UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

TÍTULO: SOLUCIÓN INFORMÁTICA DE AUTORIZACIÓN EN ENTORNOS MULTIENTIDAD Y MULTISISTEMA.

MEMORIA INDIVIDUAL PRESENTADA PARA OPTAR POR TÍTULO DE MÁSTER EN INFORMÁTICA APLICADA

Autor: Ing. Oiner Gómez Baryolo
Tutor: MsC. Yadenis Piñero Pérez

Ciudad de la Habana, julio del 2010
"Año del 52 Aniversario del Triunfo de la Revolución Cubana"

DECLARACIÓN JURADA DE AUTORÍA

Yo **Oiner Gómez Baryolo** con carné de identidad **83031215867**, declaro que soy el autor principal del resultado que expongo en la presente memoria titulada **Proceso de autorización en entornos multientidad y multisistema,** para optar por el título de Máster en Informática Aplicada. Finalmente asumo la responsabilidad moral y jurídica que se derive de este juramento profesional.

Y para que así conste, firmo la presente declaración jurada de autoría en

Ciudad de la Habana a los ____ días del mes de _____del año 2010.

_____	_____
Firma del Autor	**Firma del Tutor**
Ing. Oiner Gómez Baryolo	**MsC. Yadenis Piñero Pérez**

AGRADECIMIENTOS

SÍNTESIS

La seguridad es un factor fundamental en cualquier institución y los datos son una parte fundamental en las mismas y cualquier pérdida, desvió o mala manipulación de la información ocasionaría daños económicos y sociales irreparables. Por tanto se requiere de una buena administración y control de la información y medios materiales, así como una adecuada seguridad acorde a los intereses del país.

En la actualidad todas las aplicaciones implantadas en Cuba donde existe manejo de la información implementan su propia seguridad, lo cual ocasiona pérdida de tiempo y gastos innecesarios de cuantiosos recursos humanos y materiales. Además no se realiza una administración centralizada de los sistemas de seguridad, lo que trae consigo que sea muy difícil controlar que no ocurran violaciones en los sistemas y en caso de que ocurra poder detectarlas. Con el inicio del proyecto ERP-Cuba surge la necesidad de desarrollar un sistema que gestione la seguridad de forma centralizada, a este sistema se le denominó Acaxia el cual brindará sus servicios a todos los sistemas que se suscriban a él. Para ello implementa cinco procesos fundamentales que debe cumplir un sistema de este tipo, autenticación, autorización y auditoría e incorpora dos nuevos procesos, administración de perfiles y la administración de conexiones.

El siguiente trabajo se centra en el proceso de autorización, el mismo tiene como valor agregado la gestión centralizada en entornos multientidad y multisistema garantizando la compartimentación de la información. Para el desarrollo de la solución se realizó un estudio de los estándares internacionales más utilizados, estos aumentan la seguridad de los sistemas, la confianza de los usuarios, los niveles de integración con otras soluciones y el nivel de generalización. Con la reutilización del Sistema de Gestión Integral de Seguridad Acaxia se garantiza el cumplimiento de una parte de los requisitos desde el inicio del desarrollo de un proyecto, disminuyendo los costos, el tiempo de respuesta al usuario, el esfuerzo, aumentando la calidad y seguridad de las aplicaciones. El sistema incorpora las mejores prácticas de los sistemas de seguridad más utilizados en Cuba y el mundo, además está integrado a un sistema de estructura y composición que permite la gestión de estructuras dinámicamente, e incluye soluciones a escenarios no resueltos hasta el momento en sistemas de este tipo. Por las ventajas que brinda se reutiliza en más de veinte proyectos de la UCI (Universidad de las Ciencias Informáticas) y quince entidades desarrolladoras de software del país.

ÍNDICE DE CONTENIDOS

INTRODUCCIÓN

A medida que van pasando los años las Tecnologías de la Información y las Comunicaciones (TIC) se van desarrollando y por ende se van perfeccionando. Nuestro país, aunque se encuentra fuertemente bloqueado por el imperialismo, hace todo lo posible para estar a la altura de los países desarrollados en esta rama, llevando a cabo la informatización de todos los sectores del país. A pesar del bloqueo genocida impuesto ya por más de cincuenta años, nuestro país ha sabido buscar nuevas alternativas para contrarrestar las consecuencias que esto trae consigo, una de estas alternativas es la creación de la Universidad de las Ciencias Informáticas (UCI) por iniciativa de nuestro Comandante en Jefe Fidel Castro Ruz. Esta institución tiene como misión insertar a Cuba en el mercado mundial del software aprovechando así la capacidad intelectual e inventiva de los cubanos, produciendo soluciones y servicios informáticos.

Las aplicaciones que se realizan en la UCI requieren de una seguridad estricta y bien concebida que permita controlar y monitorear la información para evitar que las aplicaciones sean atacadas y en caso de que esto ocurra, que el ataque no sea fructífero y se logre atrapar al atacante en el menor tiempo posible. Es por este motivo que los procesos de gestión de seguridad de los sistemas informáticos son muy importantes y constituyen una parte fundamental de los mismos.

Es necesario que los sistemas conciban una política de seguridad adecuada para lograr un buen desempeño de sus objetivos. Con el desarrollo de la arquitectura de seguridad se logra una administración segura y centralizada de todos los sistemas que utilicen sus servicios. De esta forma se disminuye el tiempo de desarrollo de las aplicaciones, el costo total de los proyectos y los riesgos de seguridad.

En la actualidad la mayoría de la información valiosa de las empresas es gestionada por sistemas informáticos. La UCI en conjunto con la Unidad de Compatibilización Integración y Desarrollo de Software para la Defensa (UCID), se encuentran desarrollando una serie de aplicaciones, entre ellas el Sistema de Gestión de Entidades Cedrux del proyecto ERP-Cuba, de vital importancia para el desarrollo del país, en él se manejarán todos los recursos materiales, humanos y financieros para lograr un mejor control, distribución y planificación de los recursos. Por la importancia de la información que manejará este sistema se hizo necesario desarrollar el Sistema de Gestión Integral de Seguridad Acaxia. Este sistema no sólo prestará sus servicios al ERP sino a todos los sistemas informáticos que requieran de los mismos.

En la actualidad existen sistemas que gestionan en menor o mayor medida la seguridad y dentro de ella el proceso de autorización. Entre ellos se encuentran los marcos de trabajos, sistemas *single sign-on* y aplicaciones desarrolladas para dar solución a problemáticas muy específicas que no conciben entornos multientidad y multisistema. El aspecto multientidad está asociado a la identificación única de las entidades en un mismo nodo de despliegue, de manera que posibilite el proceso de compartimentación de la información; que consiste en restringir que cada usuario vea solo la información a la que tiene acceso aunque toda se encuentre en un mismo origen de datos. El aspecto multisistema se centra en que los usuarios puedan cumplir un conjunto de roles comunes en un conjunto de sistemas y dominio de entidades. La *Figura 1* describe con más claridad los conceptos tratados anteriormente. Los bajos niveles de seguridad con los que cuentan las aplicaciones surgen debido a que los proyectos concentran el 90% de sus esfuerzos en dar solución a los requisitos que responden al negocio que dio origen al sistema, dejando en segundo plano la seguridad de la información que se gestiona. La mayoría de las soluciones existentes son muy difíciles de extender o reutilizar porque no

cumplen con estándares internacionales, por esta razón se hace necesario obtener una solución que cubra todos los posibles escenarios en el área de la seguridad en aplicaciones de gestión. Para esta solución se requería utilizar tecnologías y herramientas libres para cumplir con el paradigma de independencia tecnológica.

Figura 1. Entorno multientidad y multisistema.

Luego de realizar un profundo análisis sobre la importancia que tiene contar con un proceso riguroso y confiable de autorización en las aplicación para mantener protegidos los datos se ha planteado el siguiente **problema a resolver:** ¿Cómo garantizar la autorización en entornos multientidad y multisistema?

Para ello tendremos como **objeto de estudio:** La administración de la seguridad en sistemas de gestión.

Teniendo como **campo de acción:** Proceso de autorización en sistemas de gestión.

Para resolver el problema planteado se ha propuesto como **objetivo general**: Desarrollar un componente que gestione de forma centralizada la autorización en entornos multientidad y multisistema.

Para dar cumplimiento al objetivo general se han trazado los siguientes **objetivos específicos:**

- ✓ Estudiar el estado del arte acerca de las principales tecnologías de desarrollo en software libre, principales conceptos definiciones del proceso de autorización y estándares internacionales relacionados con esta temática.
- ✓ Analizar y diseñar un componente que administre la autorización de forma centralizada.
- ✓ Implementar el componente de autorización.
- ✓ Aplicar la solución de autorización en entornos reales.

Para llevar a cabo esta investigación y dar cumplimiento a los objetivos propuestos, se planificaron las siguientes **tareas**:

- ✓ Estudio de la información obtenida sobre la autorización en los marcos de trabajo en PHP y otras tecnologías.
- ✓ Estudio de los estándares internacionales para este tipo de soluciones.
- ✓ Identificar escenarios que debe cubrir la solución.

- ✓ Captura y especificación de requisitos por escenarios.
- ✓ Realizar diagramas de procesos.
- ✓ Realizar el modelo conceptual.
- ✓ Diseñar los prototipos de interfaz de usuario.
- ✓ Diseño del modelo de datos.
- ✓ Implementación de la capa de presentación.
- ✓ Implementación de la capa de negocio.
- ✓ Implementación de la capa de datos.
- ✓ Descripción de los escenarios de despliegues.
- ✓ Conformar casos de pruebas.
- ✓ Validación del componente.

Se tiene como **idea a defender** de la presente investigación:

Si se desarrolla el sistema de autorización de Acaxia, se logrará centralizar la seguridad en entornos multientidad y multisistema.

- ✓ V.I. Desarrollo del sistema de autorización.
- ✓ V.D. Lograr la seguridad en entornos multientidad y multisistema.

Aporte teórico y práctico
Como resultado del trabajo se obtiene una solución de autorización bajo los principios de independencia tecnológica que cumpla con estándares internacionales y sea capaz de gestionar la seguridad en entornos multientidad y multisistema. La aplicación de la solución a proyectos de la UCI y otras entidades del país, garantiza un alto nivel de seguridad de las soluciones que se desarrollen con un ahorro de recursos y tiempo.
Capítulo 1 Estado del arte: Fundamentación teórica del trabajo, la misma incluye conceptos principales de seguridad específicamente el proceso de autorización. También se abordan estudios realizados a otras soluciones existentes en el mundo. Se tratan temas relacionados con el panorama de las tecnologías y herramientas libres usadas para el desarrollo de la solución así como los elementos de reutilización y resultados esperados de la solución propuesta.

Capítulo 2 Solución: Se describe con detalle el problema presentado y la vía de solución utilizada.

Capítulo 3 Validación: Se muestran los resultados de las pruebas realizadas al componente, la aceptación e impacto en los proyectos en los cuales se ha aplicado.

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS SOBRE SISTEMAS DE GESTIÓN DE SEGURIDAD

La gestión centralizada de la autorización tiene suma importancia para lograr un mayor control y seguridad de la información. Dicho proceso actualmente no se implementa de forma adecuada en los sistemas desarrollados en la UCI y en Cuba, por lo que este trabajo está enmarcado en el desarrollo de un componente que gestione la autorización de forma centralizada en un entorno de varias aplicaciones. En este capítulo se describe el marco conceptual que existe alrededor de la autorización en sistemas de gestión, las tecnologías y herramientas propuestas, las técnicas y los estándares existentes que permiten el desarrollo de la solución.

1.1 CONCEPTOS Y ESTÁNDARES

1.1.1. SEGURIDAD EN APLICACIONES WEB

El término seguridad informática es una generalización para un conjunto de tecnologías que ejecutan ciertas tareas relativas a la seguridad de los datos.
ISO, en su norma 7498, define la seguridad informática como una serie de mecanismos que minimizan la vulnerabilidad de bienes y recursos, donde un bien se define como algo de valor y la vulnerabilidad se define como la debilidad que se puede explotar para violar un sistema o la información que contiene. El bien más preciado por cualquier institución es la información y de ahí que se han desarrollado protocolos y mecanismos adecuados, para preservar su seguridad.
Se puede hablar en este sentido de cinco conceptos principales de la seguridad de los sistemas: autentificación, autorización, auditoría, administración de perfiles y administración de conexiones. Basándonos en estos conceptos a la hora de implementar la seguridad se lograría cumplir con la confidencialidad, integridad y el no-repudio aspectos fundamentales para cualquier sistema que gestione información. [1, 2]

A la hora de desarrollar una aplicación, generalmente nos centramos más en la funcionalidad que en la seguridad. Lo que trae como consecuencia que los atacantes se aprovechen de esto y atenten contra cualquiera de estos cuatro aspectos. En la seguridad de aplicaciones los procesos de autenticación y autorización juegan un rol fundamental, ya que permiten un mejor control en el acceso a la información.

1.1.2. AUTORIZACIÓN

Es la parte del proceso que protege los recursos del sistema permitiendo que sólo sean usados por aquellos consumidores a los que se les ha concedido autorización para ello. Los recursos incluyen archivos y otros objetos de datos, programas, dispositivos y funcionalidades provistas por aplicaciones.

1.1.3. MODELO DE CONTROL DE ACCESO BASADO EN ROLES RBAC

El concepto básico de RBAC es que los usuarios son asignados a roles, los permisos son asociados a roles y los usuarios adquieren permisos siendo miembros de roles. Las asignaciones usuario-rol y permiso-rol pueden ser muchos-a-muchos, por lo que un usuario puede pertenecer a muchos roles y un rol puede poseer muchos usuarios y de manera similar un permiso puede ser asociado a muchos roles y un rol puede tener asociado muchos permisos. RBAC también incluye el concepto de sesión, que permite la

activación y desactivación selectiva de roles, posibilitando que un usuario pueda ejercer los permisos de varios roles simultáneamente.[3]
Para que un usuario pueda ejercer los permisos para los cuales está autorizado, es necesario que este active los roles que tienen asociados estos permisos. Define un conjunto de sesiones (SESSIONS) donde cada sesión es un mapeo entre el usuario y un subconjunto de roles activos, los cuales están asignados al usuario. La utilización de sesiones facilita en gran medida la aplicación de este principio, ya que aumenta la granularidad del control de acceso.
Las decisiones de acceso son tomadas por usuario y sesión, es decir un usuario está autorizado a realizar una determinada operación sobre un determinado objeto si existe algún rol activo en dicha sesión que tenga asignado el permiso correspondiente.[4]

Sesión

Las sesiones de usuario son estructuras de datos que almacenan información referente a cada uno de los usuarios, en su mayoría sensible. La misma identifica al usuario en cada acción que ejecute sobre el sistema. Si la sesión es interceptada por algún atacante, el mismo puede ocasionar pérdidas irreparables para la institución.
Los permisos asignados a un usuario a través de los roles, solo pueden ser utilizados si estos últimos son activados. Cada usuario establece una sesión durante la cual activa algún subconjunto de los roles para los que está autorizado.
Cada sesión es un mapeo entre un usuario y sus roles activos, un usuario puede tener más de una sesión establecida de manera simultánea.
El estado de las sesiones del sistema cuentan con un conjunto de identificadores válidos de sesión llamado Session[1], existen dos formas de separar las funciones o funcionalidades de un rol, estas son:

- ✓ **SSD: Separación de deberes estáticos**

Esta política define que un usuario solo puede asumir un rol.
Con el uso de las restricciones en los roles, si el rol (padre) tiene una relación SSD con un recurso, esto implica que los hijos de este también presentan una relación SSD con ese recurso.

- ✓ **DSD: Separación de deberes dinámicos**

Esta política define que un usuario puede ser miembro de varios roles, mientras no constituya un conflicto de intereses cuando se activen independientes.[4]

Niveles de RBAC

El estándar propone cuatro niveles de seguridad y cada uno con sus restricciones, a continuación se describe cada uno de ellos.

- ✓ Nivel 0 Básico.
- ✓ Nivel 1 Jerarquía.
- ✓ Nivel 2 Restricciones.
- ✓ Nivel 3 .[5, 6].

Nivel 0 “Básico”
La Figura 2 muestra el nivel fundamental de RBAC, donde tenemos los usuarios, roles y los permisos, con sus relaciones.
Existen escenarios en los que un usuario debe cumplir un conjunto de funciones que pueden encontrarse en especialidades distintas. Estas funciones se agrupan por especialidad o roles del sistema y se le asigna permiso al usuario sobre ellas. Para cumplir con dichos requerimientos, las relaciones quedarían de la siguiente forma:

[1] (**Session).** Almacena un conjunto de datos que identifican y diferencian a los usuarios.

Usuario a Rol: mucho a mucho (viceversa).
Rol a Permisos: mucho a mucho (viceversa).

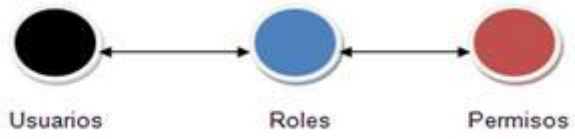

Figura 2. Nivel 1 de RBAC

Nivel 1 “Jerarquía”
La Figura 3. Nivel 1 “Jerarquía”. muestra la jerarquía entre los roles, es decir que un rol puede tener dentro de él un subconjunto de roles y están presentes los del Nivel 0.

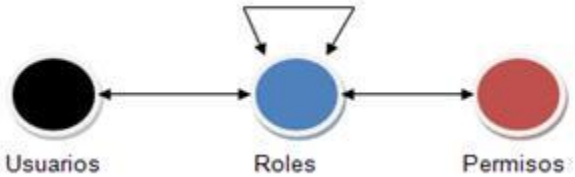

Figura 3. Nivel 1 “Jerarquía”.

Nivel 2 “Restricciones”
La Figura 4 muestra las características de los niveles 0 y 1. En él se establece restricciones a las relaciones y a la jerarquía.

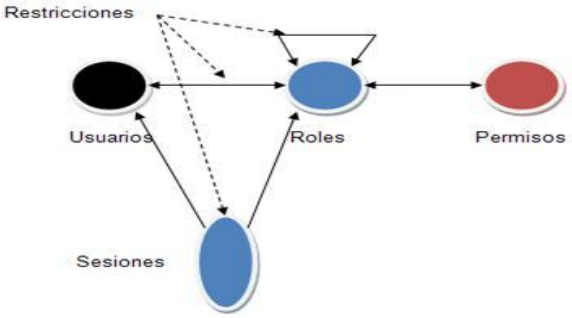

Figura 4. Nivel 2 “Restricciones”.

Esta representación sería Restricciones-SOD[2] Dinámico, para el caso de Restricciones-SOD Estático no aparecerían las sesiones.
Nivel 3 “Simétrico”
Quedaría con las características de los demás niveles, además de la restricción a la relación entre los roles y los permisos. Ver en la Figura 5 la estructura del estándar RBAC:

[2] **(SOD).** Separación de cargas donde cada función tiene que ser llevada por usuarios diferentes.

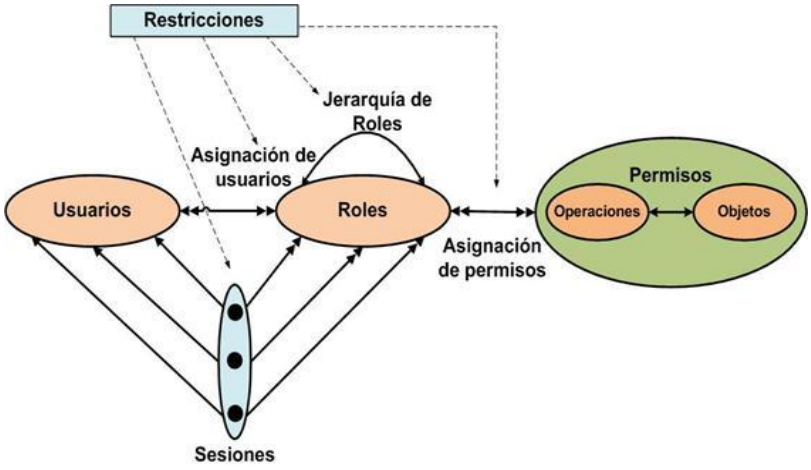

Figura 5. Estándar RBAC.

Simétrico -SOD Dinámico, para Simétrico-SOD estático, no aparecerá las sesiones. Se usará este estándar para la autorización del sistema, ya que este es muy eficiente a la hora del trabajo con roles, usuarios y permisos. Aplicando SOD separación de deberes dinámicos para asignar usuarios a roles determinados.

La figura 6 muestra la primera versión en el punto de autorización:

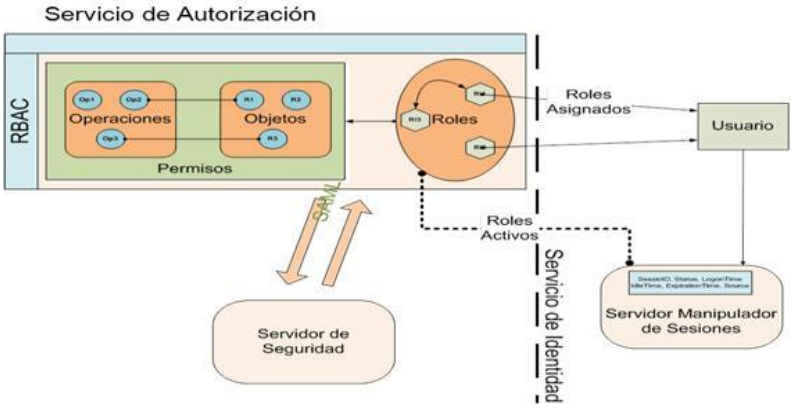

Figura 6. Autorización, primera versión.

El estudio de las formas de autorización que propone el estándar RBAC arrojó como resultado que el mismo tiene bien definido la creación de roles con permisos a los recursos, pero no incorpora el concepto entidad en el que un usuario puede tener diferentes roles, solo puede tener un rol en una entidad, se necesita incluir un concepto: entidad, la figura 7 muestra la versión final a implementar para cubrir los escenarios que requiere la solución. La entidad está relacionada con los roles, donde ella establece una relación de mucho a mucho con el rol.

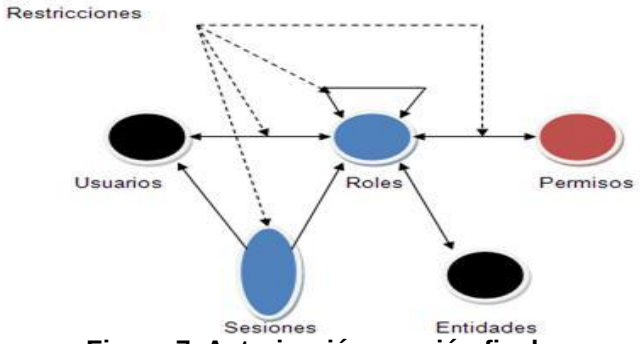

Figura 7. Autorización, versión final

1.2. SEGURIDAD DE LOS PRINCIPALES MARCOS DE TRABAJO

En la actualidad al iniciar el desarrollo de un sistema se hace un estudio del estado de las principales soluciones que existen, con el objetivo de solucionar la problemática existente. Los frameworks o marcos de trabajos integran un conjunto de componentes que resuelven muchos de los escenarios que pueden encontrarse en el transcurso de un desarrollo, estos permite agilizar el proceso y obtener resultados en breves períodos de tiempos. Como son soluciones muy genéricas, en la mayoría de los casos no abarcan todos los posibles contextos en el que se va a desempeñar una aplicación. Con la necesidad existente en nuestro país de contar con sistema ERP que mejore y agilice la gestión de los recurso, surge el proyecto ERP-Cuba perteneciente al Centro Informatización para la Gestión de Entidades CEIGE, por la premura de la tarea se necesita dar una respuesta en un período inmediato. En este caso es imprescindible el estudio de soluciones homólogas de autorización que puedan ser reutilizadas y de esta forma disminuir el tiempo de desarrollo, el esfuerzo y los costos. En el mundo existen un conjunto de marcos de trabajo que proveen un conjunto de componentes reutilizables, dentro de ellos se estudiarán los más utilizados desarrollados sobre tecnologías libres para evaluar si es factible reutilizar alguno de ellos y de esta forma dar una respuesta en un período de tiempo menor.

1.2.1. SYMFONY

Symfony es un completo marco de trabajo diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. [7]

Symfony cuenta con muchos plugin[3], un grupo de ellos es utilizado para gestionar la *seguridad* de las aplicaciones, dentro de este grupo el más usado, seguro y eficaz es: *sfGuardPlugin.*

[3]Un **complemento** (o **plug-in** en inglés) Pequeño programa que añade alguna función a otro programa, habitualmente de mayor tamaño. Un programa puede tener uno o más conectores. Son muy utilizados en los programas navegadores para ampliar sus funcionalidades.
.

