [Citado el: 16 de octubre de 2013.] http://technet.microsoft.com/es-es/library/cc732699%28v=ws.10%29.aspx.
33. **Montesino Perurena, R**. **2013**. Gestión automatizada e integrada de controles de seguridad informática. Revista de Ingeniería Electrónica, Automática y Comunicaciones, 41**.**
34. Nuevas Tendencias en Sistemas de Información: Procesos y Servicios. **R. de Soto, Adolfo y Cuervo Fernández, Eva.**
35. **O'Brien, J. A. 2001.** Sistemas de Información Gerencial. Bogotá : Irwin McGraw-Hill, 2001.
36. **Orrego, V. M.** (junio de 2011). La gestión en la seguridad de la información. Revista Pensamiento Americano, 2, 21.
37. **Pankaj Jalote.** 2000. CMM in Practice, SEI Series in Software Engineering.
38. **Pinagapani, S. 2009.** A Comparative Study of Access Control Languages. Shanghai : IEEE Computer Society, 2009.
39. **Revista Pensamiento Americano,** 2, 21.
40. **Robles, R. 2008.** Application of Role-Based Access Control for Web Environment. s.l. : IEEE Computer Society, 2008.
41. Seguridad de la Información. **Centro Nacional de Información de la Calidad. 2012.** España : s.n., 2012.
42. Seguridad de la información en un mundo sin fronteras. **Ernst & Young Global Limited. 2011.** Mancera : s.n., 2011.
43. **Solis, E. 2008.** IBM: Seguridaden ambientes SOA. Retos, patrones y soluciones. [En línea] 2008. [Citado el: 11 de noviembre de 2013.] www.cert.uy/historico/pdf/IBMSOASecurity.pdf.
44. **Tao, W. 2010.** RBAC Permission Consistency Static Analysis Framework. Jiangsu : IEEE Computer Society, 2010.
45. **Trasobares, Alejandro Hernandez.** Los sistemas de información: evolución y desarrollo. Zaragoza : s.n.
46. **Vance, A y Molyneux, B. 2012.** Reducing Unauthorized Access by Insiders through User Interface Design: Making End Users Accountable. Hawaii, USA : IEEE Computer Society, 2012.
47. **Visual Paradigm For UML. 2012.** Visual Paradigm For UML. [En línea] 2012. [Citado el: 06 de diciembre de 2013.] http://www.visual-paradigm.com/product/vpsuite.
48. **Visual Paradigm International. 2013.** Visual Paradigm. [En línea] 2013. [Citado el: 25 de octubre de 2013.] http://www.visual-paradigm.com.

ANEXOS:

Tabla 17. HU Adicionar Criterio.

HU Adicionar Criterio	
Número: 1	**Nombre HU**: Adicionar Criterio
Modificación de Historia de Usuario Número:	
Referencia: No	
Programador: Arian González Au	**Iteración Asignada**: 1
Prioridad en Negocio: Alta **(Alta/Media/Baja)**	**Puntos Estimados(días)**: 8
Riesgo en Desarrollo: Bajo	**Puntos Reales(días)**: 11
Descripción: Comienza en el instante en que el administrador del sistema decide adicionar un criterio, donde la aplicación muestra la interfaz para adicionar el nuevo criterio. Seguidamente se da paso a llenar los campos (Descripción, Tipo Proceso y Criticidad). Finalmente hace clic en el botón Adicionar. Este proceso se repite cuantas veces se desee adicionar un criterio.	
Observaciones:	

Tabla 18. HU Adicionar Especialista.

HU Adicionar Especialista	
Número: 2	**Nombre HU**: Adicionar Especialista
Modificación de Historia de Usuario Número:	
Referencia: No	
Programador: Arian González Au	**Iteración Asignada**: 1
Prioridad en Negocio: Alta **(Alta/Media/Baja)**	**Puntos Estimados(días)**: 8
Riesgo en Desarrollo: Bajo	**Puntos Reales(días)**: 12
Descripción: Comienza en el instante en que el administrador del sistema decide adicionar un especialista, donde la aplicación muestra la interfaz para adicionar el nuevo especialista. Seguidamente se da paso a llenar los campos (Nombre, Apellidos y CI). Finalmente hace clic en el botón Adicionar. Este proceso se repite cuantas veces se desee adicionar un especialista.	

Observaciones:

Tabla 19. HU Modificar Criterio.

