Universidad de las Ciencias Informáticas

Facultad 3

Título: "Solución informática para la Compartimentación de la información en los sistemas que utilizan el Sistema de Gestión Integral de Seguridad ACAXIA."

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

Autores: Thais Figueras Garcia
Miguel La Llave Iglesias

Tutores: Ing. Mileidy M. Sarduy Pérez
Ing. Yoel Hernández Mendoza

Ciudad de La Habana, Junio 2012

"Año 54 de la Revolución"

DECLARACIÓN DE AUTORÍA.

Se declara que los autores de éste trabajo autorizan al Departamento de Tecnología de la Universidad de las Ciencias Informáticas a hacer uso del mismo para su beneficio.

Para que así conste se firma la presente a los _____ días del mes de_____ del año _____.

_____ _____

Thais Figueras Garcia Miguel La Llave Iglesias

_____ _____

Ing. Mileidy M. Sarduy Pérez Ing. Yoel Hernández Mendoza

DATOS DE CONTACTO.

Síntesis de los tutores:

Ing. Mileidy M. Sarduy Pérez.
Ingeniera en Ciencias Informáticas, graduada en el año 2009. Actualmente ocupa el rol de analista principal del proyecto de Gestión Integral de Seguridad (ACAXIA).

Ing. Yoel Hernández Mendoza.
Ingeniero en Ciencias Informáticas, graduado en el 2009. Actualmente se desempeña como Jefe del proyecto Gestión Integral de Seguridad (ACAXIA).

Síntesis de los autores:

Thais Figueras Garcia
Estudiante de la Universidad de Ciencias Informáticas.
E-mail: tfigueras@estudiantes.uci.cu.

Miguel La Llave Iglesias
Estudiante de la Universidad de Ciencias Informáticas.
E-mail: mlallave@estudiantes.uci.cu.

AGRADECIMIENTOS.

De Thais:

Por lo visto, prolongar el momento de escribir los agradecimientos no ha servido de nada, porque solo de pensar en ellos y todo lo que quisiera decir me llena de lágrimas los ojos…pero bueno, YA es hora.

Primero que nada a mi mamá, por ser la persona que siempre luchó conmigo, por ser la razón de llegar hasta el final, por ser madre, padre, hermana, amiga…por todo. A mi papá, porque a pesar de la distancia siempre ha sabido aconsejarme, porque siempre se ha hecho presente, porque siempre ha confiado en mí y está más consciente que yo misma de lo que puedo llegar a hacer y ser en esta vida. Gracias a Uds. 2 es que estoy hoy aquí.

A mis tíos Ana Mª y Julio, por ayudarme y quererme tanto. A mis abuelos maternos, en especial a mi abuela Teresa que desgraciadamente ya no está entre nosotros…no sabes cómo te extraño. A Mercedes, por esa llamada cada fin de semana. A mis abues Emilia y Eduardo….su princesa después de todo lo está logrando. Al resto de mi familia, porque a pesar de que nos vemos poco los tengo siempre presente.

Agradecer a mi novio, no creo que lo hubiera logrado sin ti. Llegaste a mi vida de la forma más inesperada y fuiste el motor impulsor para seguir adelante. Gracias por ser mi compañero de estudio y de la vida. Te quiero nene.

A esa nueva familia, a los Hernández y a los Mendoza, por recibirme y quererme como una más. Yuly, Lima…gracias por tantos buenos momentos. Roly…espero poder verte crecer como un hombre de bien, siguiendo los pasos de tu hermano.

A mis amigos de la UCI, que digo amigos…a mi familia de la UCI. Yane, mi mamita querida TQMMM. Layda, bruji gracias por hacerme reír. Adnier…gracias por preocuparte siempre. A Uds. 3 gracias por esas noches de conversación interminables, por esas comidas tan ricas, por estar siempre para compartir. Lary, gracias por enseñarme la bondad de la gente, por ser mamá Lary y cuidar siempre de mí. A Julito, por ser un amigo incondicional y ayudarme tanto en los momentos más difíciles. A mi gente de la 7, fue muy lindo contar con Uds. A los que conocí en la 3, en especial a Gloria y Fidel, gracias por su ayuda.

A la gente del proyecto por recibirme, por su forma de ser, por ayudarme tanto. A mis tutores, Mile y Yoel, gracias por sus consejos.

A mi compañero de tesis…por tantas horas de trabajo y de desvelo. Sorry por el estrés causado.

A la UCI, por todos esos buenos y malos momentos, por esos sueños hechos realidad, por los recuerdos que me llevo.

A todos GRACIAS.

T.

De Miguel:

A mi mamá por el apoyo que me dio, por la confianza que depositó en mí, por darme los mejores consejos del mundo, gracias a ella por ser mi razón de ser.

A mi hermana Ori a la cual quiero muchísimo, gracias por servirme de ejemplo y por empujarme cuando le decía que la cosa estaba difícil, diciéndome que yo podía llegar también y a mi hermano Rangel, por los consejos que siempre me dio.

A mi abuela Ina por estar siempre pendiente de mí, por el apoyo que me ha dado siempre.

A mi abuelita Tomasa que la quiero muchísimo, gracias por malcriarme cuando era apenas un chiquillo.

A mis tíos que siempre me han servido de ejemplo, gracias a ellos por haber depositado tanta confianza en mí.

A mis sobrinos a los cuales quiero con el alma, espero que en un futuro les sirva este trabajo como ejemplo de sacrificio profesional y que sepan que ellos lo pueden hacer mejor que yo.

A mis amigos Lugo, Eichel, Raniel, Rojas, que más que amigos los considero mis hermanos.

A mi novia Leydi por ser mi compañera durante todo este tiempo y apoyarme cuando pensaba que no se podía llegar a la meta.

A toda mi familia que siempre ha estado al tanto de mi trayectoria y resultados.

A mi papá que ya no se encuentra conmigo pero que sé que él está orgulloso de mí, a él gracias por esa frase que siempre escuche decirle a sus compañeros de trabajo: "Lo que importa no es con cuánto te gradúas, sino cuan preparado estas después que lo haces".

A mi compañera de tesis por tantas horas que dedicó a la confección de este trabajo tan importante para los dos, a ella mi más sincero agradecimiento.

A mis compañeros de grupo, que siempre estuvieron ayudándome durante toda mi trayectoria en la universidad.

A Mile y Yoel que tanto nos han ayudado en la realización de este Trabajo de Diploma.

A Jose, Quiroga, Janier, Javier, Damian, Oiner en fin a todos los que me brindaron su apoyo durante mi estancia en el proyecto ACAXIA.

A la revolución cubana, por haberme dado la posibilidad de formarme como profesional.

DEDICATORIA.

De Thais:

A mis padres, por creer en mí y apoyar cada uno de mis pasos.

De Miguel:

Dedico esta tesis a mi mamá con el cariño más grande del mundo.

RESUMEN.

El presente trabajo tiene como objetivo desarrollar una solución que permita gestionar la compartimentación de la información hasta el nivel de usuario para lograr mayor confidencialidad e integridad en aplicaciones que utilicen el Sistema de Gestión Integral de Seguridad ACAXIA. Con dicha solución se pretende delimitar y profundizar el nivel de autorización de los permisos otorgados a usuarios con un mismo rol en una entidad en dependencia de las funcionalidades que éstos realizarán, logrando con ello un mejor control de acceso a la información.

Para su desarrollo se realizó el estudio de algunos sistemas de seguridad con el objetivo de determinar los nuevos escenarios que se deberían cubrir; además se realizó un estudio de las herramientas y tecnologías a utilizar para el desarrollo de la solución y se identificaron los requisitos funcionales con que contaría la misma mediante técnicas de captura y validación de requisitos.

En la siguiente investigación se describen los estilos arquitectónicos y patrones de diseños a utilizar para implementar la propuesta de solución y mediante pruebas funcionales o de caja negra y pruebas de caja blanca se validará que el componente puede restringir aún más el acceso a la información de los sistemas suscritos a ACAXIA.

Palabras claves:

Compartimentación de la información, recursos.

ÍNDICE DE CONTENIDO

ÍNDICE DE ILUSTRACIONES.

ÍNDICE DE TABLAS.

INTRODUCCIÓN.

Hoy en día, las Tecnologías de la Información y las Comunicaciones (TIC), están sufriendo constantes cambios, siempre en pos del mejoramiento y facilidad de uso en el trabajo diario de las personas. Las empresas apuestan cada vez más hacia las Tecnologías de Información (TI), que con su implantación permiten llevar a cabo procesos claves dentro de un marco de negocio. Una muestra de ello, es la creación de sistemas capaces de gestionar la seguridad de la información que en ellas se maneja.

Desde los inicios de la Revolución Cubana, el bloqueo económico impuesto por los Estados Unidos ha frenado el desarrollo del país en todas las esferas; pero la solidaridad de otros gobiernos ha hecho posible una evolución que aunque todavía no se puede comparar con los países del 1er mundo, si ha dado a conocer el nombre de Cuba como una nación en desarrollo. La rama de las TIC lleva varios años siendo testigo de esto, con cambios sustanciales tales como la llegada de Internet y la informatización parcial de varios sectores del país como son la salud, la educación y la economía.

La Universidad de las Ciencias Informáticas (UCI) se creó al calor de la Batalla de Ideas en el año 2002, con la misión de "*Formar profesionales (…) altamente calificados en la rama de la Informática. Producir aplicaciones y servicios informáticos, a partir de la vinculación estudio-trabajo (…). Servir de soporte a la industria cubana de la informática.*" (UCI). La UCI cuenta con varios centros desarrolladores de software entre los que se encuentra el Centro de Informatización de Gestión de Entidades (CEIGE), el cual tiene entre sus objetivos el desarrollo de un sistema integral de seguridad que gestione la misma de forma centralizada. Partiendo de este objetivo se creó el sistema ACAXIA, que maneja los procesos de Autenticación, Autorización, Auditoría, Administración de perfiles y Administración de conexiones, además de gestionar parcialmente la Compartimentación de la información.

Actualmente ACAXIA realiza la compartimentación hasta el nivel de entidad[1], lo que permite que varios usuarios que comparten un mismo rol en una entidad puedan manejar la misma información indistintamente de sus funciones. Esta situación trae consigo que personal no autorizado tenga conocimiento de la información que no le corresponde la cual podría ser alterada, borrada o copiada; significando esto una brecha de seguridad relevante que va en contra de un principio básico de la seguridad informática; el mínimo privilegio que establece que "*se deben otorgar sólo los permisos estrictamente necesarios para efectuar las acciones que se requieran, ni más ni menos de lo solicitado*". (Pfleeger, y otros, Octubre, 2006)

[1] **Entidad:** Forma parte de la estructura de un país y puede comportarse como un ministerio, entidad, área o cargo.

Analizando la situación anteriormente expuesta, se impone el siguiente **problema científico:** ¿Cómo lograr mayor confidencialidad e integridad de la información en aplicaciones que utilizan el Sistema de Gestión Integral de Seguridad ACAXIA sobre escenarios Multientidad?

Para resolver esta problemática fue **objeto de estudio**, la seguridad en software de gestión, enmarcando el **campo de acción** en el control de acceso a la información.

Basado en la idea anteriormente expuesta se puede definir como **objetivo general** del presente trabajo:
Desarrollar una solución informática que permita gestionar el control de acceso a la información hasta el nivel de usuario para lograr mayor confidencialidad e integridad en aplicaciones que utilizan el Sistema de Gestión Integral de Seguridad ACAXIA.

Se plantean además, como **objetivos específicos**:

- Crear el marco teórico de la investigación, enfatizando en la Compartimentación de la información a nivel de usuario en los sistemas de gestión de seguridad.
- Realizar análisis y diseño de la solución propuesta.
- Implementar la solución propuesta.
- Validar la solución propuesta.

Para lograr los objetivos propuestos se plantearon las siguientes **tareas generales:**

- Confeccionar el marco teórico conceptual de la investigación a partir de una búsqueda y revisión bibliográfica.
- Investigar sobre patrones de diseño para la elaboración de los modelos de diseño.
- Identificar los estándares de codificación definidos por la Línea de Arquitectura y estudiar el marco de trabajo del proyecto.
- Identificar las herramientas y tecnologías a emplear para el desarrollo de la aplicación.
- Investigar sobre soluciones de control de acceso.

Tareas a cumplir por la estudiante Analista:

- Analizar los procesos para la Compartimentación de la información.
- Realizar el modelo conceptual.
- Identificar los requisitos funcionales a resolver.
- Actualizar el documento de especificación de requisitos del proyecto ACAXIA.
- Realizar la descripción de requisitos funcionales.
- Definir los prototipos de interfaz de usuario para la validación de los requisitos.
- Realizar el diagrama de componente.

- Realizar el diagrama de clases correspondiente al componente.
- Generar matriz de trazabilidad de los datos utilizando la herramienta OSRMT.
- Realizar el modelo de datos.

Tareas a cumplir por el estudiante Desarrollador:

- Implementar las interfaces a partir de los prototipos definidos.
- Implementar los requisitos especificados para los procesos de Compartimentación de la información.
- Realizar los diseños de casos de prueba.
- Realizar pruebas de caja blanca.
- Realizar pruebas de caja negra.

Para desarrollar este trabajo se ha planteado la siguiente **idea a defender**: Si se realiza la compartimentación de la información hasta nivel de usuario, se podrá mejorar la confidencialidad e integridad de la información en aplicaciones que utilizan el sistema ACAXIA y así lograr que la información sea manipulada y accedida de forma correcta y por el personal autorizado para hacerlo.

Los siguientes **métodos teóricos** sustentan el trabajo de investigación:

- ***Analítico – Sintético****:* permitió analizar y descomponer los conocimientos generales sobre los sistemas de seguridad analizados. Resultando muy importante a la hora de enfocar la investigación en los elementos más significativos, arrojando ideas y conclusiones mucho más prácticas y concretas.
- ***Inductivo – Deductivo****:* permitió analizar la particularidad de cómo se tratan los temas de autorización y compartimentación en los sistemas de seguridad, reflejando los puntos comunes en los sistemas.
- ***Histórico – Lógico****:* su empleo permitió analizar la trayectoria de la seguridad para los sistemas de software, su evolución y las últimas tendencias.
- ***Modelación****:* permitió crear determinada correspondencia entre la compartimentación que realizan los sistemas de seguridad existentes y la que se pretende alcanzar con el componente que aquí se propone.

El documento cuenta con la siguiente estructura:

Capítulo 1: *Fundamentación Teórica*. Se realiza un estudio del estado del arte sobre el control de acceso en sistemas de seguridad, los estilos, patrones arquitectónicos y de diseño existentes y se describen las tecnologías, metodologías y herramientas utilizadas por el CEIGE para el desarrollo de la solución.

Capítulo 2: *Propuesta de solución*. Se describen los requisitos funcionales y no funcionales con que contará el componente Compartimentación de la información. Se elabora el diagrama de clases con estereotipos web correspondiente a la solución y por último se describe cada una de las clases, patrones de diseño y estilos arquitectónicos definidos para el desarrollo de la solución.

Capítulo 3: *Implementación y Pruebas*. Se especifican aspectos de interés para el proceso de implementación tales como los estándares de notificación y codificación para el código, el diagrama de componentes que conforma la solución y el diagrama de despliegue donde se identifican los nodos necesarios para desplegar la misma. Se especifican además los resultados de la validación del componente mediante métricas de diseño, pruebas funcionales y de caja blanca y mediante un caso de estudio se describen las mejoras que aporta dicha propuesta de solución al proceso de compartimentación de la información que se realizaba anteriormente en ACAXIA.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA.

INTRODUCCIÓN.

En este capítulo se aborda acerca del origen del control de acceso y los objetivos fundamentales de la Seguridad de la información. Se especifican los conceptos que se van a tratar a lo largo de ésta investigación. Además se realiza un estudio de distintos Modelos de Control de Acceso y se estudian varios sistemas de seguridad con características comunes a ACAXIA. Se puntualizan algunos de los estilos y patrones arquitectónicos y de diseño existentes y algunas técnicas para la captura y validación de requisitos. Por último, se estudian las características de la metodología, tecnologías y herramientas determinadas por el centro CEIGE para el desarrollo de soluciones en el Departamento de Tecnología.

DESARROLLO.

El constante desarrollo alcanzado en la rama de la informática y las comunicaciones ha traído consigo la digitalización de la información. Debido a las facilidades que existen para la transmisión de dicha información, es que surge la necesidad de protegerla y con ella el Control de Acceso sobre las áreas y el Control de Acceso a la información.

El Control de Acceso sobre las áreas o seguridad física es aquella que se impone para controlar el acceso a una instalación o a recursos almacenados en medios físicos. (CA. Identificación, Autenticación y Autorización.) Un ejemplo de esto es que los estudiantes y trabajadores de la UCI tienen que presentar en la entrada una credencial para acceder a la universidad. Como mejoras a ésta técnica surgieron los códigos electrónicos, tarjetas magnéticas e inteligentes y técnicas biométricas. Estas últimas son capaces de controlar mediante factores característicos propios de cada persona quién tiene acceso a un área determinada y quién no.

El Control de Acceso a la información es también de vital importancia, por lo que se hizo necesario establecer medidas de seguridad lógica para controlar quién maneja la información, para garantizar y preservar la confidencialidad, integridad y disponibilidad de la misma (**Figura 1**). Esto se logra implementando un conjunto adecuado de controles que abarcan políticas, prácticas, procedimientos, estructuras organizacionales y funciones del software (Berzosa) y es en ésta rama del control de acceso donde se hace énfasis en la investigación.

Figura 1: Objetivos de la Seguridad Informática.

En la Figura 1 se muestran los objetivos de la Seguridad Informática. (Pfleeger, y otros, Octubre, 2006) Ellos son:

Confidencialidad: se refiere a que solo personal autorizado conocerá la información.

Integridad: la información no será alterada, borrada, reordenada o copiada. Debe permanecer en su estado original, sin haber sido alterada por personal o de forma no autorizado.

Disponibilidad: solo las personas autorizadas, en un momento determinado, pueden acceder a la información.

1.1 Conceptos Fundamentales.

1.1.1 Compartimentación.

División de la información sensible y de los elementos y recursos de un sistema de información en unidades menores; basándose en la necesidad de conocer, a fin de reducir el riesgo de accesos no autorizados. (2009)

En el presente trabajo se define la compartimentación atendiendo a tres niveles fundamentales propuestos en ACAXIA:

- ***Compartimentación a nivel de dominio***: permite gestionar la accesibilidad a entidades, sistemas, roles, funcionalidades y acciones definidos en determinado dominio.
- ***Compartimentación a nivel de entidad***: permite gestionar el acceso a sistemas, roles, funcionalidades y acciones definidas en una entidad.
- ***Compartimentación a nivel de usuario***: permite gestionar el acceso de un usuario a la información que gestiona otro usuario atendiendo a un criterio determinado.

1.1.2 Control de acceso.

El control de acceso es un conjunto amplio de políticas dentro de un sistema de gestión de identidad que definen las normas de todo lo que a un usuario se le permite hacer en el ámbito de

aplicación de dicho sistema. El administrador de identidades permite que estas políticas se definan en un alto nivel independiente de los sistemas específicos, por lo que se pueden aplicar a cualquier sistema, ya que no dependen del mismo. (Pfleeger, y otros, Octubre, 2006)

1.2 Modelos de Control de Acceso.

1.2.1 Modelo de Control de Acceso Discrecional (DAC).

El modelo de control de acceso DAC[2] establece que el autor de un objeto es quien decide cómo protegerlo y está autorizado a otorgar permisos sobre el mismo a otros usuarios. Esto ofrece al usuario gran flexibilidad, ya que éste controla personalmente su objeto; (Berzosa) pero no es muy recomendable para sistemas cuyo objetivo es proteger la información en vez de compartirla ya que un sistema con este modelo no está protegido contra los abusos o errores que puedan cometer los usuarios.

