Universidad de las Ciencias Informáticas

Facultad 15

Vista de arquitectura de seguridad del proyecto ERP-Cuba

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Liset Feria González

Tutor: Ing. Oiner Gómez Baryolo
Cotutor: Ing. Darién García Tejo

Ciudad de la Habana, junio de 2010

Año 52 de la Revolución.

Declaratoria de autoría

Declaro ser autor de la presente tesis y recomiendo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los _____ días del mes de _____ del año _____.

Liset Feria González	Ing.Oiner Gómez Baryolo
_____	_____
Firma del Autor	Firma del Tutor

A mi mamá, por tratar de acortar la distancia que nos separaba y continuar dándome amor y aliento como si nos viéramos todos los días.

A mi novio por estar siempre presente, darme tanto amor y ayudarme a superarme cada día más.

A los amigos del tecnológico con los que compartí momentos importantes y dichosos, Lily, Yanelis, Adrián, Julio, Miguel, Rigoberto, Frank, gracias, por ser mis amigos en aquellos momentos y en estos.

A mis compañeros de estudio de estos 5 años por quererme con mis virtudes y defectos, por hacerme tan feliz la estancia en la universidad. Especialmente a Mailyn.

A la nueva familia que encontré, Yanet, Enma, Sintia, Ernesto; por aconsejarme en la vida y los estudios y estar presentes siempre que los necesité.

A Oiner por ser un excelente tutor y a todos los amigos que encontré en el proyecto ERP-Cuba, específicamente en la línea de arquitectura.

A toda mi familia por ser tan especial, a mi tía, prima y abuelas; muchas gracias por formar parte de mi vida y de quien soy.

También a la Revolución por darme la oportunidad de ingresar a la universidad y forjarme como una Ingeniera en Ciencias Informáticas.

Dedico esta tesis a la persona que es merecedora de todos mis logros en esta vida y mi principal motivación, la que me apoyó en todos estos años y me dio las fuerzas para superar los malos momentos, la que me brindó tanto amor y educación, y me enseñó a decidir mi camino, mi madre.

A mis hermano por quererme tanto a pesar de la distancia.

A mi novio que me ha hecho muy feliz. Que ha sabido quererme con mis virtudes y defectos, que me ha hecho parte de su vida. Y además me ha ayudado en mi formación profesional.

A la memoria de mi bisabuelo, pues me ayudó a escoger mi camino para mi formación social.

A mis abuelas, que igualmente me han llenado de amor y en especial a mi abuela Lucy, que me ha consentido, me ha orientado, y dado amor.

A mi tía María y mi prima Nely por ser estar siempre presentes.

En fin, a todas las personas que de una forma u otra me han dado apoyo y amor en estos años de estudios.

A todos gracias.

Resumen

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

Actualmente en el proyecto ERP-Cuba existe un desconocimiento del estándar de seguridad que se debe utilizar. Por esta causa implementar la seguridad resulta trabajoso, ya que cada persona la controla de forma diferente, invirtiendo tiempo y cuantiosos recursos humanos y materiales.

Este documento constituye una guía, desde el punto de vista arquitectónico, con el propósito puntual de brindar una visión de conjunto lo más estructurada posible, que establezca el papel de esta disciplina, en relación con la estrategia arquitectónica del Sistema de Gestión de Entidades Cedrux.

Existen varias razones para desarrollar este documento, partiendo del hecho de que no existe una plantilla que establezca las pautas a seguir arquitectónicamente para la seguridad; haciéndose sumamente necesario en el caso puntual del proyecto ERP-Cuba, por tener un alcance tan amplio y estar distribuido el desarrollo del software en varias líneas de desarrollo, que posteriormente se integrarán en un solo producto. De manera, que con este escrito se estandariza el desarrollo de todas las líneas, generalizando políticas, herramientas, nomenclaturas y otros aspectos que mejoran el rendimiento y la seguridad del sistema a desarrollar.

La finalidad de este documento consiste en establecer las bases para el conocimiento teórico y los fundamentos para la puesta en práctica de un modelo basado en la arquitectura de seguridad. Pues a pesar de que los aportes de esta disciplina se ponen en práctica de cierta forma en el desarrollo actual del proyecto, por la experiencia personal y por algunas normas; que de manera individual se han venido estableciendo en cada línea; de todas maneras se impone, la necesidad de un estándar que normalice el trabajo de manera general.

Palabras claves: seguridad, desconocimiento, estándar, guía, vulnerabilidad.