1.2.2. SFGUARDPLUGIN

En toda aplicación dinámica, siempre está presente la gestión de usuarios, permisos y roles. En symfony esto puede resultar muy sencillo gracias a SfGuard. Este plugin gestiona de manera muy sencilla usuarios, roles, permisos y logins.
Básicamente, la autenticación y la seguridad en una aplicación es limitar el acceso a partes de la misma. Significa que los usuarios tendrán que iniciar una sesión (autenticación) para acceder a ciertas áreas (seguridad). Diferentes usuarios pueden tener diferentes privilegios (autorización). De esta manera aseguras tu aplicación para diferentes tipos de usuarios. SfGuardPlugin te brinda el modelo (usuario, grupo y objetos de permisos). En la Figura 7 se muestra la arquitectura de seguridad de este marco de trabajo.[7]
Está compuesto por cuatro módulos:

- ✓ El módulo *sfGuardAuth* es el encargado del login y el acceso restringido.
- ✓ El módulo *sfGuardGroup* es el encargado de la gestión de grupos o roles.
- ✓ El módulo *sfGuardPermission* es el encargado de la gestión de permisos.
- ✓ El módulo *sfGuardUser* es el encargado de la gestión de usuarios.

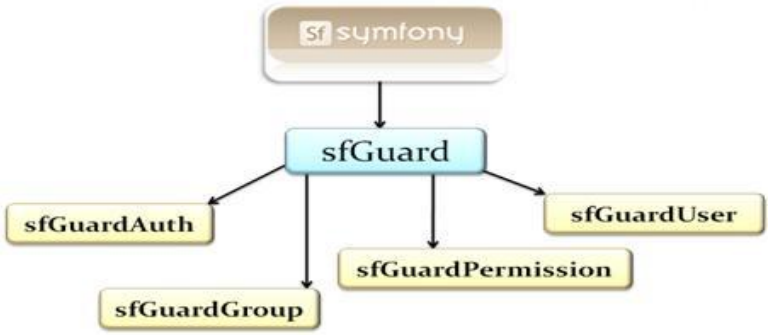

Figura 8. sfGuardPlugin.

Como se muestra en la figura anterior symfony cuenta con un componente encargado de la seguridad y dentro de ella la autorización. Este componente se enmarca fundamentalmente en roles o grupos que se le asignan un conjunto de permisos y estos roles son asignados a los usuarios, trayendo consigo que no se puedan gestionar un conjunto de privilegios y configuraciones necesarias en un entorno complejo como en el que se va a desempeñar Cedrux. Las configuraciones de los diferentes conceptos se realizan a través de xml lo que hace más engorroso el trabajo de los administradores. El estudio realizado demostró que no incorpora conceptos necesarios como la multientidad, multisistema, administración de perfiles, administración de conexiones y solo brinda seguridad a las aplicaciones desarrolladas sobre esta arquitectura.

1.2.3. ZEND FRAMEWORK

En su nivel más simple, ZendFramework es una librería de componentes escritos en PHP5, para facilitar el desarrollo de sitios web. Como está basada en PHP5 (5.1.4 es la versión mínima necesaria), eso significa que es completamente Orientada a Objetos.

Zend está formado por muchos componentes, estos están divididos en grupos, uno de estos grupos se encarga de gestionar la seguridad, ver Figura 8, los componentes de este grupo son:

Zend_Auth permite chequear y guardar credenciales de usuario de distintas maneras: utilizando la Base de Datos, el método Diguest de Apache, o autenticación http simple. Provee un Adapter de Interfaz para personalizar los mecanismos de autenticación, además del almacenamiento automático de identidad para una fácil personalización. Es simple y extensible.

A su vez **Zend_Session** trabaja como un administrador de datos de sesión, al igual que en PHP**,** solo que ofrece algo de valor agregado.

El componente **Zend_ACL** es una implementación de Listas de control de acceso (ACL)[4]en PHP que nos permite asignar roles y permisos a usuarios o grupos de usuarios en la aplicación. Soporta herencia de roles y recursos. También soporta el control de acceso condicional basado en una interfaz de declaraciones.

Finalmente el componente **Zend_Log** permite realizar un log de acciones en diferentes medios como la consola de PHP*,* archivos y base de datos mediante el anexo de varios ¨writers¨.[8]

La clase maneja diferentes niveles de criticidad:

- ✓ EMERG = 0; //Emergencia: el sistema está fuera de línea.
- ✓ ALERT = 1; //Alerta: acción debe ser atendida.
- ✓ CRIT = 2; //Crítica: condición crítica.
- ✓ ERR = 3; //Error: condición de error.
- ✓ WARN = 4; //Advertencia: condición de advertencia.
- ✓ NOTICE = 5; //Notificación: normal pero con una nota.
- ✓ INFO = 6; //Información: mensajes de información.
- ✓ DEBUG = 7; //Depuración: mensajes de depuración.

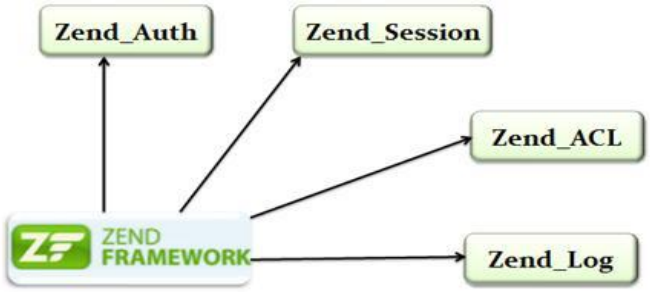

Figura 9. Seguridad en ZendFramework.

Zend es uno de los marcos de trabajos más completos, entendibles y fáciles de utilizar. Cuanta con una gran gama de componentes que facilitan el desarrollo de software. La seguridad implementada por Zend se basa fundamentalmente en lista de control de acceso y restricciones que no garantiza la protección de la información que gestiona un ERP como Cedrux, a diferencia de los demás marcos de trabajos implementa una potente solución de administración de log, de caché y sesiones de usuarios. Por esta razón se decide reutilizar solo los componentes Zend_Session, Zend_Log y Zend_Cache.

[4]La **Lista de Control de Acceso** o **ACL** (del inglés, **Access Control List**) es un concepto de seguridad informática usado para fomentar la separación de privilegios. Es una forma de determinar los permisos de acceso apropiados a un determinado objeto, dependiendo de ciertos aspectos del proceso que hace el pedido.

1.2.4. CODEIGNITER

CodeIgniter es un conjunto de herramientas para personas que construyen su aplicación web usando PHP. El objetivo de este marco de trabajo es permitirles a los usuarios desarrollar aplicaciones con mayor rapidez. Esto es posible porque provee una amplia gama de componentes reutilizables, evitando que se tenga que reimplementar componentes comunes en todas las aplicaciones de gestión. Se caracteriza por su sencillez

Es justamente restrictivo sobre las cadenas URI para ayudar a minimizar la posibilidad de ataque utilizando esta vía.

La figura 10 muestra la arquitectura de seguridad que presenta este marco de trabajo. La misma solo implementa de forma muy sencilla un conjunto de restricciones, validaciones y filtros que no garantizan la seguridad en un entorno que requiera de una seguridad estricta. Las configuraciones se realizan en formato xml, el esfuerzo para incorporar robustez es muy alto por lo que no es factible reutilizar este marco de trabajo.

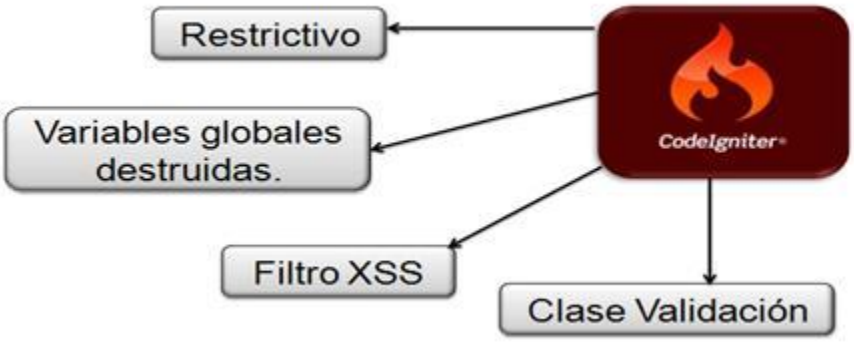

Figura 10. Seguridad en CodeIgniter.

1.2.5. KUMBIAPHP

KumbiaPHP es un framework para aplicaciones web de gestión, escrito en PHP5. Basado en las prácticas de desarrollo web como ***DRY***[5] y el principio ***KISS***[6] para software comercial y educativo. Kumbia fomenta la velocidad y eficiencia en la creación y mantenimiento de aplicaciones web, reemplazando tareas de codificación repetitivas por poder, control y placer. [9]

La implementación de la seguridad usando KumbiaPHP se lleva a cabo con el uso de las listas de control de acceso *(ACL)* como se muestra en la figura 11.

[5]El principio **No te repitas** (en inglés **Don'tRepeatYourself** o **DRY**, también conocido como **Una vez y sólo una**) es una filosofía de definición de procesos que promueve la reducción de la duplicación especialmente en informática.

[6] El principio **Manténgalo Suave** y **Breve** (en inglés, **KeepIt Short and Simple** o **KISS**) es aquel que recomienda el desarrollo empleando partes sencillas, comprensibles y con errores de fácil detección y corrección, rechazando lo enrevesado e innecesario en el desarrollo de sistemas complejos en ingeniería.

1.2.6. CAKEPHP

CakePHP es un framework para aplicaciones web escrito en PHP que provee una extensa arquitectura para el desarrollo, mantenimiento y despliegue de aplicaciones. CakePHP reduce los costos de desarrollo y ayuda a los desarrolladores a escribir menos código.

Cuenta con un módulo de autenticación de usuario único llamado 'Access Lists', ver figura 11, que se puede utilizar para dar acceso a los diferentes usuarios de diferentes partes de su sitio web con CakePHP. También cuenta con un componente de seguridad para proteger la aplicación de ataques de tipo Cross Site Request Forgery[7] (CSRF).

En el ecosistema de software de las empresas de hoy en día, la seguridad de acceso a las aplicaciones de la organización constituye un creciente desafío. Implementar y poner en marcha un modelo de seguridad de acceso implica navegar múltiples aplicaciones, en la mayoría de los casos, con muy pocas características comunes entre ellas.[10]

Los marcos de trabajos KumbiaPHP y KakePHP solo implementan listas de control de acceso que no garantizan seguridad a las aplicaciones. Configurar el acceso de los usuarios se torna extremadamente complejo ya que es necesario especificar para cada usuario los permisos que tiene sobre cada recurso sin poder agruparlos por roles. Están diseñados fundamentalmente para desarrolladores inexpertos, sistemas pequeños y de poco impacto. Por las razones expuestas no es factible utilizar ninguno de estos marcos de trabajos ya que no aportan seguridad a los sistemas que se desarrollen en el centro.

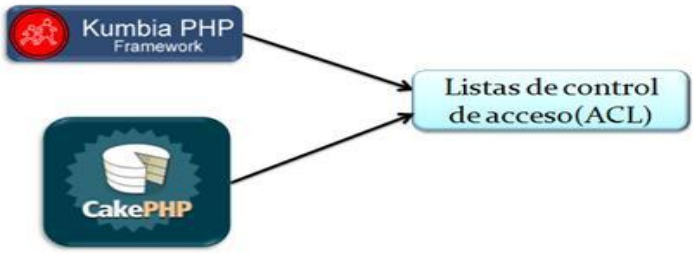

Figura 11. Seguridad en Kumbia PHP y CakePHP.

1.2.7. ACEGI SECURITY

Acegi Security es un framework Java que proporciona mecanismos de seguridad de forma declarativa para aplicaciones empresariales basadas en Spring Framework utilizando características de AOP[8] y de Inyección de Dependencias, de forma transparente para el desarrollador, sin necesidad de agregar código. Ofrece una solución completa de

[7] El **CSRF** (del inglés ***Cross-site request forgery*** o falsificación de petición en sitios cruzados) es un tipo de exploit malicioso de un sitio web en el que comandos no autorizados son transmitidos por un usuario en el cual el sitio web confía. Esta vulnerabilidad es conocida también por otros nombres como XSRF, enlace hostil, ataque de un click, cabalgamiento de sesión, y ataque automático.

[8] **AOP** siglas de Aspectorientedprogramming (Programación orientada a aspectos).Es un paradigma de programación relativamente reciente cuya intención es permitir una adecuada modularización de las aplicaciones y posibilitar una mejor separación de conceptos.

seguridad, manipulación de autenticación y autorización, tanto al nivel de peticiones en la web como al nivel de invocación de métodos. También es posible utilizarlo en aplicaciones no desarrolladas con Spring. [11]

Características principales

Acegi Security en su conjunto se puede agrupar en tres conceptos fundamentales:

Autenticación: determina si un usuario es quien dice ser.

Autorización: determina lo que el usuario está autorizado a hacer si él es realmente esa persona.

Canal de seguridad: se asegura de que nadie pueda escuchar la conversación.

Acegi Security soporta la mayoría de los procesos de autenticación existentes, además de permitir adicionar nuevos mecanismos. Es de código abierto y ofrece una seguridad no invasiva debido a que su implementación puede ser declarada sin hacer cambios dentro del código fuente de la aplicación mediante AOP. [12]

Ofrece su propio enfoque de seguridad, pero también posibilita implementaciones alternativas. Proporciona un mecanismo sencillo que le permite integrarse con códigos no estándares existentes, así como una serie de implementaciones para conectarse a mecanismos de autenticación más comunes, como LDAP[9].

Es capaz de garantizar la seguridad de las aplicaciones sin tener que escribir ningún código directamente en la lógica de la aplicación.

Gran parte de la configuración referente a la seguridad de una aplicación es totalmente independiente de los elementos que requieren asegurarse. El único componente de Acegi Security que realmente necesita saber detalles acerca del funcionamiento de la aplicación es la definición del objeto donde se asocian los recursos asegurados con las autoridades que requieren acceso a dichos recursos. [12]

Entre las características más significativas de este framework se encuentran:

Seguridad en instancias de objetos del dominio: en muchas aplicaciones es deseable definir listas de control de acceso (Access Control List, ACL[10]por sus siglas en inglés) para instancias de un objeto de dominio. Ofrece un amplio paquete ACL con características que incluyen permisos heredados (incluido el bloqueo), respaldados por un repositorio ACL basado en JDBC, caché, puntos de extensión, y un diseño basado en interfaces.

Protege peticiones HTTP: a través de filtros web garantiza el acceso a las distintas URLs y mediante rutas de Apache o de expresiones regulares realiza la autorización a los recursos solicitados.

Canal de seguridad: puede redirigir automáticamente las peticiones a través de un canal de transporte apropiado. Una característica común de un canal de seguridad es garantizar

[9] **(LDAP)**Acrónimo de Light weight Directory Access Protocol, (Protocolo Ligero de Acceso a Directorios).

[10] **(ACL).** Acrónimo de Access Control List (Lista de Control de Acceso).

que las páginas seguras sólo estén disponibles a través de HTTPS, y las públicas mediante HTTP.

Varias formas de autenticación: incluye la capacidad de recuperar el nombre de usuario y la definición de los privilegios de un archivo XML, fuente de datos JDBC o desde un fichero de propiedades.

Almacenamiento en caché: se integra con la fábrica de Cache de Spring. Esta flexibilidad significa que la base de datos (u otro tipo de repositorio de autenticación) no se encuesta varias veces solicitando la información de autenticación.

Sesiones concurrentes: posibilita configurar el número de inicios de sesión simultáneos que puede realizar un usuario.

Soporte para remoting: se integra con los protocolos remotos estándares de Spring, procesando automáticamente la autenticación básica HTTP que presentan las cabeceras de los mensajes.

Pruebas de unidad: brinda mecanismos de alto nivel para realizar pruebas de unidad a los objetos de negocio asegurados. Se puede cambiar la identidad autenticada y sus autoridades asociadas directamente dentro del método de prueba.

Componentes de Acegi Security

Como se muestra en la figura 12, la arquitectura de Acegi Security está fuertemente basada en interfaces y patrones de diseño, proporcionando las implementaciones más comúnmente utilizadas y numerosos puntos de extensión donde nuevas funcionalidades pueden ser añadidas. Emplea cinco componentes fundamentales para mantener la seguridad.

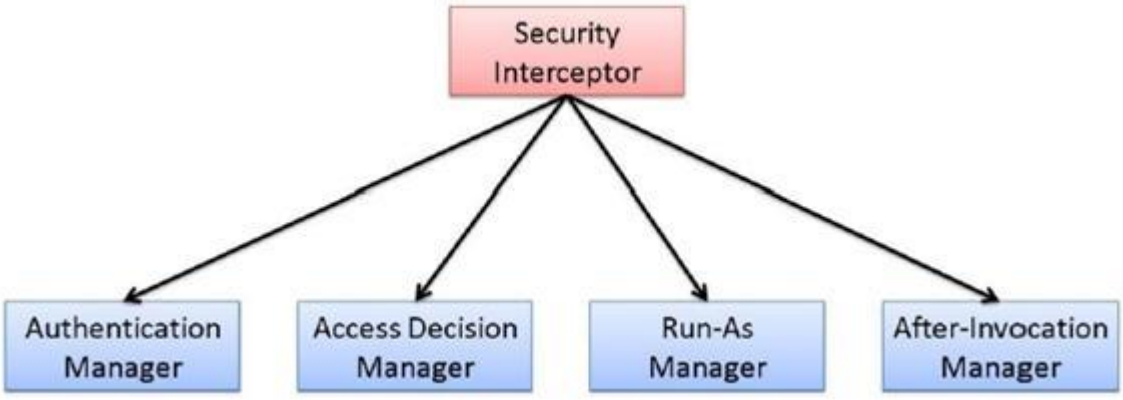

Figura 12. Acegi Security.

Es uno de los frameworks de seguridad más completos, cuenta con varios mecanismos que permiten la extensión y reutilización. Aunque brinda un grupo mayor de funcionalidades que posibilitan diferentes formas de autenticación y autorización y soluciona escenarios más específicos que aumentan los niveles de seguridad, al igual que los demás frameworks no contempla un grupo de requisitos indispensables como la auditoria de diferentes tipos de eventos, seguridad en entornos multientidad y

multisistema y no garantiza la compartimentación de la información ya que sus funciones se limitan a la seguridad a nivel de sistema no de datos. CEIGE desarrolla sus aplicaciones sobre tecnologías PHP y Acegi está implementada en Java, lo que reduce la integración solo a nivel de servicios web. Este tipo de integración para el flujo de información y concurrencia de usuarios que tendrá un sistema como Cedrux, atenta contra el rendimiento y la seguridad. Una vez concluido el análisis crítico y profundo de la solución se determinó no utilizar este sistema puesto que no cubre parte de los principales escenarios de seguridad en aplicaciones web de gestión. Estar implementada sobre una tecnología diferente a la utilizada en el centro traería dificultades en temas de integración, extensión y mantenimiento.

1.2.8. SISTEMAS SINGLE SIGN-ON

La norma hasta el momento ha sido que cada aplicación dentro de una empresa o corporación cuenta con un adecuado sistema de control de accesos y su propio repositorio de datos. En la actualidad dentro del ambiente de tecnología de cualquier empresa o corporación existen diversos métodos de autenticación y autorización de accesos que generan una gran ineficiencia. Sin embargo, con la implementación de un agente Single Sign-On (SSO) el sistema se encarga de almacenar, en una base datos o directorios protegidos, las credenciales que permiten al usuario acceder a cada una de las aplicaciones o servicios en el momento que lo desee, ya que el proceso de autenticación se realiza de manera transparente para el usuario, una vez que éste ha sido autenticado por medio de la arquitectura SSO. Se puede decir que con dicha implementación el sistema simplificaría y centralizaría el control de accesos a todas las aplicaciones de la empresa o corporación, reduciendo el costo en la administración de seguridad, lo que logra un mejor rendimiento y velocidad de los procesos de autenticación y acceso que facilita a los usuarios la interacción con los sistemas de la empresa, simplificando el manejo de claves y lo fundamental es que aumenta los niveles de seguridad, ya que contará con una plataforma central para el manejo de la seguridad en todos los procesos de autenticación y acceso a sus aplicaciones. [13]

El agente SSO se refiere al acceso a múltiples recursos por medio de un único proceso de ingreso. Gran cantidad de las arquitecturas implementadas en diferentes organizaciones han sido diseñadas con el objeto de dar acceso a los usuarios a múltiples servicios Web y aplicaciones. En la mayoría de los casos se encuentra que cada uno de los servicios o aplicaciones cuenta con su propio componente de seguridad, el cual generalmente compromete la seguridad de todo el sistema, dado el nivel de seguridad del componente más débil, el cual determina el nivel de confianza del sistema en su conjunto.[13, 14]

Existen cinco tipos principales de sistemas SSO, también conocidos como Reduced Sign-On Systems (Sistemas de Autenticación Reducida). Los cuales son:

- ✓ **Enterprise Single Sign-On (E-SSO)**: también llamado Legacy Single Sign-On, el cual funciona luego de una autenticación primaria, interceptando los requerimientos de autenticación presentados por las aplicaciones secundarias para completarlos con el usuario y la contraseña. Los sistemas E-SSO permiten interactuar con sistemas que pueden deshabilitar la presentación de la pantalla de login. [15]

- ✓ **Web Single Sign-On (Web-SSO)**: conocido como Web Access Management (Web-AM), trabaja sólo con aplicaciones y recursos que se acceden vía Web. Los accesos son interceptados con la ayuda de un servidor Proxy o de un componente instalado en el servidor Web destino. Los usuarios no autenticados que tratan de acceder son redirigidos a un servidor de autenticación y regresan sólo después de haber logrado un acceso exitoso. Se utilizan cookies, para reconocer aquellos usuarios que acceden y su estado de autenticación. [15]
- ✓ **Kerberos**: es un método popular de externalizar la autenticación de los usuarios. Los usuarios se registran en el servidor Kerberos y reciben un ticket, que luego utilizan para obtener acceso. [15]
- ✓ **Federation**: es una nueva manera de concebir este tema, también para aplicaciones web. Utiliza protocolos basados en estándares para habilitar que las aplicaciones puedan identificar los clientes sin necesidad de autenticación redundante.[15]
- ✓ **OpenID**: es un proceso de SSO distribuido y descentralizado donde la identidad se compila en una URL de forma que cualquier aplicación o servidor puede verificar.[15]

1.2.9. SISTEMAS DE SEGURIDAD UTILIZADOS EN MUNDO

Actualmente existen soluciones para el control de la seguridad de varias aplicaciones de manera centralizada, o sea, en un entorno de varias aplicaciones controlarlas a todas de igual forma, pero hay que destacar que estas propuestas son incipientes todavía por lo que en muchas ocasiones son miradas con recelos por los clientes. Una de las soluciones que proponen una seguridad centralizada es: AccessMaster IAM (Gestión de Identidades y Acceso) & SSO (Single Sign-On).[16]

Los sistemas de información corporativos incluyen un número creciente de aplicaciones heterogéneas y recursos alojados en diversos sistemas abiertos. Mientras esta variedad y heterogeneidad facilita los procesos de negocio, también constituyen un problema desde el punto de vista de la gestión de la seguridad. Se plantea la dificultad de definir e implantar una política de seguridad única, que sea aplicable a todos esos recursos y aplicaciones. El principio de las soluciones de Gestión de Identidades y Accesos (AccessMaster IAM) es el poder establecer, mediante políticas de control de acceso, qué usuarios pueden acceder a que aplicaciones y recursos, de manera que un usuario no pueda usar aplicaciones o entrar en recursos para los que no está autorizado. Otro valor añadido de esta solución es la gestión centralizada y segura de usuarios y contraseñas para los distintos servicios de la organización. El Single Sign-On (SSO) supone efectuar en lugar del usuario la operación de identificarse mediante login y password frente a las aplicaciones y recursos corporativos. Los usuarios se autentican una única vez, contra el sistema de IAM y SSO, y después este sistema se encarga de forma transparente de las autenticaciones subsiguientes en su lugar, según se van produciendo los accesos correspondientes. [16]

Este tipo de soluciones garantizan que un usuario autenticándose una vez pueda acceder a las demás aplicaciones a las que tenga acceso. Es un tipo de soluciones específicas que solo garantizan el proceso de autenticación, dejando al descubierto procesos tan importantes como la autorización, auditoría, administración de perfiles y administración de conexiones. Por esta razón se decidió solo reutilizar la filosofía que implementa para resolver el SSO.[16]

1.2.10. SISTEMAS SEGURIDAD UTILIZADOS EN CUBA Y EN LA UCI

La investigación en la UCI sobre los sistemas de gestión de seguridad evidenció la existencia del Sistema de Gestión de Sesiones basado en la arquitectura SSO cuyo objetivo es gestionar las sesiones de los usuarios en un único proceso de autenticación invisible a los ojos de los mismos, a través de un sistema con una fachada de servicios web que sea capaz de gestionar la apertura y cierre de sesiones por parte de las personas que trabajan en el dominio uci.cu.