1.2.2 Modelo de Control de Acceso Obligatorio (MAC).

El MAC [3] establece una política de seguridad multinivel, en la que los sujetos y objetos pertenecen a niveles de seguridad predefinidos por un administrador o el mismo sistema y es a éstos sistemas a los que se les concede los permisos.

Los principios comunes, utilizados para determinar los permisos de los usuarios se establecen como: ***Acceso de lectura hacia abajo,*** donde el usuario puede acceder a cualquier información que se encuentre en o por debajo de su nivel de seguridad y ***Acceso de escritura hacia arriba***, que otorga acceso y permisos de escritura sobre la información almacenada en su nivel de seguridad o superior al mismo. (Ibarra Naranjo, y otros)

Aunque los modelos MAC representan mecanismos de protección mucho más sólidos y tratan con requerimientos de seguridad más específicos que los modelos DAC no cuentan con mucha flexibilidad ni proporcionan soluciones factibles. Es por eso que su uso se ha habituado al entorno militar, donde la clasificación de la información es lo más importante. (Berzosa)

1.2.3 Modelo de Control de Acceso Basado en Roles (RBAC).

El RBAC[4] es una tecnología de forma jerárquica que describe un grupo de usuarios que pueden desempeñar un conjunto de roles y realizar operaciones en las que utilizan un conjunto de objetos como recurso. (Sánchez, y otros)

[2] **DAC** del inglés Discretionary Access Control.
[3] **MAC** del inglés Mandatory Access Control.
[4] **RBAC** del inglés Role Based Access Control.

Las relaciones (**Figura 2**) se establecen como:

- Relaciones entre usuarios y roles. Un usuario puede desempeñar varios roles y un rol puede ser desempeñado por varios usuarios a la vez.
- Los permisos son un conjunto de operaciones que se pueden realizar sobre cada uno de los objetos.
- Se determina las operaciones sobre un objeto que puede realizar un rol determinado.

RBAC incluye un conjunto de sesiones donde cada sesión es la relación entre un usuario y un sub-conjunto de roles establecidos en el momento de activar la sesión. Cada usuario tiene asociada una o más sesiones. Los permisos disponibles para un usuario son el conjunto de permisos asignados a los roles que están activados en todas las sesiones del usuario. (Sánchez, y otros)

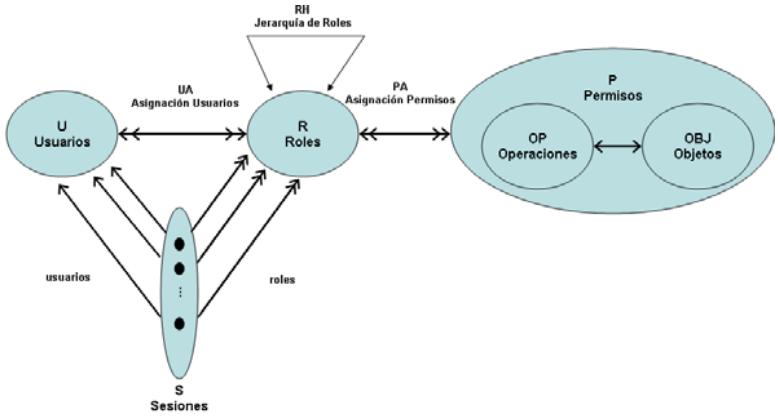

Figura 2: Elementos básicos de RBAC.

RBAC modela además una jerarquía de roles que permite a un rol específico, heredar los permisos de todos los roles que se encuentren en un nivel jerárquicamente inferior a él.

1.2.4 Valoración de los modelos estudiados.

El estudio realizado demuestra que los modelos DAC y MAC no satisfacen todas las necesidades para la gestión de seguridad en grandes sistemas de información; ya que DAC es muy débil para un control de acceso efectivo y MAC es extremadamente rígido.

Luego del estudio de los modelos de control de acceso y el tipo de seguridad que maneja ACAXIA, se llega a la conclusión que tal cual lo describe el modelo RBAC, el componente deberá ser capaz de gestionar roles para los usuarios en dependencia de la funcionalidades que se desea que ejecuten los mismos dentro del sistema y se deberá además otorgar a los usuarios que posean dichos roles los permisos necesarios, según las políticas de seguridad establecidas en ACAXIA.

1.3 Sistemas de Gestión de Seguridad.

A continuación se describen algunos sistemas de seguridad seleccionados por manejar internamente el control de acceso a la información, con el objetivo de determinar sus similitudes y diferencias con ACAXIA y determinar si manejan la compartimentación de la información hasta el nivel requerido para dar solución a la problemática planteada.

1.3.1 Tivoli Identity Manager.

IBM Tivoli Identity Manager es una solución de suministro a los usuarios automatizada y basada en políticas, que gestiona roles de usuario, identidades y derechos de acceso en toda la infraestructura de TI[5]. Se trata de un software de gestión de identidades seguro, fácil de desplegar y utilizar. Ayuda a las empresas a cumplir con las normativas, gestionar riesgos y permitir una colaboración segura. Entre sus características sale a relucir que: (IBM)

- Gestiona de manera automática roles, cuentas y derechos de acceso en todo el ciclo de vida del usuario, desde la incorporación hasta la finalización. Esto reduce los costes generales y elimina los errores manuales.
- Establece una separación de tareas para reforzar la seguridad y el cumplimiento. Asocia los requisitos que evitan conflictos empresariales con los roles y políticas de suministro que rigen los derechos de acceso de usuario.
- Corrige y elimina los derechos de acceso que no cumplen con las normativas mediante flujos de trabajo de rectificación periódicos o, de forma automática, a través de políticas de control de acceso basado en roles. Esta potente característica proporciona más detalles útiles para el auditor a fin de demostrar la conformidad.

1.3.2 OpenERP.

OpenERP es un sistema de gestión empresarial que cubre las necesidades de las áreas fundamentales en una empresa. Uno de sus principales atractivos es que maneja internamente su propia seguridad mediante permisos y restricciones, es capaz de controlar la seguridad de la información de la empresa en cuestión.

En el sistema se crean diferentes grupos que representan las diferentes funciones que se pueden desarrollar en la empresa. Según la función a desarrollar por cada grupo, son los permisos que se otorgarán a los usuarios pertenecientes al mismo. Tratando de escoger siempre el esquema que brinde mayor flexibilidad, pero a la vez, también mayor seguridad a la información del sistema. (OpenERP)

[5] **TI:** Tecnología Informática.

Cabe destacar que un mismo usuario puede pertenecer a varios grupos y contar con diferentes permisos o privilegios en cada grupo.

1.3.3 Proceso de Autenticación, Autorización y Auditoría (AAA) para aplicaciones basadas en servicios web XML. Utilizado en los proyectos del centro CESIM.

El Centro de Informática Médica (CESIM), de la UCI, es el encargado de desarrollar productos, servicios y soluciones informáticas dedicadas al área de la salud; cuyo objetivo es el mejoramiento de la calidad en la atención médica.

Para el mejoramiento de la seguridad en sus aplicaciones, el Msc. Karel Gómez Velázquez desarrolló éste producto de software capaz de proporcionar eficaces procesos de Autenticación, Autorización y Auditoría (AAA); el cual permite gestionar eficientemente usuarios, asignación de roles y privilegios de acceso para los sistemas creados en CESIM. Ofrece además un control sobre los procesos de trazabilidad y auditoría, que controlan estrictamente las operaciones en que se involucran los usuarios del sistema.

¿Cómo maneja el proceso de Autorización y Control de Acceso?
A partir de los parámetros de usuario y contraseña, el servicio devuelve un arreglo con la lista de derechos que posee el usuario autenticado. Dicho arreglo contiene por cada *componente*[6], a los que tiene acceso el identificador del componente, el nombre del componente, la dirección URL de su página principal y una lista de los roles que tienen acceso al componente. Esta lista de roles posee por cada *rol* el identificador del mismo, el nombre del rol y una lista de los niveles en los que este tiene privilegios de acceso. Por su parte la lista de niveles describe, por cada uno, el identificador del nivel, el nombre y la lista de los *servicios* que pueden ser accedidos en ese *componente*, por ese *rol* y en ese *nivel*. Cada servicio se describe a su vez mediante el identificador del servicio y su nombre. (Gómez Velázquez)

1.3.4 Valoración de los sistemas estudiados.

Cada uno de los sistemas de seguridad estudiados presentan características comunes con el sistema ACAXIA tales como:

- Gestiona roles, cuentas y derechos de acceso en todo el ciclo de vida del usuario.
- Crean diferentes grupos que representan las diversas funciones que se pueden desarrollar.

Según la función a desarrollar por cada grupo, son los permisos que se le otorgarán a los usuarios pertenecientes al mismo.

[6] **Componente:** Unidad de software ejecutable que representa el núcleo de una aplicación.

- Crean listas de derechos por cada usuario, teniendo en cuenta los roles que puede poseer el mismo y los permisos asignados a cada rol.

El estudio realizado arrojó que estos sistemas, aunque cubren escenarios comunes a los de ACAXIA, gestionan el control de acceso dependiendo de sus características propias y no brindan solución a la problemática planteada al inicio de ésta investigación.

1.4 Ingeniería de Requisitos.

La ingeniería de requisitos (**¡Error! No se encuentra el origen de la referencia.**) tiene como objetivo garantizar la correcta descripción de los requisitos para evitar futuros errores y lograr la reducción de tiempo en la construcción de un software. En ella se identifican dos aspectos muy importantes: el primero es el propósito del sistema que se va a desarrollar y el segundo, el contexto en el que será usado.

1.4.1 Técnicas de Captura de Requisitos.

Existen un número de técnicas para llevar a cabo el proceso de captura de requisitos, pero depende del equipo de desarrollo determinar cuál o cuáles son las más indicadas a usar en su caso particular. Las principales son: (Goguen, y otros, 1993) (Escalona Cuaresma, y otros, 2006/2007)

- **Entrevistas y cuestionarios:** permite al equipo de desarrollo interpretar las necesidades del cliente. Pueden ser lo mismo provechosos o no ya que dependen de las habilidades del entrevistador y los entrevistados para obtener información con la mayor calidad posible.
- **Desarrollo conjunto de aplicaciones:** proviene del inglés *JointApplicationDevelopment* (JAD) y es una técnica exploratoria que, aunque puede ser muy costosa por la cantidad de personal que involucra, es muy popular, ya que incluye a los usuarios como participantes activos en el proceso de desarrollo de sistemas. De ésta técnica puede surgir una declaración bastante fiel de los requisitos.
- **Tormenta de ideas:** consiste en la acumulación de ideas sin prejuicios y valoraciones que puedan descartarlas y aunque no evidencia los detalles concretos que se pueden necesitar del sistema, es muy común en los comienzos del proceso de ingeniería de requisitos.
- **Mapas conceptuales:** Es otra técnica bastante común, usada para la representación gráfica de las ideas y sus relaciones.
- **Escenarios:** valiosos medios para proporcionar contexto a las exigencias del consumidor. Proporcionan un marco para preguntas sobre tareas del usuario permitiendo preguntas "*y si*" y "*cómo se hace esto*".
- **Comparación de terminología:** se utiliza para complementar otras técnicas, pero consiste en identificar todos los términos con los que se trabajará durante el desarrollo del

sistema. Para su correcta utilización, es necesario identificar el uso de términos diferentes para los mismos conceptos (correspondencia), misma terminología para diferentes conceptos (conflictos) o cuando no hay concordancia ni en el vocabulario ni en los conceptos (contraste).

- **Reuniones:** el propósito de éstas es intentar alcanzar un efecto aditivo por el que un grupo de gente puede obtener más penetración en los requisitos del software que trabajando individualmente. Ellos pueden inspirarse y refinar las ideas que pueden ser difíciles de traer a la superficie usando entrevistas. Otra ventaja es que dejan a los stakeholders[7] reconocer donde hay requisitos en conflicto. Cuando se aplica bien, esta técnica puede resultar en un rico y constante sistema de requisitos que difícilmente sería realizable de otro modo.
- **Observación:** los ingenieros de software aprenden sobre las tareas del usuario sumergiéndose en la observación de cómo los usuarios obran recíprocamente con su software. Estas técnicas son relativamente costosas, pero son instructivas porque ilustran que muchas tareas del usuario y procesos del negocio son demasiado sutiles y complejos para que sus agentes los describan fácilmente.

1.4.2 Técnicas de Validación de Requisitos

Las técnicas de validación de requisitos mencionadas a continuación son las que se encuentran en el libro "*Ingeniería de Software*" v.7 de Ian Sommerville: (Sommerville, 2005)

- **Revisiones:** los requisitos son analizados sistemáticamente por un equipo de revisores, en busca de anomalías y/u omisiones.
- **Prototipos**: se muestra un modelo ejecutable del sistema a los usuarios finales y los clientes, para que puedan ver si dicho modelo cumple con sus necesidades reales.
- **Generación de casos de prueba:** la elaboración de casos de prueba puede revelar los problemas con que puede contar un requisito y es parte fundamental de la programación externa.

La actividad de validación tiene como entrada el documento de especificación de requisitos, prototipos del software y listas de chequeo de la especificación de requisitos de software y como salida se obtiene una lista de problemas identificados y la aprobación del documento de especificación de requisitos de software.

1.4.3 Técnicas de Captura y Validación de requisitos seleccionadas.

Durante el proceso de ingeniería de requisitos se propone, como Técnicas de Captura de requisitos realizar varias **reuniones**, en las que mediante **mapas conceptuales** y **tormenta de ideas** se pueda llegar a la conclusión de cuáles serían los requisitos funcionales de software del componente Compartimentación de la información.

[7] **Stakeholders:** Persona o entidad interesada en la realización de un proyecto o tarea.

Teniendo en cuenta la importancia del proceso de Validación de Requisitos se propone realizar varias **Revisiones** en las que estén presentes la gran mayoría del personal involucrado en la solución y se propone la creación de los **Prototipos de Interfaz de Usuario** y la **Generación de casos de prueba** para probar cada requisito identificado.

1.5 Diseño de Software.

El diseño de software consiste en aplicar distintas técnicas y principios, con el objetivo de definir un producto con los suficientes detalles como para permitir su realización física. Es el proceso mediante el cual se traducen los requisitos a una representación técnica del software que se va a desarrollar. El diseño debe proporcionar una idea completa de lo que es el software; debe ser una guía que puedan leer y entender los desarrolladores y todo el personal encargado de realizar las pruebas y el mantenimiento del software. (Diseño y Programación Orientada a Objetos, 2004/2005)

1.5.1 Arquitectura de Software.

La arquitectura de software es una forma abstracta de representar el software. Tiene disímiles definiciones entre ellas:

"*La arquitectura del software de un programa o sistema de computación es la estructura o estructuras del sistema que comprende los elementos del software, las propiedades externamente visibles de esos elementos, y las relaciones entre ellos*". (Bass, y otros, 2003)

1.5.2 Estilos arquitectónicos

Los estilos arquitectónicos son transformaciones impuestas al diseño de todo un sistema y tienen como objetivo fundamental establecer una estructura para todos los componentes del mismo y una forma de organizar su arquitectura. (Reynoso, y otros, Marzo, 2004)

1.5.3 Patrones.

Un patrón es una regla que consta de tres partes, y expresa una relación entre un contexto, un problema y una solución. Por lo general, sigue el siguiente esquema: (Buschmann, y otros, 1996)

- *Contexto*: situación de diseño en la que aparece un problema de diseño.
- *Problema*: conjunto de fuerzas que aparecen repetidamente en el contexto.
- *Solución*: configuración que equilibra estas fuerzas. Abarca:
  - Estructura con componentes y relaciones.
  - Comportamiento a tiempo de ejecución: aspectos dinámicos de la solución, como la colaboración entre componentes y la comunicación entre ellos.

Un patrón, al igual que un estilo, se enfoca en imponer transformaciones a la arquitectura de un software, mientras que soporta el desarrollo, mantenimiento y evolución de sistemas complejos.

1.5.3.1Patrones arquitectónicos.

La selección de un patrón arquitectónico es una decisión a tomar fundamental para el desarrollo de un sistema. Ya que dichos patrones expresan, según Buschmann (Buschmann, y otros, 1996), el esquema de organización estructural fundamental, proveen un conjunto de subsistemas predefinidos, especifican sus responsabilidades e incluyen reglas y pautas para la organización de las relaciones entre ellos. Buschmann propone además que son plantillas para arquitecturas de software concretas, que especifican las propiedades estructurales de una aplicación y tiene un impacto en la arquitectura de subsistemas.

1.5.3.2 Patrones de Diseño.

Un patrón de diseño es la solución a determinado problema de diseño que se presente en el desarrollo de un sistema. Entre sus características debe incluir la habilidad de ser reutilizable y aplicable a diferentes problemas de diseño en distintas circunstancias.

Patrones GRASP.

General Responsibility Assignment Software Patterns (GRASP), en español, Patrones generales de software para asignar responsabilidades; describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

Patrones GoF.

Los patrones de diseño GoF se dieron a conocer a principios de los años 90 con el libro "*Design Patterns. Elements of Reusable Object-Oriented Software.*". En dicho libro se hace una recopilación de 23 patrones de diseño comunes, clasificados en tres grupos de acuerdo a su naturaleza:

- *Creacionales*: concierne al proceso de creación de objetos.
- *Estructurales*: tratan la composición de clases y/o objetos.
- *De Comportamiento*: caracterizan las formas en las que interactúan y reparten responsabilidades las distintas clases u objetos.

1.6 Modelo de Desarrollo de Software.

La elección de un modelo o metodología de desarrollo de software, depende de varios factores tales como: tipo de proyecto o sistema a desarrollar, herramientas a utilizar en el mismo, las personas que lo van a desarrollar, los recursos disponibles y las especificaciones solicitadas por el cliente para el que se esté trabajando. Ya que estos factores no se muestran de la misma forma en todos los sistemas, se ha hecho muy difícil establecer modelos de desarrollo estándares a seguir, ya que el mismo debe ser un proceso adaptable y configurable, en correspondencia con el sistema a desarrollar.

Como política del CEIGE, en el Dpto. de Tecnología se utiliza el "*Modelo de Desarrollo Orientado a Componentes del proyecto ERP-Cuba*" (Producción), el cual tiene como características:

- ***Orientado a componentes:*** sistema compuesto por varios componentes desarrollados independientemente pero que al unirlos constituye dicho sistema.
- ***Iterativo e incremental:*** plantea que ese mismo sistema puede tener tantas iteraciones como sean necesarias y que en cada versión se obtendrá un incremento y mejoramiento de las funcionalidades del sistema.

1.7 Tecnologías y Herramientas para el desarrollo.

Como parte de la investigación se estudiaron las tecnologías y herramientas definidas por el CEIGE para el desarrollo de software de gestión. Estas son:

1.7.1 Servidor Web Apache 2.0.

El Servidor Apache HTTP es un servidor Web de tecnología Open Source que implementa el protocolo HTTP/1.1. Es un servidor altamente configurable de diseño modular. Presenta bases de datos LDAP para la autenticación y negociado de contenido. (Wiki)

1.7.2 Gestor de Base de Datos.

PostgreSQL 8.3.8.

PostgreSQL es un sistema de base de datos objeto-relacional de almacenamiento y manipulación de datos muy común. Entre sus características están: (PostgreSQL)

- Soporta distintos tipos de datos: datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP), cadenas de bits, entre otros. También permite la creación de tipos propios.
- Incorpora una estructura de datos *array*.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, entre otras.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.
- Incluye herencia entre tablas (aunque no entre objetos), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

MongoDB 1.2.2.