Índice de contenido

Índice de figuras

Introducción

Las empresas desarrolladoras de software sea cual sea su tamaño, capital o presencia en el mercado persiguen dentro de sus objetivos que el software desarrollado sea inmune a ataques informáticos. Este es un reto difícil de lograr si se tiene en cuenta que las demandas de los clientes en muchos casos son superiores a las capacidades productivas de las empresas y que la producción de software de forma industrial es un mito que solo pocas entidades pueden respaldar. La solución a este conflicto se ha buscado en todas las direcciones, se han perfeccionado los procesos y las metodologías de desarrollo, se trabaja fuertemente en la capacitación de los recursos humanos y constantemente se desarrollan nuevas tecnologías.

En la actualidad muchos países desarrollan sistemas para la gestión empresarial para la automatización e integración de prácticas de negocio, asociadas a aspectos operativos o productivos de una empresa. Este tipo de sistemas, suelen presentar una arquitectura modular, donde cada módulo gestiona las funciones de un área empresarial diferente, como pueden ser: nóminas, finanzas, gestión de proyectos, sistema de gestión geográfica, contabilidad, logística. Estos sistemas son integrales, es decir, una agrupación de todos los módulos que los componen y que agrupan a su vez todos los procesos de gestión de una empresa. La seguridad de estos es muy compleja, puesto que la gestión empresarial engloba un sin número de áreas que gestionan gran cantidad de información y en algunos casos es confidencial.

Cuba a pesar de ser un país del tercer mundo, no quiere mantenerse alejada de la revolución tecnológica a la cual se ha girado el mundo actual. Es por ello que, entre sus objetivos por lograr un avance tecnológico, se ha trazado la meta de desarrollar un sistema de gestión integral, tecnología que sea competitiva a nivel mundial, que reutilice los principales avances del mundo, y con la inclusión del conocimiento endógeno que tenga en cuenta las características nacionales. En este tipo de sistema por su extensión durante el proceso de desarrollo, se conciben numerosos artefactos en cada una de las áreas de trabajo y por tanto, es necesaria la seguridad de cada uno de ellos.

La tarea de realizar un sistema de gestión integral, se le otorgó a la Universidad de Ciencias Informáticas (UCI), creándose el proyecto ERP- Cuba (Enterprise Resource Planning, en español Planificación de Recursos Empresariales), el cual es el encargado de construir las tecnologías que subsiguientemente serán usadas por las diferentes áreas de desarrollo, durante este proceso se generan componentes los cuales tienen una importancia medular para todo el sistema y por tanto resulta imprescindible mantenerlos ordenados y centralizados.

Actualmente se ha generado conocimiento en la arquitectura del proyecto ERP-Cuba, pero en ocasiones, el desconocimiento del estándar a utilizar para la seguridad provoca un atraso en las líneas de desarrollo,

debido a que si no se usa este estándar el tiempo es más prolongado cuando llega la hora de integrar distintas soluciones, complicando el trabajo ya realizado, teniéndose que reestructurar y recodificar lo antes desarrollado, atrasando así los cronogramas planificados. Además al realizar evoluciones y actualizaciones se corren grandes riesgos con cambios, pues se pueden obviar u olvidar algunos aspectos necesarios para la seguridad, provocándose así una posible brecha de seguridad, debilitándose el sistema.

El problema de la estandarización de la seguridad se debe principalmente a que no existe una guía para realizarla, por lo que se hace difícil mantener organizado los aspectos tecnológicos de los sistemas, así como su reutilización por otros proyectos que implementan soluciones integrales.

Debido a la situación polémica que trae aparejada la falta de estandarización de la seguridad en los sistemas, se plantea la siguiente interrogante como problema a resolver: ¿Cómo estandarizar los procesos de codificación, comunicación y configuración segura del proyecto ERP-Cuba?
Para darle solución al problema que se plantea, el objeto de estudio lo constituye: la seguridad en sistemas de gestión, cuyo campo de acción queda enmarcado en: la seguridad del entorno de desarrollo, de despliegue y de las aplicaciones del proyecto ERP-Cuba.

Para la realización del trabajo de diploma se traza como objetivo general realizar una guía para usuarios y desarrolladores que les sirva para estandarizar los procesos de codificación, comunicación y configuración segura del proyecto ERP-Cuba

Para darle cumplimiento al objetivo general planteado se trazan los siguientes objetivos específicos:

- Realizar un estudio del estado del arte de las principales herramientas y tecnologías relacionadas con la investigación.
- Seleccionar las herramientas de apoyo a la seguridad.
- Desarrollar el estándar de configuración segura para los servidores y el lenguaje de programación.
- Especificar los estándares de validación y nomenclatura.
- Definir las políticas de seguridad necesarias.
- Establecer el estándar de seguridad a nivel de sistema.
- Crear listas de chequeo para el control y seguimiento del desarrollo y el despliegue del software.