Además en la facultad 10 se llevó a cabo el desarrollo de un software de autenticación y control centralizado para la corporación de PDVSA capaz de mapear las credenciales de los usuarios hacia todas las aplicaciones de dicho sistema y otros sistemas heterogéneos, proporcionando una infraestructura para simular un login único para el usuario corporativo frente a una plataforma tecnológica heterogénea dentro del ambiente de aplicativos de integración, en que los usuarios puedan acceder a diferentes aplicaciones con solo un conjunto de credenciales. Está aplicación trajo como resultado principal la importancia de mapeos entre diferentes entornos que manejan variados mecanismos de seguridad.

En la UCI existen algunas soluciones que incorporan un porciento alto de los escenarios propuestos por los estándares para sistemas de seguridad. Entre las soluciones estudiadas se encuentran las implementadas por el proyecto de salud de la facultad 7, por la UCID, por identidad de la facultad 1, por la facultad 10 y por SOFTEL. En el caso de la solución de la facultad 1 a pesar de contar con un conjunto grande de funcionalidades, se decidió no utilizarla porque está implementada sobre tecnologías propietarias y su campo de acción está encaminado a las aplicaciones de escritorio. La propuesta de la facultad 7 está implementada sobre tecnología Java, este aspecto es muy importante porque la integración con este sistema se reduce solo a nivel de servicios web atentando contra el rendimiento de Cedrux. Otro factor por el que no se decide su reutilización es porque está solución no es un componente o módulo independiente del sistema de salud. Las aplicaciones de la UCID, la facultad 10 y SOFTEL son soluciones hechas a la medida que solo incorporan los requerimientos particulares de cada uno de los usuarios y en ninguno de los casos incorporan el multisistema, el multiidioma, el multitema, el multiescritorio la multientidad, la administración de conexiones ni la auditoría de sistema. Se puede concluir que dichas soluciones no cubren los escenarios identificados por el departamento de tecnología, por las razones expuestas no es recomendable reutilizar ninguno de estos componentes.

1.2.11. VALORACIÓN DE LAS SOLUCIONES ESTUDIADAS

Los componentes de seguridad estudiados presentan soluciones particulares para escenarios muy específicos, no tienen en cuenta que en el mundo se desarrollan sistemas tan complejos como los ERP que necesitan restricciones desde niveles o agrupaciones de información muy pequeñas hasta niveles estructurales. Los más avanzados incorporan los conceptos de roles, objetos o sistemas, ningún caso incluye los entornos multientidad y multisistema, manteniendo la compartimentación de la información hasta los niveles más básicos. Se basan fundamentalmente en reglas o flujos que en su mayoría hay que configurarlas de forma manual, a medida que aumenta el nivel de restricciones, aumenta la complejidad de las configuraciones. Este proceso se torna tedioso y complejo para administradores que gestionen la seguridad de varias aplicaciones ya que si un usuario tiene acceso a varias aplicaciones tendría que repetir la configuración en cada uno de estos. Los sistemas de autorización estudiados desarrollados fuera del país son muy

costosos ejemplo de ello es la plataforma v-Go Single ON, producida por la compañía de Passlogix, la distribución de esta plataforma en conjunto con IBM tiene un costo de 75 USD por usuario que incluye un valor de licencia única y su mantenimiento por un año, si se compra su versión multilingüe y multiplataforma que incluye el CD-ROM de instalación costará entonces 140 USD por usuario y los servicios de instalación y configuración tendrán un valor de 3.60 USD por usuario. Nuestro país con la situación existente no puede dedicar recurso para adquirir productos de este tipo.
Concluido el análisis se llegó a la conclusión que se hace necesario desarrollar un componente que gestione la seguridad y dentro de ella el proceso de autorización de forma centralizada para que pueda ser reutilizado por varias aplicaciones en entornos multientidad y multisistema. El desarrollo del mismo se ejecutará sobre el marco de trabajo Sauxe y guiado por el modelo de desarrollo propuesto para componentes de este tipo.

1.3. METODOLOGÍA PROPUESTA

En años recientes se ha comenzado a dar una mayor importancia al concepto de Arquitectura de Software dentro de la Ingeniería de Software. Una de las definiciones más comunes que se pueden encontrar con respecto a este concepto es aquella que provee los especialistas del Software Engineering Institute (SEI[11]), quienes actualmente se ubican entre los contribuyentes principales a la investigación en el tema. Se realizó un estudio minucioso de todas las corrientes, estilos, modelos y metodologías más utilizadas en el mundo para este tipo de actividad. Se obtuvo como resultado que las metodologías como RUP, XP, SCRUM, entre otras, centran su atención en la especificación, descripción de los procesos de negocios comunes en una entidad, ninguna de ellas norma el diseño, especificación y evaluación de un componente arquitectónico o tecnológico.[17]
Por esta razón se decidió reutilizar lo mejor de cada metodología para formar un modelo que se ajuste al desarrollo tecnológico. Para ello se tomaron las mejores prácticas propuestas por el SEI quienes proponen la arquitectura basada en escenarios es la corriente más nueva. Se trata de un movimiento predominantemente europeo con centro en Holanda. Recupera el nexo de la Arquitectura de Software con los requerimientos y la funcionalidad del sistema, ocasionalmente borroso en la arquitectura estructural clásica. En esta corriente suele utilizarse diagramas de casos de uso UML como herramienta ocasional, dado que los casos de uso son uno de los escenarios posibles y que no están orientados a objeto. Los autores vinculados con esta modalidad han sido, aparte de los codificadores de ATAM, CBAM, QASAR y demás métodos del SEI, los arquitectos holandeses de la Universidad Técnica de Eindhoven, de la Universidad Brije, de la Universidad de Groningen y de Philips Research: MugurelIonita, DieterHammer, Henk Obbink, Hans de Bruin, Hans Van Vliet, EelkeFolmer, Jilles Van Gurp y Jan Bosch. La presencia de holandeses es significativa; la Universidad de Eindhoven es, incidentalmente, el lugar en el que surgió lo que P. I. Sharp proponía llamar la escuela arquitectónica de Dijkstra. [18]
En los últimos años, el Software Engineering Institute ha dedicado una de sus líneas de investigación a los aspectos relacionados con la arquitectura de software. De estas investigaciones ha emergido un proceso enfocado al diseño y evaluación de las arquitecturas de software que se enmarca en un contexto organizacional. El proceso de diseño y evaluación de la arquitectura se descompone en varias etapas que incluyen:

[11] **(SEI):** Instituto de Ingeniería de Software, es un instituto federal estadounidense de investigación y desarrollo, fundado por el Congreso de los Estados Unidos en 1984 para desarrollar modelos de evaluación y mejora en el desarrollo de software.

Métodos de SEI para el diseño de una Arquitectura de Software

- ✓ **QAW (Quality Attribute Workshop):** Es un taller enfocado a la captura de requerimientos orientados a la arquitectura. Estos requerimientos son los atributos de calidad que son descritos a través de un esquema llamado "escenario" que incluye información relativa a la fuente, estímulo, artefacto, entorno, respuesta y métrica de respuesta.[18]
- ✓ **ADD (Attribute Driven Design):** Una vez que se han capturado escenarios y se han priorizado, se procede a realizar el diseño de la arquitectura siguiendo un enfoque iterativo de descomposición del sistema en componentes cada vez más pequeños. Durante ADD, se van tomando escenarios y se van realizando elecciones de diseño (usando soluciones conceptuales llamadas "patrones" y "tácticas") que permitan satisfacer los requerimientos descritos por los escenarios.[18]

Método del SEI para la documentación de la arquitectura

- ✓ **VaB (Views and Beyond):** Como resultado de ADD, se obtienen distintas estructuras del sistema, formadas de componentes y sus relaciones. Estas estructuras se documentan a través de "vistas" que muestra una perspectiva particular del sistema. Dado que la comunicación de la arquitectura reviste un papel fundamental en el desarrollo, los aspectos relacionados con su documentación son primordiales.[18]

Métodos del SEI para evaluar sistemas y arquitecturas de sistemas

- ✓ **ATAM (Architecture Trade off Analysis Method):** El método ATAM permite revelar qué también logra satisfacer los atributos de calidad la arquitectura y revela además que riesgos, puntos sensibles y compromisos están involucrados en la arquitectura.[19]
- ✓ Cost-Benefit Analysis Method (CBAM): este método constituye guías de los ingenieros de sistemas y otros stakeholders para la utilidad económica y ventajas asociadas a las decisiones arquitectónicas. [19]
- ✓ Active Reviews for Intermediate Designs (ARID): utilizado para evaluar diseños tempranos o porciones del diseño para su viabilidad en satisfacción con los stakeholders que les concierne.[19]

A medida que crece la complejidad de las aplicaciones, y que se extiende el uso de sistemas distribuidos y sistemas basados en componentes, los aspectos arquitectónicos del desarrollo de software estén recibiendo un interés cada vez mayor, tanto desde la comunidad científica como desde la propia industria del software. Como resultado del estudio el departamento de tecnología de CEIGE creó un modelo de desarrollo que incluye las mejores prácticas implementadas por metodologías como RUP, XP, SCRUM y los modelos propuestos por el SEI bajo un enfoque de desarrollo ágil. Este modelo describe un conjunto de actividades que especifican como construir y documentar un componente tecnológico basándose en los escenarios arquitectónicos. El modelo propuesto permitirá aumentar el nivel de integración y reutilización de la arquitectura.[17]

1.4. TECNOLOGÍAS PROPUESTAS

1.4.1. SAUXE

El marco de trabajo Sauxe da solución a un sin número de escenarios o aspectos arquitectónicos como: gestión y configuración dinámica de cache, integración de componentes de forma distribuida o no distribuida, acceso a bases de datos a través de una capa de abstracción, gestión de concurrencia de recursos, administración centralizada de transacciones, gestión dinámica de las trazas generadas por los sistemas, implementación de mecanismos de autenticación y autorización, implementación de mecanismos de mensajería y control de excepciones, gestión y configuración dinámica de precondiciones, poscondiciones y validaciones de variables, gestión y configuración de flujos de trabajo, visualización de las funcionalidades de un sistema, entre otros escenarios de alta complejidad, garantizando los atributos de calidad de los sistemas que se desarrollen con el mismo. Cuenta con una arquitectura en capas como se muestra en la Figura 13 que a su vez presenta en su capa superior un MVC[12].
Sauxe 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. Siguiendo el paradigma de independencia tecnológica por el cual apuesta el país, reutiliza las siguientes tecnologías libres:

Figura 13. Arquitectura de Sauxe.

1.4.2. EXTJS

En el mercado actualmente existen múltiples librerías de Javascript que permiten realizar todo tipo de maravillas en el navegador web. ExtJS permite que con pocas líneas de código sea posible realizar interfaces amigables para los usuarios. Es la librería más avanzada para el desarrollo rápido de aplicaciones con una apariencia totalmente novedosa y una arquitectura flexible. ExtJS es una librería Javascript que permite construir aplicaciones complejas en Internet.[20]
Esta librería incluye:

- ✓ Componentes UI del alto performance y personalizables.
- ✓ Modelo de componentes extensibles.
- ✓ Un API fácil de usar.
- ✓ Licencias Open Source y comerciales.

[12] **(MVC).Modelo Vista Controlador** es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

¿Qué tiene de bueno?

Antes de poder entrar a examinar ExtJS primero se tiene que hablar sobre RIA, acrónimo de Rich Internet Applications (Aplicaciones Ricas en Internet). Lo que RIA intenta proveer es aquello de lo que siempre ha adolecido la web, una experiencia de usuario muy parecida o igual a la que se tiene en las aplicaciones de escritorio.
Las aplicaciones web tradicionales tienen problemas como la recarga continua de las páginas cada vez que el usuario pide nuevo contenido, o la poca capacidad multimedia, para lo cual se han hecho necesarios plug-ins[13] externos.[20]
Junto con el reto de llevar la experiencia RIA a los usuarios comenzó el debate sobre cuál sería el mejor modo de atacar el problema. La historia de los últimos años ha traído diversas tecnologías, basadas en Flash (Adobe), Java (Sun), Silverlight (MS). Todas muy interesantes, pero con la desventaja de necesitar algún tipo de extensión en los navegadores que podría no estar presente. Ha sido esta limitante lo que le ha dado la victoria (al menos por el momento) al casi dejado de lado Javascript y la "nueva" tecnología conocida como AJAX.[20]
ExtJS encaja dentro de este esquema como un motor que permite crear aplicaciones RIA mediante Javascript. Si se enmarca a ExtJS dentro del desarrollo RIA, éste sería el render de la aplicación que controla el cliente y que ese encarga de enviar y obtener información del servicio.
Una de las grandes ventajas de utilizar ExtJS es que permite crear aplicaciones complejas utilizando componentes predefinidos así como un manejador de layouts similar al que provee Java Swing, gracias a esto provee una experiencia consistente sobre cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada uno (Firefox, IE, Safari, etc.).[20]
Además la ventana flotante que provee ExtJS es excelente por la forma en la que funciona. Al moverla o redimensionarla solo se dibujan los bordes haciendo que el movimiento sea fluido lo cual le da una ventaja tremenda frente a otros.
Usar un motor de render como ExtJS, permite tener además estos beneficios:

- ✓ Existe un balance entre Cliente – Servidor. La carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo.
- ✓ Comunicación asíncrona. En este tipo de aplicación el motor de render puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente se dé cuenta.[20]
- ✓ Eficiencia de la red. El tráfico de red puede disminuir al permitir que la aplicación elija que información desea transmitir al servidor y viceversa, sin embargo la aplicación que haga uso de la pre-carga de datos puede que revierta este beneficio por el incremento del tráfico.[20]

1.4.3. ZEND FRAMEWORK

Se trata de un framework para desarrollo de aplicaciones web y servicios web con PHP, brinda soluciones para construir sitios web modernos, robustos y seguros. Además es Open Source y trabaja con PHP 5. A diferencia de CakePHP que trabaja con PHP 4 y

[13](**plug-in).** Es un módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.

PHP 5. Este framework está formado por una serie de métodos estáticos y componentes que usarán estos métodos. Los componentes son varios y variados y aunque alguno es posible que no lo usemos nunca, hay otras que puede que las usemos hasta la saciedad, por ejemplo el componente para la BD. Entre los componentes de vital importancia se encuentran: Zend_Config para temas de configuración de aplicaciones web, Zend_Db para tratar con bases de datos, Zend_Search o Zend_Feed, entre otros. La instalación es sencilla, tan solo tendremos que añadir en el fichero de configuración php.ini, el path hasta la carpeta library del framework con la instrucción include_path. [21]

Características:

- ✓ Trabaja con MVC (Model View Controller)
- ✓ Cuenta con módulos para manejar archivos PDF, canales RSS, Web Services (Amazon, Flickr, Yahoo), etc
- ✓ El Marco de Zend también incluye objetos de las diferentes bases de datos, por lo que es extremadamente simple para consultar base de datos, sin tener que escribir ninguna consulta SQL.
- ✓ Una solución para el acceso a base de datos que balancea el ORM con eficiencia y simplicidad.
- ✓ Completa documentación y tests de alta calidad.
- ✓ Soporte avanzado.
- ✓ Un buscador compatible con Lucene.
- ✓ Robustas clases para autenticación y filtrado de entrada.
- ✓ Clientes para servicios web, incluidos Google Data APIs y StrikeIron.
- ✓ Muchas otras clases útiles para hacerlo tan productivo como sea posible.[21, 22]

1.4.4. DOCTRINE

Doctrine es un potente y completo sistema ORM (object renlationanl mapper) para PHP 5.2+ con un DBAL (database abstraction layer) incorporado. Se está empezando a ver su potencial, pero de la documentación se puede decir que tiene todas las características necesarias para ser funcional en casi cualquier proyecto. Entre muchas otras cosas brinda la posibilidad de exportar una base de datos existente a sus clases correspondientes y también a la inversa, es decir convertir clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos. Por otro lado, como la librería es bastante grande ésta tiene un método para ser 'compilada' al pasar a producción. [23]

Cuando se trabaja con Doctrine, necesitamos informar a su motor interno de cuál es el modelo de la aplicación, para ello se puede hacer ingeniería inversa de la base de datos existente, o si empezamos la aplicación desde 0, crear el modelo en la sintaxis específica que nos propone Doctrine y luego generar toda la base de datos.

Para crear el modelo, doctrine brinda dos alternativas, hacer una clase por tabla e indicarle mediante PHP el tipo de datos que almacenaremos en él.

Ventajas que nos facilitan enormemente tareas comunes y de mantenimiento:

- ✓ **Reutilización:** La principal ventaja que aporta un ORM es la reutilización permitiendo llamar a los métodos de un objeto de datos desde distintas partes de la aplicación e incluso desde diferentes aplicaciones.

- ✓ **Encapsulación:** La capa ORM encapsula la lógica de los datos pudiendo hacer cambios que afectan a toda la aplicación únicamente modificando una función.
- ✓ **Portabilidad:** Utilizar una capa de abstracción nos permite cambiar en mitad de un proyecto de una base de datos MySQL a una Oracle sin ningún tipo de complicación. Esto es debido a que no utilizamos una sintaxis MySQL, Oracle o SQLite para acceder a nuestro modelo, sino una sintaxis propia del ORM utilizado que es capaz de traducir a diferentes tipos de bases de datos.
- ✓ **Seguridad:** Los ORM suelen implementar mecanismos de seguridad que protegen nuestra aplicación de los ataques más comunes como SQL Injection.
- ✓ **Mantenimiento del código:** Gracias al correcto ordenamiento de la capa de datos, modificar y mantener nuestro código es una tarea sencilla.[23, 24]

1.5. HERRAMIENTAS PROPUESTAS

1.5.1. ZEND STUDIO

Programa de la casa Zend, uno de los mayores impulsores de PHP, orientada a desarrollar aplicaciones web. Zend Studio es un editor de texto para páginas PHP que proporciona un buen número de ayudas desde la creación y gestión de proyectos hasta la depuración del código.
A diferencia de las versiones anteriores a la v5.5.1.281, ya no se trata de un IDE[14] desarrollado en Java (excesiva lentitud y consumo de memoria en algunos casos), ahora está basado en Eclipse, es una plataforma de software de código abierto independiente de una plataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido". Soporta varias librerías como: ExtJS, Yahoo UI y JQuery para la presentación así como la inclusión de Framework para el desarrollo de la web pudiendo combinarlas fácilmente en una aplicación. ZendStudio está disponible como una aplicación independiente, se puede encontrar para dos plataformas fundamentales: Windows y GNU/Linux. Entre sus principales características se encuentran: [25]

- ✓ Soporta PHP4 y PHP5.
- ✓ Asistente de código.
- ✓ Plantillas (PHP, PHPDoc, Nuevo archivo).
- ✓ Código plegable (clases, funciones y PHPDoc).
- ✓ Tiempo real la detección de errores.
- ✓ PHP (Project) Vista Exploratoria.
- ✓ Plataforma de Integración.
- ✓ Plataforma API.
- ✓ Galerías de código.
- ✓ ZendGuardIntegration.
- ✓ PHPDocumentor.
- ✓ Fácil de depuración.

[14] Entorno de Desarrollo Integrado en inglés, Integrated Development Environment.

1.5.3. APACHE

Apache es un servidor web gratuito, potente y que ofrece un servicio estable y sencillo de mantener y configurar. Es indiscutiblemente uno de los mayores logros del Software Libre. Alguna de sus características:

- ✓ Es multiplataforma, aunque idealmente está preparado para funcionar bajo Linux.
- ✓ Muy sencillo de configurar.
- ✓ Es Open-source.
- ✓ Amplias librerías de PHP y Perl a disposición de los programadores.
- ✓ Posee diversos módulos que permiten incorporarle nuevas funcionalidades, estos son muy simples de cargar.
- ✓ Es capaz de utilizar lenguajes como PHP, TCL, Python, entre otros.[26]

Apache tiene amplia aceptación en la red: desde 1996, es el servidor HTTP más usado en el mundo y alcanzó su máxima cuota de mercado en el 2005 siendo el servidor empleado en el 70% de los sitios web en el mundo.[27] Por las características anteriormente expuestas se decidió utilizar este servidor web para la publicación de Acaxia.

1.5.4. POSTGRESQL

Es un servidor de base de datos relacional, libre. Tiene soporte total para transacciones, disparadores, vistas, procedimientos almacenados, almacenamiento de objetos de gran tamaño. Se destaca en ejecutar consultas complejas, consultas sobre vistas, subconsultas y joins[15]de gran tamaño. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Como toda herramienta de software libre PostgresSQL tiene entre otras ventajas las de contar con una gran comunidad de desarrollo en Internet, su código fuente está disponible sin costo alguno y algo muy importante es que dicha herramienta es multiplataforma. Además de sus ofertas de soporte, cuenta con una importante comunidad de profesionales y entusiastas de PostgreSQL de los que los centros de desarrollos pueden obtener beneficios y contribuir. Está disponible en casi cualquier Unix (34 plataformas en la última versión estable), y una versión nativa de Windows está actualmente en estado beta de pruebas. [28]Por todos los beneficios anteriormente expuestos se convierte en la mejor alternativa para el desarrollo del producto.

1.5.5. VISUAL PARADIGM

Visual Paradigm para UML es una herramienta UML libre y profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML.

[15] **(JOIN)**. En lenguaje SQL permite combinar registros de dos o más tablas en una base de datos relacional.

Este es un sistema muy completo que permitirá aumentar la productividad al momento de desarrollar. Para los ingenieros y arquitectos de software es excelente ya que permite centralizar y planificar las ideas, además de mejorar la productividad en el desarrollo y mantenimiento del software, aumentar la calidad del mismo, reducir el tiempo de desarrollo y mantenimiento, mejorar la planificación de un proyecto, realizar una gestión global en todas las fases de desarrollo de software con una misma herramienta. Esta herramienta puede ayudar en todos los aspectos del ciclo de vida del software. La mayor parte de los equipos de desarrollo del CEIGE cuentan con un dominio medio o alto de la herramienta, aspecto que le da un valor a agregado a la decisión de adoptar esta solución para el modelado del sistema.[29]

1.5. CONCLUSIONES PARCIALES

De la investigación realizada sobre el proceso de autorización en los sistemas de seguridad existentes en el mundo, se obtuvo como resultado que cada uno de ellos gestionaba la autorización de forma diferente, sin garantizar que la misma se administrará de forma centralizada, por lo que la implementación de esta tendría que ser adaptada a las características del sistema a desarrollar. Además se debían configurar algunos ficheros manualmente, si este proceso fuese realizado por desarrolladores sin experiencia provocaría que se dejasen brechas de seguridad.

Adquirir una herramienta de este tipo existente en el mercado, se estaría incorporando un riesgo de seguridad inminente ya que no se tendría el dominio del funcionamiento interno del sistema y en vez de disminuir las brechas de seguridad se estaría aumentando el nivel de incertidumbre y vulnerabilidad de nuestras aplicaciones. Los sistemas de seguridad desarrollados en la universidad que implementan el proceso de autorización están desarrollados basándose en requisitos específicos para cada sistema, por lo que ninguno trata el tema de la multientidad y el multisistema. La no disponibilidad de soluciones que cubran los escenarios identificados tanto en el contexto nacional como en el exterior hace necesario la implementación de una solución que gestione la autorización de forma centralizada y que implementen la solución de multientidad y multisistema.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

El siguiente capítulo recoge los resultados obtenidos durante el proceso de desarrollo de la solución, así como algunos de los artefactos generados por el mismo. Primeramente son descritos los procesos de negocio relativos al campo de acción y los requisitos funcionales de la solución para lograr un entendimiento de lo que se quiere lograr. Posteriormente se realiza una descripción del diseño elaborado por el autor para alcanzar las metas trazadas, además de la estrategia a seguir durante el proceso de implementación.