MongoDB es un sistema de base de datos multiplataforma orientado a documentos, de esquema libre, lo que significa que cada entrada o registro puede tener un esquema de datos diferente, con atributos o "columnas" que no tienen por qué repetirse de un registro a otro. Sus principales características son la escalabilidad, velocidad, alto rendimiento y funcionalidad y posee un sistema de consultas dinámicas y respuesta rápida. (Paramio, 2011)

1.7.3 Herramienta CASE.

Las herramientas CASE[8] (Ingeniería de Software Asistida por computadora) son aplicaciones que ayudan la labor de los analistas, ingenieros de software y desarrolladores durante la planeación, análisis, diseño e implementación de un software. (Introducción a herramientas CASE y System Architect)

Visual Paradigm 8.0.

Es una herramienta de diseño UML y una herramienta CASE UML diseñada para ayudar el desarrollo de software. Soporta los principales estándares de la industria, tales como: UML, SysML, BPMN, XMI, entre otros. Ofrece un conjunto de herramientas de los equipos de desarrollo de software, necesarios para la captura de requisitos, planificación del software, planificación de pruebas, elaborar modelos de clases y de datos, entre otros artefactos. (Visual-Paradigm)

1.8 Herramienta para el desarrollo colaborativo.

1.8.1 Tortoise SVN 1.7.0.

TortoiseSVN es un cliente gratuito de código abierto para el sistema de control de versiones Subversion. Maneja ficheros que se almacenan en un repositorio central que registra todos los cambios hechos a los ficheros y directorios. Permite recuperar versiones antiguas de los mismos y examinar las trazas de los datos, así como quién accedió a ellos. (Consultoría de áreas de conocimiento)

1.9 Framework de Desarrollo.

FrameWork se refiere a "*ambiente de trabajo, y ejecución*". En general, son soluciones completas que contemplan herramientas de apoyo a la construcción (ambiente de trabajo o desarrollo) y motores de ejecución (ambiente de ejecución). (2009)

Aquí se presentan los utilizados en el CEIGE, necesarios para el desarrollo de este trabajo.

[8] **CASE:** Computer Aided Software Engineering.

1.9.1 Zend Framework 1.7.

Zend Framework es una librería de código abierto de componentes escritos en PHP5, orientada a Objetos (OO), que facilita el desarrollo de sitios web. Zend Framework hace hincapié en la calidad del código**,** a través de una batería de test unitarios, utilizando PHPUnit, cubriendo alrededor del 85% del código escrito para el Framework. (2008)

1.9.2 Marco de trabajo Sauxe 1.5.

Sauxe es un Marco de Trabajo que contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. (Msc. Oiner Gómez Baryolo, 2008)

1.9.3 Doctrine 0.9.

Doctrine es un mapeador de objeto relacional (ORM[9]) para PHP 5.2.3+ que se encuentra en la parte superior de una capa poderosa de abstracción de base de datos (DBAL por sus siglas en inglés). Una de sus principales características es la opción de escribir las consultas de base de datos en un dialecto orientado a objetos de propiedad SQL llamado Doctrine QueryLanguage (DQL), lo que proporciona a los desarrolladores una alternativa a SQL, que mantiene la flexibilidad sin necesidad de duplicar el código innecesario. (Doctrine)

1.9.4 ExtJS 2.2.

ExtJS es una librería JavaScript ligera y de alto rendimiento, construida para el desarrollo veloz de aplicaciones Web, compatibles con la mayoría de navegadores. Utiliza técnicas como Ajax, DHTML y manipulación del DOM. ExtJS incluye un conjunto de controles para el desarrollo de interfaces en los desarrollos web. (ExtJS)

1.9.5 Acl.

Las ACL[10] o Listas de Control de Acceso determinan, mediante reglas, cuales usuarios tienen acceso a cierta parte de la información (Ibarra Naranjo, y otros). A través de la especificación y uso de ACL, una aplicación puede controlar cómo los objetos solicitantes (roles[11]) han obtenido acceso a objetos protegidos (recursos[12]). (2011)

Zend Framework cuenta con un componente Zend_Acl, el cual provee la implementación de un sistema simple y flexible de Listas de Control de Acceso para la administración de privilegios;

[9] **ORM:** Object Relation Mapper.
[10] **ACL** del inglés Access Control Lists.
[11] **Rol:** Objeto que puede solicitar acceso a un recurso.
[12] **Recurso:** Objeto al cual el acceso está controlado.

pero en el caso particular de la solución, se decidió implementar nuevas ACL para almacenar en ellas la relación existente entre: recursos compartimentados, roles, permisos y criterios y de esta relación obtener la autorización o negación de una determinada regla.

1.10 Lenguajes de Modelado y Desarrollo.

Los lenguajes de modelado y desarrollo son aquellos sistemas capaces de indicar distintas instrucciones que permiten modelar una solución a un problema, de forma tal que todo el que conozca el lenguaje pueda entender la solución. (Definicion.de) A continuación se enuncian los lenguajes de programación escogidos para el desarrollo del componente.

1.10.1 Unified Modeling Language (UML) 5.0.

UML[13] es un lenguaje que permite modelar, construir y documentar los elementos que forman un software orientado a objetos; es la sucesión de una serie de una serie de métodos de análisis y diseño. (Cornejo) Se ha convertido en el estándar de facto de la industria, ya que, a pesar de no tener un respaldo formal de una autoridad institucional o un organismo de normalización, es ampliamente usado.

1.10.2 PHP5.2.6.

PHP es un lenguaje de scripting de propósito general ampliamente utilizado que es especialmente adecuado para el desarrollo web y puede ser embebido en páginas HTML. La mayoría de su sintaxis es similar a C, Java y Perl. Su meta es permitir escribir a los creadores de páginas web, páginas dinámicas de una manera rápida y fácil. (2010 - 2011)

1.11 IDE de programación.

1.10.1 NetBeans 7.0.

El IDE NetBeans es un entorno de desarrollo integrado disponible para Windows, Mac, Linux y Solaris. Consiste en un IDE de código abierto y una plataforma de aplicaciones que permiten a los desarrolladores crear rápidamente aplicaciones web, empresariales, de escritorio y aplicaciones móviles utilizando la plataforma Java, así como PHP, JavaScript y Ajax, entre otros. (NetBeans.org)

CONCLUSIONES DEL CAPÍTULO.

Teniendo en cuenta los resultados obtenidos en el estudio de los sistemas de seguridad expuestos anteriormente, surge la necesidad de diseñar una solución que lleve a cabo la compartimentación de la información hasta nivel de usuario en ACAXIA. Para dicha solución será necesario hacer uso de listas de control de acceso y las mismas serán almacenadas en una

[13] **UML:** Lenguaje de Modelado Unificado.

base de datos no relacional, sobre el gestor MongoDB, atendiendo a los beneficios del mismo antes expuesto. Otras de las herramientas y tecnologías a utilizar en el desarrollo serán las definidas por el CEIGE para el desarrollo en el Departamento de Tecnología, ellas son Apache, PostgreSQL, Visual Paradigm y PHP y como framework de desarrollo el Marco de trabajo Sauxe, Doctrine, ExtJS y Zend Framework.

CAPÍTULO II: PROPUESTA DE SOLUCIÓN.

INTRODUCCIÓN.

En éste capítulo se realiza una propuesta de solución para el desarrollo del componente Compartimentación de la información. Dicha propuesta va a estar conformada por la identificación y descripción de los requisitos funcionales y no funcionales y los artefactos generados durante las fases de Análisis y Diseño del componente.

DESARROLLO.

2.1 Propuesta de Solución

Con el componente *Compartimentación de la información* se pretende controlar de forma más estricta el acceso a la información que manejan los sistemas suscritos a ACAXIA. Teniendo en cuenta la Multientidad que gestiona el sistema se decide establecer reglas a los permisos que tienen asignados usuarios con un mismo rol en una entidad determinada. Estas reglas evitarían que un usuario manipule información perteneciente a otro usuario con un mismo rol en una misma entidad.

2.1.1 Modelo Conceptual.

Los modelos conceptuales se construyen a partir de las definiciones fundamentales que se van a tratar a lo largo de la investigación. En ellos se representa cómo se relacionan dichos conceptos entre sí para hacer más fácil su entendimiento y contribuir con esto a la solución del problema. (ongei.gob.pe)

A continuación se definen algunos conceptos a tratar. Los demás fueron tomados del Modelo Conceptual (Pérez) del proyecto ACAXIA:

Recurso: cualquier elemento que se maneje a nivel de negocio y que esté representado en una tabla de la BD.

Criterio: atributo de los recursos que representan columnas en la tabla del recurso.

Permiso: tipo de autorización que puede tener un rol o usuario sobre determinado recurso.

Regla: especificación que se le agrega a los permisos.

Usuario: cualquier persona que interactúa con el sistema y juega un rol determinado en el mismo. (Pérez)

Rol: agrupa una serie de permisos sobre sistemas, funcionalidades y acciones que se le asignarán a un conjunto de usuarios. (Pérez)

Entidad: Forma parte de la estructura de un país y puede comportarse como un ministerio, entidad, área o cargo. (Pérez)

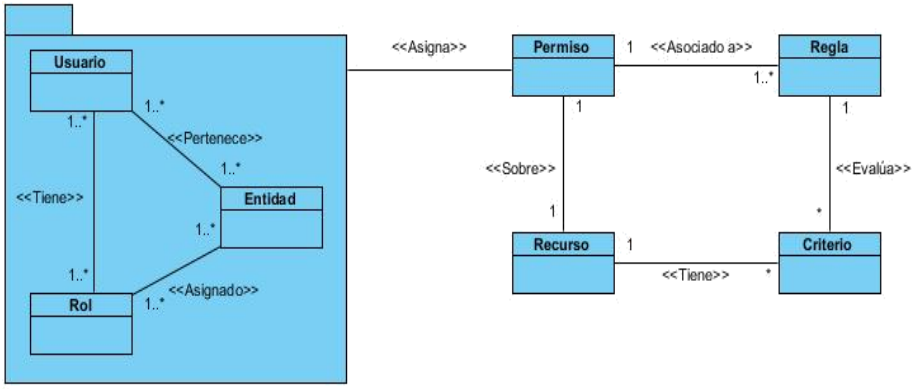

Figura 3. Modelo Conceptual

2.2 Requisitos de software.

En la 7ma edición del libro Software Engineering, Ian Sommerville clasifica los Requisitos Funcionales (RF) en 2 grupos fundamentales (Sommerville, 2005):

- *Requisitos de usuarios*: son declaraciones en un lenguaje natural y diagramas de los servicios que se espera proporcione el sistema y las limitaciones sobre las que debe operar.
- *Requisitos de sistema*: describe detalladamente las funciones, servicios y limitaciones operacionales del sistema.

2.2.1 Requisitos Funcionales del sistema.

Siguiendo la clasificación descrita anteriormente por Sommerville, los requisitos funcionales del sistema son:

RF1: Gestionar permisos.

RF 1.1: Listar usuarios.

Tabla 1: 1RF Listar usuarios.

Precondiciones	Deben existir usuarios previamente registrados en el sistema.
Flujo de eventos	
Flujo básico Listar roles	
1	Se selecciona en el menú la opción Compartimentar información
2	El sistema muestra una interfaz subdividida en varios grid en el primero de los cuales se visualiza un listado con los usuarios registrados en el sistema.
Pos-condiciones	
	N/A
Flujos alternativos	
N/A	
Pos-condiciones	

N/A		
Validaciones		
N/A		
Conceptos	**Usuarios**	Visibles en la interfaz: Denominación. Utilizados internamente: N/A.
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Prototipo elemental de interfaz gráfica de usuario.

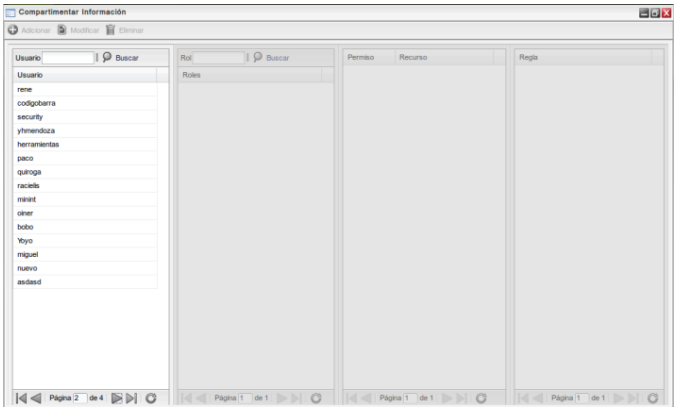

Figura 4: Interfaz del 1RF Listar usuarios.

RF 1.2: Listar roles.

Tabla 2: 2RF Listar roles.

Precondiciones	Deben existir usuarios y roles previamente registrados en el sistema.
Flujo de eventos	
Flujo básico Listar usuarios	
1	Se selecciona en el menú la opción Compartimentar información
2	El sistema muestra una interfaz subdividida en varios grid en el primero de los cuales se visualiza un listado con los usuarios registrados en el sistema.
3	Se selecciona un usuario.
4	El sistema habilita el 2do grid y muestra los roles asociados al usuario seleccionado.
Pos-condiciones	
1	N/A
Flujos alternativos	
N/A	
Pos-condiciones	
N/A	
Validaciones	
N/A	

Conceptos	**Usuarios**	Visibles en la interfaz: Denominación. Utilizados internamente: N/A.
	Roles	Visibles en la interfaz: Denominación. Utilizados internamente: N/A.
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Prototipo elemental de interfaz gráfica de usuario.

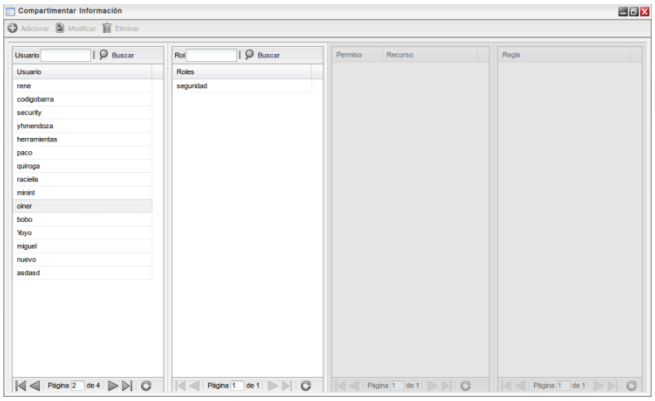

Figura 5: Interfaz del 2RF Listar roles.

RF 1.3: Listar entidades.

Tabla 3: 3RF Listar entidades.

Precondiciones	Deben existir usuarios, roles y entidades previamente registradas en el sistema.
Flujo de eventos	
Flujo básico Listar usuarios	
1	Se selecciona en el menú la opción Compartimentar información
2	El sistema muestra una interfaz subdividida en varios grid en el primero de los cuales se visualiza un listado con los usuarios registrados en el sistema.
3	Se selecciona un usuario.
4	El sistema habilita el 2do grid y muestra los roles asociados al usuario seleccionado.
5	Se selecciona un rol.
6	El sistema muestra una nueva ventana con el listado de entidades en las que dicho rol tiene permisos.
Pos-condiciones	
1	N/A
Flujos alternativos	
N/A	

Pos-condiciones		
N/A		
Validaciones		
N/A		
Conceptos	**Usuarios**	Visibles en la interfaz: Denominación. Utilizados internamente: N/A.
	Roles	Visibles en la interfaz: Denominación. Utilizados internamente: N/A.
	Entidades	Visibles en la interfaz: Denominación. Utilizados internamente: N/A.
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Prototipo elemental de interfaz gráfica de usuario.

Figura 6: Interfaz del 4RF Listar entidades.

RF 1.4: Listar permisos.

Tabla 4: 4RF Listar permisos.

Precondiciones	Deben existir usuarios, roles y entidades previamente registradas en el sistema. Deben habérsele otorgado permisos sobre algún recurso al rol escogido, perteneciente al usuario determinado. Debe haberse definido en un XML los recursos que son compartimentables, los criterios tanto generales como específicos de los mismos.
Flujo de eventos	
Flujo básico Listar recursos	
1	Se selecciona en el menú la opción Compartimentar información
2	El sistema muestra una interfaz subdividida en varios grid en el primero de los

	cuales se visualiza un listado con los usuarios registrados en el sistema.	
3	Se selecciona un usuario.	
4	El sistema habilita el 2do grid y muestra los roles asociados al usuario seleccionado.	
5	Se selecciona un rol.	
6	El sistema muestra una nueva ventana con el listado de entidades en las que dicho rol tiene permisos.	
7	Se selecciona una entidad.	
8	El sistema habilita un 3er grid mostrando en el mismo el listado de permisos que tiene el usuario sobre los recursos de la entidad seleccionada atendiendo al rol que se escogió previamente.	
Pos-condiciones		
1	N/A	
Flujos alternativos		
N/A		
Pos-condiciones		
1	N/A	
Validaciones		
	N/A	
Conceptos	**Usuarios**	Visibles en la interfaz: Denominación. Utilizados internamente: N/A.
	Roles	Visibles en la interfaz: Denominación. Utilizados internamente: N/A.
	Entidades	Visibles en la interfaz: Denominación. Utilizados internamente: N/A.
Conceptos	**Permiso**	Visibles en la interfaz: Denominación. Permiso. Recurso. Regla. Utilizados internamente: N/A.
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Prototipo elemental de interfaz gráfica de usuario.

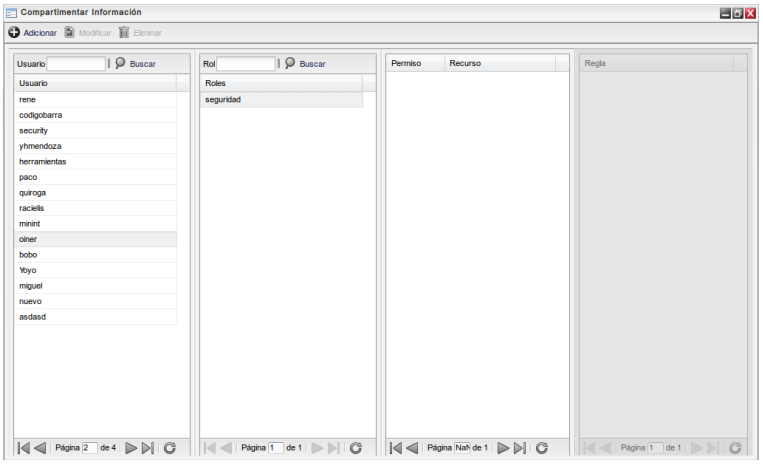

Figura 7: Interfaz del 5RF Listar permisos.

RF 1.5: Listar reglas.

Tabla 5: 5RF Listar reglas.

Precondiciones	Deben existir usuarios, roles y entidades previamente registradas en el sistema. Deben habérsele otorgado permisos sobre algún recurso al rol escogido, perteneciente al usuario determinado. Debe haberse definido en un XML los recursos que son compartimentables, los criterios tanto generales como específicos de los mismos.
Flujo de eventos	
Flujo básico Listar recursos	
1	Se selecciona en el menú la opción Compartimentar información
2	El sistema muestra una interfaz subdividida en varios grid en el primero de los cuales se visualiza un listado con los usuarios registrados en el sistema.
3	Se selecciona un usuario.
4	El sistema habilita el 2do grid y muestra los roles asociados al usuario seleccionado.
5	Se selecciona un rol.
6	El sistema muestra una nueva ventana con el listado de entidades en las que dicho rol tiene permisos.
7	Se selecciona una entidad.
8	El sistema habilita un 3er grid mostrando en el mismo el listado de permisos que tiene el usuario sobre los recursos de la entidad seleccionada atendiendo al rol que se escogió previamente.
9	Se selecciona un permiso.
10	El sistema habilita el 4to y último grid, donde se muestra un listado con las reglas asociadas al permiso seleccionado
Pos-condiciones	
1	N/A
Flujos alternativos	
N/A	

Pos-condiciones		
2 N/A		
Validaciones		
N/A		
Conceptos	**Usuarios**	Visibles en la interfaz: Denominación. Utilizados internamente: N/A.
	Roles	Visibles en la interfaz: Denominación. Utilizados internamente: N/A.
	Entidades	Visibles en la interfaz: Denominación. Utilizados internamente: N/A.
Conceptos	**Permiso**	Visibles en la interfaz: Permiso. Recurso. Utilizados internamente: N/A.
	Regla	Visibles en la interfaz: Regla. Utilizados internamente: N/A.
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Prototipo elemental de interfaz gráfica de usuarios.