HU Modificar Criterio	
Número: 3	**Nombre HU**: Modificar Criterio
Modificación de Historia de Usuario Número:	
Referencia: No	
Programador: Arian González Au	**Iteración Asignada**: 1
Prioridad en Negocio: Alta **(Alta/Media/Baja)**	**Puntos Estimados(días)**: 10
Riesgo en Desarrollo: Bajo	**Puntos Reales(días)**: 15
Descripción: Comienza en el instante en que el administrador del sistema decide modificar un criterio, donde la aplicación muestra la interfaz para modificar el criterio seleccionado. Se muestra una tabla donde se selecciona un criterio y pulsamos el botón Modificar, seguidamente se muestran los datos actuales del criterio para que sean modificados. Finalmente hace clic en el botón Aceptar. Este proceso se repite cuantas veces se desee modificar un criterio específico.	
Observaciones:	

Tabla 20. HU Modificar Especialista.

HU Modificar Especialista	
Número: 4	**Nombre HU**: Modificar Especialista
Modificación de Historia de Usuario Número:	
Referencia: No	
Programador: Arian González Au	**Iteración Asignada**: 1
Prioridad en Negocio: Alta **(Alta/Media/Baja)**	**Puntos Estimados(días)**: 12
Riesgo en Desarrollo: Bajo	**Puntos Reales(días)**: 15
Descripción: Comienza en el instante en que el administrador del sistema decide modificar un especialista, donde la aplicación muestra la interfaz para modificar el especialista seleccionado. Se	

muestra una tabla donde se selecciona un especialista y pulsamos el botón Modificar, seguidamente se muestran los datos actuales del especialista para que sean modificados. Finalmente hace clic en el botón Aceptar. Este proceso se repite cuantas veces se desee modificar un especialista específico.
Observaciones:

Tabla 21. HU Eliminar Criterio.

HU Eliminar Criterio	
Número: 5	**Nombre HU**: Eliminar Criterio
Modificación de Historia de Usuario Número:	
Referencia: No	
Programador: Arian González Au	**Iteración Asignada**: 1
Prioridad en Negocio: Media **(Alta/Media/Baja)**	**Puntos Estimados(días)**: 7
Riesgo en Desarrollo: Bajo	**Puntos Reales(días)**: 8
Descripción: Comienza en el instante en que el administrador del sistema decide eliminar un criterio, donde la aplicación muestra la interfaz para eliminar el criterio seleccionado. Se muestra una tabla donde se selecciona un criterio y pulsamos el botón Eliminar. Este proceso se repite cuantas veces se desee eliminar un criterio específico.	
Observaciones:	

Tabla 22. HU Eliminar Especialista.

HU Eliminar Especialista	
Número: 6	**Nombre HU**: Eliminar Especialista
Modificación de Historia de Usuario Número:	
Referencia: No	
Programador: Arian González Au	**Iteración Asignada**: 1
Prioridad en Negocio: Media **(Alta/Media/Baja)**	**Puntos Estimados(días)**: 7
Riesgo en Desarrollo: Bajo	**Puntos Reales(días)**: 9
Descripción:	

Comienza en el instante en que el administrador del sistema decide eliminar un especialista, donde la aplicación muestra la interfaz para eliminar el especialista seleccionado. Se muestra una tabla donde se selecciona un especialista y pulsamos el botón Eliminar. Este proceso se repite cuantas veces se desee eliminar un especialista específico.
Observaciones:

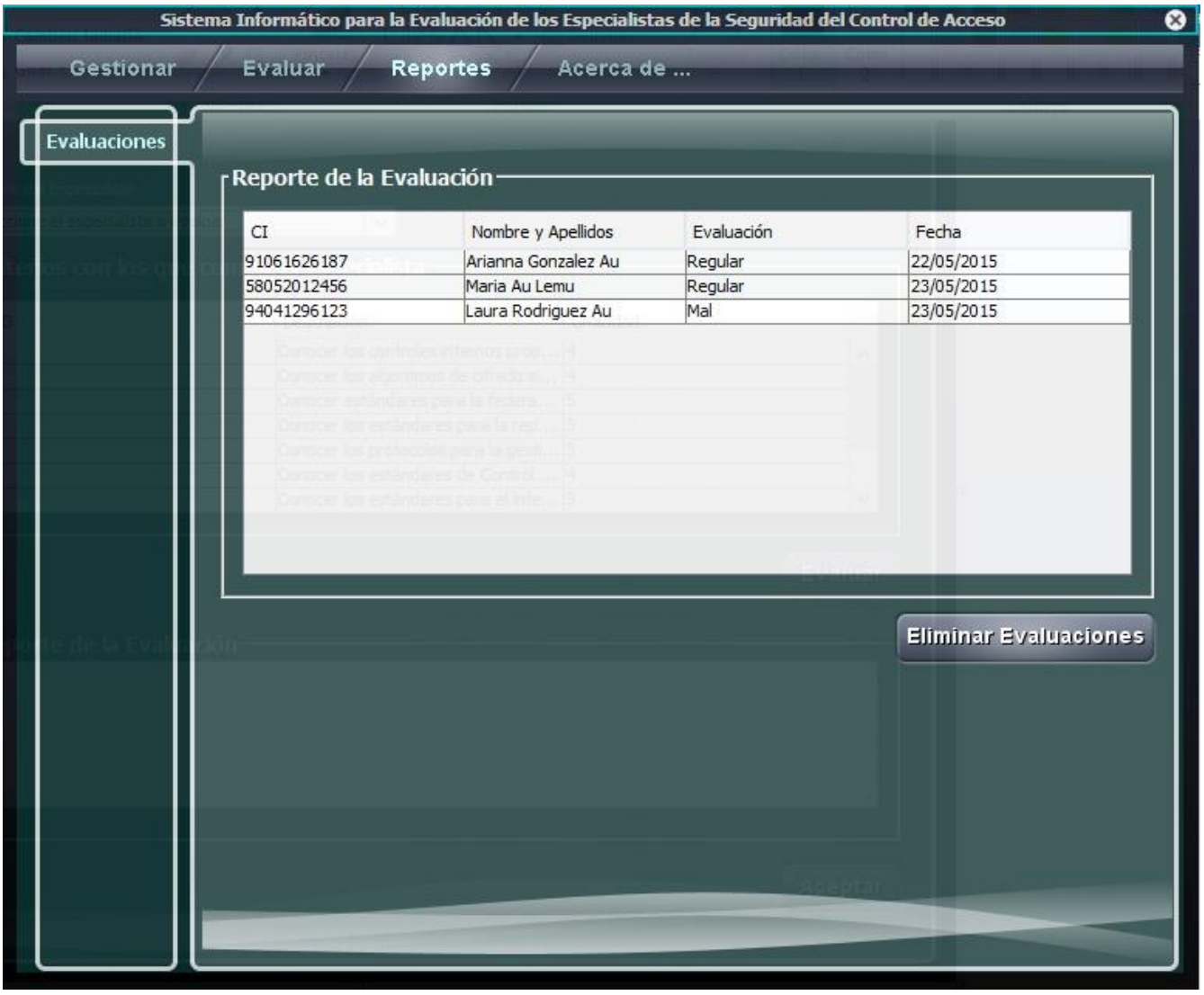

Figura 24. Interfaz Reportes de Evaluaciones.