2.1. DESCRIPCIÓN DE LA SOLUCIÓN

La solución de autorización debe permitir restringir los permisos de los usuarios en un entorno multientidad y multisistema, para ello debe gestionar las estructuras de los sistemas. Un sistema puede estar compuesto por subsistemas, funcionalidades y acciones. Se debe restringir el acceso a los servicios publicados por los diferentes sistemas, para ello se debe permitir la gestión dinámica de los servicios que brindan o consumen los mismos, los servicios están compuestos por funciones y estas a su vez necesitan parámetros de entrada para su ejecución.
Los permisos sobre los diferentes sistemas, funcionalidades y acciones deben de estar agrupados por roles y estos a su vez deberán ser asignados a usuarios en un conjunto de entidades. Los usuarios deben contener un grupo de datos asociados a su perfil entre los que se encuentran el tema, el idioma, el rango de IP desde donde puede conectarse, el tipo de escritorio y el dominio de entidades a los cuales tienen acceso.
El concepto dominio se encarga de agrupar un conjunto de estructuras, estas estructuras pueden ser ministerios, agrupaciones, entidades, áreas, estructuras internas o cargos. Debe restringirse el acceso sobre los diferentes niveles estructurales y de sistemas, de esta forma los administradores solo podrán asignar permisos a partir de los que le fueron asignados.

A continuación la Figura 14 muestra un mapa conceptual que relaciona los principales conceptos identificados en la investigación, además se muestran los diagramas de procesos fundamentales con una breve descripción para que el equipo de desarrollo comprenda el negocio que debe cubrir la aplicación. Es necesario que se traten puntos claves del desarrollo como la arquitectura de la aplicación, patrones de diseño utilizados, algunos artefactos que propone la metodología seleccionada y el diseño de clases del sistema.

Para lograr la calidad del proceso de desarrollo será guiado por los estándares y normativas dictadas por el departamento de tecnología del CEIGE. Estas normativas regulan el diseño de interfaces de usuarios, la nomenclatura de las clases, componentes y objetos a nivel de datos, estilos de codificación, validaciones a todos los niveles y la forma en la que se deben describir cada uno de los componentes desarrollados para facilitar el mantenimiento y la reutilización.

Modelo conceptual

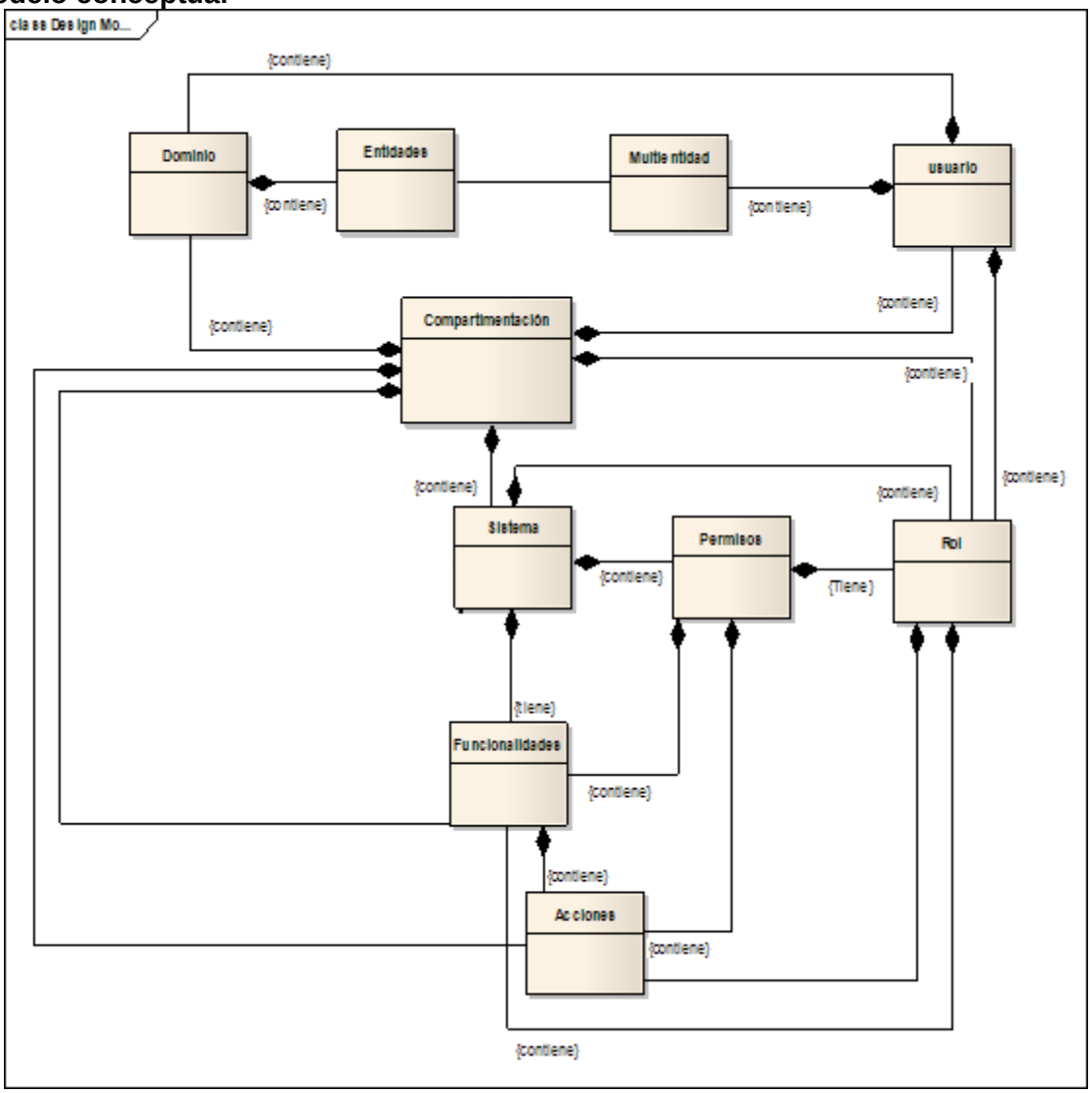

Figura 14. Mapa conceptual

Dominio: está compuesto por un grupo de estructuras o entidades a las que pueden tener acceso uno o varios usuarios.
Entidades: forma parte de la estructura de un país y puede comportarse como un ministerio, entidad, área o cargo.
Multientidad: es el concepto que une los usuarios o sistemas con las estructuras.
Usuario: cualquier persona que interactúa con el sistema y juega un rol determinado en el mismo.
Compartimentación: es el concepto que regula el acceso a la información.
Sistema: es el producto al que se le desea brindar seguridad.
Funcionalidad: agrupa un conjunto de acciones del sistema.
Acción: es la operación que realiza el usuario en el sistema.
Rol: es el concepto que agrupa una serie de permisos sobre sistemas, funcionalidades y acciones que se le asignarán a un conjunto de usuarios.
Permiso: es el concepto que une los roles, los usuarios y las diferentes estructuras jerárquicas y del sistema.

2.2. DESCRIPCIÓN DE LOS PROCESOS DE NEGOCIO

Un proceso de negocio es un conjunto de tareas relacionadas lógicamente llevadas a cabo para lograr un resultado de negocio definido. La descripción de los procesos de negocio hace más viable el paso a las actividades del análisis ya que posibilita una compresión más clara de los procesos en cuestión y contribuye a que los requisitos que se definan satisfagan las necesidades del usuario. El autor, teniendo en cuenta la opinión de especialistas y de los usuarios finales, ha identificado seis procesos de negocio que deben ser asistidos por la solución. Estos procesos de negocio son:
Gestión de dominios, Gestión de sistemas, Gestión de funcionalidades, Gestión de acciones, Gestión de roles y Gestión de usuarios que se describen en los epígrafes 2.2.1, 2.2.2, 2.2.3, 2.2.4, 2.2.5, 2.2.6 respectivamente.

2.2.1. DESCRIPCIÓN DEL PROCESO: GESTIONAR DOMINIOS

Un domino no es más que un conjunto de estructuras agrupadas por algún criterio común. Las estructuras pueden ser ministerios, agrupaciones, entidades, áreas o cargos. Se debe restringir el acceso de los usuarios sobre la información que se gestiona en cada una de las estructuras mencionadas. Para ello de se debe permitir gestionar los dominios de estructuras que serán asignados a los usuarios atendiendo al rol que juega dentro de cada entidad. Los dominios se deben gestionar desde el sistema de autorización y el procesamiento de los datos quedará en manos del sistema de Estructura y Composición con el que existirá integración a nivel de servicios internos. La Figura 15 muestra la descripción del proceso para entender mejor el negocio.

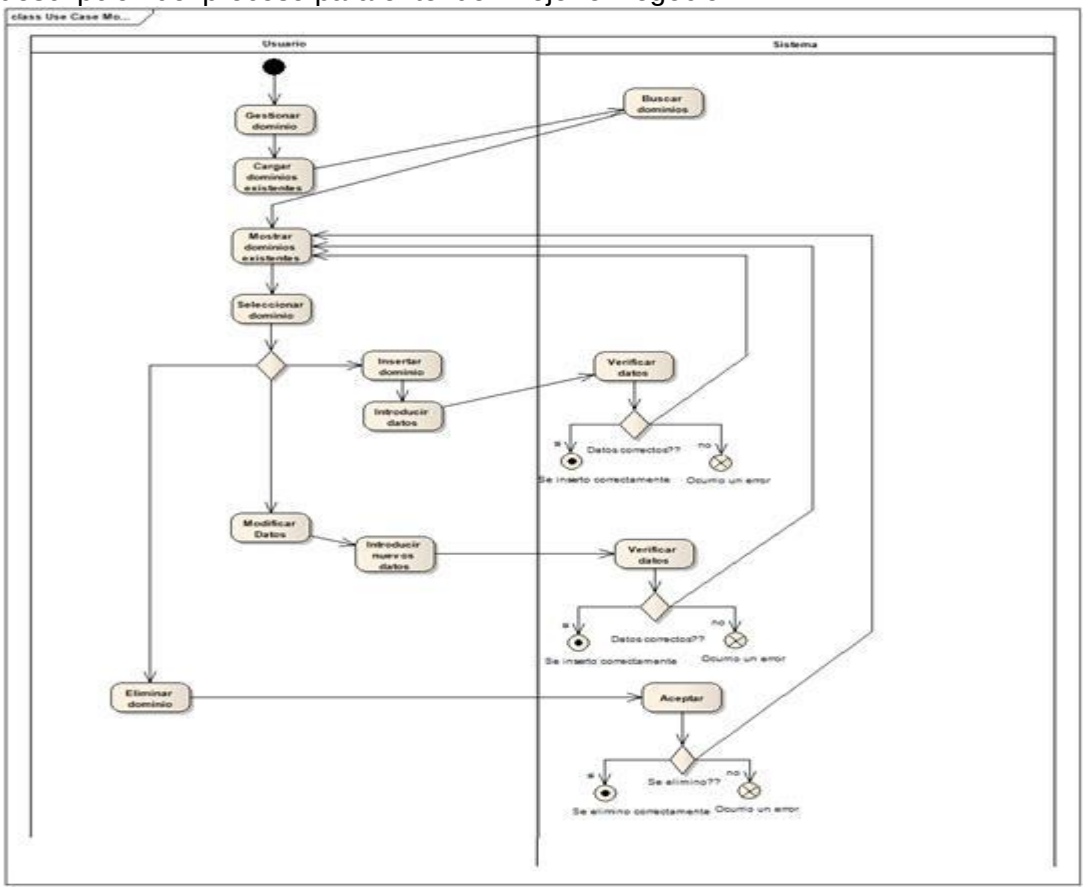

Figura 15. Gestionar dominios

2.2.2. DESCRIPCIÓN DEL PROCESO: GESTIONAR SISTEMAS

Este proceso debe permitir gestionar las estructuras de los sistemas a las los cuales se les desea brindar seguridad. Un sistema puede estar compuesto por uno o varios subsistemas y así sucesivamente para cada subsistema, de esta forma se crea una estructura arbórea. La estructura de los sistemas debe representar la distribución física del mismo. Iniciando este proceso se le debe mostrar al usuario las estructuras de los sistemas existentes, de esta forma el mismo podrá seleccionar si desea registrar un nuevo sistema o subsistema, modificar, eliminar algún sistema o subsistema existente. Como vía de interoperabilidad debe darse la posibilidad de exportar o importar las estructuras de los sistemas. Para lograr un mayor entendimiento del negocio, la Figura 16 muestra el diagrama que describe este proceso.

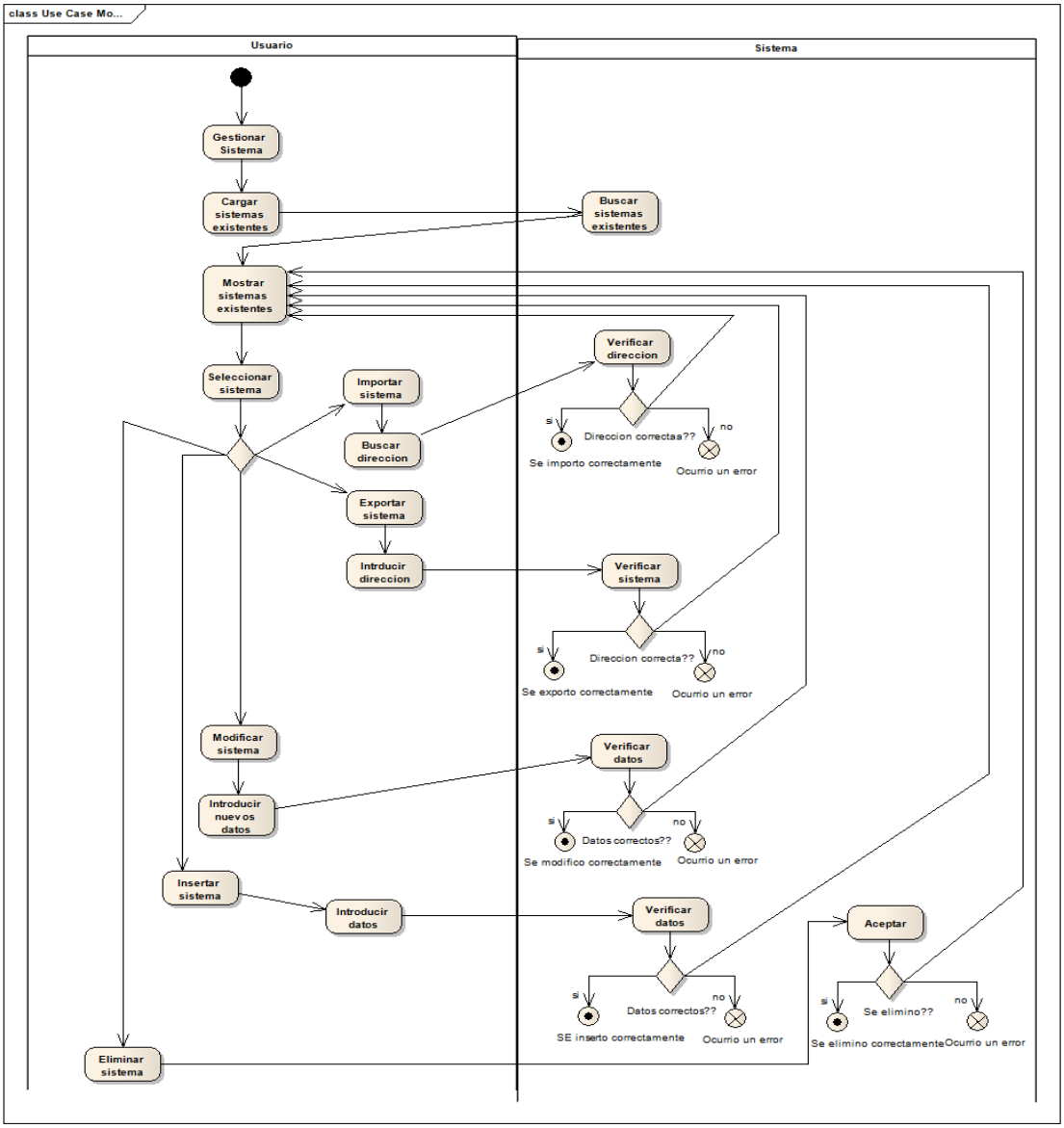

Figura 16. Gestionar sistema

2.2.3. DESCRIPCIÓN DEL PROCESO: GESTIONAR FUNCIONALIDADES

El sistema debe permitir restringir el acceso a las funcionalidades de cada uno de los sistemas o subsistemas, para ello debe mostrársele al usuario todos los sistemas registrados para que el mismo gestione las funcionalidades de cada uno de ellos. Las funcionalidades no son más que conceptos que agrupan un conjunto de acciones tales como insertar, modificar entre otras. Entre los datos que se gestionan en este proceso el más importante es la referencia, este campo representa la dirección o función que se desea ejecutar cuando el usuario seleccione la funcionalidad sobre la que desea trabajar. Para un mejor entendimiento de este proceso ver la Figura 17 en la que se muestra el flujo de las operaciones.

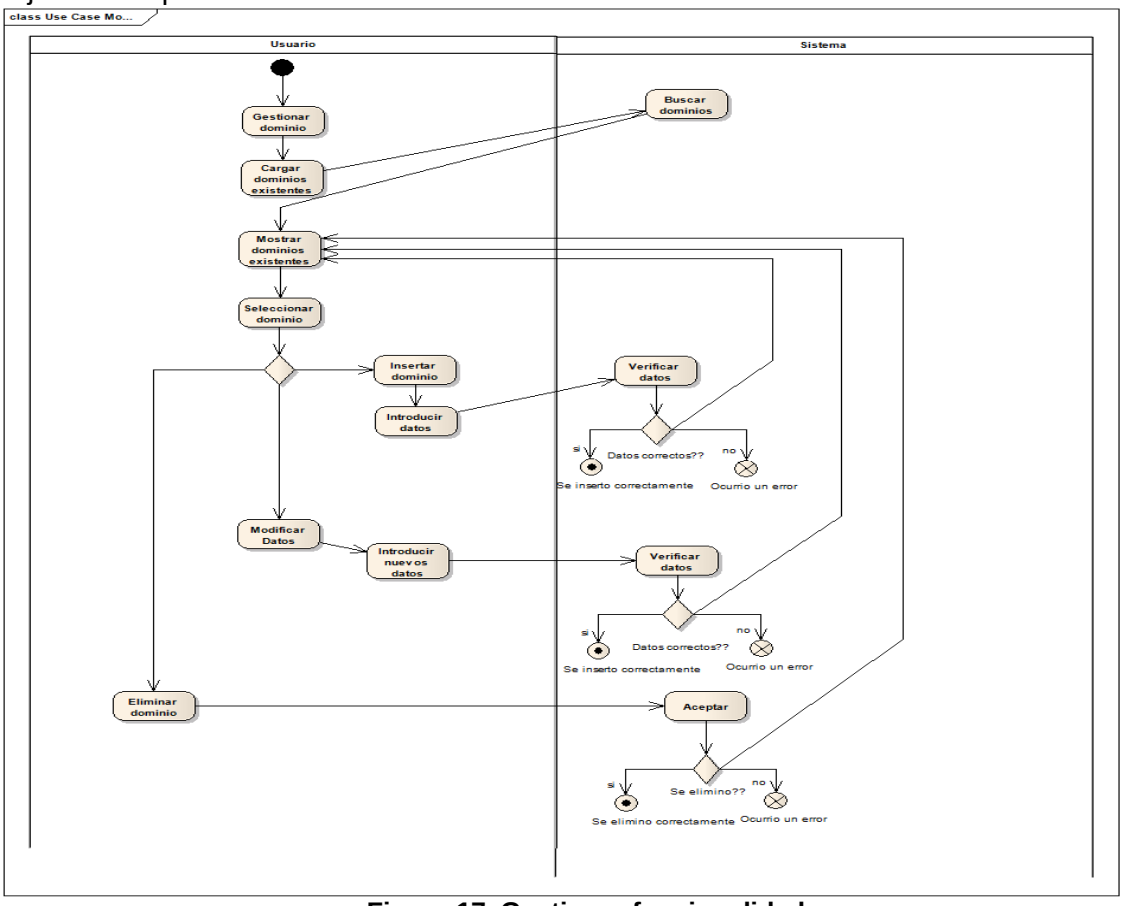

Figura 17. Gestionar funcionalidad

2.2.4. DESCRIPCIÓN DEL PROCESO: GESTIONAR ACCIONES

Las acciones son evento u operaciones que ejecuta un usuario en un sistema, ejemplo de ellas pueden ser insertar, modificar, eliminar entre otras. Como se explicó en el proceso anterior una funcionalidad contiene un conjunto de acciones y para restringir el acceso a ellas es necesario mostrarle al usuario los sistemas y funcionalidades registradas para que el mismo seleccione a qué funcionalidad desea gestionarle las acciones. Entre los datos más significativos de las acciones se encuentra la abreviatura, que debe coincidir con el id del componente que ejecuta la acción a nivel de interfaz o con algún objeto al cual se le pueda restringir el acceso. De esta forma el usuario solo podrá ver los componentes a los cuales tiene acceso en la funcionalidad que se encuentra trabajando.

Para un mejor entendimiento del proceso se puede observar la Figura 18 que describe el flujo de las operaciones.

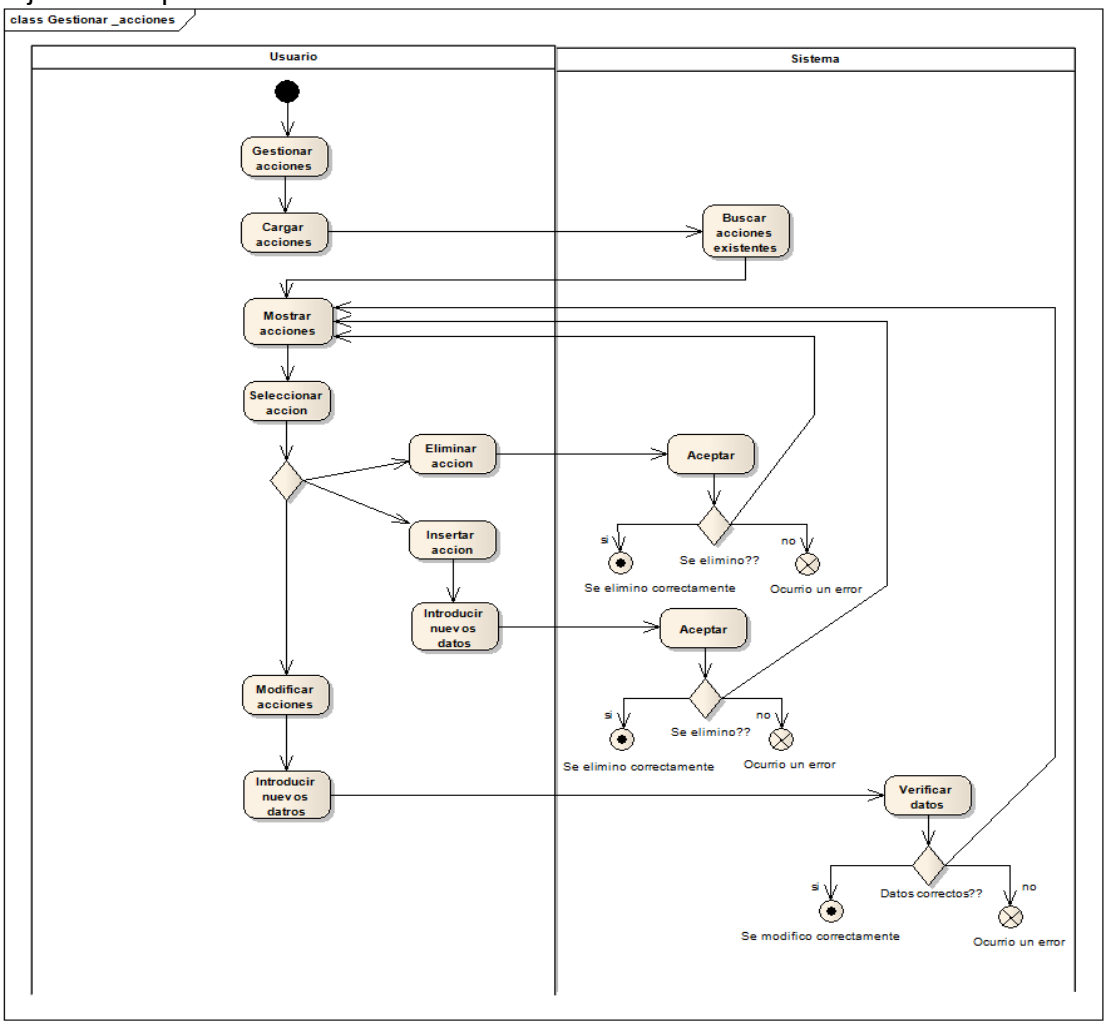

Figura 18. Gestionar Acciones

2.2.5. DESCRIPCIÓN DEL PROCESO: GESTIONAR ROLES

Los roles no son más que la agrupación de un conjunto de permisos sobre las diferentes estructuras de un sistemas. Para gestionar este concepto es necesario que al inicio se le muestre al usuario los roles registrados y que se le dé la posibilidad de adicionar un nuevo rol con permisos sobre una o varias estructuras de los sistemas, modificar o eliminar un rol existente y regularle las acciones que podrá realizar un usuario. Es necesario tener en cuenta que siempre debe existir un rol creado para que no se pierda el acceso al sistema. Para comprender mejor este proceso puede ver la Figura 19 en la que se muestran las posibles acciones a seguir.