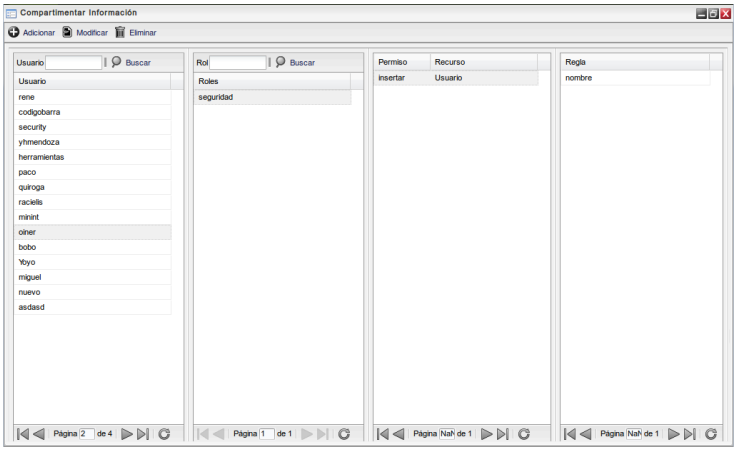

Figura 8: Interfaz del 5RF Listar reglas.

RF 1.6: Buscar usuario.

Tabla 6: 6RF Buscar usuario.

Precondiciones	Deben existir usuarios previamente registrados en el sistema. Debe haberse definido en un XML los recursos que son compartimentables, los criterios tanto generales como específicos de los mismos.	
Flujo de eventos		
Flujo básico Buscar roles		
1	Se selecciona en el menú la opción Compartimentar información.	
2	El sistema muestra una interfaz subdividida en varios grid en el primero de los cuales se visualiza un listado con todos los usuarios registrados en el sistema.	
3	En el campo de texto “Usuario” se escribe el nombre del usuario que se desea buscar.	
4	Se presiona el botón “Buscar”.	
5	El sistema muestra los usuarios que cumplen con el criterio de búsqueda especificado.	
Pos-condiciones		
1	N/A	
Flujos alternativos		
Flujo alternativo 3.a En el campo de texto “Usuario” se escriben caracteres inválidos.		
4	El sistema no permite insertar caracteres inválidos, dejando el textbox en blanco.	
Flujo alternativo 3.b En el campo de texto “Usuario” se escriben solo números.		
4	El sistema marca el textbox en rojo, informando que “Este valor es incorrecto.”	
Flujo alternativo 4.a Se presiona el botón “Buscar” sin haber insertado el nombre del usuario.		
5	El sistema muestra todos los usuarios registrados en el sistema.	
Flujo alternativo 5.a No existe el usuario especificado.		
5	El sistema muestra el grid de usuarios en blanco.	
Pos-condiciones		
1	N/A	
Validaciones		
1	N/A	
Conceptos	**Usuarios**	Visibles en la interfaz: Denominación Utilizados internamente: N/A
Requisitos especiales	N/A	
Asuntos pendientes	N/A	
Prototipo elemental de interfaz gráfica de usuario.		

Figura 9: Interfaz del 6RF Buscar usuario.

RF 1.7: Buscar rol.

Tabla 7: 7RF Buscar rol.

Precondiciones		Deben existir usuarios y roles previamente registrados en el sistema. Debe haberse definido en un XML los recursos que son compartimentables, los criterios tanto generales como específicos de los mismos.
Flujo de eventos		
Flujo básico Buscar usuarios		
1	Se selecciona en el menú la opción Compartimentar información.	
2	El sistema muestra una interfaz subdividida en varios grid en el primero de los cuales se visualiza un listado con los usuarios registrados en el sistema.	
3	Se selecciona un usuario.	
4	El sistema habilita el 2do grid y muestra los roles asociados al usuario seleccionado.	
5	En el campo de texto “Rol” se escribe el nombre del rol que se desea buscar.	
6	Se presiona el botón “Buscar”.	
7	El sistema muestra los roles que cumplen con el criterio de búsqueda especificado	
Pos-condiciones		
1	N/A	
Flujos alternativos		
Flujo alternativo 5.a En el campo de texto “Rol” se insertan caracteres inválidos.		
6	El sistema no permite insertar dichos caracteres y deja el textbox en blanco.	
Flujo alternativo 5.b En el campo de texto “Rol” se insertan solo números.		
6	El sistema marca el textbox en rojo e informa “Este valor es incorrecto”.	
Flujo alternativo 7.a No existe el rol especificado para el usuario seleccionado.		
8	El sistema muestra el grid de roles en blanco.	
Pos-condiciones		
1	N/A	
Validaciones		
N/A		
Conceptos	**Usuarios**	Visibles en la interfaz: Denominación Utilizados internamente: N/A
	Roles	Visibles en la interfaz: Denominación Utilizados internamente: N/A
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Prototipo elemental de interfaz gráfica de usuario.

Figura 10: Interfaz del 7RF Buscar rol.

RF 1.8: Adicionar permiso.

Tabla 8: 8RF Adicionar permiso.

Precondiciones	Deben existir usuarios, roles y entidades previamente registradas en el sistema. Debe haberse definido en un XML los recursos que son compartimentables, los criterios tanto generales como específicos de los mismos.
Flujo de eventos	
Flujo básico Adicionar permisos	
1	Se selecciona en el menú la opción Compartimentar información.
2	El sistema muestra una interfaz subdividida en varios grid en el primero de los cuales se visualiza un listado con los usuarios registrados en el sistema.
3	Se selecciona un usuario.
4	El sistema habilita el 2do grid y muestra los roles asociados al usuario seleccionado.
5	Se selecciona un rol.
6	El sistema muestra una nueva ventana con el listado de entidades sobre las cuales tiene permiso dicho rol.
7	Se selecciona una entidad.
8	El sistema habilita el 3er grid mostrando en el mismo el listado de permisos que tiene el usuario sobre los recursos de la entidad seleccionada atendiendo al rol que se escogió previamente.
9	Se presiona el botón “*Adicionar*”.
10	El sistema muestra la ventana “*Nuevo permiso*”.
11	Se selecciona el tipo de permiso y recurso que se desean otorgar al permiso.
12	Se presiona el botón “Agregar” para asociarle una o más reglas al permiso.
13	El sistema muestra la ventana “*Nueva regla*”.
14	Se registran los datos necesarios para crear una regla.
15	Se presiona el botón “*Aceptar*”.
16	El sistema valida los datos registrados y muestra el mensaje “Regla adicionada satisfactoriamente”.
17	Se muestra en la interfaz “*Nuevo permiso*” la/las reglas adicionadas.
18	Se presiona el botón “Aceptar”.
19	El sistema valida los datos insertados.
20	El sistema muestra un mensaje confirmando el éxito de la operación: “*Permiso adicionado satisfactoriamente*”.
Pos-condiciones	
	Se adiciona un permiso al rol escogido en la entidad seleccionada.
Flujos alternativos	
Flujo alternativo 12.a Se presiona el botón “Agregar” para asociar una regla sin haber llenado los campos o alguno de los campos relacionados con el permiso.	
13	Se muestra un mensaje informando “Existen campos vacíos”.
	Volver al paso 11 del flujo básico.
Flujo alternativo 18.a Se presiona el botón “Aceptar” sin haber asociado una regla.	
19	Se muestra un mensaje informando “Debe adicionar al menos una regla”.
	Volver al paso 12 del flujo básico.

Flujo alternativo *.a Se cancela la acción.		
1 Concluye el requisito.		
Pos-condiciones		
1 No se registran los datos.		
Validaciones		
Se validan los datos según lo establecido en el Modelo Conceptual CIG-SEG-N-i1301.		
Conceptos	**Permiso**	Visibles en la interfaz: Permiso. Utilizados internamente: N/A.
	Recurso	Visibles en la interfaz: Recurso. Utilizados internamente: N/A.
	Regla	Visibles en la interfaz: Criterio. Comparación. Valor. Utilizados internamente: N/A.
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Prototipo elemental de interfaz gráfica de usuario.

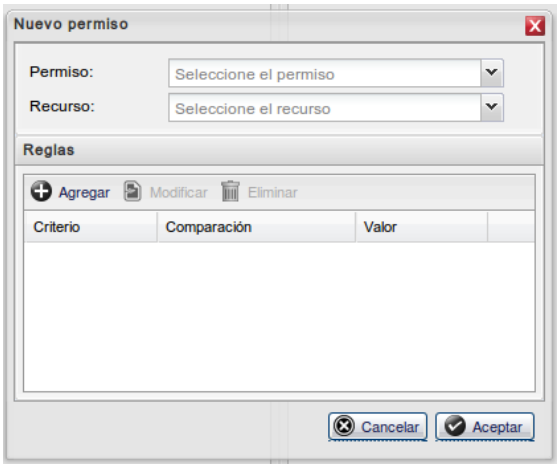

Figura 11: Interfaz del 8RF Adicionar permiso.

RF 1.9: Modificar permiso.

Tabla 9: 9RF Modificar permiso.

Precondiciones	Deben existir usuarios, roles y entidades previamente registradas en el sistema. Deben habérsele otorgado permisos sobre algún recurso al rol perteneciente al usuario seleccionado. Debe haberse definido en un XML los recursos que son compartimentables, los criterios tanto generales como

	específicos de los mismos.
Flujo de eventos	
Flujo básico Modificar permisos	
1	Del listado de permisos que tiene el usuario sobre los recursos de la entidad seleccionada atendiendo al rol que se escogió previamente, se selecciona el permiso que se desea modificar.
2	Se selecciona el botón "*Modificar*".
3	Se muestra la interfaz "*Modificar permiso*".
4	Se modifican los valores deseados (Permiso, Recurso).
5	El sistema elimina automáticamente la/las reglas definidas para el permiso anterior (de existir alguna regla asociada al nuevo permiso, se muestra en el listado de reglas). De ser necesario, se asocia una nueva regla al permiso.
	Ver descripción de requisito *incluido Gestionar reglas*.
6	Se presiona el botón "*Aceptar*".
7	El sistema valida los datos modificados.
8	El sistema muestra un mensaje confirmando el éxito de la operación: "*Permiso modificado satisfactoriamente*".
Pos-condiciones	
1	Se modifica el permiso seleccionado.
Flujos alternativos	
Flujo alternativo 6.a Se presiona el botón "Aceptar" sin asociar una regla.	
7	El sistema muestra el mensaje de error "Debe adicionar al menos una regla".
	Volver al paso 5 del Flujo Básico.
Flujo alternativo *.a Se cancela la acción.	
1	Concluye el requisito.
Pos-condiciones	
1	No se modifican los datos.
Validaciones	
	Se validan los datos según lo establecido en el Modelo Conceptual CIG-SEG-N-i1301.
Conceptos	**Permiso**: Visibles en la interfaz: Permiso. Utilizados internamente: N/A.
	Recurso: Visibles en la interfaz: Recurso. Utilizados internamente: N/A.
	Regla: Visibles en la interfaz: Criterio. Comparación. Valor. Utilizados internamente: N/A.

Requisitos especiales	N/A
Asuntos pendientes	N/A

Prototipo elemental de interfaz gráfica de usuario.

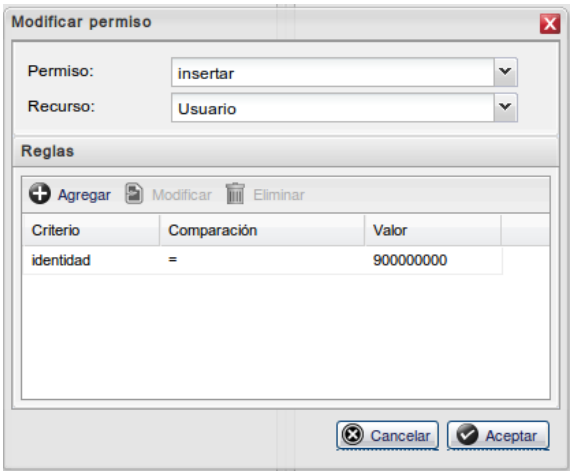

Figura 12: Interfaz del 9RF Modificar permiso.

RF 1.10: Eliminar permiso.

Tabla 10: 10RF Eliminar permiso.

Precondiciones	Deben existir usuarios, roles y entidades previamente registradas en el sistema. Deben habérsele otorgado permisos sobre algún recurso al rol perteneciente al usuario seleccionado. Debe haberse definido en un XML los recursos que son compartimentables, los criterios tanto generales como específicos de los mismos.
Flujo de eventos	
Flujo básico Eliminar permisos	
1	Del listado de permisos sobre recursos asociados a un rol perteneciente al usuario seleccionado, se selecciona el que se desea eliminar.
2	Se presiona el botón "*Eliminar*"
3	Se muestra una ventana solicitando confirmación de la acción
4	Se presiona el botón "Si"
Pos-condiciones	
1	Se elimina el permiso seleccionado.
Flujos alternativos	
Flujo alternativo *.a Se cancela la operación	
1	Concluye el requisito.
Pos-condiciones	
3	No es eliminado el permiso.
Validaciones	
	N/A

Conceptos	Visibles en la interfaz: N/A. Utilizados internamente: N/A.
Requisitos especiales	N/A
Asuntos pendientes	N/A

Prototipo elemental de interfaz gráfica de usuario.

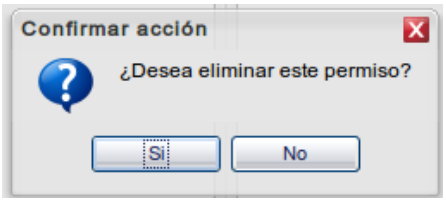

Figura 13: Interfaz del 10RF Eliminar permiso.

RF 2: Gestionar reglas.

RF 2.1: Listar reglas.

Tabla 11: 11RF Listar reglas.

Precondiciones	Deben existir usuarios, roles y entidades previamente registradas en el sistema. Deben existir permisos sobre algún recurso al rol perteneciente al usuario seleccionado. Debe haberse definido en un XML los recursos que son compartimentables, los criterios tanto generales como específicos de los mismos.
Flujo de eventos	
Flujo básico Listar reglas	
1	Se selecciona un usuario, un rol correspondiente a dicho usuario y una entidad perteneciente a dicho rol y el sistema muestra la lista de permisos otorgados al mismo.
2	Se selecciona un permiso.
3	Se presiona el botón "*Modificar*"
4	El sistema muestra la ventana "*Modificar permiso*"
5	Dentro de dicha ventana, se muestra un listado con las reglas asociadas al permiso en cuestión.
Pos-condiciones	
2	N/A
Flujos alternativos	
N/A	
Validaciones	
2	N/A

Conceptos	**Regla**	Visibles en la interfaz: Criterio. Comparación. Valor. Utilizados internamente: N/A.
	Permiso	Visibles en la interfaz: Permiso. Utilizados internamente: N/A.
	Recurso	Visibles en la interfaz: Recurso. Utilizados internamente: N/A.
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Prototipo elemental de interfaz gráfica de usuario.

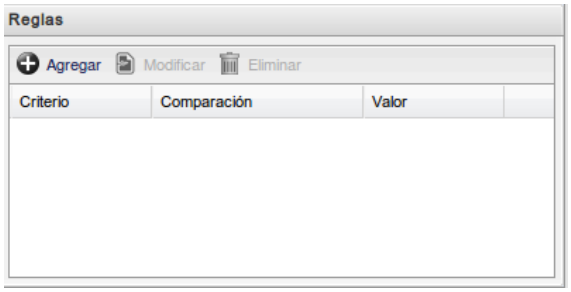

Figura 14: Interfaz del 11RF Listar reglas.

RF 2.2: Adicionar regla.

Tabla 12: 12RF Adicionar regla.

Precondiciones	Deben existir usuarios, roles y entidades previamente registradas en el sistema. Deben existir permisos sobre algún recurso al rol perteneciente al usuario seleccionado. Debe haberse definido en un XML los recursos que son compartimentables, los criterios tanto generales como específicos de los mismos.
Flujo de eventos	
Flujo básico Adicionar regla	
1	Se presiona el botón "*Adicionar*" o "*Modificar*" permiso.
2	El sistema muestra la ventana correspondiente.
3	Se selecciona el botón "*Agregar*" regla.
4	El sistema muestra la ventana "*Nueva regla*"
5	Se especifica el nombre de la regla en el campo "*Regla*".

6	Se selecciona un criterio
7	Se escoge el valor de comparación
8	Una vez especificado el campo Criterio, se habilita un campo valor que se carga dinámicamente, en dependencia del tipo de datos del criterio especificado. El mismo puede tomar los valores: Fecha, Cadena, Número y Valor. Se escoge un valor.
9	Se presiona el botón “Aceptar”.
10	El sistema valida los datos registrados.
11	El sistema muestra el mensaje “Regla adicionada satisfactoriamente”.
Pos-condiciones	
1	Se adiciona una nueva regla para restringir el acceso a un determinado recurso.
Flujos alternativos	
Flujo alternativo 5.a En el nombre de la regla se insertan caracteres inválidos.	
6	El sistema no permite registrar dichos caracteres y deja el textbox en blanco.
Flujo alternativo 9.a Se adiciona una regla dejando campos vacíos.	
10	Se muestra un mensaje informando que “Existen campos vacíos”
	Volver al paso 5 del flujo básico.
Flujo alternativo 10.a Se adiciona una regla que ya existe.	
11	Se muestra un mensaje informando que “Existe una regla con ese nombre asociada a ese recurso”.
	Volver al paso 5 del flujo básico.
Flujo alternativo 10.b Se adiciona una regla sobre un criterio que ya existe.	
11	Se muestra un mensaje de error “existe una regla asociada al criterio seleccionado.”.
	Volver al paso 6 del Flujo básico.
Flujo alternativo *.a Se cancela la operación	
1	Concluye el requisito.
Pos-condiciones	
1	No se crea una nueva regla.
Validaciones	
1	Se validan los datos según lo establecido en el Modelo conceptual CIG-SEG-N-i1301.
Conceptos	**Regla** Visibles en la interfaz: Regla Criterio Comparación Valor o Cadena Utilizados internamente: N/A
Requisitos especiales	N/A
Asuntos pendientes	N/A

Prototipo elemental de interfaz gráfica de usuario.

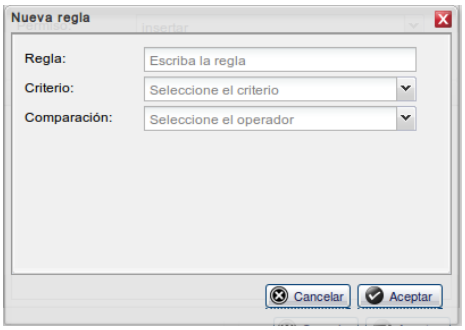

Figura 15: Interfaz del 12RF Adicionar regla.

RF 2.3: Modificar regla.

Tabla 13: 13RF Modificar regla.