La investigación cuenta con la siguiente idea a defender:
Si se realiza una guía para usuarios y desarrolladores se logrará estandarizar los procesos de codificación, comunicación y configuración segura del proyecto ERP-Cuba.

Estructura del trabajo

El trabajo está estructurado en 3 capítulos.

Capítulo 1

El primer capítulo contiene la fundamentación teórica del trabajo, el mismo incluye términos y conceptos más importantes empleados en la seguridad de sistemas de gestión, así como estándares, herramientas, y tecnologías más utilizados en el mundo y en el Sistema de Gestión de Entidades Cedrux.

Capítulo 2

El segundo capítulo presenta la primera parte de la descripción de la solución. En el mismo se puede encontrar cómo debe realizarse la configuración segura del servidor web y el de base de datos, así como del lenguaje de programación y qué tener en cuenta para el entorno de despliegue. Además de las herramientas de apoyo a la seguridad seleccionadas, las políticas definidas y los estándares de validación y nomenclatura por los que se debe regir la seguridad del Centro de Informatización de la Gestión de Entidades.

Capítulo 3

El tercer capítulo brinda la última parte de la solución propuesta, en él aparece cómo utilizar el sistema Acaxia y los estándares definidos para los procesos de autenticación, autorización y encriptación. También posee las listas de chequeo para el control y seguimiento del sistema en el entorno de desarrollo y despliegue.

1

CAPÍTULO I Fundamentación teórica

1.1. Introducción

El bien más preciado por cualquier institución es la información y de ahí que se han desarrollado protocolos o mecanismos adecuados para preservarla ,logrando un mayor control y seguridad de la misma. Este capítulo incluye términos y conceptos más importantes empleados en la seguridad de sistemas de gestión. Las herramientas y tecnologías definidas por el Centro de Informatización de la Gestión de Entidades. Los estándares internacionales más utilizados para el desarrollo de sistemas de gestión integral de seguridad, así como un estudio de otras soluciones usadas en el mundo.

1.2. Principales términos y conceptos

Autentificación o autenticación: es el proceso de verificar formalmente la identidad de las entidades participantes en una comunicación o intercambio de información. Por entidad se entiende tanto personas, como procesos o computadoras. (1)

Autorización: es la parte del sistema que protege los recursos del mismo, 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 dato, programas, dispositivos y funcionalidades provistas por aplicaciones. (1)

Activo: es el conjunto de los bienes y derechos tangibles e intangibles de propiedad de una persona natural o jurídica que por lo general son generadores de renta o fuente de beneficios, en el ambiente informático llámese activo a los bienes de información y procesamiento, que posee la institución. Recurso del sistema de información o relacionado con éste, necesario para que la organización funcione correctamente y alcance los objetivos propuestos. (2)

Amenaza: es un evento que puede desencadenar un incidente en la organización, produciendo daños materiales o pérdidas inmateriales en sus activos. (2)

Ataque: evento, exitoso o no, que atenta sobre el buen funcionamiento del sistema. (2)

Auditoría: es la capacidad de determinar qué acciones o procesos se están llevando a cabo en el sistema, así como quién y cuándo las realiza. (3)

Confidencialidad: es la propiedad de la seguridad que permite mantener en secreto la información y solo los usuarios autorizados pueden manipularla. Igual que antes, los usuarios pueden ser personas, procesos, programas. (1)

Disponibilidad: que los recursos de información sean accesibles, cuando estos sean necesitados. (2)

Desastre o contingencia: interrupción de la capacidad de acceso a información y procesamiento de la misma a través de computadoras necesarias para la operación normal de un negocio. (2)

Integridad: es la característica que hace que su contenido permanezca inalterado a menos que sea modificado por personal autorizado y esta modificación sea registrada para posteriores controles o auditorias. (3)

Impacto: medir la consecuencia al materializarse una amenaza. (2)

Política de seguridad: representa para una aplicación, su conjunto de requerimientos de seguridad, incluyendo, aproximación al control de accesos, identificación de que necesita ser protegido, identificar anticipadamente usuarios, sus privilegios,etc. (4)

Riesgo: posibilidad de que se produzca un impacto determinado en un activo, en un dominio o en toda la organización. (2)

Seguridad informática: es una disciplina que se encarga de proteger la integridad y la privacidad de la información almacenada en un sistema informático. (5)

Vulnerabilidad: posibilidad de ocurrencia de la materialización de una amenaza sobre un activo. (2)

1.3. Objetivos y principios de la seguridad informática

Los activos son los elementos que la seguridad informática tiene como objetivo proteger. Son tres elementos los que conforman los activos estos son: la información, los equipos que la soportan (software, hardware y organización) y los usuarios. Generalmente, la seguridad informática consiste en garantizar que el material, los recursos de software y los sistemas de información de una organización se usen únicamente para los propósitos para los que fueron creados y dentro del marco previsto. Por tanto, la seguridad informática se resume, por lo general, en tres objetivos principales: la integridad, la confidencialidad y la disponibilidad.