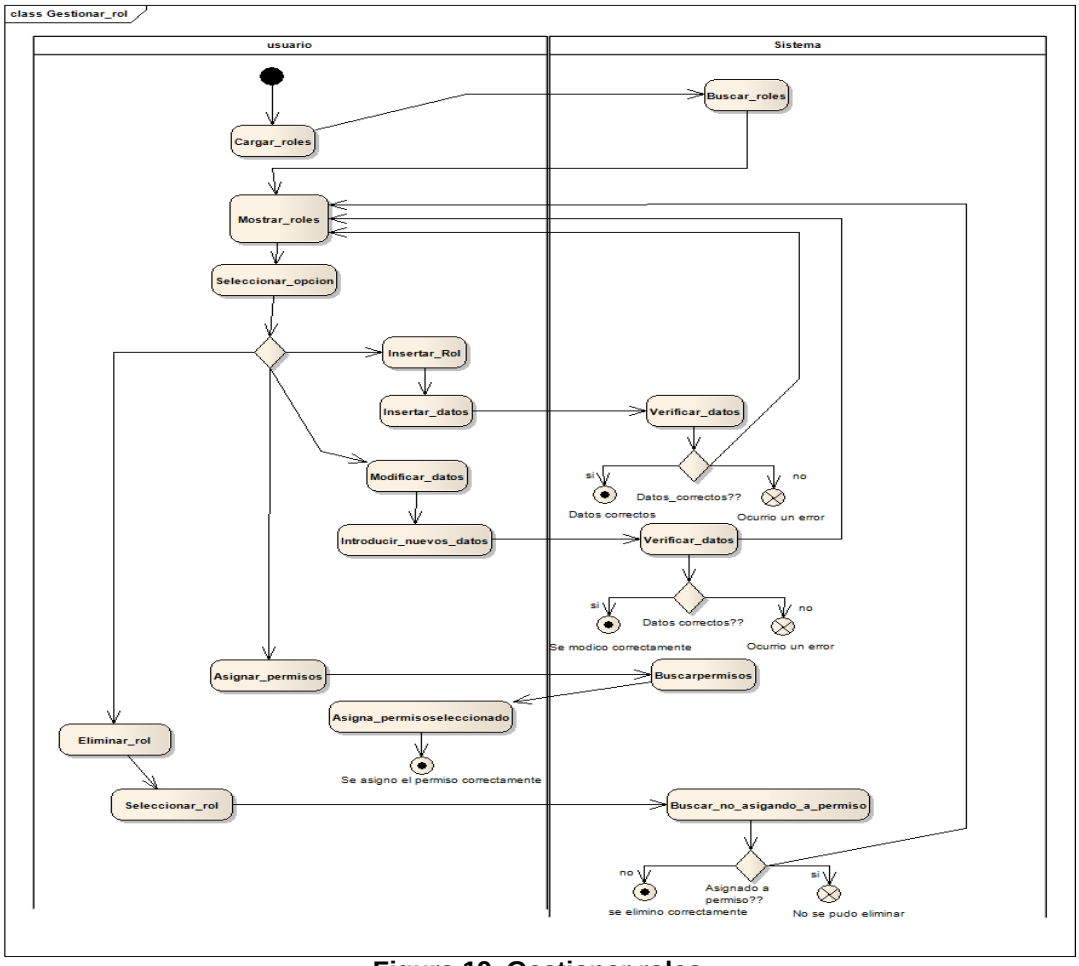

Figura 19. Gestionar roles

2.2.6. DESCRIPCIÓN DEL PROCESO: GESTIONAR USUARIOS

Este es uno de los procesos más complejos e importantes del sistema ya que se debe encargar de unir los demás procesos para crear las restricciones asociadas a cada usuario en las diferentes estructuras ya sean de sistema o de composición. Para lograr una gestión eficiente de este proceso es necesario mostrarle al administrador los usuarios registrados, una vez mostrada la lista de usuarios debe darse la posibilidad de adicionar un nuevo usuario, el mismo debe contener un grupo de datos asociados a su perfil entre los que se encuentran el tema, el idioma, el rango de IP desde donde puede conectarse, el tipo de escritorio y el dominio de entidades al cual tiene acceso. Además se debe dar la posibilidad de modificar o eliminar un usuario existente, teniendo en cuenta siempre las validaciones pertinentes para que no se pierdan datos de valor y que se garantice la disponibilidad del sistema. Para que los usuarios puedan acceder al sistemas necesitan tener al menas un rol en alguna de las entidades del dominio que le fue asignado al crearse, para lograr este objetivo debe mostrarse el dominio de entidades que posee el usuario y los roles creados para que la administración decida qué rol tendrá sobre las diferentes estructuras. El usuario debe estar inicialmente desactivado y solo podrá ser activado por la persona que tenga esta autoridad en la entidad y además debe proveerse un espacio donde pueda cambiar su contraseña. El sistema debe brindar todos los servicios necesarios para la autorización, gestión de perfil, comprobación de rangos de IP

desde donde puede conectarse un usuario, adicionar, modificar o eliminar usuarios a nivel de IoC o servicios web. Para entender mejor el negocio de este proceso tan complejo se realizó el diagrama que se muestra en la Figura 20 que describe las acciones que puede realizar un usuario para cubrir todos los posibles escenarios.

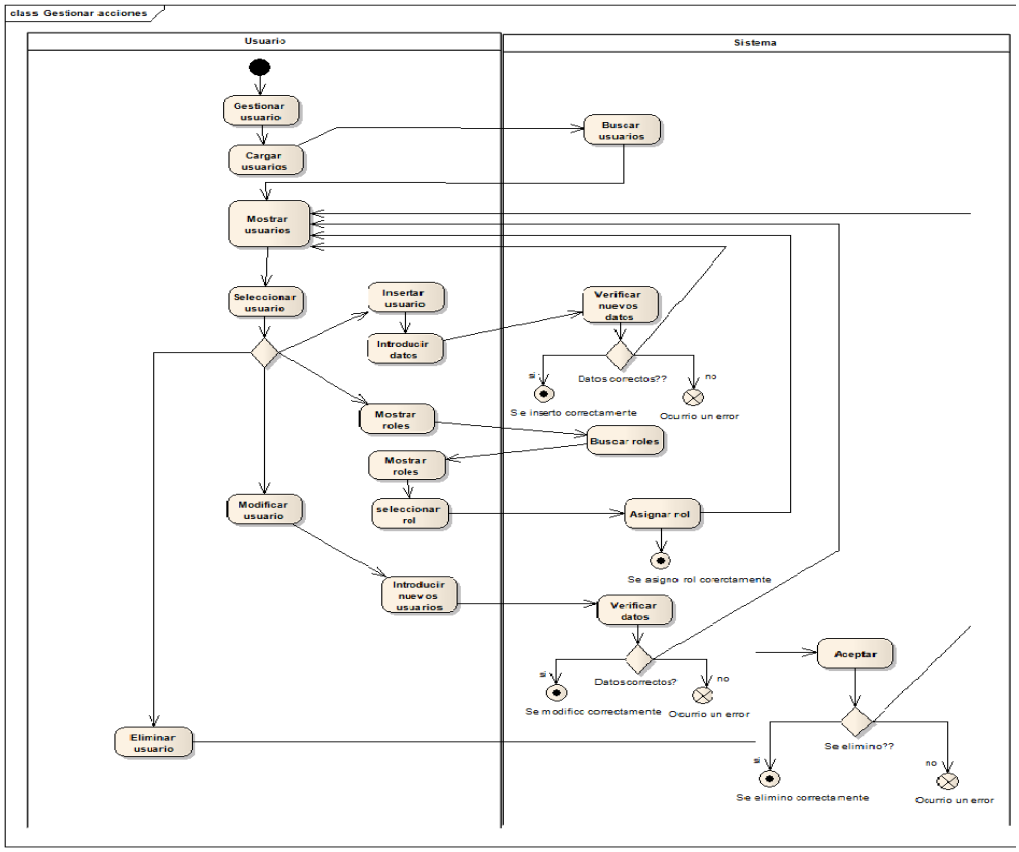

Figura 20. Gestionar usuarios

2.3. REQUISITOS DE SOFTWARE

Uno de los elementos más importantes en un proceso de desarrollo de software lo constituyen los requisitos, pues estos permiten una comunicación efectiva entre los usuarios y el equipo de desarrollo, con el objetivo de llegar a un entendimiento de lo que hay que realizar, siendo así la clave del éxito en la producción de un software.
Existen distintas definiciones de requisito de software dadas por diversos autores entre las que podemos citar:

- ✓ Los requisitos son expresiones de las necesidades de stakeholders para alcanzar una meta particular.[30]
- ✓ Un requisito es una condición o capacidad necesaria dada por un usuario con el objetivo de resolver un problema o alcanzar un objetivo.[31]
- ✓ Los requisitos expresan las necesidades y restricciones atribuibles a un producto de software que contribuye a la solución de algún problema del mundo real.[32]

Los requisitos de software se clasifican en funcionales y no funcionales. Los requisitos funcionales describen qué es lo que el sistema debe hacer para dar soporte a las funciones y objetivos del usuario. Los requisitos no funcionales imponen restricciones de cómo los requisitos funcionales deben ser implementados.[33]

2.3.1. REQUISITOS FUNCIONALES

Los requisitos funcionales son los que definen las funciones que el sistema será capaz de realizar. Los mismos se centran en qué y no cómo se deben hacer esas funciones. Una vez descritos los procesos de negocios, sé identificaron los requisitos a cumplir por el sistema, los cuales son listados a continuación:

R.1. Requisito funcional Gestionar sistema
- R.1.1. Listar sistemas.
- R.1.2. Registrar sistema.
- R.1.3. Modificar sistema.
- R.1.4. Eliminar sistema.
- R.1.5. Importar sistema.
- R.1.6. Exportar sistema.

R.2. Requisito funcional Gestionar funcionalidad.
- R.2.1. Listar funcionalidades.
- R.2.2. Registrar funcionalidad.
- R.2.3. Modificar funcionalidad.
- R.2.4. Eliminar funcionalidad.
- R.2.5. Buscar funcionalidad.

R.3. Requisito funcional Gestionar acciones.
- R.3.1. Listar acciones.
- R.3.2. Registrar acción.
- R.3.3. Modificar acción.
- R.3.4. Eliminar acción.
- R.3.5. Buscar acción.

R.4. Requisito funcional Gestionar rol.
- R.4.1. Listar roles.
- R.4.2. Registrar rol.
- R.4.3. Modificar rol.
- R.4.4. Regular acciones.
- R.4.5. Eliminar rol.

R.5. Requisito funcional Gestionar usuario.
- R.5.1. Listar usuarios.
- R.5.2. Registrar usuario
- R.5.3. Modificar usuario.
- R.5.4. Eliminar usuario.
- R.5.5. Asignar rol.
- R.5.6. Cambiar contraseña.
- R.5.7. Buscar usuarios.

R.6. Requisito funcional Gestionar nomenclador de dominio.
- R.6.1. Listar dominios existentes.
- R.6.2. Registrar dominio.
- R.6.3. Modificar dominio.
- R.6.4. Eliminar dominio.
- R.6.5. Buscar dominio.

2.3.2. ESPECIFICACIÓN DE REQUISITOS

La Tabla 1. Especificación del requisito Listar sistemas. muestra la especificación del requisito funcional Listar sistemas.

Tabla 1. Especificación del requisito Listar sistemas.

Precondiciones	Se ha registrado al menos un sistema en el sistema

Flujo de eventos		
Flujo básico		
1	El sistema muestra un listado de los sistemas. Se muestra la Denominación.	
2	Concluye el requisito.	
Pos-condiciones		
1	N/A	
Flujos alternativos		
Flujo alternativo		
1	N/A	
Pos-condiciones		
1	N/A	
Validaciones		
1	N/A	
Relaciones	**Requisitos Incluidos**	N/A
	Extensiones	N/A
Conceptos	**Sistema**	Visibles en la interfaz: Denominación Abreviatura Icono Descripción Utilizados internamente: N/A
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

La Figura 21. Listar sistemas muestra el prototipo de interfaz del RF Listar sistemas.

Figura 21. Listar sistemas

La Tabla 2 muestra la especificación del requisito funcional Adicionar sistema.

Tabla 2. Especificación de requisito Adicionar sistema.

Precondiciones		N/A
Flujo de eventos		
Flujo básico		
1	Se selecciona el subsistema donde se adicionará el nuevo sistema	
2	Se introducen los datos del sistema: Denominación Abreviatura Icono Descripción Servidor de bases de datos Gestor de bases de datos Base de Datos Esquema Externo	
3	Si el sistema es externo se especifica el servidor web que utiliza.	
4	El sistema valida (ver validación 1) los datos introducidos.	
5	Si los datos son correctos el sistema los registra.	
6	El sistema confirma el registro de los datos.	
7	Concluye el requisito.	
Pos-condiciones		
1	Se registró en el sistema un nuevo sistema.	
Flujos alternativos		
Flujo alternativo 5.a Información errónea		
1	El sistema señala los datos erróneos y permite corregirlos.	
2	El usuario corrige los datos.	
3	Volver al paso 4 del flujo básico.	
Pos-condiciones		
1	N/A	
Flujo alternativo 5.b Información incompleta		
1	El sistema señala los datos vacíos y permite corregirlos.	
2	El usuario corrige los datos.	
3	Volver al paso 4 del flujo básico.	
Pos-condiciones		
1	N/A	
Flujo alternativo *.a El usuario cancela la acción		
1	Concluye el requisito.	
Pos-condiciones		
1	No se registran los datos.	
Validaciones		
1	Se validan los datos según lo establecido en el Modelo conceptual CSG-ERP-N-SEG-i2201.	
Relaciones	**Requisitos Incluidos**	N/A
	Extensiones	N/A

Conceptos	**Sistema**	Visibles en la interfaz: Denominación Abreviatura Icono Descripción Utilizados internamente: N/A
	Servidor de Bases de datos	Visibles en la interfaz: Descripción Utilizados internamente: N/A
	Gestor de Bases de Datos	Visibles en la interfaz: Nombre Puerto Descripción Utilizados internamente: N/A
	Base de Datos	Visibles en la interfaz: Nombre Utilizados internamente: N/A
	Esquema	Visibles en la interfaz: Nombre Utilizados internamente: N/A
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

La Figura 22 muestra el prototipo de interfaz de usuario del RF Adicionar sistema.

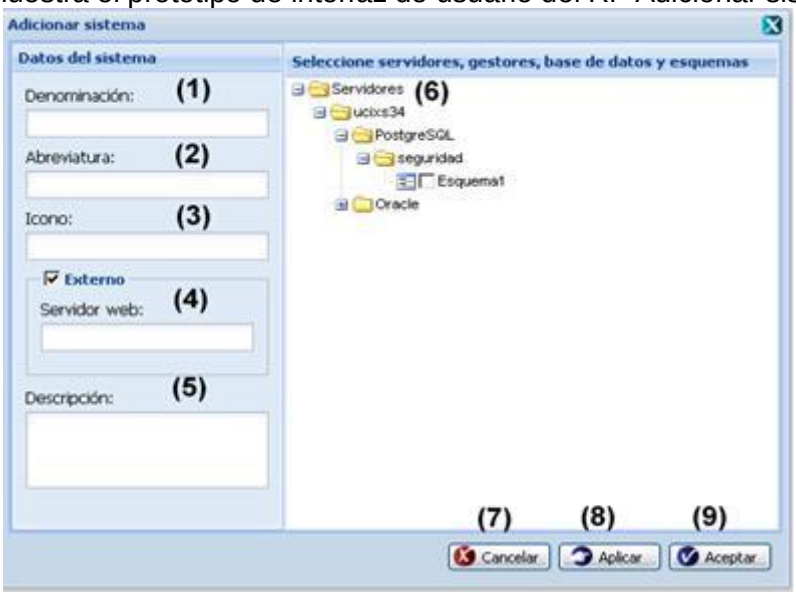

Figura 22. Adicionar sistema.

La Tabla 3 muestra la especificación del requisito funcional Modificar sistema.

Tabla 3. Especificación del requisito Modificar sistema.

Precondiciones		Se ha registrado al menos un sistema en el sistema
Flujo de eventos		
Flujo básico		
1	Se selecciona el sistema a modificar.	
2	El sistema muestra y permite editar los datos del sistema.	
3	Se introducen los datos del sistema Denominación Abreviatura Icono Descripción Servidor de bases de datos Gestor de bases de datos Base de Datos Esquema Externo	
4	El sistema valida (ver validación 1) los datos introducidos.	
5	Si los datos son correctos el sistema los registra.	
6	El sistema confirma el registro de los datos.	
7	Concluye el requisito.	
Pos-condiciones		
•	Se modificaron los datos del sistema.	
Flujos alternativos		
Flujo alternativo 5.a Información errónea		
1	El sistema señala los datos erróneos y permite corregirlos.	
2	El usuario corrige los datos.	
3	Volver al paso 4 del flujo básico.	
Pos-condiciones		
1	N/A	
Flujo alternativo 5.b Información incompleta		
1	El sistema señala los datos vacíos y permite corregirlos.	
2	El usuario corrige los datos.	
3	Volver al paso 4 del flujo básico.	
Pos-condiciones		
1	N/A	
Flujo alternativo *.a El usuario cancela la acción		
1	Concluye el requisito.	
Pos-condiciones		
1	No se registran las modificaciones realizadas.	
Validaciones		
1	Se validan los datos según lo establecido en el Modelo conceptual CSG-ERP-N-SEG-i2201.	
Relaciones	**Requisitos Incluidos**	Se selecciona el sistema a modificar: Listar sistemas, en la agrupación Gestionar sistemas.
	Extensiones	N/A

Conceptos	**Sistema**	Visibles en la interfaz: Denominación Abreviatura Icono Descripción Utilizados internamente: N/A
	Servidor de Bases de datos	Visibles en la interfaz: Descripción Utilizados internamente: N/A
	Gestor de Bases de Datos	Visibles en la interfaz: Nombre Puerto Descripción Utilizados internamente: N/A
	Base de Datos	Visibles en la interfaz: Nombre Utilizados internamente: N/A
	Esquema	Visibles en la interfaz: Nombre Utilizados internamente: N/A
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

La Figura 23 muestra el prototipo de interfaz de usuario del RF Modificar sistema.

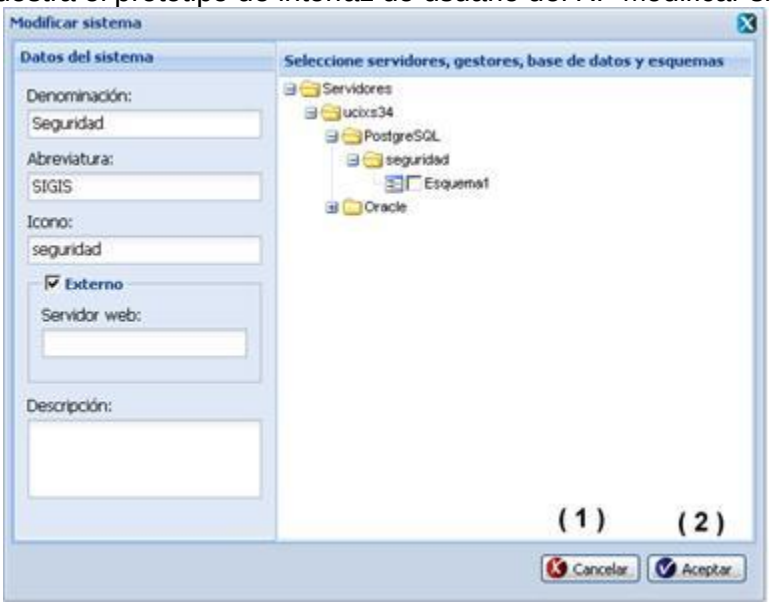

Figura 23. Modificar sistema.

La Tabla 4 muestra la especificación del requisito funcional Eliminar sistema.

Tabla 4. Especificación del requisito Eliminar sistema.

Precondiciones		Se ha registrado al menos una aplicación en el sistema.
Flujo de eventos		
Flujo básico		
1	Se selecciona el sistema a eliminar.	
2	El sistema verifica (ver validación 1) que se pueda eliminar el sistema.	
3	Se solicita confirmación para eliminar el sistema.	
4	Si el usuario confirma se elimina el sistema.	
5	El sistema confirma la eliminación.	
6	Concluye el requisito.	
Pos-condiciones		
1	Se eliminó el sistema.	
Flujos alternativos		
Flujo alternativo 3.a Existe un usuario que tiene un rol en el sistema		
1	El sistema notifica por qué no puede eliminarse sistema.	
Pos-condiciones		
1	N/A	
Flujo alternativo *.a El usuario cancela la acción		
1	Concluye el requisito.	
Pos-condiciones		
1	No se elimina el sistema.	
Validaciones		
1	No se puede eliminar un sistema si un usuario tiene asignado un rol en él.	
Relaciones	**Requisitos Incluidos**	Se selecciona el sistema a eliminar: Listar sistemas, en la agrupación Gestionar sistemas.
	Extensiones	N/A
Conceptos	**Sistema**	Visibles en la interfaz: Denominación Abreviatura Icono Descripción Utilizados internamente: N/A
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

La Figura 24 muestra el prototipo de interfaz de usuario del RF Eliminar sistema.

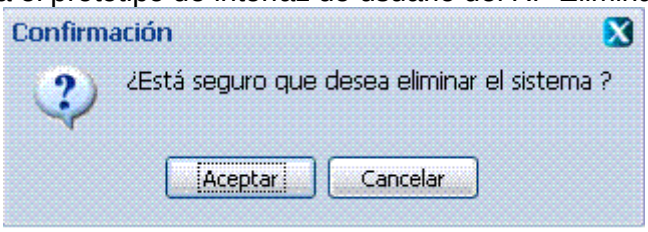

Figura 24. Eliminar sistema.

La Tabla 5 muestra la especificación del requisito funcional Importar sistema.

Tabla 5. Especificación del requisito Importar sistema.

Precondiciones		Se ha registrado al menos un sistema.
Flujo de eventos		
Flujo básico		
1	Se selecciona el sistema donde se importará el nuevo sistema.	
2	El sistema permite seleccionar el fichero a importar.	
3	El sistema valida el fichero (ver validación 1).	
4	El sistema importa y muestra una confirmación de importación.	
5	Concluye el requisito.	
Pos-condiciones		
1	N/A	
Flujos alternativos		
Flujo alternativo 3.a Los datos no son válidos para su importación.		
1	El sistema notifica al usuario que no se pueden importar los datos.	
Pos-condiciones		
1	N/A	
Flujo alternativo *.a El usuario cancela la acción		
1	Concluye el requisito.	
Pos-condiciones		
1	N/A	
Validaciones		
1	El fichero está en un formato válido para importarlo al sistema. El formato válido es: XML	
Relaciones	**Requisitos Incluidos**	Se selecciona el sistema donde se importará el nuevo sistema: Listar sistemas, en la agrupación Gestionar sistemas.
	Extensiones	N/A
Conceptos	**Sistema**	Visibles en la interfaz: Denominación Abreviatura Icono Descripción Utilizados internamente: N/A
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

La Figura 25 muestra el prototipo de interfaz de usuario del RF Importar sistema.

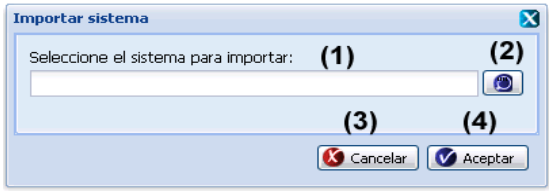

Figura 25. Importar sistema.

La Tabla 6 muestra la especificación del requisito funcional Exportar sistema.

Tabla 6. Especificación del requisito Exportar sistema.