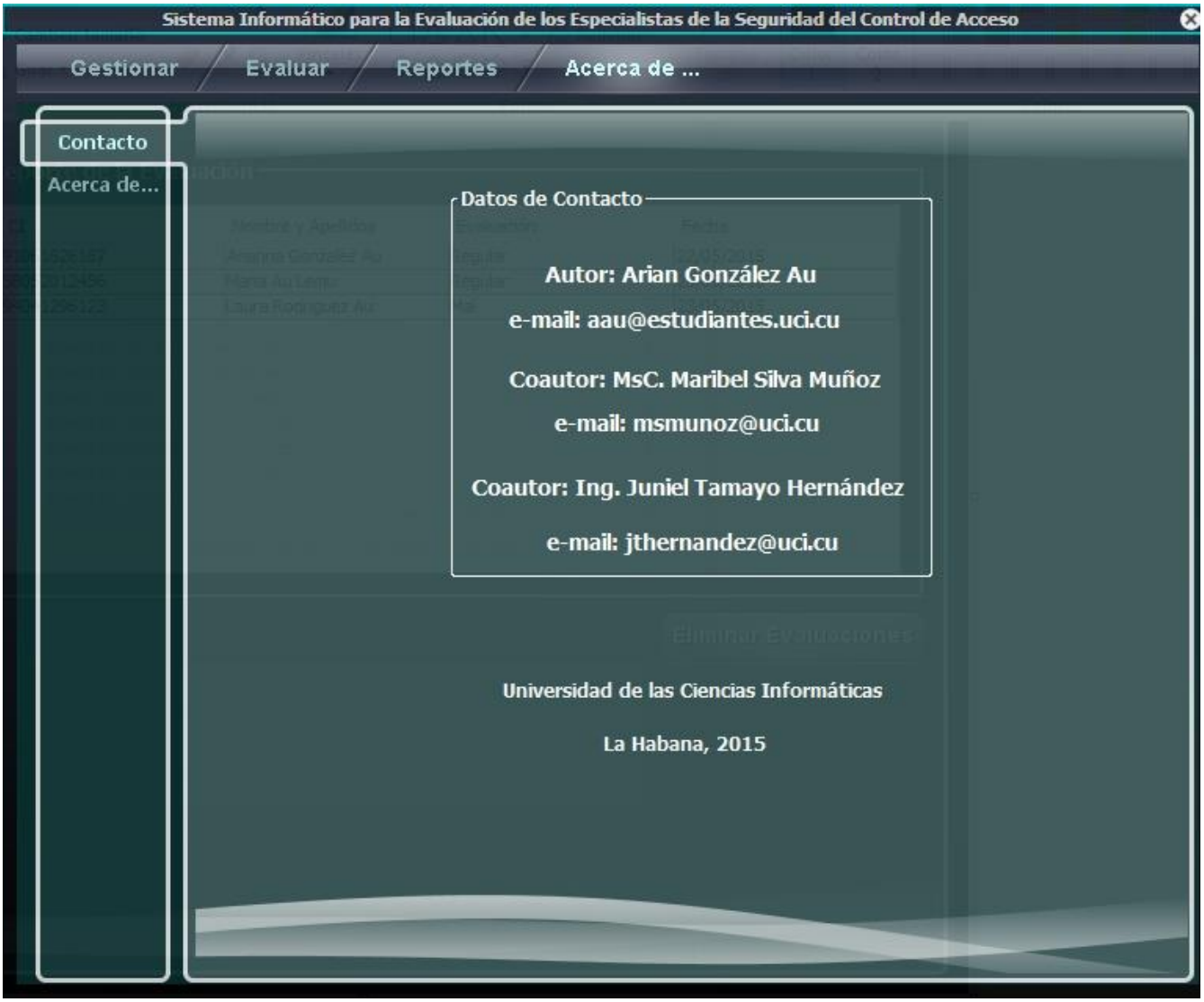

Figura 25. Interfaz de Contacto.

GLOSARIO DE TERMINOS

API: Applications Programing Interfaces. Conjunto de subrutinas que aportan los sistemas operativos a los programas de aplicación para el acceso a los recursos y servicios prestados por el ordenador.

Base de Datos: Conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. Una base de datos relacional es una base de datos que cumple con el modelo relacional, que no es más que un modelo de datos basado en la lógica de predicados y en la teoría de conjuntos que su idea fundamental es el uso de “relaciones”.

CASE: Acrónimo de Computer Aided Software Engineering (Ingeniería de Software Asistida por Ordenador), son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

Clases: Conjunto de objetos que comparten atributos, operaciones, relaciones y semántica, las mismas representan los conceptos fundamentales del sistema.

Empresa: Una empresa es el organismo social integrado por elementos humanos, técnicos y materiales cuyo objetivo natural y principal es la obtención de utilidades.

GRASP: General Responsibility Assignment Software Patterns de sus siglas en inglés, traducido como patrones generales de asignación de responsabilidades.

IDE: Integrated Development Environment (Ambiente Integrado de Desarrollo). Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

IEEE: Corresponde a las siglas de The Institute of Electrical and Electronics Engineers (el Instituto de Ingenieros Eléctricos y Electrónicos), es la asociación técnica y profesional, sin fines de lucro, más grande del mundo formada por profesionales de todas las disciplinas de la ingeniería. Su trabajo es promover la creatividad, el desarrollo y la integración, compartir y aplicar los avances en las tecnologías de la información, electrónica y ciencias en general para beneficio de la humanidad y de los mismos profesionales.

Lenguaje de Programación: Un lenguaje de programación es aquel elemento dentro de la informática que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis.

Linux: Sistema operativo libre tipo Unix.

Multiplataforma: Término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas. Por ejemplo, una aplicación multiplataforma podría ejecutarse en Windows en un procesador x86, en GNU/Linux en un procesador x86, y en Mac OS X en uno x86.

Programación Orientada a Objetos (POO): Es un paradigma de programación que define los programas en términos de "clases de objetos", objetos que son entidades que combinan estado (propiedades o datos), comportamiento (procedimientos o métodos) e identidad (propiedad del objeto que lo diferencia del resto). Expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar.

Software Libre: Es la denominación del software que respeta la libertad de los usuarios sobre su producto adquirido y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, modificado y redistribuido libremente.

SQL: Son las siglas de Structured Query Language, Lenguaje de consulta estructurado, es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar de forma sencilla la información de interés de una base de datos, así como también hacer cambios sobre la misma.

TIC: Las tecnologías de la información y la comunicación (TIC) agrupan los elementos y las técnicas utilizadas en el tratamiento y la transmisión de las informaciones, principalmente de informática, Internet y telecomunicaciones.

UML: lenguaje de modelado visual consistente en el cual se expresan los resultados de numerosas metodologías de orientación a objetos existentes.