La seguridad informática tiene como objetivo:

- Restringir el acceso (de personas de la organización y de las que no lo son) a los programas y archivos. Asegurar que los operadores puedan trabajar pero que no puedan modificar los programas ni los archivos que no correspondan (sin una supervisión minuciosa). (6)
- Asegurar que la información transmitida sea la misma que reciba el destinatario al cual se ha enviado y que no le llegue a otro. (6)
- Asegurar que existan sistemas y pasos de emergencia alternativos de transmisión entre diferentes puntos. (6)
- Organizar a cada uno de los empleados por jerarquía informática, con claves distintas y permisos bien establecidos, en todos y cada uno de los sistemas o aplicaciones empleadas. (6)
- Actualizar constantemente las contraseñas de accesos a los sistemas de cómputo. (6)

Existen algunos principios y estrategias a seguir para mantener adecuadamente la seguridad informática de una institución. A continuación se mencionan estos principios:

- **Mínimo privilegio:** se deben otorgar los permisos estrictamente necesarios para efectuar las acciones que se requieran, ni más, ni menos de lo solicitado. (7)
- **Eslabón más débil:** la seguridad de un sistema es tan fuerte como su parte más débil. Un atacante primero analiza cuál es el punto más débil del sistema y concentra sus esfuerzos en ese lugar. (7)
- **Proporcionalidad:** las medidas de seguridad deben estar en correspondencia con lo que se protege y con el nivel de riesgo existente. No sería lógico proteger con múltiples recursos un activo informático que no posee valor o que la probabilidad de ocurrencia de un ataque sobre el mismo sea muy baja. (7)
- **Dinamismo**: la seguridad informática no es un producto, es un proceso. No se termina con la implementación de los medios tecnológicos, se requiere permanentemente monitoreo y mantenimiento. (7)
- **Participación universal:** la gestión de la seguridad informática necesita de la participación de todo el personal de una institución. La seguridad que puede ser alcanzada mediante medios técnicos es limitada y debiera ser apoyada por una gestión y procedimientos adecuados, que involucren a todos los individuos. (7)

1.4. Riesgos en la seguridad informática

Los riesgos**,** en términos de seguridad, se caracterizan por lo general mediante la siguiente ecuación.

$$\text{Riesgo} = \frac{\text{Amenaza x Vulnerabilidad}}{\text{Contramedida}} \quad (8)$$

La **amenaza** representa el tipo de acción que tiende a ser dañina, mientras que la **vulnerabilidad** (conocida a veces como falencias (flaws) o brechas (breaches)), representa el grado de exposición a las amenazas en un contexto particular. Finalmente, la **contramedida,** representa todas las acciones que se implementan para prevenir la amenaza. Las contramedidas que deben implementarse no sólo son soluciones técnicas, sino también reflejan la capacitación y la toma de conciencia por parte del usuario, además de reglas claramente definidas. (8)

Para que un sistema sea seguro, deben identificarse las posibles amenazas y por lo tanto, conocer y prever el curso de acción del enemigo. Existe un viejo dicho en la seguridad informática que dicta: "lo que no está permitido debe estar prohibido" (9) y ésta debe ser la meta perseguida.

Elementos de un análisis de riesgo: Cuando se pretende diseñar una técnica para implementar un análisis de riesgo informático, se pueden tomar los siguientes puntos como referencia a seguir:

- Construir un perfil de las amenazas que esté basado en los activos de la organización. (10)
- Identificar (activos de la organización, amenazas de cada uno de los activos listados, requerimientos de seguridad de la organización, vulnerabilidades dentro de la infraestructura tecnológica, vulnerabilidades de la organización). (10)
- Conocer las prácticas actuales de seguridad. (10)
- Detección de los componentes claves. (10)
- Desarrollar planes y estrategias de seguridad que contengan (riesgo para los activos críticos, medidas de riesgos, estrategias de protección, planes para reducir los riesgos). (10)

1.5. Amenazas en la seguridad informática

Un ataque no es más que la realización de una amenaza. Las categorías generales de amenazas o ataques son las siguientes:

Intercepción: Una entidad no autorizada consigue acceso a un recurso. Este es un ataque contra la confidencialidad. La entidad no autorizada podría ser una persona, un programa o un ordenador. Ejemplos de este ataque son: pinchar una línea para tomar los datos que circulen por la red y la copia ilícita de ficheros o programas (intercepción de datos). (11)