Precondiciones		Se ha registrado al menos un sistema en el sistema.
Flujo de eventos		
Flujo básico		
1	El sistema permite seleccionar la ubicación hacia donde se exportará la información, el formato y registrar el nombre.	
2	El sistema valida los datos.	
3	El sistema exporta los datos.	
4	Concluye el requisito.	
Pos-condiciones		
1	Se ha creado en la ubicación especificada un fichero con el nombre y formato indicados.	
Flujo alternativo 3.a Información errónea		
1	El sistema señala los datos erróneos y permite corregirlos.	
2	El usuario corrige los datos.	
3	Volver al paso 2 del flujo básico.	
Pos-condiciones		
1	N/A	
Flujo alternativo 3.b Información incompleta		
1	El sistema señala los datos vacíos y permite corregirlos.	
2	El usuario corrige los datos.	
3	Volver al paso 2 del flujo básico.	
Pos-condiciones		
1	N/A	
Flujo alternativo *.a El usuario cancela la acción		
1	Concluye el requisito.	
Pos-condiciones		
1	No se crea el fichero.	
Flujos alternativos		
Flujo alternativo *.b No se pueden exportar los datos.		
1	El sistema notifica al usuario que los datos no pueden ser exportados.	
Pos-condiciones		
1	No se crea el fichero.	
Validaciones		
1	Se validan los datos según lo establecido en el Modelo conceptual CSG-ERP-N-SEG-i2201.	
Relaciones	**Requisitos Incluidos**	N/A
	Extensiones	N/A
Conceptos	**Sistema**	Visibles en la interfaz: Denominación Abreviatura Icono Descripción Utilizados internamente: N/A
Requisitos especiales	N/A	

Asuntos pendientes	N/A

La Figura 26 muestra el prototipo de interfaz de usuario del RF Exportar sistema.

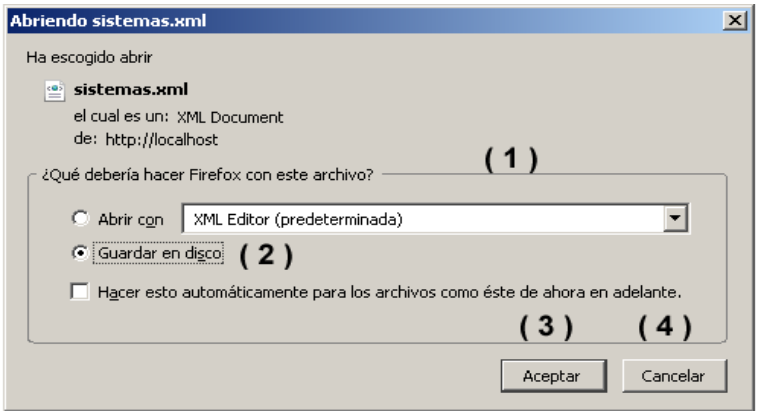

Figura 26. Exportar sistema.

La especificación de los demás requisitos se puede encontrar en la memoria general del proyecto Acaxia.[34]

2.3.2. REQUISITOS NO FUNCIONALES

Como se ha mencionado con anterioridad, el presente trabajo forma parte de un proceso productivo iniciado por el CEIGE y los resultados que se obtengan formarán parte del marco de trabajo desarrollado en el mismo. De esta manera los requisitos no funcionales a los que se debe acoger la aplicación a desarrollar son los que fueron establecidos por el centro al inicio del proceso de desarrollo, a continuación son descritos algunos de los más importantes.

Usabilidad (USB)

- ✓ El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora.

Rendimiento (REN)

- ✓ 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.

Seguridad (SEG)

- ✓ Autenticación (Contraseña de acceso.)
- ✓ Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.
- ✓ 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.
- ✓ Verificación sobre las acciones irreversibles (eliminaciones).

Portabilidad (POR)

✓ El sistema debe ser multiplataforma.

Soporte (SOP)

✓ 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.

Para el correcto funcionamiento del sistema se deben cumplir un conjunto de requisitos no funcionales de software y hardware los cuales se describen en la Tabla 7.

Requisitos Técnicos

Tabla 7. Requisitos de software y hardware.

Requerimientos	Cliente	Servidor (1)	Servidor (2)
Software	Mozilla Firefox 2.0.17	• Ubuntu Server. • Apache 2.0 • PHP 5	• Ubuntu Server. • PostgreSQL 8.3.
Hardware	• Procesador: 1.40 GHZ. • RAM: 256 MB (recomendado 512 Mb). • Tarjeta de Red: 1	• Procesador: 3.00 GHZ. • RAM: 1GB. • Disco duro: 160 GB. • UPS: 1. • Lector de CD: 1. • Tarjeta de Red: 1.	• Procesador: 3.00 GHZ. • RAM: 1GB. • Disco duro: 160 GB. • UPS: 1. • Lector de CD: 1. • Tarjeta de Red: 1.

2.4. MODELO DE DISEÑO

El Modelo de diseño es una abstracción del Modelo de Implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño. Es utilizado como entrada esencial en las actividades relacionadas a la implementación. El Modelo de Diseño puede contener: diagramas, clases, paquetes, subsistemas, cápsulas, protocolos, interfaces, relaciones, colaboraciones y atributos. Dentro de este epígrafe se verán los elementos que se tuvieron en cuenta durante el diseño de la solución, dando paso así a la exposición de los resultados de este flujo de trabajo, que refleja cómo será implementado el sistema en términos de clases del diseño.

2.4.1. TAXONOMÍA ESTRUCTURAL

La arquitectura es basada en capas aunque en las capas superiores, implementa un Modelo Vista Controlador (MVC). Está compuesto básicamente por cinco niveles o capas:
1. Capa de presentación: en esta capa se emplea las facilidades que brinda el Marco de Trabajo ExtJS para la construcción de interfaces amigables a la vista de los usuarios. Ext centra su desarrollo en tres componentes fundamentales JS-File, CSS-File y Client-page y Server-Page.
2. Capa de control o negocio: en esta capa se emplea el patrón de arquitectura Modelo Vista Controlador (MVC). De forma vertical al modelo descrito hasta este momento,

estarán los aspectos fundamentales del sistema así como el encargado del tratamiento de la seguridad a nivel de aplicación Web.
3. Capa de acceso a dato: en esta capa estará presente el ORM (Object Relational Mapping) Doctrine, como Marco de Trabajo de persistencia para la comunicación con el servidor de datos mediante el protocolo PDO, también estará un persistidor de configuración que es el encargado de comunicarse vía XML con los ficheros de configuración del sistema denominado FastResponse.
4. Capa de datos: en esta capa estará ubicado como servidor de base de datos PostgreSQL y un conjunto de ficheros de configuración de la Arquitectura Tecnológica.
5. Capa de servicio: en esta última capa se encuentran todos los subsistemas que prestan y consumen servicios entre sí.

Esta gran estructura descrita, se comunica con una series de servicios Web mediante protocolos SOAP, e IoC, que interactúan con el sistema proporcionándole un conjunto de funcionalidades tanto de seguridad como de negocio. Para obtener mayor información sobre los conectores, responsabilidad arquitectónica y demás aspectos pude dirigirse al Libro de Ayuda de la Arquitectura Tecnológica del ERP-Cuba.[35] La arquitectura descrita constituye la base para el desarrollo del sistema de autorización, la misma provee un conjunto de componentes genéricos reutilizables que garantizan un conjunto de escenarios arquitectónicos. La Figura 27 muestra la distribución de los componentes por niveles o capas, enfatizando en la integración entre componentes.

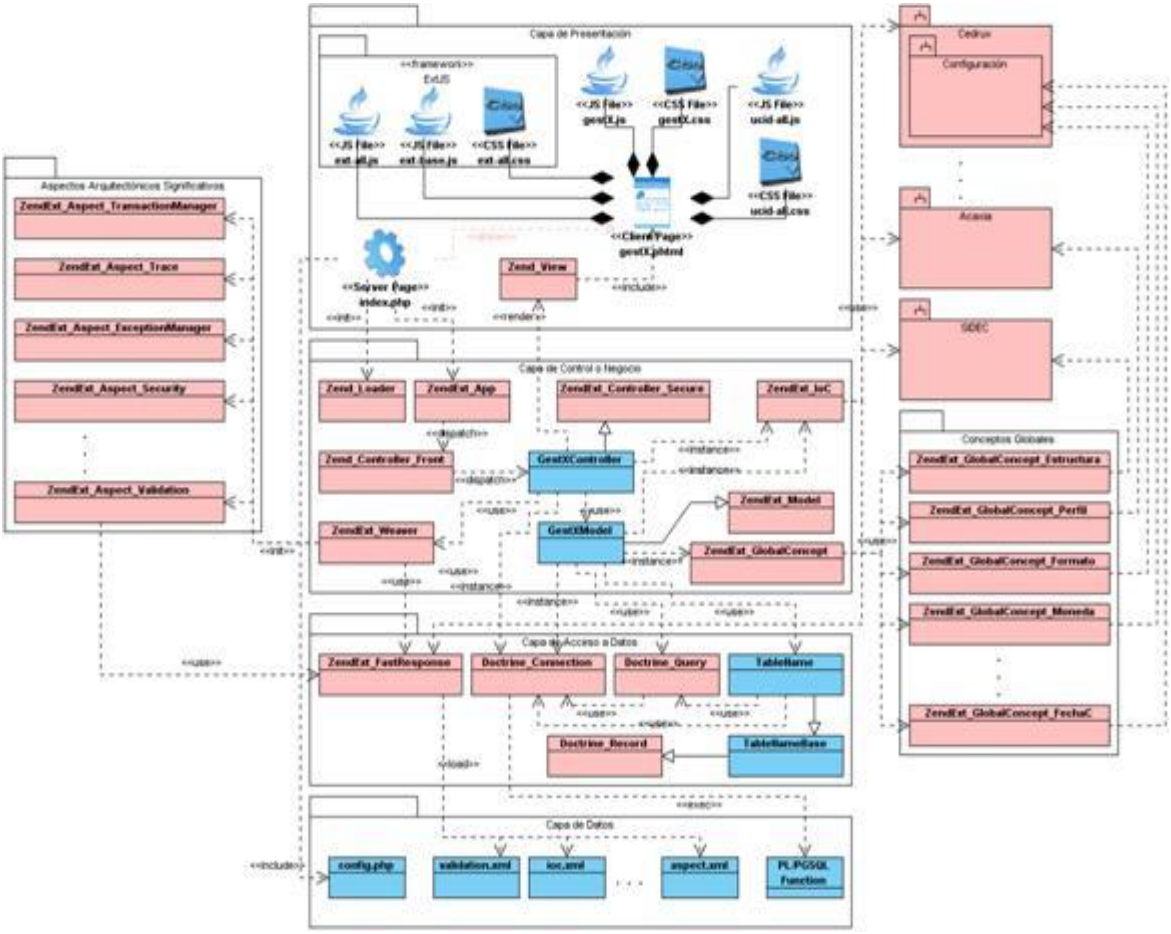

Figura 27. Taxonomía Arquitectónica

2.4.2. DIAGRAMA DE CLASES

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases del software y de las interfaces en una aplicación, contiene información como clases, asociaciones, atributos, métodos y dependencias. A continuación la Figura 28 muestra el diagrama de clases del diseño del requisito funcional Gestionar sistema realizado durante el proceso de desarrollo.

Diagrama de clases de Gestionar sistema

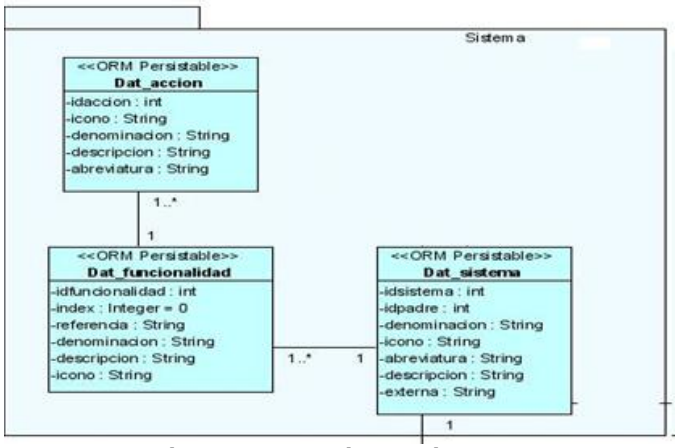

Figura 28. Gestionar sistema.

Para consultar el diagrama de clases de los demás procesos pude dirigirse la memoria colectiva del proyecto Acaxia.[34]

2.4.3. MODELO DE DATOS

Un modelo de datos es una colección de conceptos bien definidos matemáticamente que ayudan a expresar las propiedades estáticas y dinámicas de una aplicación con un uso de datos intensivo.[36]

En este epígrafe se muestra el modelo de datos. Este estará divido en tres partes que se corresponden con cada módulo del sistema, Configurar Sistemas, Configurar Usuarios, Configurar Nomencladores. Con este diagrama podemos comprender como está diseñada la Base de Datos.

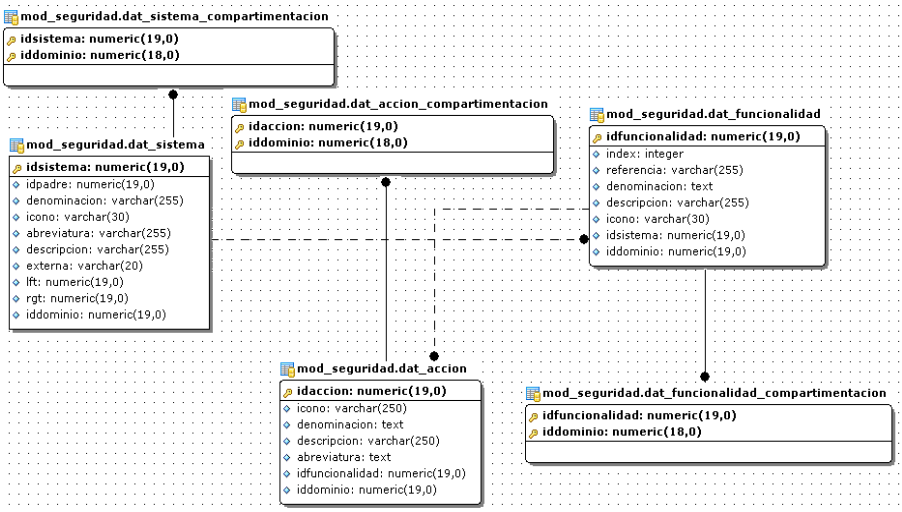

Figura 29. Configurar Sistemas

Para consultar el modelo de datos de los demás módulos puede dirigirse al documento ERP-ARQ-Modelo de datos de Acaxia.[37]

2.5. 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 las aplicaciones de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros, además de los recursos necesarios para poder ejecutar el sistema desarrollado.

2.5.1. DIAGRAMA DE COMPONENTES

La solución propuesta consta de una serie de componentes que interactúan entre sí. A continuación la Figura 30 muestra el diagrama de componentes elaborado. Para consultar el diagrama de componentes general del sistema, se puede consultar la memoria colectiva del proyecto Acaxia.[34]

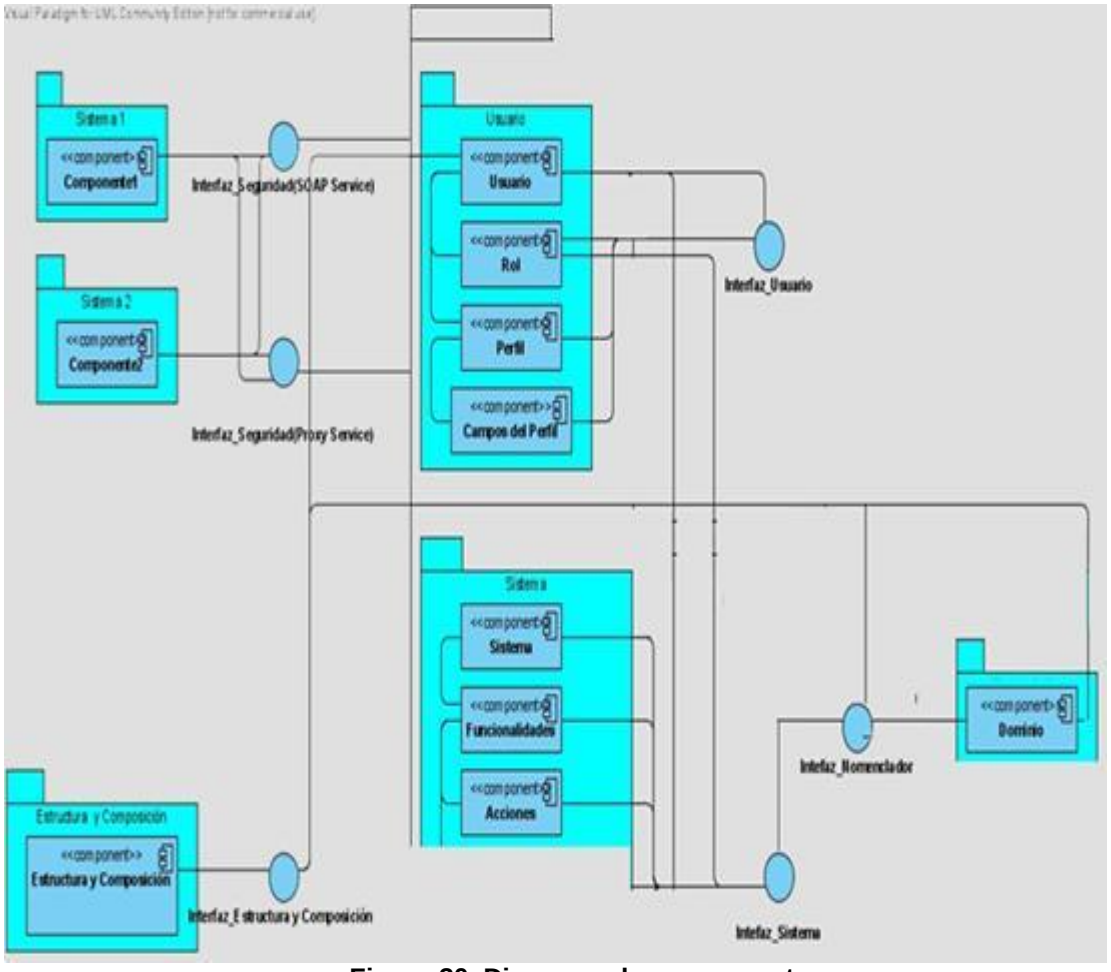

Figura 30. Diagrama de componentes.

2.6. CONCLUSIONES PARCIALES

En este capítulo fueron expuestos los artefactos generados durante el diseño de la solución propuesta. En el mismo se identificaron y describieron los requisitos funcionales y no funcionales, se conformó la arquitectura base que rige el diseño, el modelo de datos, el diagrama de clases del diseño propuesto por los autores, el diagrama de componentes. El diseño e implementación del sistema se tornó complejo, puesto que no se contaba con especialistas funcionales que dominaran el tema. El equipo de desarrollo se comportó como usuario y desarrollador, los requisitos fueron identificándose en el transcurso del desarrollo. Por la necesidad imperiosa de dar una respuesta inmediata, se dejaron de implementar algunas funcionalidades de menor nivel de complejidad tales como otros drivers de autenticación, gestión de servicios y procesos dinámicos. Con el desarrollo de este sistema se adquirieron conocimientos sobre la seguridad informática y el desarrollo de aplicaciones web de gestión. Una vez concluido el diseño e implementación de la solución puede darse paso a la validación de la solución.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN

En el presente capítulo se realizarán las pruebas de liberación, se evaluará el diseño del sistema aplicando métricas de software, se realizará un estudio de los beneficios que ha traído consigo la aplicación de la solución en entornos reales, se hará un estudio del impacto económico y social producto de la reutilización, ahorrando recursos, tiempo y esfuerzo en el desarrollo de nuevas soluciones o brindándole seguridad a sistemas existentes. Por último se mostrarán los resultados obtenidos en eventos y publicaciones científicas. Este capítulo tiene como objetivo evaluar y validar la calidad del sistema aplicando un conjunto de técnicas como las que se explicarán a continuación.

3.1. PRUEBAS DE LIBERACIÓN

Las pruebas del software son un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación. Dichas pruebas son realizadas con el objetivo de detectar errores en el sistema, por lo que se llevan a cabo durante todo el ciclo de vida del producto. Los casos de prueba especifican una forma de probar el sistema, incluyendo las entradas con las que se ha de probar, las condiciones bajo las que ha de probarse, así como los resultados esperados.[38]

Para la validación del sistema se utilizó el método de validación por medio de pruebas de caja negra realizadas por el equipo de calidad del CEIGE y Calisoft. Estas pruebas se basan en la especificación del programa o componente a ser probado para elaborar los casos de prueba. El componente es visto como una “caja negra” cuyo comportamiento es desconocido y sólo puede ser evaluado estudiando sus entradas y las salidas obtenidas.

La realización de los casos de pruebas es un tipo de prueba de caja negra y tiene como objetivo demostrar al cliente la reacción que corresponderá por parte del sistema luego de realizar alguna acción en el mismo.[39] Para un mejor entendimiento de las respuestas o posibles funcionalidades que brindará el sistema, según la necesidad del usuario. Para ello se diseñaron un conjunto de escenarios de pruebas utilizados por el equipo de calidad de CEIGE y Calisoft con el objetivo de evaluar el cumplimiento de los requisitos del sistema. A continuación se muestra la descripción de los escenarios que prueban cada uno de los requisitos funcionales.

Diseño de caso de prueba: Adicionar sistema

La Tabla 8 describe los escenarios de prueba y dentro de ellos cada uno de los flujos que pueden desencadenarse para probar el correcto funcionamiento de un requisito funcional. En este caso se probará el requisito funcional Registrar sistema.

Condiciones de ejecución

- Se tienen los permisos necesarios para realizar esta operación.
- El usuario se debe encontrar en el subsistema Seguridad, en el módulo Configurar sistemas, en la interfaz Sistemas.
- El sistema que se desea adicionar no ha sido adicionado antes al sistema.

Tabla 8. Requisitos a probar.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario

1: Registrar sistema	Se adiciona un nuevo sistema.	EP 1.1: Registrar sistema	– Se escoge el subsistema al que se le desea agregar el subsistema. – Se presiona el botón **Adicionar.** – Se insertan todos los datos. – Se presiona el botón **Aceptar**.
		EP 1.2: Registrar sistema dejando campos requeridos en blanco.	– Se escoge el subsistema al que se le desea agregar el subsistema. – Se presiona el botón **Adicionar**. – Se introducen los datos dejando campos requeridos en blanco. – Se presiona el botón **Aceptar**.
		EP 1.3: Registrar sistema introduciendo error en los datos.	– Escoger el subsistema al que se le desea agregar el subsistema. – Se presiona el botón **Adicionar**. – Se introducen los datos del sistema que se desea adicionar en el formulario introduciendo errores en los datos. – Se presiona el botón **Aceptar**.
		EP 1.4 Aplicar	– Escoger el subsistema al que se le desea agregar el subsistema. – Se presiona el botón **Adicionar**. – Se introducen los datos del sistema que se desea adicionar en el formulario. – Se presiona el botón **Aplicar**. – Se presiona el botón **Aceptar**.
		EP 1.5: Cancelar	– Escoger el subsistema al que se le desea agregar el subsistema. – Se presiona el botón **Adicionar**. – Se introducen o no los datos en el formulario. – Se presiona el botón **Cancelar**.

La Tabla 9 describe cada una de las variables de entrada del requisito Registrar sistema.

Tabla 9. Descripción de variable

No	Nombre de campo	Clasificación	Puede ser nulo	Descripción
1	Denominación	Campo de texto	No	Combinación de letras.
2	Abreviatura	Campo de texto	No	Combinación de letras.
3	Icono	Campo de texto	Si	Combinación de letras.
4	Servidores	Campo de texto	Si	Combinación de números.
5	Descripción	Campo de texto	Si	Combinación de letras.
6	Esquema	Árbol	Si	Cuadro de selección

La Tabla 10 especifica los posibles escenarios a ser ejecutados sobre el sistema. De cada uno de los escenarios se describen los datos de entrada y la respuesta del sistema.

Tabla 10. Juegos de datos a probar

Id del escenario	Escenario	Denominación	Abreviatura	Icono	Servidores	Descripción	Esquema	Respuesta del sistema
EP1.1	Registrar sistema.	V(sist seguridad)	V(seg)	V(seg34)	V(Ldap)	V(seguridad)	V(publict)	Se adiciona el sistema y se guarda la información adicionada en el módulo de configurar sistemas.