Precondiciones	Deben existir usuarios, roles y entidades previamente registradas en el sistema. Deben existir permisos sobre algún recurso al rol perteneciente al usuario seleccionado. Debe haberse definido en un XML los recursos que son compartimentables, los criterios tanto generales como específicos de los mismos.
Flujo de eventos	
Flujo básico Adicionar regla	
1	Se presiona el botón “*Modificar*” o “Adicionar” permiso.
2	El sistema muestra la ventana correspondiente.
3	Del listado de reglas asociadas al permiso se escoge la que se desea modificar.
3	Se presiona el botón “*Modificar*” regla.
4	El sistema muestra la ventana “*Modificar regla*”.
5	Se modifican los valores deseados para la regla escogida.
6	Se presiona el botón “Aceptar”.
7	El sistema valida los datos introducidos y muestra el mensaje “Regla modificada satisfactoriamente”.
8	Se modifican los datos de la regla asociada al permiso en cuestión.
Pos-condiciones	
1	Se modifican los valores de la regla seleccionada.
Flujos alternativos	
Flujo alternativo 5.a Modificar nombre de la regla, introduciendo caracteres inválidos.	
6	El sistema no permite escribir dichos caracteres, dejando el textbox en blanco.
Flujo alternativo 7.a Se modifica una regla dejando campos vacíos.	
8	Se muestra un mensaje informando que “Existen campos vacío”.
	Volver al paso 5 del flujo básico.
Flujo alternativo 7.b Modificar una regla sobre un criterio que ya existe.	
8	El sistema muestra el mensaje “Existe una regla asociada al criterio seleccionado”.

	Volver al paso 5 del flujo básico.	
Flujo alternativo *.a Se cancela la operación		
1	Concluye el requisito.	
Pos-condiciones		
1	No se modifica la regla.	
Validaciones		
1	Se validan los datos según lo establecido en el Modelo conceptual CIG-SEG-N-i1301.	
Conceptos	**Regla**	Visibles en la interfaz: Regla Criterio Comparación Valor o Cadena Utilizados internamente: N/A
	Permiso	Visibles en la interfaz: Permiso Recurso Utilizados internamente: N/A
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Prototipo elemental de interfaz gráfica de usuario.

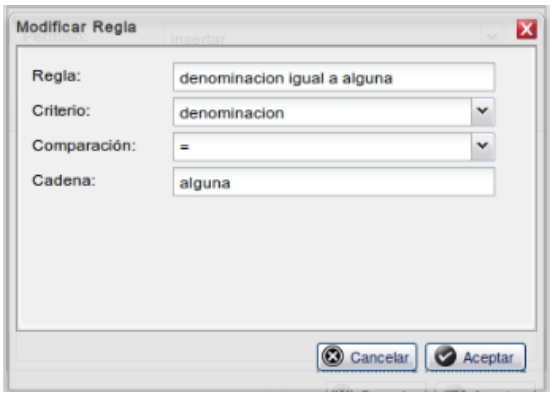

Figura 16: Interfaz del 13RF Modificar regla.

RF 2.4: Eliminar regla.

Tabla 14: 14RF Eliminar regla.

Precondiciones	Deben existir usuarios, roles y entidades previamente registradas en el sistema. Deben existir permisos sobre algún recurso al rol perteneciente al usuario seleccionado. Debe haberse definido en un XML los recursos que son

		compartimentables, los criterios tanto generales como específicos de los mismos.
Flujo de eventos		
Flujo básico Adicionar regla		
1	Se presiona el botón “*Modificar*” permiso.	
2	El sistema muestra la ventana correspondiente.	
3	Del listado de reglas asociadas al permiso se escoge la que se desea eliminar.	
4	Se selecciona el botón “*Eliminar*”.	
5	El sistema muestra una ventana solicitando confirmación.	
6	Se presiona el botón “Si”.	
Pos-condiciones		
1	Es eliminada la regla previamente seleccionada.	
Flujos alternativos		
Flujo alternativo *.a Se cancela la operación		
1	Concluye el requisito.	
Pos-condiciones		
1	No se elimina la regla.	
Validaciones		
N/A		
Conceptos	**Regla**	Visibles en la interfaz: Regla. Criterio. Comparación. Valor o Cadena. Utilizados internamente: N/A.
	Permiso	Visibles en la interfaz: Permiso. Recurso. Utilizados internamente: N/A.
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Prototipo elemental de interfaz gráfica de usuario.

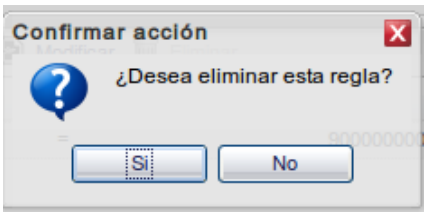

Figura 17: Interfaz del 14RF Eliminar regla.

RF 3: Especificar recurso compartimentable.

Tabla 15: 15RF Especificar recurso compartimentable.

Precondiciones	N/A	
Flujo de eventos		
Flujo básico Listar reglas		
Se almacenará en un fichero XML los recursos a compartimentar y los criterios generales y específicos de cada uno, criterios por los que se podrán crear reglas para un mayor grado de compartimentación.		
1	Se especifica el recurso compartimentable.	
2	Para cada recurso se definen los criterios específicos por los que se podrá compartimentar el mismo	
3	Concluye el requisito.	
Pos-condiciones		
3	N/A	
Flujos alternativos		
N/A		
Pos-condiciones		
4	N/A	
Validaciones		
3	N/A	
Conceptos	**Reglas**	Visibles en la interfaz: Comparación. Valor. Utilizados internamente: N/A.
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

2.2.2 *Requisitos No Funcionales.*

Durante el proceso de captura de requisitos no se identificaron nuevos Requisitos No Funcionales (RnF) y se determinó que para la construcción del componente se tendrían en cuenta los mismos RnF detectados para ACAXIA y descritos en el documento "*Especificación de Requisitos de Software*", el cual se puede encontrar en el expediente de documentación del sistema. (Pérez, 2011)

Usabilidad.

RnF 1: El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora.

Eficiencia.

RnF 2: Los tiempos de respuesta y velocidad de procesamiento de la información serán rápidos, no mayores de 5 segundos para las actualizaciones y 20 para las recuperaciones.

Soporte.

RnF 3: La aplicación contará antes de su puesta en marcha con un período de pruebas, se le dará mantenimiento, configuración y se brindará el servicio de instalación.

RnF 4: Los desarrolladores deben utilizar para el desarrollo de la solución el estándar de codificación elaborado en el Departamento de Tecnología.

Restricciones de diseño.

RnF 5: El lenguaje de programación a utilizar para desarrollar el sistema debe ser PHP para la lógica del negocio y ExtJS para la capa de presentación.

RnF 6: Las herramientas a utilizar para desarrollar el sistema deben ser PostgreSQL 8.3 como gestor de base de datos y Pgadmin III como cliente, Doctrine para la capa de acceso a datos y ZendExt Framework para la lógica del negocio.

RnF 7: El sistema debe ser multiplataforma, haciendo énfasis en Linux y Windows.

Requisitos Legales, de Derecho de Autor y otros.

RnF 8: El sistema está avalado por los tres documentos rectores emitidos en el país para la certificación y validación de los sistemas contables:

- La Resolución Orden #4 del Ministro de las Fuerzas Armadas Revolucionarias.

Estándares Aplicables.

RnF 9: El sistema debe utilizar el estándar SAML para la estandarización del proceso de autenticación.

RnF 10: El sistema debe utilizar el estándar RBAC para la estandarización del proceso de autorización.

Políticos Culturales (CUL).

RnF 11: El sistema solo podrá ser utilizado en territorio cubano y por las entidades autorizadas por el Ministerio de las FAR.

RnF 12: El producto no debe contener palabras en otros idiomas.

RnF 13: El producto debe respetar los términos empleados normalmente por los especialistas en el tema de la esfera que se automatiza.

Confiabilidad.

RnF 14: El sistema debe utilizar contraseña de acceso para llevar a cabo la autenticación del usuario

RnF 15: El sistema debe garantizar protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.

RnF 16: La atención al sistema incluyendo, el mantenimiento de las bases de datos así como la salva de la información se realizarán de forma centralizada por el administrador.

RnF 17: El sistema debe realizar verificaciones sobre las acciones irreversibles (eliminaciones).

Ambiente.

Para que el sistema funcione es necesario garantizar los siguientes requisitos de software:

Para el cliente:

- RnF 2: Navegador Mozilla Firefox.
- RnF 3: Sistema operativo Windows 98 o superior o Linux.

Para el servidor:

- RnF 4: Sistema operativo Windows Advancer Server (2000 o superior) o Linux en cualquiera de sus distribuciones.
- RnF 5: Un servidor Apache 2.0 o superior con módulo PHP 5.0 disponible, este debe estar configurado con la extensión "pgsql" incluida.
- RnF 6: Un servidor de base de datos PostgreSQL 8.0 o superior.
- Para que el sistema funcione es necesario garantizar los siguientes requisitos de hardware:

  Para el servidor:
- RnF 7: Requerimientos mínimos: Procesador Pentium III a 1GHz de velocidad de procesamiento y 1Gb de memoria RAM.
- RnF 8: Al menos 40Gb de espacio libre en disco duro.
- RnF 9: Tarjeta de red.

  Para el cliente:
- RnF 10: Requerimientos mínimos: Procesador Pentium II a 133Mhz con 128 Mb de memoria RAM.
- RnF 11: Tarjeta de red.

2.3 Modelo de Diseño.

2.3.1 Estilos arquitectónicos.

El estilo arquitectónico definido por el Departamento de Tecnología (Mena López, y otros, 2008), donde se desarrolla el sistema ACAXIA es el **Estilo N capas**. Donde se identificaron 3 niveles o capas fundamentales, como se describe a continuación y se muestra en la **Figura 18**:

- **Capa Datos:** donde estarán ubicados los servidores de base de datos PostgreSQL y MongoDB y además, un conjunto de Ficheros de Configuración del sistema.
- **Capa de Acceso a Datos:** donde estará presente el framework Doctrine, como persistidor para la comunicación con el servidor de datos mediante el protocolo PDO[14], también estará un Persistidor de Configuración que es el encargado de comunicarse vía XML con los

[14]**PDO:** Process Data Objects. Protocolo de comunicación especialmente adecuado para la transmisión rápida de datos; éste modelo define un productor y uno o varios consumidores PDO.

Ficheros de Configuración del sistema, y el último elemento de esta capa es la Fachada de Transferencia de Acceso Rápido cuya función es el acceso rápido tanto a los ficheros de comunicación como al servidor de datos, a través del Persistidor de Configuración y Doctrine para cada caso en específico.

- **Capa de Funcionamiento:** en esta capa se emplea el patrón de arquitectura Modelo Vista Controlador (MVC), así como el elemento de Acceso de Respuesta Rápida el cual brinda informaciones de forma vertiginosa violando las restricciones que impone el estilo en capas propuesto. De forma vertical al modelo descrito hasta este momento, estará el módulo de cacheo del sistema así como el encargado del tratamiento de la seguridad a nivel de aplicación Web.

Esta gran estructura descrita, se comunica con una series de servicios Web mediante protocolos SOAP[15], REST[16], HTTP[17] y HTTPS[18], que interactúan con el sistema proporcionándole un conjunto de funcionalidades tanto de seguridad como de negocio. (Mena López, y otros, 2008)

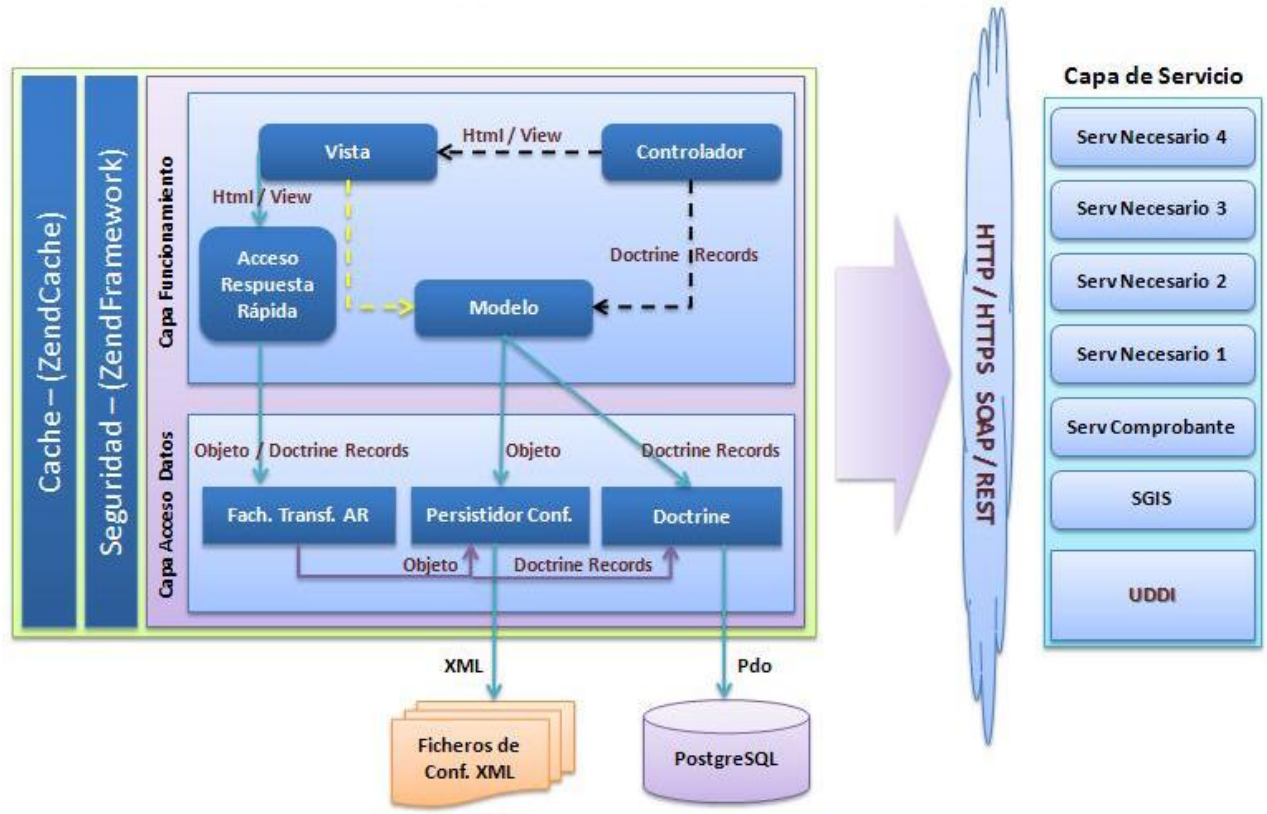

Figura 18: Gráfico estructural. Arquitectura N Capas.

[15] **SOAP:** Simple Object Access Protocol. Permite la comunicación entre aplicaciones a través de mensajes por medio de Internet.

[16] **REST:** Representational State Transfer o Transferencia Representacional de Estados. Sirve para acceder a servicios sencillamente a través de URLs.

[17] **HTTP:** Hypertext Transfer Protocol o Protocolo de Transferencia de Hipertexto. Protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP.

[18] **HTTPS:** Versión segura del protocolo HTTP que implementa un canal de comunicación seguro y basado en SSL (Secure Socket Layers) entre el navegador del cliente y el servidor HTTP.

2.3.2 *Patrón arquitectónico utilizado.*

Como se explicó en el epígrafe anterior, el patrón arquitectónico **Modelo-Vista-Controlador (MVC)** se emplea en la capa de funcionamiento y es el encargado de separar los datos de la aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos:

- **Modelo:** está compuesto por: datos, reglas de negocio y las funcionalidades correspondientes para la comunicación con el persistidor de datos Doctrine.
- **Controlador:** gestiona las entradas del usuario.
- **Vista:** muestra la información del modelo usuario.

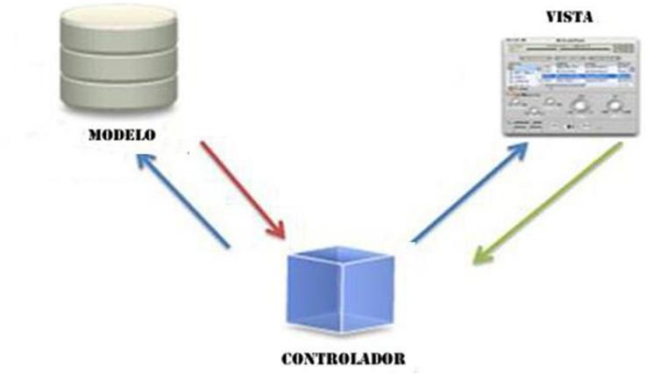

Figura 19: Modelo-Vista-Controlador (MVC).

2.3.3 *Patrones de diseño utilizados.*

Durante el desarrollo de la solución se utilizarán varios de los patrones generales de software para asignar responsabilidades o GRASP. Ellos son:

- **Experto:** se puede ver el uso de este patrón en todas las clases a utilizar en el componente ya que cada clase conoce su información y es la encargada de implementar las funcionalidades que brindan información de las mismas.
- **Creador:** su uso se puede ver en la clase controladora *GestcompartimentacionrecursosController*, ya que ella es la encargada de la creación de objetos de varias clases como son *SegUsuario, SegRol, ZendExt_Mongo*.
- **Controlador:** la clase controladora *GestcompartimentacionrecursosController* se encarga de llevar el control de todos los eventos relacionados con el negocio. Implementa las funcionalidades que dan respuesta a las peticiones del usuario.
- **Alta Cohesión:** el uso de éste patrón indica que la información almacenada en las clases debe ser coherente y relacionada a lo que se maneja en dicha clase y para esta solución se ve su uso igualmente en la clase controladora.
- **Bajo Acoplamiento:** el uso de éste patrón se evidencia en la poca relación existente entre las clases que conforman el componente.

De los patrones GoF, se usarán:

- El patrón creacional **Singleton** o Solitario. Se puede ver su aplicación en el uso del IoC[19], donde se garantiza que cada clase tenga una instancia única y permite que la misma sea accedida desde cualquier parte del sistema. .
- El patrón estructural **Fachada** se encarga de, mediante una única interfaz, relacionar varias clases de un subsistema, para facilitar su uso (Gamma, y otros). En el componente Compartimentación de la información se puede ver el uso de dicho patrón en la clase *SeguridadProxyService*, la cual se utiliza como interfaz de acceso a las demás interfaces y componentes del sistema y garantiza la comunicación con sistemas externos.
- El patrón de comportamiento **Mediador** es el encargado de definir un objeto que encapsula cómo interactúan un conjunto de objetos. Estimula la pérdida de acoplamiento ocultando las referencias explícitas entre los objetos, permitiendo variar su interacción de forma independiente. (Gamma, y otros) Éste patrón se puede evidenciar en la clase DatEntidadSegUsuarioSegRol, la cual garantiza la comunicación entre varios objetos de distintas clases como por ejemplo las clases *SegUsuario y SegRol.*

El uso de estos patrones garantizó asignar a cada clase la responsabilidad que le corresponde, obtener el menor número de relaciones y dependencias entre clases y aumentar las posibilidades de reusabilidad de las mismas. Todos estos elementos ayudaron a la confección de un diseño de la solución de gran claridad y entendimiento.

2.3.4 Diagrama de Clases.

Un diagrama de Clases representa las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas (Osmosislatina.com). A continuación (**Figura 20**) se muestra el diagrama de clases con estereotipos web identificado para el desarrollo del componente Compartimentación de la información.

[19] **IoC:** Inversión de control. Fichero XML donde se describen los servicios que brindan todos los sistemas.