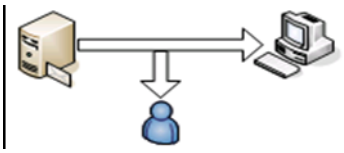


Figura 1: Ejemplo visual de intercepción. (11)

Modificación: Una entidad no autorizada no sólo consigue acceder a un recurso, sino que es capaz de manipularlo. Este es un ataque contra la integridad. Ejemplos de este ataque es: el cambio de valores en un archivo de datos, alterar un programa para que funcione de forma diferente y modificar el contenido de mensajes que están siendo transferidos por la red. (11)

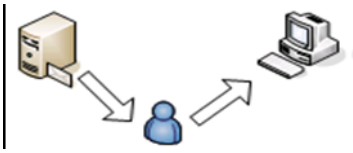


Figura 2: Ejemplo visual de modificación. (11)

Interrupción: Un recurso del sistema es destruido o se vuelve no disponible. Este es un ataque contra la disponibilidad. Ejemplos de este ataque son: la destrucción de un elemento hardware, como un disco duro, cortar una línea de comunicación o deshabilitar el sistema de gestión de ficheros. (11)

Figura 3: Ejemplo visual de interrupción. (11)

Fabricación: Una entidad no autorizada inserta objetos falsificados en el sistema. Este es un ataque contra la autenticidad. Ejemplos de este ataque son: la inserción de mensajes ilegítimos en una red o añadir registros a un archivo. Estos ataques se pueden asimismo clasificar de forma útil en términos de ataques pasivos y ataques activos. (11)

- **Ataques pasivos:** En los ataques pasivos el atacante no altera la comunicación, sino que únicamente la escucha o monitoriza, para obtener información que está siendo transmitida. Sus objetivos son la intercepción de datos y el análisis de tráfico, una técnica más sutil para obtener información de la comunicación. (11)
- **Ataques activos:** Estos ataques implican algún tipo de modificación del flujo de datos transmitido o la creación de un falso flujo de datos, pudiendo subdividirse en cuatro categorías: (11)

- **Suplantación de identidad:** El intruso se hace pasar por una entidad diferente. Normalmente incluye alguna de las otras formas de ataque activo. Por ejemplo, secuencias de autenticación pueden ser capturadas y repetidas, permitiendo a una entidad no autorizada acceder a una serie de recursos privilegiados suplantando a la entidad que posee esos privilegios, como al robar la contraseña de acceso a una cuenta. (11)
- **Reactuación:** Uno o varios mensajes legítimos son capturados y repetidos para producir un efecto no deseado, como por ejemplo ingresar dinero repetidas veces en una cuenta dada. (3)
- **Modificación de mensajes:** Una porción del mensaje legítimo es alterada, o los mensajes son retardados o reordenados, para producir un efecto no autorizado. Por ejemplo, el mensaje "Ingresa un millón de pesos en la cuenta A" podría ser modificado para decir "Ingresa un millón de pesos en la cuenta B". (11)
- **Degradación fraudulenta del servicio:** Impide o inhibe el uso normal o la gestión de recursos informáticos y de comunicaciones. Por ejemplo, el intruso podría suprimir todos los mensajes dirigidos a una determinada entidad o se podría interrumpir el servicio de una red inundándola con mensajes espurios. (11)

1.6. Tecnologías utilizadas

Para el desarrollo de este trabajo de diploma se estudiaron las tecnologias definidas y utilizadas por el Sistema de Gestión de Entidades Cedrux, las cuales se describen a continuación.

1.6.1. PHP- Hypertext Pre-processor

Es un lenguaje de programación interpretado diseñado originalmente para la creación de páginas web dinámicas[1]. Es un lenguaje interpretado de propósito general, ampliamente usado, que puede ser incrustado dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. (13)

Ventajas de PHP

- Es un lenguaje multiplataforma. (13)
- Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos. (13)

[1] *Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.* (12)

- El código fuente escrito en PHP es invisible al navegador y al cliente, ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable. (13)
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL. (13)
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones). (13)
- Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda. (13)
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos. (13)
- Permite aplicar técnicas de programación orientada a objetos. (13)
- Biblioteca nativa de funciones sumamente amplia e incluida. (13)
- No requiere definición de tipos de variables, aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución. (13)
- Tiene manejo de excepciones (desde PHP5). (13)

Inconvenientes

- La ofuscación de código es la única forma de ocultar los fuentes. (13)

Diferencias de PHP con respecto a otros lenguajes de programación

- Es software libre. (14)
- Es muy rápido. (14)
- Su sintaxis está inspirada en C. (14)
- Su librería estándar es realmente amplia. (14)
- PHP es relativamente multiplataforma. (14)
- El acceso a las bases de datos de PHP es muy heterogéneo. (14)
- PHP es suficientemente versátil y potente. (14)

1.6.2. EXT JS

¿Qué es ExtJS?