EP1.2	Registrar sistema dejando campos requeridos en blanco.	I(vacío)	V(seg)	V(seg34)	V(Ldap)	V(seguridad)	V(publict)	Se muestra el campo de texto en rojo que indica que no se pueden dejar campos en blanco y se mantiene en la ventana Adicionar sistema. Los campos Denominación y Abreviatura son los únicos que no pueden ser nulos.
		V(sist seguridad)	I(vacío)	V(seg34)	V(Ldap)	V(seguridad)	V(publict)	
		V(sist seguridad)	V(seg)	I(vacío)	V(Ldap)	V(seguridad)	V(publict)	
		V(sist seguridad)	V(seg)	V(seg34)	I(vacío)	V(seguridad)	V(publict)	
		V(sist seguridad)	V(seg)	V(seg34)	V(Ldap)	I(vacío)	V(publict)	
		V(sist seguridad)	V(seg)	V(seg34)	V(Ldap)	V(seguridad)	I(vacío)	
EP 1.3	Registrar sistema introduciendo error en los datos.	I(-*/)	V(seg)	V(seg34)	V(Ldap)	V(seguridad)	V(publict)	Se muestra el campo de texto en rojo que indica que no se pueden entrar caracteres inválidos y se mantiene en la ventana Adicionar sistema. El campo descripción admite todo tipo de caracteres.
		V(sist seguridad)	I(-*/)	V(seg34)	V(Ldap)	V(seguridad)	V(publict)	
		V(sist seguridad)	V(seg)	I(-*/)	V(Ldap)	V(seguridad)	V(publict)	
		V(sist seguridad)	V(seg)	I(vacío)	V(120)	V(seguridad)	V(publict)	
		V(sist seguridad)	V(seg)	V(seg34)	V(Ldap)	I(/*-)	V(publict)	

		V(sist seguridad)	V(seg)	V(seg34)	V(Ldap)	V(seguridad)	I(250)	
EP1.4	Aplicar	V(sist seguridad)	V(seg)	V(seg34)	V(Ldap)	V(seguridad)	V(publict)	Se adiciona el sistema y se guarda la información adicionada en el módulo de configurar sistemas.
		I(vacío)	V(seg)	V(seg34)	V(Ldap)	V(seguridad)	V(publict)	Se muestra el campo de texto en rojo que indica que no se pueden dejar campos en blanco y se mantiene en la ventana Adicionar sistema. Los campos Denominación y Abreviatura son los únicos que no pueden ser nulos.
		V(sist seguridad)	I(vacío)	V(seg34)	V(Ldap)	V(seguridad)	V(publict)	
		V(sist seguridad)	V(seg)	I(vacío)	V(Ldap)	V(seguridad)	V(publict)	
		V(sist seguridad)	V(seg)	V(seg34)	I(vacío)	V(seguridad)	V(publict)	
		V(sist seguridad)	V(seg)	V(seg34)	V(Ldap)	I(vacío)	V(publict)	
		V(sist seguridad)	V(seg)	V(seg34)	V(Ldap)	V(seguridad)	I(vacío)	
		I(-*/)	V(seg)	V(seg34)	V(Ldap)	V(seguridad)	V(publict)	Se muestra el campo de texto en rojo que indica que no se pueden entrar caracteres inválidos y se mantiene en la ventana Adicionar sistema. Los campos Denominación y Abreviatura son los únicos que no pueden
		V(sist seguridad)	I(-*/)	V(seg34)	V(Ldap)	V(seguridad)	V(publict)	
		V(sist seguridad)	V(seg)	I(-*/)	V(Ldap)	V(seguridad)	V(publict)	

		V(sist seguridad)	V(seg)	I(vacío)	V(120)	V(seguridad)	V(publict)	ser nulos.
		V(sist seguridad)	V(seg)	V(seg34)	V(Ldap)	I(/*-)	V(publict)	
		V(sist seguridad)	V(seg)	V(seg34)	V(Ldap)	V(seguridad)	I(250)	
EP1.5	Cancelar	NA	NA	NA	NA	NA	NA	Se cancela la operación.

Para consultar los demás escenarios de pruebas, puede dirigirse al capítulo 3 de la memoria general de Acaxia.[34]
En las pruebas realizadas al sistema y al manual se detectaron un total de 97 no conformidades, la Figura 31. No conformidades por iteración. muestra una gráfica donde se refleja la cantidad de no conformidades detectadas por iteración. El sistema fue liberado en la tercera iteración y el manual en la quinta iteración. Finalmente CALISOFT emitió su certificado de liberación que se puede encontrar en la memoria colectiva del proyecto Acaxia.[34]

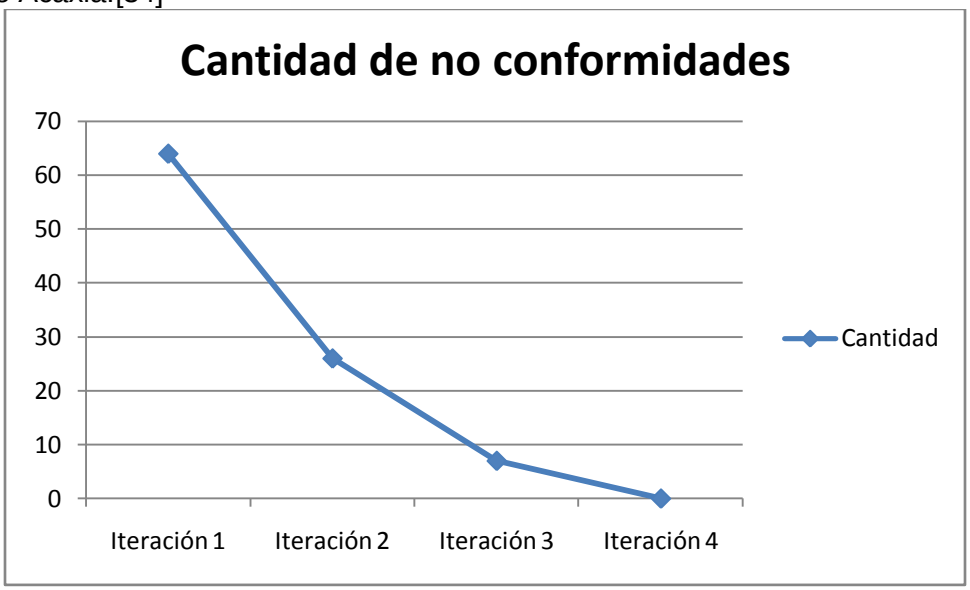

Figura 31. No conformidades por iteración.

Una vez liberado el sistema se inició el proceso de registro para proteger el derecho de autoría en el CENDA, para consultar los documentos emitidos por esta entidad registradora puede remitirse a los anexos de la memoria colectiva de Acaxia.[34]

3.2. EVALUACIÓN DEL DISEÑO APLICANDO MÉTRICAS DE SOFTWARE

Las métricas de software son una medida cuantitativa que permite a los desarrolladores tener una visión profunda de la eficacia del proceso del software y de los proyectos que

dirigen utilizando el proceso como un marco de trabajo. Se reúnen los datos básicos de calidad y productividad. Estos datos son entonces analizados, comparados con promedios anteriores, y evaluados para determinar las mejoras en la calidad y productividad. Las métricas son también utilizadas para señalar áreas con problemas de manera que se puedan desarrollar los remedios y mejorar el proceso del software.[39]
Las métricas empleadas están diseñadas para evaluar los siguientes atributos de calidad:

- ✓ **Responsabilidad.** Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- ✓ **Complejidad de implementación:** consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- ✓ **Reutilización:** consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- ✓ **Acoplamiento:** consiste en el 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:** consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirectamente, pero fuertemente en los costes y la planificación del proyecto.
- ✓ **Cantidad de pruebas:** consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (unidad) del producto (componente, módulo, clase, conjunto de clases, entre otras) diseñado.

Las métricas concebidas como instrumento para evaluar la calidad del diseño del Sistema de Gestión Integral de Seguridad y su relación con los atributos de calidad definidos en este trabajo son las siguientes:
Tamaño operacional de clases (TOC): está dado por el número de métodos asignados a una clase.
La Tabla 11 describe los atributos que se evalúan al aplicar la métrica TOC.

Tabla 11.Tamaño operacional de clase (TOC).

Atributo de calidad	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Relaciones entre clases (RC): esta dado por el número de relaciones de uso de una clase con otra.
La Tabla 12 describe los atributos de calidad que se miden al evaluar la métrica RC.

Tabla 12.Relaciones entre clases (RC).

Atributo de calidad	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del Acoplamiento de la clase.
Complejidad de mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
Reutilización	Un aumento del RC implica una disminución en

	el grado de reutilización de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

3.2.1. RESULTADOS DE LA APLICACIÓN DE LA MÉTRICA TOC

Ver instrumentos y tabla de resultados del instrumento de medición de la métrica tamaño operacional de clase (TOC) en los anexos de la memoria colectiva del proyecto Acaxia.[34]

La Figura 32 muestra la representación de los resultados obtenidos en el instrumento agrupados en los intervalos definidos. El gráfico refleja que la mayoría de las clases tienen de uno a cinco procedimientos. Este resultado demuestra que el funcionamiento general del sistema está distribuido equitativamente entre la mayoría de componentes y es necesario analizar los componentes que contienen la mayor cantidad de métodos o responsabilidad para tratar de restarle funcionalidades y distribuirlas entre los demás componentes y así garantizar que ningún componente sea demasiado crítico para en caso de fallo sea menor el número de funcionalidades que queden fura de servicios.

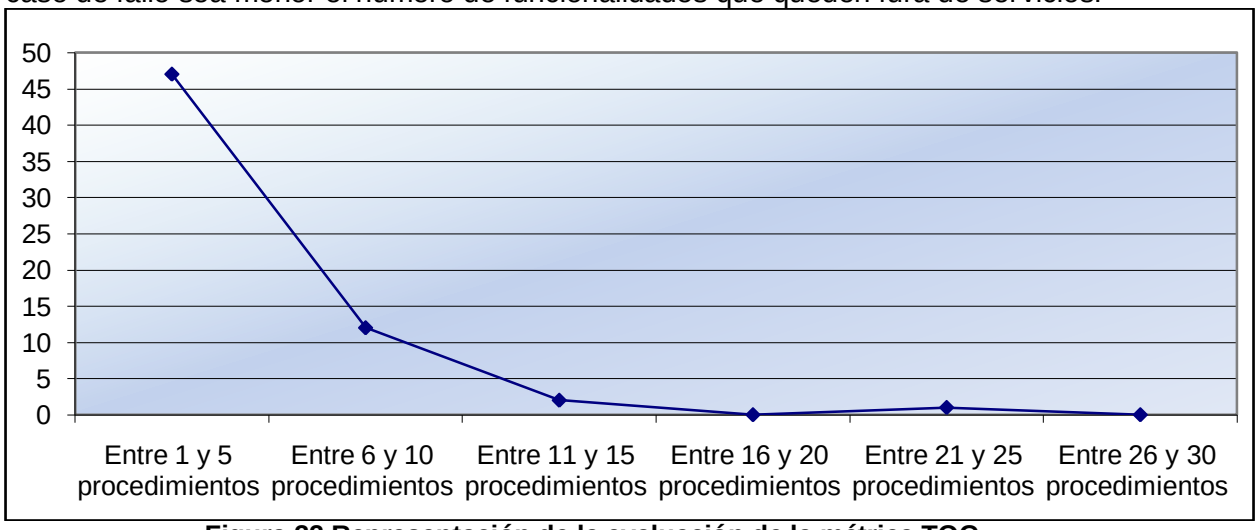

Figura 32.Representación de la evaluación de la métrica TOC.

La Figura 33 muestra la representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

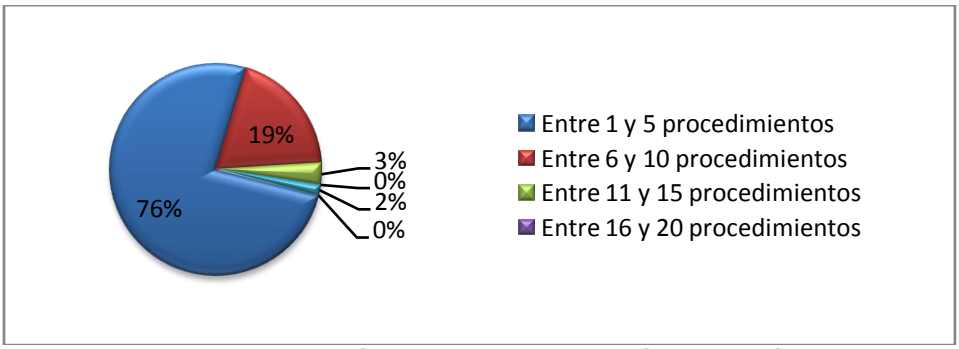

Figura 33. Representación en % de la evaluación de la métrica TOC.

La Figura 34 muestra la representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo responsabilidad. Quedó demostrado que el 68% de las clases tienen una baja responsabilidad ya que este tributo se distribuyó equitativamente entre todas las clases del sistema. Esta característica permite que en caso de fallos como la responsabilidad está distribuida de forma equilibrada ningún componente sea demasiado crítico como para dejar fuera de servicio el sistema. Es necesario especificar que existen componentes con responsabilidad fuera de la media que deben analizarse para tratar de distribuir la misma entre todos los componentes.

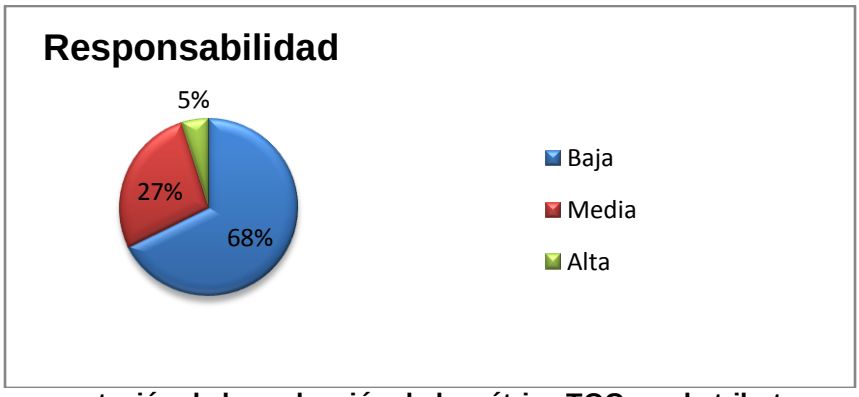

Figura 34. Representación de la evaluación de la métrica TOC en el atributo responsabilidad.

La Figura 35 muestra la representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación. El gráfico muestra que la mayoría de las clases tienen una baja complejidad, este atributo está distribuido equitativamente. Esta característica permite mejorar el mantenimiento y soporte de estos componentes. Existe un 28% de los componentes que deben ser analizados para tratar de dividir y distribuir sus funciones y así disminuir su complejidad.

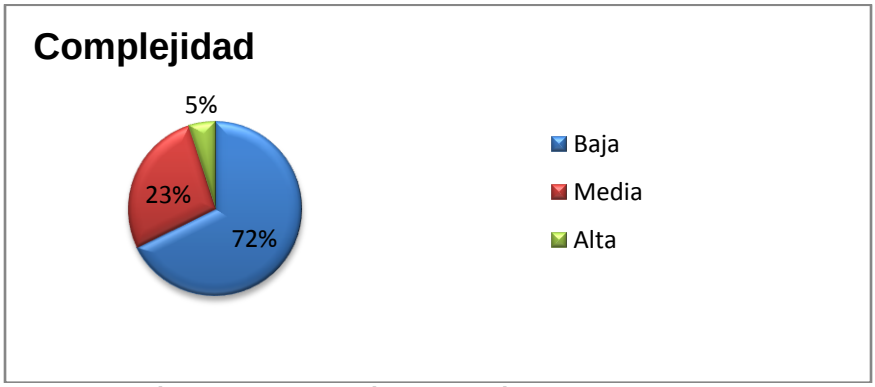

Figura 35. Representación de la evaluación de la métrica TOC en el atributo complejidad de implementación.

La Figura 36 muestra la representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo reutilización. Queda demostrado que el diseño de la solución es eficiente ya que todos los componentes tienen un alto grado de reutilización.

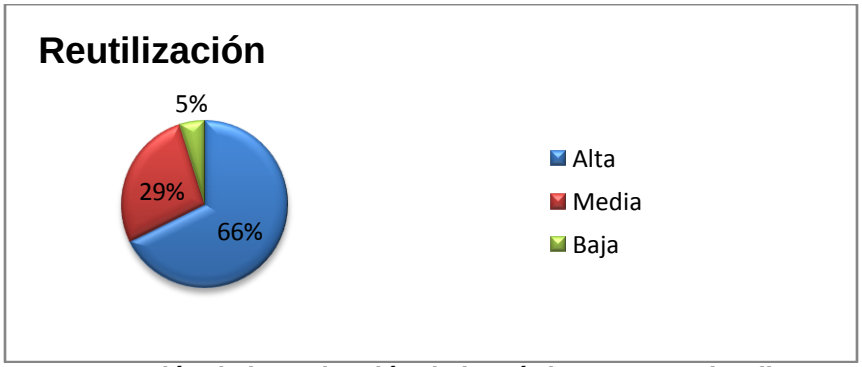

Figura 36. Representación de la evaluación de la métrica TOC en el atributo reutilización.

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica TOC, se puede concluir que el diseño del Sistema de Gestión Integral de Seguridad tiene una calidad aceptable teniendo en cuenta que el 93 % de las clases incluidas en este sistema posee menos cantidad de operaciones que la mitad del valor máximo registrado en las mediciones. Además el 97% de las clases poseen evaluaciones positivas en los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización).

3.2.2. RESULTADOS DE LA APLICACIÓN DE LA MÉTRICA RC

Ver instrumentos y tabla de resultados de la medición de la métrica Relaciones entre clases (RC) en la memoria colectiva del proyecto Acaxia.[34]
La Figura 37 muestra la representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos. Se demuestra que el 69% de las clases tienen una dependencia, encapsulan en sí mismo la responsabilidad del funcionamiento. Existe un 31% que dependen de otros componentes para su funcionamiento, luego de un análisis quedó demostrado que no es posible eliminar estas dependencias por la responsabilidad arquitectónica que deben mantener los componentes.

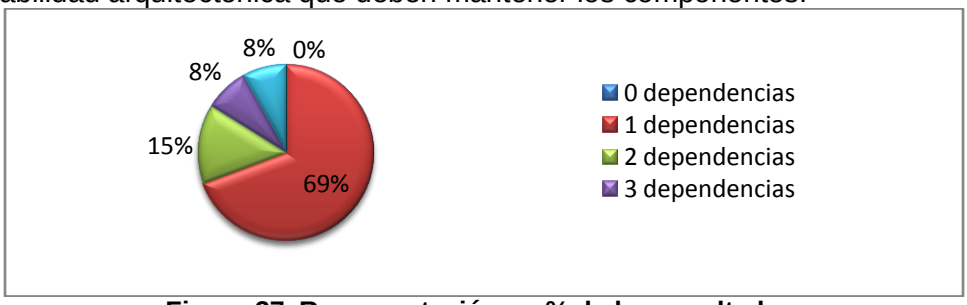

Figura 37. Representación en % de los resultados.

La Figura 38 muestra la representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento. Se evidencia un diseño eficiente al quedar reflejado que los componentes cuentas con un bajo acoplamiento. Aunque existe un 36% que debe ser analizado profundamente para identificar si es posible o no disminuir el acoplamiento y así ganar en modularidad.

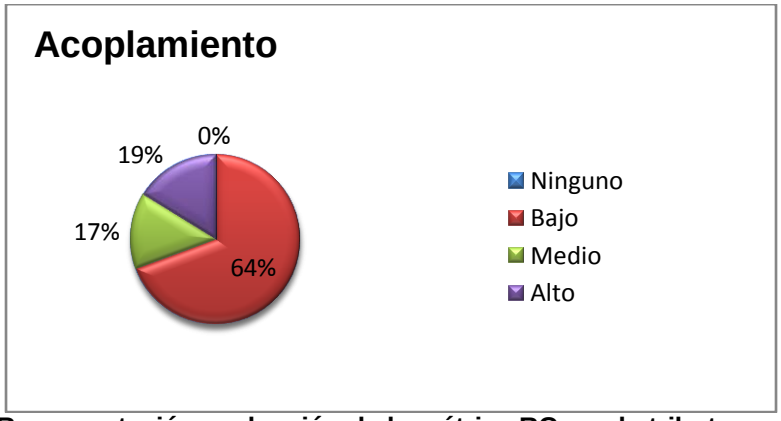

Figura 38. Representación evaluación de la métrica RC en el atributo acoplamiento.

La Figura 39 muestra la Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento. Queda demostrada la eficiencia de la arquitectura del sistema al evidenciarse una baja complejidad de mantenimiento. Después de concluir el análisis de los aspectos evaluados en la métrica TOC este atributo puede mejorar su comportamiento y así disminuir la complejidad del soporte o mantenimiento.

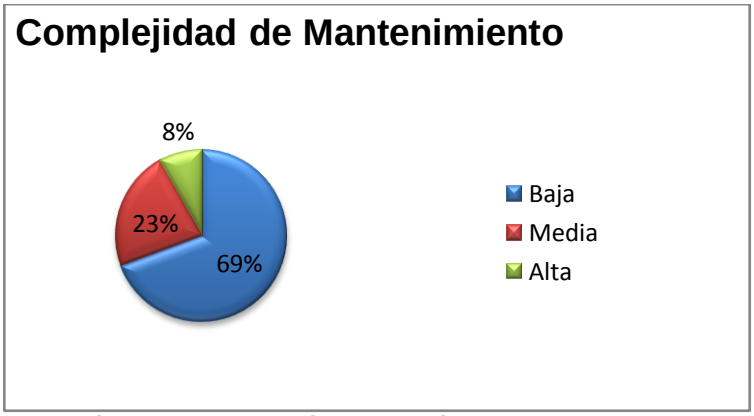

Figura 39. Representación de la evaluación de la métrica RC en el atributo complejidad de mantenimiento.

La Figura 40 muestra la representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas.

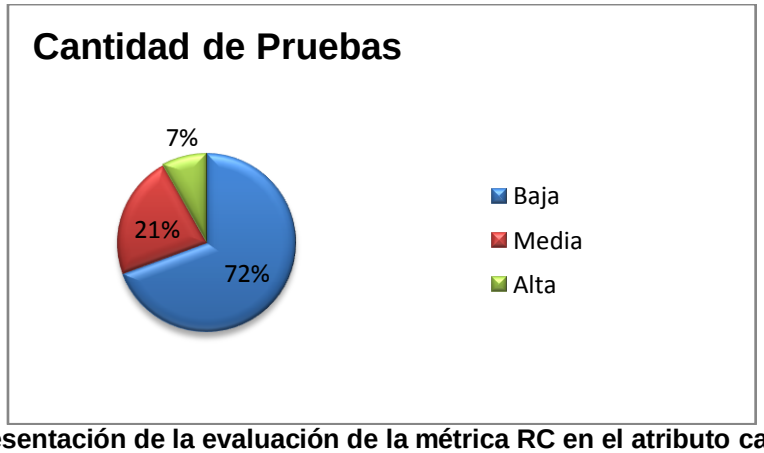

Figura 40. Representación de la evaluación de la métrica RC en el atributo cantidad de pruebas.

La Figura 41 muestra la representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización.

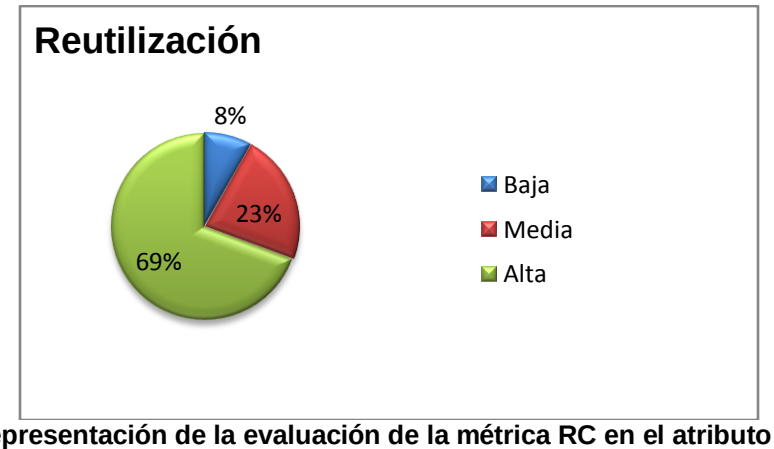

Figura 41. Representación de la evaluación de la métrica RC en el atributo reutilización.

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño del Sistema de Gestión Integral de Seguridad tiene una calidad aceptable teniendo en cuenta que el 90 % de las clases incluidas en estos subsistemas posee menos de 3 dependencias de otras clases. Además el 95% de las clases posee índices aceptables en cuanto a Acoplamiento. Así mismo los atributos de calidad Complejidad de Mantenimiento, Cantidad de Pruebas y Reutilización se comportan satisfactoriamente en un 96 % de las clases.

3.2.3. MATRIZ DE INFERENCIA DE INDICADORES DE CALIDAD

La matriz inferencia de indicadores de calidad, también llamada matriz de cubrimiento, es una representación estructurada de los atributos de calidad y métricas utilizadas para evaluar la calidad del diseño propuesto. Dicha matriz permite conocer si los resultados obtenidos de las relaciones atributo/métrica es positivo o no, llevando estos resultados a una escalabilidad numérica donde, si los resultados son positivos se le asigna el valor de 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.

A continuación se muestran los resultados obtenidos.