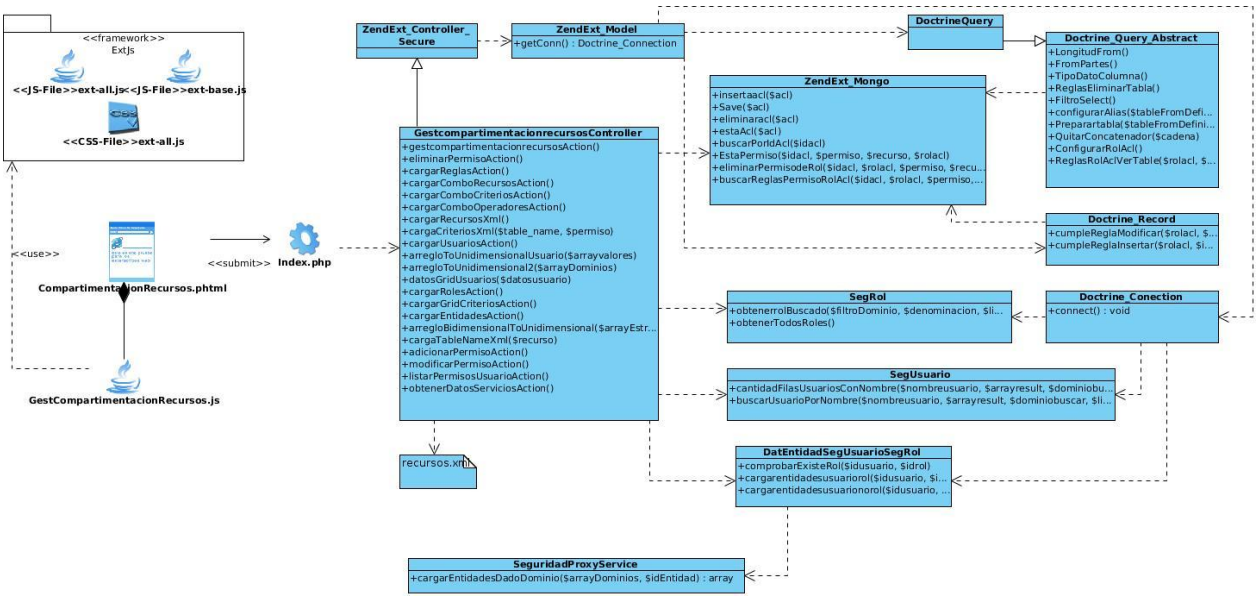

Figura 20: Diagrama de Clases.

Descripción de las clases del diseño asociadas a la solución.

Tabla 16: Descripción de clases

Clase	*Descripción*
ZendExt_Mongo	Es la clase encargada de establecer la conexión de la base de datos con el gestor Mongodb. Cuenta con funcionalidades para realizar operaciones de adición, eliminación y modificación de datos almacenados en la misma.
Doctrine_Record	Esta clase crea referencias de un objeto almacenado en una tabla de la base de datos, puede ser utilizada para conocer si un usuario puede modificar dicho objeto a través de los métodos mostrados en el diagrama.
Doctrine_Query_Abstract	Es la clase usada para realizar consultas a la base de datos y presenta funcionalidades para filtrar los resultados de la consulta realizada y para determinar si un usuario tiene derecho a realizar la eliminación de un objeto de la base de datos, entre otras.
Doctrine_Conection	Permite la conexión a la base de datos.
DoctrineQuery	Permite la creación de consultas hacia la base de datos.
ZendExt_Controller_Secure	Clase de donde heredan todas las clases controladoras.
GestcompartimentacionrecursosController	Clase controladora que hereda de *ZendExt_Controller_Secure*. Encargada de

	realizar todas las funcionalidades del componente.
recursos.xml	Fichero XML que almacenará los recursos a compartimentar y los criterios generales y específicos de cada uno.
CompartimentacionRecursos.phtml	Plantilla HTML encargada de mostrarle al usuario la interfaz correspondiente al componente con la que va a trabajar.
SegUsuario	Clase con toda la información concerniente a los usuarios registrados en el sistema.
SegRol	Clase donde se almacena la información de los roles del sistema.
DatEntidadSegUsuarioSegRol	Es la encargada de establecer las relaciones de cada usuario con los roles asociados a él en una entidad determinada, la cual se obtiene a través de la clase *SeguridadProxyService*.
GestCompartimentacionRecursos.js	Formulario js a través del cual se maneja la información que introducen los usuarios del sistema.
<<Framework>> ExtJs	Librería JavaScript que utiliza la clase *CompartimentacionRecursos.phtml*.
SeguridadProxyService	Esta clase sirve como fachada para la comunicación entre los componentes del marco de trabajo Sauxe.
ZendExt_Model	Entre las características de esta clase se pueden mencionar que obtiene las conexiones activas en el sistema cuando se trata de acceder a una clase determinada del modelo.
Index.php	Se encarga de determinar cuál es la clase controladora que le corresponde a cada página cliente.

2.3.5 Modelo de Datos.

Un modelo de datos es un lenguaje utilizado para la descripción de una base de datos. Por lo general, permite describir el tipo de datos que incluye la base y la forma en que se relacionan, las restricciones de integridad y las operaciones de manipulación de los datos. (Definición.de)

Para el componente Compartimentación de la información se generan 2 modelos de datos distintos, el primero para el gestor de base de datos PostgreSQL y el segundo perteneciente a la base de datos creada en MongoDB.

En PostgreSQL (**Figura 21**) se encuentran las tablas:

***seg_usuario*:** almacena toda la información concerniente a los usuarios registrados en el sistema.

***seg_rol*:** contiene la información almacenada de los roles asignados a cada uno de los usuarios.

dat_entidad_seg_usuario_seg_rol: del servicio perteneciente al sistema Estructura y Composición se obtiene la terna resultante de la relación que existe entre las tablas *seg_usuario* y *seg_rol*, donde se especifica que un usuario puede tener un solo rol en una entidad.

recursoxml: fichero XML donde se almacenan los recursos a compartimentar y los criterios de cada uno, criterios por los que se podrán crear reglas para un mayor grado de compartimentación.

En MongoDB (**Figura 22**) se encuentra la tabla ***acl***, que almacena la información referente a la relación creada luego de la asignación de permisos a determinados usuarios que desempeña determinado rol en una entidad sobre determinado recurso.

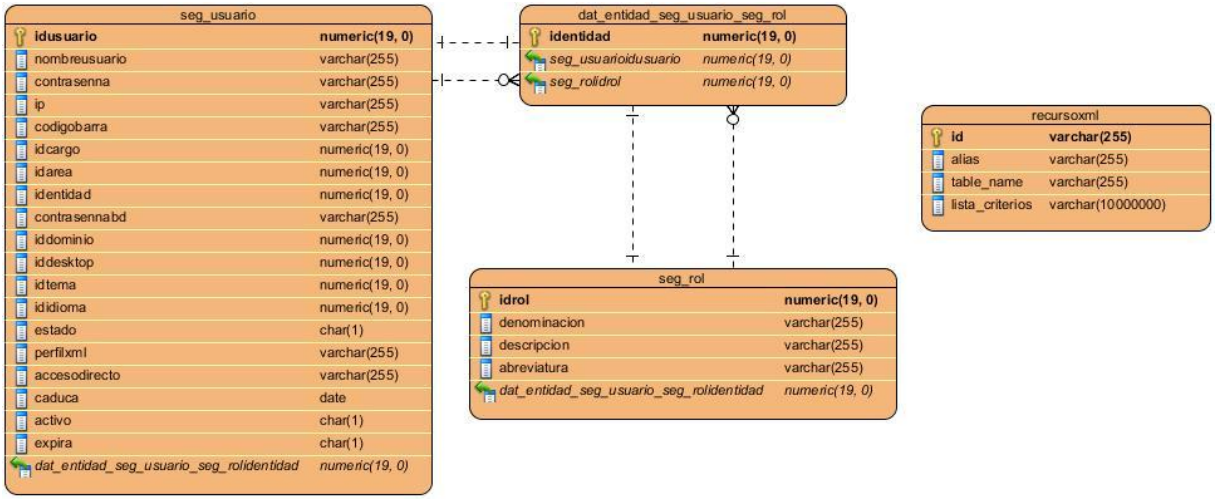

Figura 21: Modelo de Datos PostgreSQL.

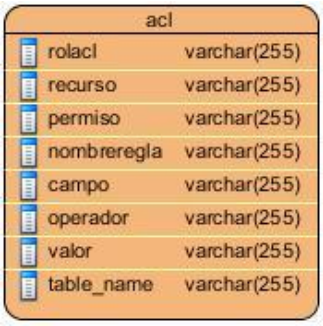

Figura 22: Modelo de Datos MongoDB.

CONCLUSIONES DEL CAPÍTULO.

En éste capítulo se elaboró el modelo conceptual de la solución, donde se identificó la relación que debía existir entre cada uno de los conceptos identificados. Se realizó la descripción de los requisitos funcionales y no funcionales identificados durante el proceso de Ingeniería de requisitos. Como parte del modelo de diseño elaborado, se estudió el estilo arquitectónico definido para el desarrollo en el departamento y se identificaron los patrones de diseño a utilizar en el desarrollo de la solución. Por último, se elaboró y describió el diagrama clases y los modelos de datos con que contará la solución.

CAPÍTULO III: IMPLEMENTACIÓN Y PRUEBAS.

INTRODUCCIÓN.

En éste capítulo se describirán los estándares de codificación a utilizar para garantizar un buen entendimiento y legibilidad del código, y los componentes necesarios para realizar la implementación y despliegue de la solución. Además se mostrarán los resultados de la evaluación mediante métricas de diseño, un caso de prueba y pruebas funcionales y de caja blanca; métodos escogidos para validar la solución.

1.1 Modelo de Implementación.

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje de programación utilizados; y cómo dependen los componentes unos de otros. (Brito Acuña)

1.1.1 Diagrama de componentes.

Los diagramas de componentes describen los elementos físicos (o componentes) del sistema y sus relaciones, mientras que los componentes representan todo tipo de elementos que serán desarrollados; pueden ser simples archivos, paquetes, entre otros.

Para el desarrollo del componente Compartimentación de la información es necesaria la interacción con otros componentes del marco de trabajo como son: ZendExt, Doctrine y ExtJS. El mismo hace uso de servicios internos mediante el IoC para obtener información de otros subsistemas del marco de trabajo, entre ellos los que brinda Estructura y Composición. A continuación se muestra el diagrama de componentes elaborado.

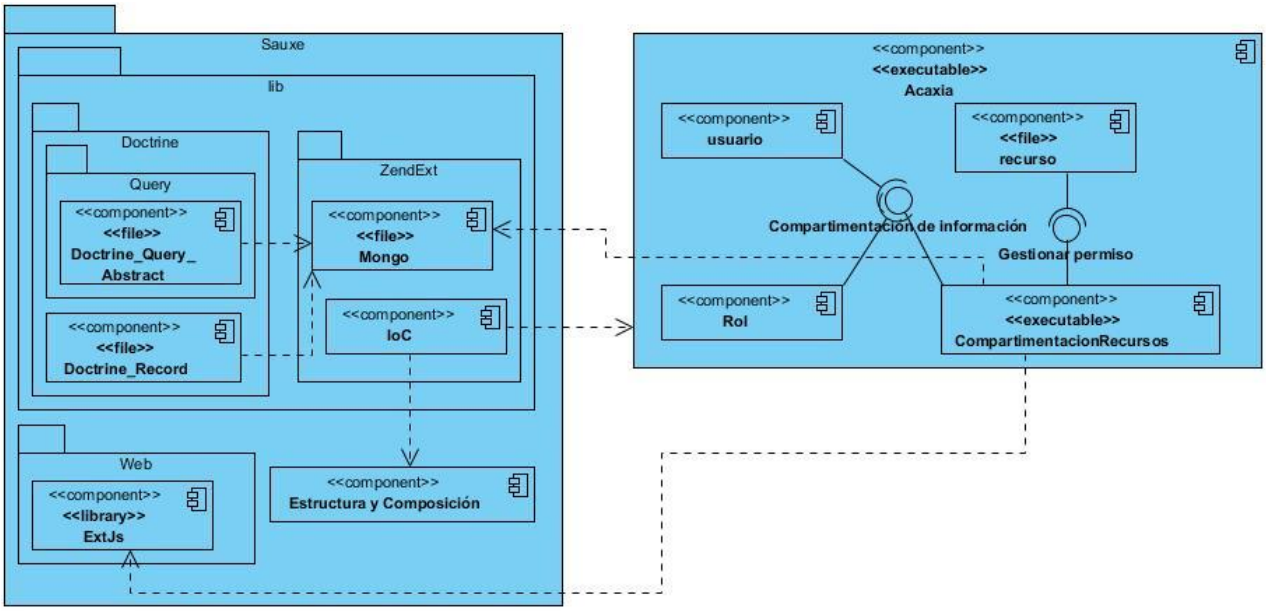

Figura 23: Diagrama de Componentes.

1.1.2 Diagrama de Despliegue.

El Diagrama de Despliegue se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes (nodos). Los nodos son objetos físicos que existen en tiempo de ejecución y que representan algún tipo de recurso computacional, también pueden ser dispositivos del sistema. (Sistem)

El usuario accede, desde su puesto de trabajo al sistema que se encuentra instalado en el servidor de aplicaciones y dicho servidor se conecta a los servidores de base de datos, para realizar las consultas y obtener los resultados deseados; en éste caso obtener los permisos otorgados a usuarios y/o roles sobre los recursos. (**Figura 24**)

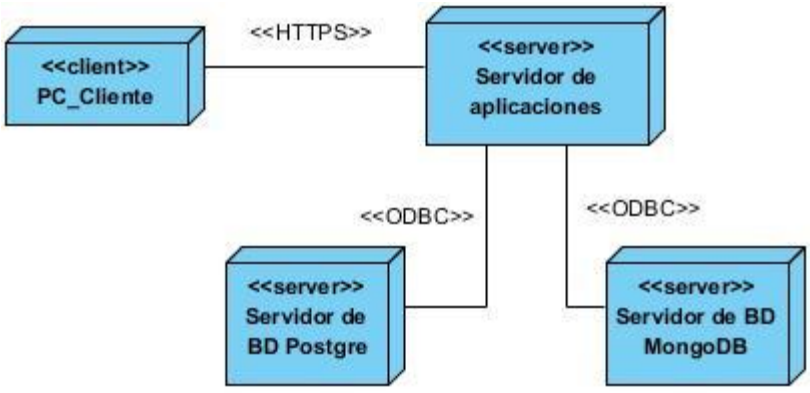

Figura 24: Diagrama de Despliegue.

1.1.3 Estándares de codificación.

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. (Alfonso, 2008)

Para el desarrollo del componente compartimentación de la información se utilizarán algunos de los estándares de codificación y normas propuestos como parte de la Línea de arquitectura determinada para el desarrollo del ERP Cuba. (Alfonso, 2008) .

Estándares de nomenclatura.

- **Nomenclatura de las clases.**

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación *PascalCasing**. Con sólo leerlo se reconoce el propósito de la misma.

Ejemplo: *GestionarUsuario.*

- **Nomenclatura según el tipo de clases.**

1. **Clase controladora.**

Las clases controladoras después del nombre llevan la palabra: “Controller”.

Ejemplo: *GestionarUsuarioController,*

- **Nomenclatura de las funciones.**

El nombre a emplear para las funciones se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación *CamelCasing**, y con sólo leerlo se reconoce el propósito de la misma.

Ejemplo: *insertarMoneda*.

- **Nomenclatura de las variables.**

El nombre a emplear para las variables se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación *CamelCasing*,* y comenzando con un prefijo según el tipo de datos.

Ejemplo: *arrMoneda*.

- **Prefijos para los tipos de datos.**

Los prefijos a utilizar en la creación de variables serán los siguientes:

Tabla 17: Prefijos para tipos de datos.

Tipos de Datos	Prefijos
Arreglos	arr
Objetos	obj

Normas de comentariado.

Es una necesidad comentar todo lo que se haga dentro del desarrollo, es decir, establecer las pautas que conlleven a lograr un código más legible y reutilizable, de manera que se pueda aumentar su mantenibilidad a lo largo del tiempo.

Los comentarios deben ser lo bastante claro y preciso de forma tal que se entienda el propósito de lo que se está desarrollando.

- **En las clases.**

Antes de la declaración de una clase se escribe una breve descripción donde se explique el propósito de la misma. Que se escribe de la siguiente forma:

```
/**
* Nombre de la clase *
* Descripcion *
* @author *
* @package *(módulo)
* @subpackage *(sub módulo)
* @copyright *
* @version (versión - parche) */
```

- **En las funciones.**

Antes de la declaración de la función se escribe una breve descripción donde se explique el propósito de la misma. Se escribe de la siguiente forma:

```
/**
* Nombre de la función *
* Descripcion *
* @author * (en caso de que no sea el autor de la clase)
* @param *(los parámetros que se le pasan a la función con su descripción)
* @throws *(en caso de que dispare una excepción)
* @return *(se pone lo que devuelve la función y un comentario)
*/
```

Estilo del código.

En la implementación cuando se escriba una sentencia en PHP la forma de utilizar los tab del mismo es la siguiente:

```
<?php
//código
?>
```

Figura 25: Estilo del código.

- **Sangría o indexado.**

La política de sangría a utilizar en la implementación es por **tab**.

```php
<?php
/**
 * Indentation
 */
class Example {
    var $theInt = 1;
    function foo($a, $b) {
        switch ( $a) {
            case 0 :
                $Other->doFoo ();
            break;
            default :
                $Other->doBaz ();
        }
    }
    function bar($v) {
        for($i = 0; $i < 10; $i ++) {
            $v->add ( $i );
        }
    }
}
?>
```

Figura 26: Estilo del código: Sangría o indexado.

- **Brazas o llaves.**

En la declaración de clases o interfaces, métodos, bloques y switch, la apertura de llaves se hace en la misma línea.

Ejemplo:

```php
<?php
/**
 * Braces
 */
interface EmptyInterface {
}

class Example {
    function bar($p) {
        for($i = 0; $i < 10; $i ++) {
        }
        switch ( $p) {
            case 0 :
                $fField->set ( 0 );
            break;
            case 1 :
                {
                    break;
                }
            default :
                $fField->reset ();
        }
    }
}
?>
```

Figura 27: Estilo del código: brazas o llaves.

- **Espacios en blanco.**

1. Arreglos.

La declaración de los espacios en blanco en los arreglos es como se muestra en el ejemplo.

Ejemplo:

```php
<?php
list ( $a, $b ) = array (1, 2, 3 );
$array = array (1 => 2, 2 => 3 );
$array [$i]->foo ();
$array [] = 'first cell';
?>
```

Figura 28: Estilo de código: Espacio en blanco_Arreglos.

1.2 Métricas de Diseño.

La evaluación de un producto, mediante métricas, es un aspecto fundamental a tener en cuenta; ya que, aunque las métricas del producto el software no suelen ser absolutas, brindan la posibilidad de evaluar la calidad a partir de varias reglas definidas claramente. Permiten detectar y corregir los posibles problemas que se puedan presentar durante el proceso de desarrollo y no después de terminado el producto.

IEEE (IEEE, 2007) define las métricas como: "*una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado*".

Según Pressman (Pressman), la calidad del diseño se puede evaluar aplicando métricas básicas para la calidad del diseño orientado a objetos. Los atributos de calidad involucrados son:

- ***Responsabilidad:*** responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- ***Complejidad de implementación:*** grado de complejidad que posee la implementación de un diseño de clases específico.
- ***Reutilización:*** grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- ***Acoplamiento:*** grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.
- ***Complejidad del mantenimiento:*** grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.
- ***Cantidad de pruebas:*** número o grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (componente, modulo, clase, conjunto de clases, entre otros) diseñado.