Es una librería JavaScript que permite construir aplicaciones complejas en internet. Esta librería incluye:

- Componentes interfaz de usuario (UI) del alto performance y personalizables. (15)
- Modelo de componentes extensibles. (15)

- Un API [2] fácil de usar. (15)
- Licencias Open source y comerciales. (15)

¿Cuáles son sus beneficios?

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 [3] externos. ExtJS encaja dentro de este esquema como un motor que permite crear aplicaciones RIA [4] mediante JavaScript. Si se enmarca a ExtJS dentro del desarrollo RIA, éste sería el render [5] de la aplicación que controla el cliente y que se 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 (diseños) 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). (15)

Usar un motor de render como ExtJS permite tener además estos beneficios:

- Existe un balance entre Cliente – Servidor. (15)
- Comunicación asíncrona. (15)
- Eficiencia de la red. (15)

Inconvenientes:

- Necesita una plataforma. (15)

---

[2] *Una interfaz de programación de aplicaciones o API (del inglés application programming interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Usados generalmente en las bibliotecas.* (16)

[3] *Un plug-in es un módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande. La idea es que el nuevo componente se* enchufa *simplemente al sistema existente.* (17)

[4] *Rich Internet Applications (Aplicaciones de Internet Enriquecidas): son un nuevo tipo de aplicaciones con más ventajas que las tradicionales aplicaciones web. Esta surge como una combinación de las ventajas que ofrecen las aplicaciones web y las aplicaciones tradicionales.* (18)

[5] *Renderizado (render en inglés): es un término usado en jerga informática para referirse al proceso de generar una imagen desde un modelo*. (19)

- Descargas lentas. (15)
- Problemas con los motores de búsqueda. (15)
- Su sistema de licenciamiento no contempla la licencia LGPL[6], el código debe ser 100% GPL [7] o se debe pagar por su licencia de desarrollo. (15)

1.6.1. Doctrine

Doctrine es un ORM [8] (Object – Relational Mapping) también conocido como O/RM, O/R mapping, posee una poderosa capa de abstracción de base de datos (BD). Una de sus características es la opción de escribir consultas de BD en un objeto apropiado orientado al dialecto SQL (Structured Query Language) y que se le denomina Lenguaje de Consulta de Doctrine o DQL (del inglés Doctrine Query Language), inspirado por el Lenguaje de Consulta de Hibernate o HQL (del inglés Hibernate Query Language). Este proporciona a los desarrolladores una poderosa alternativa al SQL que mantiene la flexibilidad sin requerir duplicación de código innecesario. (21)

Principales características: Doctrine es un framework para el mapeo objeto – relacional para PHP que está dividido en dos capas principales, la DBAL (del inglés Database Abstraction Layer) y el ORM. La figura 4 refleja cómo las capas de Doctrine trabajan juntas. (21)

Ventajas y desventajas: Una de las ventajas de utilizar estas capas de abstracción de objetos/relacional es que evita utilizar una sintaxis específica de un sistema de bases de datos concreto. Esta capa transforma automáticamente las llamadas a los objetos en consultas SQL optimizadas para el sistema gestor de bases de datos que se está utilizando en cada momento. De esta forma, es muy sencillo cambiar a otro sistema de bases de datos completamente diferente en mitad del desarrollo de un proyecto. Su principal ventaja es el rendimiento en ejecución y la forma tan concisa en la que se pueden escribir consultas muy complejas. Otra de las ventajas

---

[6] *Licencia LGPL: es una licencia que es prácticamente igual a la GPL, pero permite que los software con esta licencia estén integrado en software privativos.* (20)

[7] *Licencia GPL: .Se puede copiar, regalar o vender a terceros el software, sin tener la obligación de pagar por ello. El software modificado no debe tener costo por la licencia. Tiene que incluir el código fuente. Un programa con licencia GPL que ha sido modificado automáticamente es publicado con licencia GPL.* (20)

[8] *Mapeo Objeto-Relacional: es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional. En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional.* (21)

de Doctrine es que se puede utilizar las herramientas de sincronización y generación durante todo el proceso de desarrollo. (21)

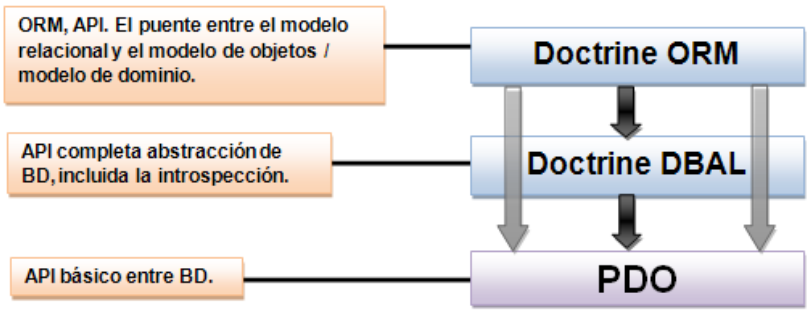


Figura 4: Trabajo de las capas de Doctrine. (21)