Tabla 13. Resultados de la evaluación de la relación atributo/métrica.

Atributos/Métricas	TOC	RPC	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

Tabla 14. Rango de valores para la evaluación técnica de los atributos de calidad evaluados por cada métrica.

Categoría	Rango de valores
Malo	<= 0.4
Regular	>0.4 y <=0.7
Bueno	>0.7

La Figura 42 muestra la gráfica de los resultados obtenidos de los atributos de calidad evaluados en las métricas aplicadas anteriormente, donde todos los atributos de calidad mantienen un buen comportamiento.

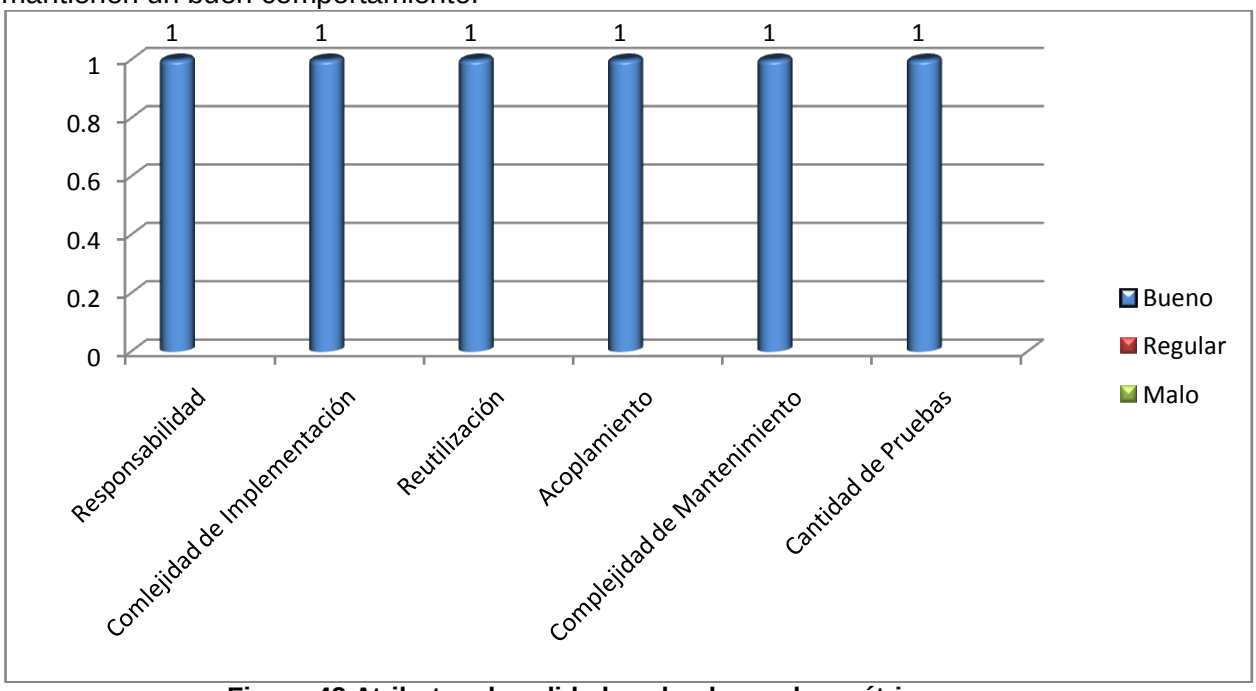

Figura 42.Atributos de calidad evaluados en las métricas.

3.3. IMPACTO Y APORTE EN LA PRODUCCION D SOFTWARE

Como se había especificado anteriormente, este componente forma parte del sistema Acaxia, el mismo se encuentra generalizado en un conjunto de 23 entidades en el contexto nacional. La versión 1.0 beta del sistema se concluyó el 24 de febrero del 2009, desde ese momento se ha generalizado en una gran cantidad de entidades y proyectos de desarrollo de software. La utilización del sistema durante este tiempo ha permitido identificar un conjunto de requisitos que se han venido incorporando en la versión 1.5 alfa y 2.0 alfa.

Cada entidad en la que se encuentra generalizada la aplicación tiene diferente función social, entrono tecnológico y otras diferencias que pueden atentar contra el correcto funcionamiento del sistema. A estas se les puede adicionar la complejidad que representa que en muchas de las entidades existe una gran variedad de sistemas desarrollados sobre diferentes tecnologías y la mayoría de ellos incorpora su propio mecanismo de seguridad, esto trae como consecuencia que los usuarios tengan que memorizar los usuarios y contraseñas para acceder a ellos.

En el caso que una entidad se encargue de brindar la infraestructura tecnológica a otra para que esta pueda gestionar algún proceso utilizando para ello un sistema y este se utilice en las dos entidades, es necesario crear dos espacios diferentes a nivel de sistema y de base de datos. Esta situación es un escenario no resuelto por el sistema de seguridad de la aplicación que no es capaz de mantener la compartimentación de la información en un entorno multientidad y multisistema.

La solución propuesta permite centralizar la seguridad de todas las aplicaciones existentes, de esta forma la responsabilidad de proteger la información recae sobre un único sistema. La versatilidad y capacidad de adaptación de Acaxia ha permitido que las

entidades y proyectos que a continuación se mencionan puedan centralizar la seguridad en ambientes heterogéneos.

- ✓ Entidades
  - ICID.
  - Yuri Gagari.
  - Centro de Gestión.
  - CubaTaxi.
  - Rafael Trejo.
  - UCI.
  - MINFAR.
  - UCID.
  - MEP.
  - MAC.

Además se usan en 12 entidades desarrolladoras de software del país con el objetivo de que los sistemas que desarrollan cuenten con una seguridad robusta, confiable y altamente configurable. A continuación se listan algunas de las entidades y proyectos que reutilizan la solución con el fin de desarrollar nuevos sistemas.

- ✓ UCI
  - Cedrux (CEIGE).
  - GINA (CEIGE).
  - Generador Dinámico de Reportes GDR (Datec).
  - Sistema de Gestión Estadística (Datec).
  - Minería de datos (Datec).
  - Sistema de auditoría y control (SIGAC) del MAC (Facultad 2).
  - Fuerza de Trabajo Calificada del MEP (Facultad 1).
- ✓ UCID-FAR
  - Sistema de apoyo a la toma de decisiones SIMEM.
  - Sistema de representación geoespacial LiberGIS.
  - Sistema de supervisión y control de los PSI (Hoyo).
  - Sistema para la gestión de las transportaciones en las FAR (DITRANS).
- ✓ Entidades
  - Aduana.
  - TRANSOFT.
  - DESOFT (Camagüey, Habana, Holguín, etc.).
  - MININT.
  - CIGB.
  - MINFAR.
  - Academia de las FAR.
  - Centro de desarrollo de Holguín.

La reutilización del sistema en los proyectos y entidades mencionados ha permitido ahorrar cuantiosos recursos humanos y materiales puesto que adquirir un sistema de este tipo en el mercado internacional resultaría costoso y si se decide implementar un sistema que garantice la seguridad de cada una de las aplicaciones que se desarrollen gastaríamos millones de dólares innecesariamente y nunca se lograría estandarizar este proceso. A continuación se mencionan algunas de las ventajas que han permitido que el sistema tenga una gran aceptación.

- ✓ Garantiza la seguridad de numerosos sistemas que manejan información valiosa.
- ✓ Ha tenido un gran impacto social en la formación de desarrolladores.
- ✓ Ha posibilitado a los usuarios finales del sistema una rápida adquisición de conocimientos sobre la seguridad.

- ✓ Incorpora seguridad desde los inicios del desarrollo.
- ✓ Cumple con estándares internacionales para sistemas de este tipo.
- ✓ Cuanta con una configuración sencilla y flexible.
- ✓ Permite la configuración dinámica del lenguaje en el que se va a mostrar el sistema.
- ✓ Incorpora los conceptos multientidad, multisistema, multi-tema y multi-escritorio.
- ✓ Incorpora la administración de perfiles de usuarios.
- ✓ Incorpora la administración de conexiones.
- ✓ Permite la autenticación contra LDAP, OpenLDAP y el sistema.
- ✓ Permite configurar la integración con otros sistemas de forma visual.

Los anexos de la memoria colectiva de Acaxia[34] muestran las actas de aceptación firmadas por proyectos y entidades en las que se reflejan las ventajas que brinda reutilizar este sistema.

3.3.1. IMPACTO ECONÓMICO

En el presente año, el CEIGE ha firmado y espera firmar un conjunto de convenios importantes que deben aportar al país grandes sumas de dinero. Acaxia y dentro de ella el componente de autorización formará parte de cada una de las soluciones a comercializar para garantizar la seguridad del sistema y de los datos. A continuación se realiza un análisis de costo y beneficio que evidencia la factibilidad de esta solución a partir de los proyectos de exportación en ejecución y las perspectivas futuras.

Proyectos de exportación en ejecución 2010

- ✓ Misión UCI en Venezuela
  - Mantenimiento Venezuela. (1.3 millones).
  - Sistema para gestión de medicamentos. (1.9 millones).
  - Sistema de gestión para Angola. (9 millones).

Proyectos Potenciales 2010 – 2011

- ✓ Misión UCI en Venezuela
  - EM. Guardián del Alba. Sistema logístico.
  - EM SIME-PDVSA.
  - Despliegue Cedrux fase 1.
  - Despliegue Cedrux fase 2.

Si se toma en cuenta que como se había expresado anteriormente, Acaxia formará parte de cada una de las soluciones a comercializar, que el costo de desarrollo de la misma es de 220103.5 CUP que significa un 1.8 % del total de los ingresos de negocios concertados y que Acaxia representa al menos un 8% del total de las soluciones a entregar a los clientes, se puede inferir que desde el primer convenio se sufragan todos los gastos y se obtienen ganancias. Si a este análisis se le incorpora el valor por reducción de importaciones, informatización de la sociedad y reutilización, el valor de la solución aumenta vertiginosamente. Para obtener más detalles de los recursos empleados en el desarrollo del proyecto puede consultar la Tabla 15.

Tabla 15. Recursos empleados.

Elementos de Gastos	Año 2008	Año 2009	Año 2010	Total
Salario	$17990	$15420	$10280	$43690
Salario complementario (9,09 % del salario total anual	$1635.3	$1401.7	$934.5	$3971.5
Seg. Social (hasta 14% del total de salarios).	----------	----------	----------	----------

Subtotal	**$19625.3**	**$16821.7**	**$11214.5**	**$47661.5**
Recursos materiales	$177750			$177750
Subcontrataciones	$4000			$4000
Otros recursos Nota 2	-------			-------
Subtotal	**$181750**			**$181750**
Total Gastos Directos Nota 3	**$218197**			**$218197**
Gastos Indirectos Nota 4	0.04*$19625.3= $785	0.04*$16821.7= $672.9	0.04*$11214.5= $448.6	**$1906.5**
Total Gastos Nota 5	**$220103.5**			**$220103.5**

3.4. RESULTADOS EN EVENTOS

El sistema obtuvo premio relevante en el Fórum Científico Técnico a nivel de Universidad y a nivel municipal y destacado a nivel provincial, además fue presentado en UCIENCIA 2010, 1er Taller nacional de Interoperabilidad organizado por el CEIGE, entre otros. Para consultar los certificados y publicaciones puede dirigirse a los anexos de la memoria colectiva de Acaxia.[34]

3.5. CONCLUSIONES PARCIALES

Del proceso de pruebas de funcionamiento al que fue sometido el sistema se obtuvo como resultado que existía desconocimiento por parte del equipo de desarrollo de los estándares definidos para el diseño de interfaz de usuario, por esta razón en la primera revisión fueron identificadas 64 no conformidades. A partir de este momento el equipo realizó un estudio profundo de los estándares definidos y los aplicó en la resolución de las no conformidades. Las 26 no conformidades detectadas en la segunda iteración y las 7 detectadas en la tercera, demostraron que se gana en experiencia y responsabilidad. Los resultados al aplicar las métricas TOC, RC y la matriz de cubrimiento o matriz de inferencia de indicadores de calidad evidenció que existen componentes que sostienen relaciones innecesarias con otros componentes que puedes ser eliminadas en futuras versiones para aumentar la modularidad y disponibilidad del sistema, además existen componentes que cuentan con un gran número de métodos que pueden distribuirse en otros componentes para disminuir la responsabilidad y de esta forma en caso que falle este componente el número de funcionalidades que queden fuera de servicio será menor. Se expusieron los resultados obtenidos por medio de la generalización del producto en los proyectos de la UCI y demás entidades del país, además se realizó un análisis del aporte de la solución en los ingresos por concepto de exportaciones y por disminución de importaciones.

La validación principal de un sistema es la aceptación de los usuarios, los avales que se exponen en los anexos de la memoria colectiva dan fe de ello y para validar la solución en el ámbito científico novedoso se hace referencia a los eventos en los que se ha expuesto y los resultados obtenidos en cada uno de ellos.

CONCLUSIONES GENERALES

Al realizar un estudio del estado del arte de los sistemas de autorización tanto dentro como fuera del país, arrojó como resultado que la mayoría presentan soluciones particulares para escenarios muy específicos. Los más avanzados incorporan los conceptos de roles y objetos o sistemas, ningún caso incluye los entornos multientidad y multisistema, manteniendo la compartimentación de la información hasta los niveles más básicos.

Los sistemas de autorización estudiados desarrollados fuera del país son muy costosos ejemplo de ello es la plataforma v-Go Single ON, producida por la compañía de Passlogix, la distribución de esta plataforma en conjunto con IBM tiene un costo de 75 USD por usuario que incluye un valor de licencia única y su mantenimiento por un año, si se compra su versión multilingüe y multiplataforma que incluye el CD-ROM de instalación costará entonces 140 USD por usuario y los servicios de instalación y configuración tendrán un valor de 3.60 USD por usuario. Nuestro país con la situación existente no puede dedicar recurso para adquirir productos de este tipo.

Concluido el análisis de las soluciones existentes se decidió desarrollar un componente que gestiona la seguridad y dentro de ella el proceso de autorización de forma centralizada que actualmente es reutilizado por varias aplicaciones en entornos multientidad y multisistema. Incorporando las mejores prácticas identificadas en las soluciones estudiadas por lo que se convierte en un sistema integral que ejecuta los procesos de forma automática. El desarrollo del mismo se ejecutó sobre las herramientas y tecnologías libres, guiado por los modelos y estándares más utilizados para componentes de este tipo.
La solución fue sometida a un proceso de pruebas de funcionamiento guiado por casos de pruebas. Las pruebas fueron ejecutadas por la línea de calidad del CEIGE y Calisoft donde se verificó que se cumplieran todos los requisitos funcionales identificados hasta finalmente ser liberado en cuarta iteración. Para evaluar la calidad del diseño del sistema se le aplicaron un conjunto de métricas. El resultado de este proceso demostró que la responsabilidad del funcionamiento del sistema se encuentra distribuida de forma equilibrada en todos sus componentes, de esta forma si uno de ellos falla no solo afectará las funciones que implementa este componente. Otro resultado obtenido de la aplicación de las métricas es el bajo nivel de dependencia que existe entre los componentes, esta característica garantiza que en caso que falle algún componente los demás mantengan la disponibilidad de la aplicación. El resultado obtenido fue presentado en el Fórum de Ciencia y Técnica del 2009 obteniendo resultado relevante a nivel UCI, relevante a nivel municipal, destacado a nivel provincial y por la calidad del mismo fue seleccionado para el nivel nacional en el próximo año.
De forma general la solución se reutiliza en 14 entidades desarrolladores de software y 9 entidades clientes en las que se encuentra instalado Cedrux. En lo que va de año ha contribuido con el ingreso al país de 1.3 millones de dólares y hay perspectivas potenciales de aumentar esta cifra a 3.2 millones antes de que termine el año.

RECOMENDACIONES

Se recomienda incluir en próximas versiones los siguientes aspectos:

- ✓ Incluir la herencia entre roles para facilitar el trabajo de los administradores.
- ✓ Lograr la integración con alguna PKI para incorporar el uso de certificados digitales.
- ✓ Permitir que el tema multientidad sea desconectable.
- ✓ Gestionar la integración con otras soluciones sin necesidad de realizar cambios en la implementación.
- ✓ Implementar la auditoria a nivel de datos.
- ✓ Posibilitar la creación de las estructuras físicas a medida que se vayan registrando las estructuras lógicas del sistema en Acaxia.
- ✓ Modularizar el sistema de tal forma que se pueda reutilizar por partes, ejemplo solo la autenticación.
- ✓ Analizar los componentes más críticos para identificar si es posible restarle funcionalidades para distribuirlas entre los demás componentes.
- ✓ Analizar los componentes con mayor dependencia para tratar de eliminarlas.

REFERENCIAS BIBLIOGRÁFICAS

1. Atica, Normativa para el Desarrollo de Aplicaciones Web Seguras. 2007.
2. Pakala, S., Preguntas Frecuentes sobre Seguridad en Aplicaciones Web (OWASP FAQ). Enero 25, 2005.
3. Av. Pellegrini 250, R., República Argentina, Especificación Formal del Modelo RBAC en el Cálculo de Construcciones Inductivas. Octubre 2008.
4. Rosa, C.D.L.a.C.D., Análisis Formal del Estándar NIST para Modelos RBAC. 2009.
5. Ravi S. Sandhu , E.J.C., Role-Based Access Control Models. October 26, 1995.
6. Skarmeta, D.A.F.G., Definición de una infraestructura de control de acceso basada en la arquitectura AAA y el uso de credenciales de autorización. Julio de 2006.
7. Potencier, F.a.Z., Francois, The Definitive Guide to Symfony. 2008.
8. Rigazzi, P., Zend Framework Playground.
9. Tejeda, D., KumbiaPHP Framework. 2009.
10. Corporation, Cake Development. . 2007.
11. Minter, D., Beginning Spring 2. 2008.
12. Walls, C.y.B., Ryan, Spring in Action Second Edition. 2008.
13. Marín Mártires, L.A.a.A.C., Luis Ramón, Sistema de autenticación y autorización centralizado. 2008.
14. Junco, A.R.A., Consideraciones Generales en la Implementación de Sistemas Single SignOn en Ambientes Open Source. Junio 2004.
15. Eliurkis, DeepinPHP. 2006.
16. Rincón, R.B., Guía práctica para la implantación de Gestión de Identidades y Accesos y Single Sign-On (IAM & SSO). 2005.
17. MileidyMagalysSarduy Pérez, S.H.C., Propuesta de modelo de desarrollo de software tecnológico del centro de soluciones de gestión. Julio 2009.
18. SEI, Software EngenneringInstitute. 2008.
19. CEIGE, E.A.d., Especificación Técnica para el marco de la arquitectura. 2008.
20. Frederick, S., Ramsay, Colin y Blades, Steve 'Cutter'. Learning Ext JS. 2008.
21. Esser, S., Secure Programming with the Zend-Framework. 2009.
22. Pedro Boda, K.N., Javier Martinez, Benjamín Gonzales, Zend Framework Manual en Español. 14 de Noviembre del 2009.
23. Aquino, A.y.L., Yasser, Implementación del módulo de Contabilidad General del Sistema Integral de Gestión Cedrux. 2009.
24. Smith, L., Introduction to the Doctrine Object Relational Mapper. Apr 2009.
25. Technologies, Z., User Guide Zend Studio 7.0. 2009.
26. Linux., Á.t.d., 2009.
27. Santiago, I.S.C.B.M., Programación web II. Noviembre 2009.
28. PostgreSQL, E.e.d.d.d., Manual del usuario de PostgreSQL. 2008.
29. Prada Nicot, H.y.S.G., Kenner, Desarrollo de los componentes Puesto de Trabajo y Pagos Adicionales del subsistema Capital Humano integrado al sistema integral de gestión CEDRUX. 2009.
30. Nuseibeh, B.y.E., Steve, Requirements engineering: a roadmap. Proceedings of the Conference on The Future of Software Engineering. 2000.
31. IEEE, Compilation of IEEE Standard Computer Glossaries. 1991.
32. Kotonya, G.y.S., Ian, Requirements Engineering: Processes and Techniques. 1998.

33. Sommerville, I.y.S., Pete, Requirements Engineering: A good practice guide. Lancaster University. 1997.
34. Ing. Oiner Gómez Baryolo, I.D.G.T., Ing. Noel Jesús Rivero Pino., Memoria colectiva del proyecto Acaxia. 2010.
35. O.Gómez , Y.C., M.Tenrero, Libro de Ayuda de la Arquitectura Tecnológica del ERP-Cuba, en la versión 2.0 del Marco de Trabajo. 2010.
36. Hernández González, D.A., Un método para el diseño de la base de datos a partir del modelo orientado a objetos. 2004.
37. O.Gómez , D.T., ERP-ARQ-Modelo de datos de Acaxia. 2008.
38. Mercedez Ruiz Carreira, I.R.R., Estimación del Coste de la Calidad del Software através de la Simulación del Procesos de Desarrollo. 2008.
39. Pressman, R.S., Ingeniería del Software, Un enfoque práctico. Tercera edición. McGraw-Hill Companies, 2002.

BIBLIOGRAFÍA

Microsoft Corporation. Microsoft TechNet Seguridad. [Online] 2009. http://www.microsoft.com/latam/technet/seguridad/articulos/bpsegcorp.mspx.
Delitosinformáticos.com. Delitosinformáticos.com. Seguridad. [Online] Marzo 25, 2001.[Cited:Febrero12,2009.] http://www.delitosinformaticos.com/seguridad/clasificacion.shtml.
Potencier, Fabier and Francois, Zaninotto.The Definitive Guide to Symfony.s.l.:Apress, 2008. ISBN-13: 978-1590597866.
Zend Framework Playground.Zend Framework Playground. [Online] WordPress, Septiembre 2008, 2008.[Cited: Abril 21, 2009.] http://spanish.zendfw.com/.
EllisLab, inc.CodeIgniter. [Online] EllisLab, 2001. http://www.codeigniter.com.
KumbiaPHP Framework. KumbiaPHP Framework. [Online] 2007. http://www.kumbiaphp.com.
Cake Development Corporation. Cake Development Corporation. CakePHP. [Online] Cake Development Corporation, 2007. http://www.cakedc.com/.
Marín Mártires, Lázaro Antonio and Capriles Alvarado, Luis Ramón. Sistema de autenticación y autorización centralizado. Ciudad Habana: Universidad de las Ciencias Informáticas., 2008.
Eliurkis.DeepinPHP.[Online] 2006.[Cited: Mayo 20, 2009.] http://www.deepinphp.com/2007/09/01/single-sign-on-sistema-de-autenticacion-unico.
UCI. Teleformación. [Online] Universidad de las Ciencias Informáticas. http://teleformacion.uci.cu.
UCI. Biblioteca. [Online] Universidad de las Ciencias Informáticas. http://biblioteca.uci.cu/.
Symfony.es. Symfony.es[Online] 2009. [Cited: Febrero 12, 2009.] http://www.symfony.es.
SICV. SeguridadInformatica.es. [Online] Seguridad Informática y Creación Visual, 2009. http://www.seguridadinformatica.es/.
SeguInfoNews. SeguInfoNews. [Online] Enero 29, 2009. [Cited: Mayo 20, 2009.] http://blog.segu-info.com.ar/2009/01/ataque-que-es-cross-site-request.html.
Ramos, Kenyie Araya. Desarrolloweb. [Online] 2006. [Cited: Febrero 10, 2009.] http://www.desarrolloweb.com/articulos/introduccion-cross-site-scripting.html.
Pressman, R. Ingenería del software: Un enfoque práctico. McGraw Hill: s.n., 2000.
PHP, Comunidad. Comunidad_PHP. [Online] Universidad de las Ciencias Informáticas. http://php.uci.cu/.
Lazo Ochoa, Rene and Yzquierdo Herrera, Raykenler. El modelo de diseño del sistema HyperWeb.Módulos de Tratamiento Farmacológico y Configuración. Ciudad Habana: Universidad de las Ciencias Informáticas, 2007.
Fundation, OpenLDAP.OpenLDAP.[Online] net Boolean, Marzo 3, 2008. http://www.openldap.org.
Fundation, Ext. Ext JS. [Online] EXT; LLC, 2006. http://www.extjs.com/.
FormatoWeb.FormatoWeb.XSS.[Online] 2007. http://www.formatoweb.com.ar/blog/2007/10/10/ataques-xss-el-peligro-de-hacer-echo-get-var/.
Abrams, Brad. Brad Abrams. [Online] Microsoft Corporation, 2009. http://blogs.msdn.com/brada/archive/2004/02/03/67024.aspx.
Symfony. Open Source PHP Web Framework. [Online] Symfony & Sensio Labs, 2009. http://www.symfony-project.org/.
Microsoft Corporation. Microsoft Solution Framework.2006.