Las métricas seleccionadas para evaluar la calidad del diseño del componente Compartimentación de la información, para el sistema ACAXIA son:

1.2.1 *Tamaño Operacional de Clases (TOC)*

Está dado por el número de métodos u operaciones (de instancia privada y heredada) que están encapsulados dentro o por una clase. Evalúa los siguientes atributos de calidad:

Tabla 18: Atributos de calidad que evalúa TOC.

Atributo de calidad	*Modo en que lo afecta*
Responsabilidad	Aumento del TOC provoca aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Aumento del TOC provoca aumento de la complejidad de implementación de la clase.
Reutilización	Aumento del TOC provoca disminución del grado de reutilización de la clase.

Para la evaluación de dichos atributos de calidad, se definieron los siguientes criterios y categorías de evaluación:

Tabla 19: Criterios de evaluación de la métrica TOC.

Atributo	*Categoría*	*Criterio*
Responsabilidad	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Complejidad de implementación	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio

Resultados obtenidos

Evaluando la cantidad de operaciones de cada una de las clases con que cuenta el componente Compartimentación de la información, según los criterios establecidos en la **Tabla 20**, se obtuvo que un 67% de las clases cuentan con Responsabilidad y Complejidad: Baja y Reutilización: Alta.

Tabla 20: Instrumento de evaluación de la métrica TOC.

Clase	*Cantidad de Procedimientos*	*Responsabilidad*	*Complejidad*	*Reutilización*
Gestcompartimentacion recursosController	20	Baja	Baja	Alta
SegUsuario	42	Media	Media	Media
SegRol	25	Baja	Baja	Alta

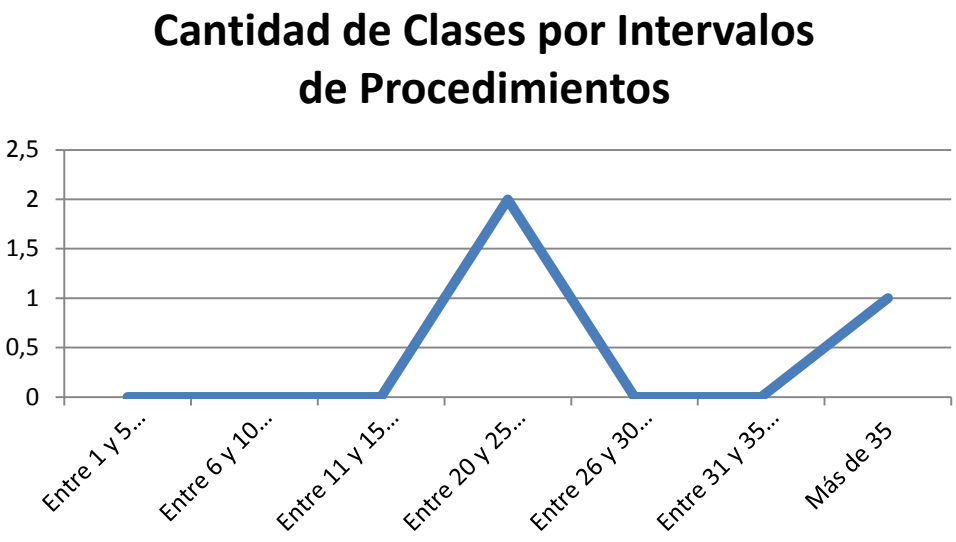

Figura 29: Representación de las clases según la cantidad de operaciones.

Teniendo en cuenta el umbral de cantidad de operaciones con que cuenta cada clase, está establecido que un tamaño de clase pequeño es aquel que tiene un valor menor o igual que 3, un tamaño medio para aquellas clases que tengan entre 4 y 10 operaciones y un tamaño de clase grande es aquel que es mayor o igual que 16. Se concluye que el componente cuenta con 3 clase fundamentales cuyo promedio de cantidad de operaciones equivale a 29. Los valores de tamaño quedan distribuidos de la siguiente manera:

Tabla 21: Tamaño de clases.

Umbral	*Tamaño*	*Cantidad de Clases*
Pequeño	<=3	0
Medio	>=4 y <=10	0
Grande	>=16	3

Como se puede observar el 100% de las clases están clasificadas de Grande lo cual se considera un resultado positivo según los parámetros de calidad Responsabilidad, Complejidad de Implementación y Reutilización propuestos para esta métrica.

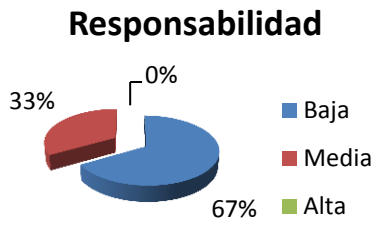

Figura 30: Resultados de la evaluación de la métrica TOC para el atributo de calidad Responsabilidad.

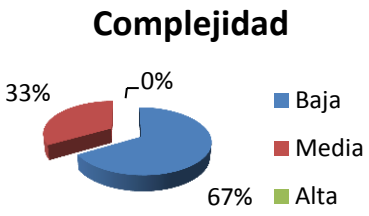

Figura 31: Resultados de la evaluación de la métrica TOC para el atributo de calidad Complejidad.

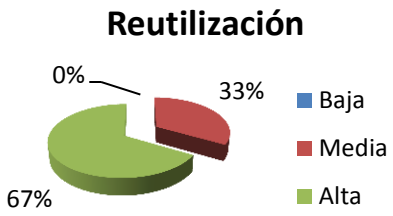

Figura 32: Resultados de la evaluación de la métrica TOC para el atributo de calidad Reutilización.

1.2.2 Relación entre Clases (RC)

Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

Tabla 22: Atributos de calidad que evalúa RC.

Atributo de calidad	*Modo en que lo afecta*
Acoplamiento	Aumento del RC provoca aumento del Acoplamiento de la clase.
Complejidad de mantenimiento	Aumento del RC provoca aumento de la complejidad del mantenimiento de la clase.
Reutilización	Aumento del RC provoca disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Aumento del RC provoca aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Para la evaluación de dichos atributos de calidad, se definieron los siguientes criterios y categorías de evaluación:

Tabla 23: Criterios de evaluación de la métrica RC.

Atributo	*Categoría*	*Criterio*
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de mantenimiento	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio
Cantidad de pruebas	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio

Resultados obtenidos

Evaluando la cantidad de relaciones entre clases con que cuenta el componente Compartimentación de la información, según los criterios establecidos en la **Tabla 24**, se obtuvo que un 67% de las clases cuentan con Bajo Acoplamiento y Media Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas.

Tabla 24: Instrumento de evaluación de la métrica RC.

Clase	*Cantidad de Relaciones de Uso*	*Acoplamiento*	*Complejidad de Mantenimiento*	*Reutilización*	*Cantidad de Pruebas*
GestcompartimentacionrecursosController	0	Ninguno	Baja	Alta	Baja
SegUsuario	1	Baja	Media	Media	Media
SegRol	1	Baja	Media	Media	Media

Teniendo en cuenta la cantidad de dependencias entre las clases mostrada, está establecido que una cantidad de relaciones pequeña será igual a 1, una cantidad media será entre 1 y 3 relaciones y grande será más de 3 relaciones.

Tabla 25: Cantidad de dependencias por clases.

Intervalos	*Cantidad de Clases*
0 dependencias	0
1 dependencias	2
> 1 dependencias	0

Tabla 26: Evaluación según cantidad de relaciones.

Umbral	*Relaciones*	*Cantidad de Clases*
Pocas	<=1	0
Medias	>1 y <= 3	2
Muchas	>3	0

Como se puede observar el 100% de las clases están clasificadas de Media según sus relaciones, lo cual representa un resultado positivo según los atributos de Acoplamiento, Complejidad del Mantenimiento, Reutilización y Cantidad de Pruebas.

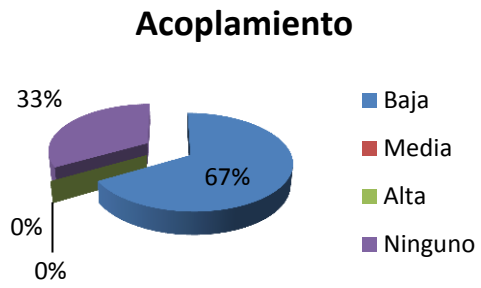

Figura 33: Resultados de la evaluación de la métrica RC para el atributo de calidad Acoplamiento.

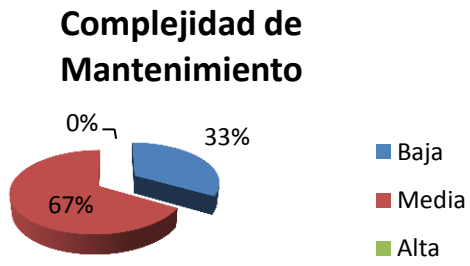

Figura 34: Resultados de la evaluación de la métrica RC para el atributo de calidad Complejidad de Mantenimiento.

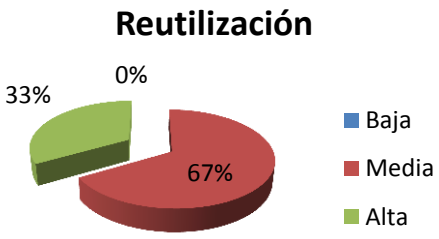

Figura 35: Resultados de la evaluación de la métrica RC para el atributo de calidad Reutilización.

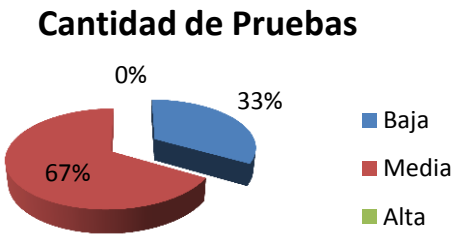

Figura 36: Resultados de la evaluación de la métrica RC para el atributo de calidad Cantidad de Pruebas.

1.2.3 Matriz interferencia de indicadores de calidad

En ésta investigación se define que más del 60% de las clases con Responsabilidad, Complejidad y Acoplamiento clasificados de *Bajo*; más del 60% de las clases con Complejidad de mantenimiento y Cantidad de pruebas clasificados de *Medio* y más del 60% de las clases con Reutilización *Alta* o *Media*, serán considerados resultados Favorables y por debajo de 60%, No Favorables.

La matriz de interferencia de indicadores de calidad es una representación de los atributos de calidad y las métricas utilizadas para evaluar la calidad del diseño propuesto para el componente. Con ella se puede conocer si los resultados obtenidos de las relaciones atributo/métrica son positivo o no, llevando estos resultados a una escalabilidad numérica donde, si los resultados son positivos se le asigna el valor ‘1’, si son negativos toma valor ‘0’ y si no existe relación es considerada como nula y es representada con un guión simple (-). Luego se puede obtener un resultado general para cada atributo promediando todas sus relaciones no nulas.

Tabla 27: Resultados de la evaluación de la relación Atributo/Métrica.

Atributo/Métrica	*TOC*	*RC*	*Promedio*
Responsabilidad	1	-	1
Complejidad de Implementación	1	-	1
Reutilización	1	1	1
Acoplamiento	-	1	1
Complejidad de Mantenimiento	-	1	1
Cantidad de pruebas	-	1	1

Teniendo en cuenta los resultados obtenidos al aplicar las métricas de diseño TOC y RC, que se muestran en las **Tabla 20** y **Tabla 24**, se construye el siguiente gráfico (**Figura 37: Resultados de la evaluación.**), donde se muestran los resultados obtenidos al evaluar los atributos de calidad involucrados en cada una de las métricas evaluadas.

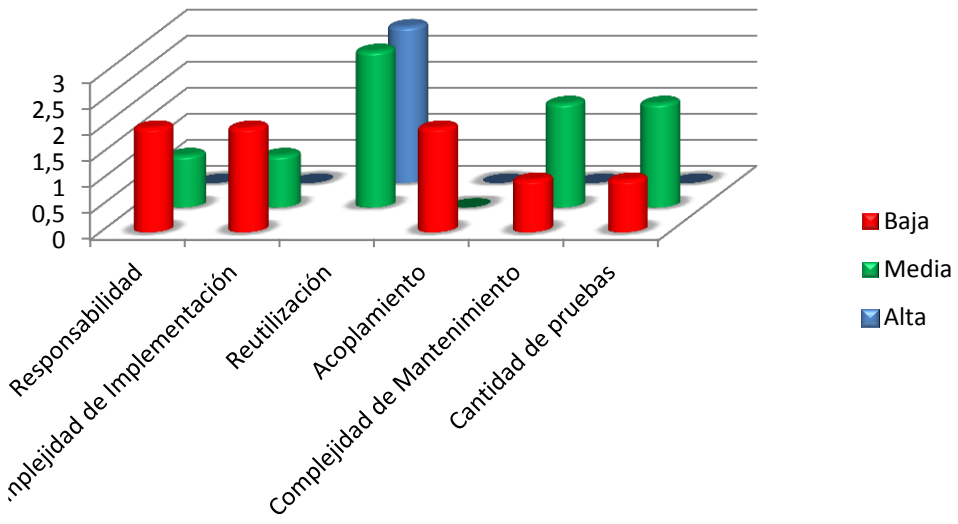

Figura 37: Resultados de la evaluación.

1.3 Pruebas de Software.

Las pruebas de software son un instrumento para determinar el status de la calidad de un software. Tienen como objetivo, además de descubrir errores, medir el grado en que el software cumple con los requerimientos definidos. (pruebasdesoftware.com, 2005)

1.3.1 Pruebas de Caja Negra

Las pruebas de caja negra son las que se aplican a la interfaz del software y se centran en los requisitos funcionales del sistema; permitiendo derivar conjuntos de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa. (Pressman)

Para realizarle las pruebas de caja negra al componente Compartimentación de la información, se elaboraron Casos de pruebas específicos para cada uno de los RF descritos en la sección **2.2.1** de éste documento. Dichos casos de prueba fueron evaluados y aprobados por el Departamento de Calidad del CEIGE y por el equipo de trabajo del Dpto. de Tecnología encargado de revisar y liberar el componente para ser usado como parte de ACAXIA.

1.3.2 Pruebas de Caja Blanca

Las pruebas de caja blanca son un método de diseño que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de prueba. Al emplear sus métodos, se puede garantizar que, al menos una vez, se ejecuten todas las rutas independientes del módulo. (Pressman)

La prueba del **Camino Básico** es una técnica que permite derivar casos de prueba a partir de un conjunto de caminos independientes por los cuales puede circular el flujo de control. (Pressman) Para obtener el conjunto de caminos independientes se construye el *Grafo de Flujo* (**Figura 40**) asociado y se calcula su *Complejidad ciclomática*.

Para poder elaborar el Grafo de Flujo, primero se deben enumerar las sentencias del código:

```
public function modificarPermisoAction() {
    $idAcl = $this->global->Perfil->iddominio;   1
    $idRol = $this->_request->getPost("idrol");
    $idUsuario = $this->_request->getPost("idusuario");
    $permisoViejo = $this->_request->getPost("permisoviejo");
    $recursoViejo = $this->_request->getPost("recursoviejo");
    $permisoNuevo = $this->_request->getPost("permisonuevo");
    $recursoNuevo = $this->_request->getPost("recursonuevo");
    $arrReglas = json_decode(stripslashes($this->_request->getPost('arrayReglas')));
    $idEntidad = $this->_request->getPost('identidad');
    $arrMongo = array();
    $arrMongo['_id'] = (string) $idacl;


    $rolAcl = $idUsuario . '_' . $idRol . '_' . $idEntidad;


        foreach ($arrReglas as $reg) {              2
            $tupla['rolacl'] = (string) $rolAcl;    3
            $tupla['permiso'] = (string) $permisoNuevo;
            $tupla['recurso'] = (string) $recursoNuevo;
            $tupla['nombreregla'] = $reg->nombreregla;
            $tupla['campo'] = $reg->campo;
            $tupla['operador'] = $reg->operador;
            $tupla['valor'] = $reg->valor;
            $tupla['table_name']=$this->cargaTableNameXml($recursoNuevo);
            if (!in_array($tupla, $arrMongo)){      4
                $arrMongo[] = $tupla;               5
            }                                       6
        }
```

Figura 38: Código fuente de la funcionalidad Modificar permiso 1.

```
$rolEliminar = $idUsuario . '_' . $idRol . '_' . $idEntidad;    7
$objmongo = new ZendExt_Mongo();
$objmongo->eliminarPermisodeRol($idAcl, $rolEliminar, $permisoViejo, $recursoViejo);
$arrDatos = $objmongo->buscarPorIdAcl($idAcl);

    foreach ($arrDatos as $obj) {      8
        $tup['rolacl'] = $obj['rolacl'];          9
        $tup['permiso'] = $obj['permiso'];
        $tup['recurso'] = $obj['recurso'];
        $tup['nombreregla'] = $obj['nombreregla'];
        $tup['campo'] = $obj['campo'];
        $tup['operador'] = $obj['operador'];
        $tup['valor'] = $obj['valor'];
        $tup['table_name']=$obj['table_name'];
        $objeto = $tup;
        if (!in_array($objeto, $arrMongo)) {      10

            $arrMongo[] = $objeto;                11
        }                                         12
    }
     $objmongo->Save($arrMongo);      13
    $this->showMessage('Permiso modificado satisfactoriamente');


}
```

Figura 39: Código fuente de la funcionalidad Modificar permiso 2.

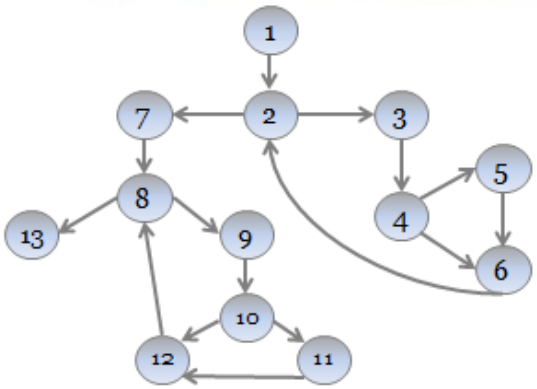

Figura 40: Grafo de flujo asociado a la funcionalidad Modificar permiso.

La complejidad ciclomática es la métrica de software con que se define la cantidad de caminos independientes de cada una de las funcionalidades del programa y provee el límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. (Pressman)

Según Pressman la Complejidad ciclomática se basa en la teoría gráfica y se calcula de 3 maneras distintas, donde, para que el cálculo sea correcto, todas deben arrojar el mismo resultado:

1. El número de regiones corresponde a la complejidad ciclomática "*V (G)*".
   V (G) = R Donde R es la cantidad total de regiones.
   V (G) = 5
2. **V (G) = E – N + 2** Donde E es el número de aristas y N número de nodos.
   V (G) = 14 – 11 + 2

V (G) = 5

3. **V (G) = P + 1** Donde P es el número de nodos predicados.

V (G) = 4 + 1

V (G) = 5

Teniendo en cuenta el valor obtenido de la complejidad de la funcionalidad Modificar permiso, se obtuvo la cantidad de posibles caminos independientes y cantidad de pruebas que se deben realizar para comprobar que cada sentencia se ejecute al menos una vez.

El cálculo arrojó que V (G) = 5, por lo que los posibles caminos básicos son:

- Camino básico Nº 1: 1, 2, 7, 8, 13.
- Camino básico Nº 2: 1, 2, 7, 8, 9, 10, 11, 12, 8, 13.
- Camino básico Nº 3: 1, 2, 7, 8, 9, 10, 12, 8, 13.
- Camino básico Nº 4: 1, 2, 3, 4, 5, 6, 2, 7, 8, 13.
- Camino básico Nº 5: 1, 2, 3, 4, 6, 2, 7, 8, 13.

Derivación de casos de prueba para la funcionalidad modificarPermisoAction

Luego de elaborados el Grafo de Flujo y los caminos básicos a recorrer, se elaboran los casos de prueba correspondiente a cada camino con los que se garantizará que cada sentencia de código se ejecute al menos una vez en cada caso identificado.