1.6.2. Zend Framework

Zend Framework (ZF) es un framework [9] de código abierto para desarrollar aplicaciones web y servicios web con PHP5. ZF es una implementación que usa código 100% orientado a objetos. La estructura de los componentes de ZF es algo único; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado. A menudo se refiere a este tipo de diseño como "use-at-will" (uso a voluntad). Aunque se pueden utilizar de forma individual, los componentes de la biblioteca estándar de ZF conforman un potente y extensible framework de aplicaciones web al combinarse. (23)

Principales características:

- Trabaja con MVC (Model View Controller). (24)
- Cuenta con módulos para manejar archivos PDF, canales RSS, Web Services (Amazon, Flickr, Yahoo). (24)
- Una solución para el acceso a base de datos que balancea el ORM con eficiencia y simplicidad. (24)
- El Marco de Zend también incluye objetos de las diferentes bases de datos, por lo que es extremadamente simple para consultar su base de datos, sin tener que escribir ninguna consulta SQL. (24)
- Completa documentación y test de alta calidad. (24)

[9] *En el desarrollo de software, un framework, es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.* (22)

- Soporte avanzado para i18n (internacionalización). (24)
- Un buscador compatible con Lucene[10]. (24)
- Robustas clases para autenticación y filtrado de entrada. (24)
- Clientes para servicios web, incluidos Google Data APIs y StrikeIron. (24)
- Muchas otras clases útiles para hacerlo tan productivo como sea posible. (24)

1.6.3. Sauxe

Sauxe, es un marco de trabajo que contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. Siguiendo el paradigma de independencia tecnológica por el cual apuesta el país, utiliza las siguientes herramientas libres: (26)

- Postgres versión 8.3 ó superior. (26)
- Apache HTTP Server versión 2.0 ó superior. (26)
- Visual Paradigm versión 6.3. (26)
- SVN versión 1.4.6 ó superior. (26)
- Mozilla Firefox versión 3.0.0.12. (26)

Está desarrollado sobre lenguaje de programación PHP versión 5.2.4, aunque en la capa de presentación se utilice JavaScript y HTML y en la capa de acceso a datos el lenguaje de consulta DQL que implementa Doctrine. Implementa una arquitectura en capas aunque en una de sus capas contiene un modelo vista controlador. En la capa de presentación utiliza ExtJs por la gran gama de componentes que se pueden reutilizar para agilizar el proceso de desarrollo y mostrarle al usuario una interfaz más amigable y funcional.
En la capa de negocio utiliza Zend Framework, por su nivel de flexibilidad en la integración con la capa superior, inferior y vertical de la arquitectura (ver Anexo 1**: Aspectos de Sauxe.**). Cuenta con un componente llamado Traza que permite gestionar todas las acciones que se ejecutan dentro de un sistema. Con esta información no solo se pueden identificar ataques, sino que también permite evaluar el rendimiento de cada acción para re-implementarla en caso que el tiempo de respuesta sea muy grande. (26)

---

[10] *Lucene, es un API de código abierto para recuperación de información. Tiene versiones para otros lenguajes incluyendo Delphi, Perl, C#, C++, Python, Ruby y PHP .Es útil para cualquier aplicación que requiera indexado y búsqueda a texto completo.* (25)

1.7. Herramientas utilizadas

A continuación se describen las herramientas definidas y utilizadas por el Sistema de Gestión de Entidades Cedrux, las cuales se estudiaron para el desarrollo de este trabajo de diploma.

1.7.1. Apache

Apache es un servidor web gratuito, potente y que ofrece un servicio estable y sencillo de mantener y configurar. Destacar las siguientes características: es multiplataforma (aunque idealmente está preparado para funcionar bajo Linux), muy sencillo de configurar, es Open-source (código abierto), muy útil para proveedores de servicios de internet que requieran miles de sitios pequeños con páginas estáticas, 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) y es capaz de utilizar lenguajes como PHP, TCL, Python, entre otros. (27)

Ventajas que posee:

- **Fiabilidad:** Alrededor del 90% de los servidores con más alta disponibilidad funcionan con él. (27)
- **Gratuidad**: es totalmente gratuito y se distribuye bajo la licencia Apache Software License, que permite la modificación del código. (27)
- **Extensibilidad**: se pueden añadir módulos para ampliar las ya de por sí, amplias capacidades de Apache. (27)