Caso de prueba para camino básico Nº 1 (1, 2, 7, 8, 13).

Si $arrReglas = vacío y $arrDatos = vacío.

Caso de prueba para camino básico Nº 2 (1, 2, 7, 8, 9, 10, 11, 12, 8, 13).

Si $arrReglas = vacío y $arrDatos ≠ vacío y $objeto no está en $arrMongo.

Caso de prueba para camino básico Nº 3 (1, 2, 7, 8, 9, 10, 12, 8, 13).

Si $arrReglas = vacío y $arrDatos ≠ vacío y $objeto está en $arrMongo.

Caso de prueba para camino básico Nº 4 (1, 2, 3, 4, 5, 6, 2, 7, 8, 13).

Si $arrReglas ≠ vacío y $tupla no está en $arrMongo.

Caso de prueba para camino básico Nº 5 (1, 2, 3, 4, 6, 2, 7, 8, 13).

Si $arrReglas ≠ vacío y $tupla está en $arrMongo.

Resultados obtenidos.

Al ejecutar cada uno de los caminos básicos identificados se comprobó que cada una de las sentencias de código de cada camino será ejecutada al menos una vez, siendo éstos los resultados esperados de la validación mediante pruebas de caja blanca.

Ésta misma técnica de validación fue aplicada a todas las funcionalidades del componente obteniendo igual resultado.

1.4 Prueba del componente Compartimentación de la información como solución al problema científico de la investigación basados en un caso de estudio.

En el CEIGE se está desarrollando la versión 2.0 de ACAXIA, con el objetivo de minimizar las brechas de seguridad existentes en la versión anterior del producto y a partir del principio básico de la seguridad informática: *Mínimo privilegio*, se listan una serie de requerimientos que debe cumplir esta versión del producto:

1. Se necesita que usuarios del sistema con un mismo rol en una entidad tengan acceso a información diferenciada teniendo en cuenta las especificidades de su puesto de trabajo o de su área.
2. Los usuarios deben ser capaces de listar, insertar, modificar y/o eliminar recursos atendiendo a criterios especificados.

1.4.1 Caso de estudio.

En ACAXIA se encuentran registrados los usuarios "*Yoel*" y "*Oiner*", los mismos desempeñan el rol de "seguridad" en la entidad "UCI". Hasta el momento "*Yoel*" y "*Oiner*" pueden gestionar (listar, insertar, modificar y eliminar) usuarios en el sistema, lo cual constituye un **problema** ya que cada uno de ellos realiza funciones diferentes dentro de dicha entidad. Es por eso que para gestionar la información de manera más segura y **mejorar la confidencialidad e integridad** de la misma se deben tener en cuenta las siguientes reglas del negocio que se establecieron por la dirección de seguridad informática de la UCI:

- Listar los usuarios de manera diferenciada atendiendo al dominio al que pertenecen, "*Oiner*" debe ver los usuarios que pertenecen al dominio uci y "*Yoel*" a los que pertenecen al dominio uci que no sean de la entidad UCI.
- "*Oiner*" solo debe insertar usuarios que pertenezcan la entidad UCI y "*Yoel*" a los que pertenezcan a FACULTAD.
- "*Oiner*" solo debe modificar los usuarios que pertenecen a la entidad UCI y "*Yoel*" solo debe modificar los usuarios que pertenecen a la entidad FACULTAD.

- "*Oiner*" solo debe eliminar los usuarios que pertenecen a la entidad UCI y "*Yoel*" solo deben eliminar los usuarios que pertenecen a la entidad FACULTAD.

Situación antes de la existencia del componente Compartimentación de la información.

Antes de la implementación del componente Compartimentación de la información los usuarios "*Oiner*" y "*Yoel*" tenían acceso a manipular cualquier información referente a otros usuarios dentro del sistema por lo que podían realizar cualquier operación (Listar, Insertar, Modificar, Eliminar) sobre los usuarios del sistema.

Situación después de la existencia del componente Compartimentación de la información.

Luego de realizar las implementaciones correspondientes en el sistema para gestionar la información teniendo en cuenta las reglas expuestas, se obtuvo como resultado un componente capaz de manipular los permisos a cada usuario; donde el administrador del sistema podrá adicionar, modificar y/o eliminar los permisos para cada usuario teniendo en cuenta el rol que desempeñen en una entidad determinada.

Conclusiones validación del componente mediante un caso de estudio.

Una vez terminado el componente se puede observar cómo los usuarios Oiner y Yoel pueden manipular la información de forma diferenciada teniendo en cuenta determinadas reglas, lo cual evidencia que el componente Compartimentación de la información contribuye al mejoramiento de la confidencialidad e integridad de la información almacenada en ACAXIA. Por lo que se llega a la conclusión de que el sistema es considerado una solución capaz de resolver el problema planteado en la introducción del trabajo de tesis realizado.

CONCLUSIONES DEL CAPÍTULO.

En éste capítulo se realizó el modelo de implementación y como parte de él se obtuvieron el Diagrama de Componentes y de Despliegue de la solución y se documentaron los estándares de codificación por los que se guió la implementación del componente. Para la aprobación y liberación del mismo, el Dpto. de Calidad del CEIGE efectuó 2 iteraciones de pruebas, obteniendo en la última 0 (cero) no conformidades.

Por último, para probar que el componente sí constituye una mejora para ACAXIA y los sistemas suscritos a él, se elaboró un caso de estudio, donde se evalúa la situación existente antes de la implementación del componente y cómo dicho componente resuelve los conflictos de seguridad que se planteaba el sistema.

CONCLUSIONES.

Al concluir el desarrollo del componente Compartimentación de la información se pudo arribar a las siguientes conclusiones:

- Luego del estudio sobre el proceso de Compartimentación de la información y el Control de Acceso en los sistemas de gestión de seguridad expuestos, se obtuvo como resultado que hasta el momento ninguno soluciona la situación problemática expuesta al inicio de la investigación.
- Luego de un análisis de las tendencias tecnológicas actuales y las herramientas definidas para el desarrollo en el Dpto. de Tecnología, se hizo uso de un nuevo gestor de base de datos MongoDB para la implementación de la solución.
- Luego de realizados los procesos de Análisis, Diseño, Implementación y Pruebas se obtuvo un producto funcional acorde a los requisitos identificados.
- Como constancia, la aplicación fue probada por el Dpto. de Calidad del CEIGE y por el grupo de trabajo del Dpto. de Tecnología, los cuales constataron que el componente cumple los objetivos determinados al inicio de esta investigación.

Por lo anteriormente expuesto se concluye que el componente, efectivamente sí constituye una mejora de la integridad y confidencialidad de la información para los sistemas suscritos al Sistema de Gestión Integral de Seguridad ACAXIA.

RECOMENDACIONES.

- Implementar una solución que permita configurar el XML de los recursos a compartimentar desde una interfaz visual.
- A otros sistemas de gestión se les recomienda hacer uso del Sistema de Gestión Integral de Seguridad ACAXIA una vez liberada su próxima versión.

REFERENCIAS BIBLIOGRÁFICAS.

Alfonso, Damian Pérez. 2008. *Normas y estándares de codificación.* Universidad de las Ciencias Informáticas. 2008. Normas y estándares de Codificación del ERP.

Baryolo, Msc. Oiner Gómez. 2010. *Proyecto de Investigación. Seguridad en Sistemas de Gestión.* [Digital] Ciudad de La Habana, Cuba : Universidad de las Ciencias Informáticas, 2010.

Bass, Len, Clements, Paul y Kazman, Rick. 2003. *Software Architecturee in Practice. 2nd Edition.* s.l. : Addyson Wesley, 2003.

Berzosa, Luis Rodríguez. Control de Acceso: De la era Mainframe a las PKI. [En línea] [Citado el: 30 de Enero de 2012.] http://www.iec.csic.es/CRIPTONOMICON/articulos/expertos69.html.

Brito Acuña, Karenny. eumed.net. [En línea] [Citado el: 9 de Abril de 2012.] http://www.eumed.net/libros/2009c/584/RUP%20Diseno%20e%20implementacion%20del%20sistema.htm.

Buschmann, F., y otros. 1996. *Pattern - Oriented Software Architecture. A System of Ptterns. .* Inglaterra : John Wiley & Sons, 1996.

CA. Identificación, Autenticación y Autorización. **Colectivo de autores. Dpto. Sist. Digitales.**

Camacho, Erika, Cardeso, Fabio y Nuñez, Gabriel. 2004. *Arquitectura de software. Guía de estudio.* 2004.

Consultoría de áreas de conocimiento. *TortoiseSVN: Manual de usuario.* [Digital] s.l. : Ejie S.A.

Cornejo, José Enrique González. DocIRS. *¿Qué es UML?* [En línea] [Citado el: 25 de Mayo de 2012.] http://www.docirs.cl/uml.htm.

Definicion.de. Definicion.de. *Lenguaje de Programación.* [En línea] [Citado el: 28 de Mayo de 2012.] http://definicion.de/lenguaje-de-programacion/.

Definición.de. Definición.de. [En línea] [Citado el: 5 de Abril de 2012.] http://definicion.de/modelo-de-datos/.

Diseño y Programación Orientada a Objetos. **2004/2005.** 2004/2005.

Doctrine, Web. Doctrine. [En línea] [Citado el: 13 de Diciembre de 2011.] http://www.doctrine-project.org/projects/orm/1.2/docs/manual/introduction/en.

Escalona Cuaresma, Dra. María José y González Romano, Dr. José Mariano. 2006/2007. *Metodología y Técnicas en proyectos software para la Web. .* Universidad de Sevilla : s.n., 2006/2007.

ExtJS, Comunidad. ExtJS Nuestra comunidad en español. *Qué es ExtJS?* [En línea] [Citado el: 25 de Mayo de 2012.] http://www.extjs.mx/2011/11/post-con-cursos-generales/.

Gamma, Erich, y otros. *Design Patterns, Elements of Reusable Object-Oriented Software.*

Goguen, J. y Linde, C. 1993. *Techniques for Requirements Elicitation.* 1993.

Gómez Velázquez, Msc. Karel. *Propuesta de procesos de Autenticación, Autorización y Auditoría (AAA) para aplicaciones basadas en Servicios WebXML.* [Digital] Ciudad de La Habana : Universidad de las Ciencias Informáticas.

2009. Guía de Seguridad de las TIC. [En línea] Centro Criptológico Nacional (CCN) , 24 de Agosto de 2009. [Citado el: 6 de Diciembre de 2011.] España. https://www.ccn-cert.cni.es/publico/serieCCN-STIC401/index.html.

Ibarra Naranjo, Hyldeé M y Mañas Argemí, José A. *RBAC: Alternativa actual para la realización de Conrol de Accesos a gran escala.* [Digital] Madrid, España : Dpto. Ingeniería en Sistemas Telemáticos. Universidad Politécnica de Madrid.

IBM. IBM. *Tivolti Identity Manager.* [En línea] IBM. [Citado el: 7 de Diciembre de 2011.] http://www-142.ibm.com/software/products/es/es/identity-mgr.

IEEE. 2007. *Introducción a la Ingeniería de Requisitos. Ingeniería de software.* [Digital] 2007.

Introducción a herramientas CASE y System Architect. s.l. : Universidad Politécnica de valencia.

Larman, Craig. 1999. *UML y Patrones. Introducción al Análisis y Diseño orientado a objetos.* Mexico : Prentice Hall, 1999.

Mena López, Grette Leydi y Tenrrero Cabrera, Marianela. 2008. *Arquitectura ERP.* 2008.

Mendoza, Ing. Yoel Hernández. 2011. *Proyecto Técnico. Sistema de Gestión Integral de Seguridad ACAXIA.* [Digital] Ciudad de La Habana : Universidad de las Ciencias Informáticas, 2011.

Msc. Oiner Gómez Baryolo, Nemuris Silega Martínez. 2008. *Plantilla Registro de la Propiedad Intelectual (Sauxe).* [Digital] Ciudad de La Habana : Universidad de las Ciencias Informáticas, 2008.

Narváez, Daniel Estiven Valdivieso. 2009. *Análisis comparativo entre las técnicas utilizadas en la Ingeniería de Requisitos, permitiendo evaluar dichas técnicas frente a las características de los proyectos de software.* Ecuador : s.n., 2009.

NetBeans.org. NetBeans.org. [En línea] [Citado el: 8 de Junio de 2012.] http://netbeans.org/community/releases/70/.

ongei.gob.pe. www.ongei.gob.pe. [En línea] [Citado el: 08 de Junio de 2012.] http://www.ongei.gob.pe/publica/metodologias/Lib5011/cap2-3.htm.

OpenERP. OpenERP. [En línea] [Citado el: 24 de Enero de 2012.] http://doc.openerp.com/v6.0/book/8/8_20_Config/8_20_Config_accessRights.html.

Osmosislatina.com. Osmosislatina.com. [En línea] [Citado el: 8 de Junio de 2012.] http://www.osmosislatina.com/lenguajes/uml/clasesob.htm.

Paramio, Carlos. 2011. www.genbetadev.com. *Una introducción a MongoDB.* [En línea] 10 de Mayo de 2011. [Citado el: 26 de Enero de 2012.] http://www.genbetadev.com/bases-de-datos/una-introduccion-a-mongodb.

Pérez, Ing. Mileidy M. Sarduy. *Modelo Conceptual (CIG-SEG-N-i1301).* Departamento de Tecnología, Universidad de las Ciencias Informáticas. . Modelo conceptual del sistema ACAXIA.

Pérez, Ing. Mileidy Sarduy. 2011. *Especificación de requisitos de software.* Ciudad de La Habana : s.n., 2011.

Pérez, Javier Eguíluz. librosweb.es. *Introducción a JavaScript.* [En línea] [Citado el: 13 de Diciembre de 2011.] http://www.librosweb.es/javascript/.

Pfleeger, Charles P. y Pfleeger, Shari Lawrence. Octubre, 2006. *Security in Computing, 4ta Edición.* s.l. : Prentice Hall, Octubre, 2006.

2010 - 2011. php.net. [En línea] The PHP group, 2010 - 2011. [Citado el: 13 de Diciembre de 2011.] http://www.php.net/.

PostgreSQL. postgresql.org. *What is PostgreSQL?* [En línea] PostgreSQL Global Development Group. [Citado el: 8 de Febrero de 2012.] http://www.postgresql.org/docs/8.3/static/intro-whatis.html.

Pressman, Roger S. *Software Engineering. A practitioner´s Approach.* 7th Edition.

Producción, Equipo de. *Modelo de Desarrollo Orientado a Componentes del Proyecto ERP - Cuba.* Centro de desarrollo de soluciones empresariales (CEDRUX).

pruebasdesoftware.com. 2005. pruebasdesoftware.com. *Gestión de Calidad y Pruebas de Software.* [En línea] 2005. [Citado el: 30 de Mayo de 2012.] http://pruebasdesoftware.com/laspruebasdesoftware.htm.

2011. Referencia de Zend Framework Parte III. *Capítulo 12. Zend_Acl.* [En línea] Zend Technologies. Inc., 16 de Noviembre de 2011. [Citado el: 9 de Diciembre de 2011.] http://manual.zfdes.com/es/zend.acl.html.

Reynoso, Carlos y Kiccillof, Nicolás. Marzo, 2004. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* v 1.0. Universidad de Buenos Aires : s.n., Marzo, 2004.

Sánchez, M., Jiménez, B. y Gutiérrez, F. L. *Estudio del Control de Acceso en un Sistema Colaborativo.*

Sevilla, Escuela Técnica Superior de Ing. Informática. *Manual Validación.* [Digital] Sevilla : Departamento de Lenguajes y Sistemas Informáticos. Escuela Técnica Superior de Ingeniería Informática. Universidad de Sevilla.

Sistem, SPARX. SPARX Sistem. [En línea] [Citado el: 23 de Febrero de 2012.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html.

2009. SOA Agenda. *Qué son los frameworks.* [En línea] 25 de Octubre de 2009. [Citado el: 13 de Diciembre de 2011.] http://soaagenda.com/journal/articulos/que-son-los-frameworks/.

Society, Software Engineering Standards Committee of the IEEE Computer. 2000. *IEEE STD 1471-2000.* 2000.

Sommerville, Ian. 2005. *Ingeniería de Software.* 7ma. 2005.

UCI. Portal de la Universidad de las Ciencias Informáticas. *Misión.* [En línea] [Citado el: 9 de Abril de 2012.] http://www.uci.cu/mision.

2002. Universidad de las Ciencias Informáticas. *Misión.* [En línea] UCI, 2002. http://www.uci.cu/mision.

Visual-Paradigm. Visual-Paradigm. *Visual Paradigm for UML 8.3 Model-Code-Deploy Platform.* [En línea] [Citado el: 13 de Diciembre de 2011.] http://www.visual-paradigm.com/product/vpuml/.

Wiki, Httpd. Httpd Wiki. FaQ. *What is Apache?* [En línea] [Citado el: 24 de Mayo de 2012.] http://wiki.apache.org/httpd/FAQ#What_is_Apache.3F.

2008. Zend Framework. *Qué es Zend Framework?* [En línea] 9 de Septiembre de 2008. [Citado el: 13 de Diciembre de 2011.] http://spanish.zendfw.com/que-es-zend-framework/.

GLOSARIO DE TÉRMINOS.

A.

Artefactos: Productos tangibles del proyecto que son creados, modificados y usados dentro de las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.

B.

Brecha de seguridad: vía por la que se puede atacar la seguridad de un sistema.

C.

Compartimentación de la información: División de la información sensible y de los elementos y recursos de un sistema de información en unidades menores; basándose en la necesidad de conocer, a fin de reducir el riesgo de accesos no autorizados.

Componente: Unidad de composición de aplicaciones de software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio.

Confidencialidad: Se refiere a que solo personal autorizado conocerá la información.

Control de acceso: Conjunto de políticas dentro de un sistema de gestión de identidad que definen las normas de todo lo que un usuario se le permite hacer en el ámbito de aplicación de dicho sistema.

Criterio: atributo de los recursos que representan columnas en la tabla del recurso.

E.

Entidad: Forma parte de la estructura de un país y puede comportarse como un ministerio, entidad, área o cargo.

F.

Funcionalidad: Coherencia entre las necesidades detectadas y los resultados que se obtienen con el uso del material. Es lo que un producto puede hacer. Probar la funcionalidad significa asegurar que el producto funciona tal como estaba especificado.

H.

Herramienta: Es un objeto elaborado a fin de facilitar la realización de una tarea mecánica.

I.

Integridad: La información no será alterada, borrada, reordenada o copiada. Debe permanecer en su estado original, sin haber sido alterada por personal o de forma no autorizado.

M.

Métrica: Medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado.

P.

Permiso: tipo de autorización que puede tener un rol o usuario sobre determinado recurso.

R.

Recurso: cualquier elemento que se maneje a nivel de negocio y que esté representado en una tabla de la BD.

Regla: especificación que se le agrega a los permisos.

Reusabilidad: Capacidad de un componente y un subsistema para ser usado por otras aplicaciones en otros escenarios. Esta minimiza la duplicación de componentes así como el tiempo de implementación.

Rol: agrupa una serie de permisos sobre sistemas, funcionalidades y acciones que se le asignarán a un conjunto de usuarios.

S.

Sistema: Es un conjunto organizado de objetos o partes interactuantes e interdependientes, que se relacionan formando un todo unitario y complejo.

Software: Se refiere al equipamiento lógico o soporte lógico de una computadora digital, y comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de una tarea específica.

U.

Usuario: cualquier persona que interactúa con el sistema y juega un rol determinado en el mismo.

V.

Validación: Confirmación mediante el suministro de evidencia objetiva de que se han cumplido los requisitos para una utilización o aplicación específica prevista.