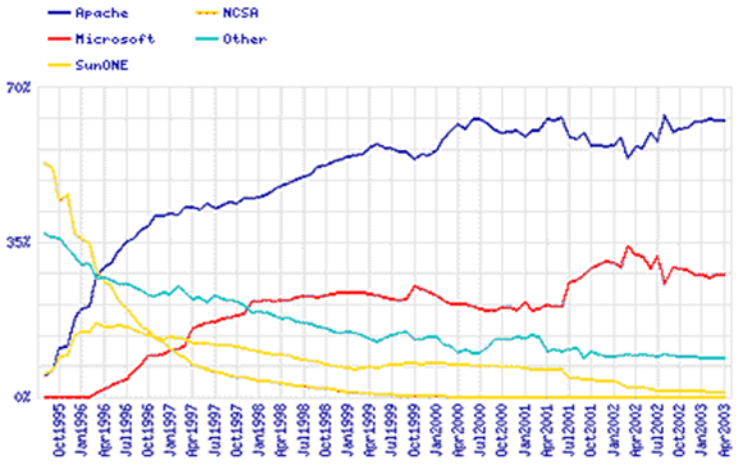


Figura 5: Gráfico del despunte y avance de apache. (27)

1.7.2. PostgreSQL

PostgreSQL, es un potente motor de bases de datos, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de bases de datos comerciales. Es más completo que MySQL ya que permite métodos almacenados, restricciones de integridad, vistas, aunque en las últimas versiones de MySQL se han hecho grandes avances en ese sentido. (28)

PostgreSQL ofrece muchas ventajas respecto a otros sistemas de bases de datos estas son: instalación ilimitada, mejor soporte que los proveedores comerciales, ahorros considerables en costos de operación, estabilidad y confiabilidad legendarias, extensible, multiplataforma, diseñado para ambientes de alto volumen, herramientas gráficas de diseño y administración de bases de datos. (29)

1.7.3. Subversion

Subversion es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS (Concurrent Versions System). Es software libre bajo una licencia de tipo Apache/BSD y se le conoce también como svn por ser ese el nombre de la herramienta de línea de comandos. Una característica importante de Subversion es que, a diferencia de CVS, los archivos versionados no tienen cada uno un número de revisión independiente. Subversion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintos ordenadores. (30)

Ventajas

- Se sigue la historia de los archivos y directorios a través de copias y renombrados. (30)
- Las modificaciones (incluyendo cambios a varios archivos) son atómicas. (30)
- La creación de ramas y etiquetas es una operación más eficiente; tiene costo de complejidad constante (O(1)) y no lineal (O(n)) como en CVS. (30)
- Se envían sólo las diferencias en ambas direcciones (en CVS siempre se envían al servidor archivos completos). (30)
- Puede ser servido mediante Apache, sobre WebDAV/DeltaV. Esto permite que clientes WebDAV utilicen Subversion en forma transparente. (30)
- Maneja eficientemente archivos binarios (a diferencia de CVS que los trata internamente como si fueran de texto). (30)
- Permite selectivamente el bloqueo de archivos. Se usa en archivos binarios que, al no poder fusionarse fácilmente, conviene que no sean editados por más de una persona a la vez. (30)
- Cuando se usa integrado a Apache permite utilizar todas las opciones que este servidor provee a la hora de autentificar archivos (SQL, LDAP, PAM). (30)

Carencias

- El manejo de cambio de nombres de archivos no es completo. Lo maneja como la suma de una operación de copia y una de borrado. (30)
- No resuelve el problema de aplicar repetidamente parches entre ramas, no facilita el llevar la cuenta de qué cambios se han trasladado. Esto se resuelve siendo cuidadoso con los mensajes de commit. (30)

1.7.4. Mozilla Firefox

Mozilla Firefox es un navegador web libre descendiente de Mozilla Application Suite, desarrollado por la Corporación Mozilla, la Fundación Mozilla y un gran número de voluntarios externos. Es un navegador multiplataforma y está disponible en varias versiones de Microsoft Windows, Mac OS X, GNU/Linux y algunos sistemas basados en Unix. Las características que incluye Mozilla Firefox son la navegación por pestañas, corrector ortográfico, marcadores, bloqueador de ventanas emergentes, atajos del teclado, soporte para motores de búsqueda y un gestor de descargas. Los usuarios pueden personalizar Firefox con las extensiones y temas. Firefox proporciona un entorno para los desarrolladores web, en el que se puede utilizar herramientas incorporadas. (31)

Firefox usa un sistema de seguridad sandbox. Utiliza el sistema SSL/TLS para proteger la comunicación con los servidores web, utilizando fuerte criptografía cuando se utiliza el protocolo Https. También proporciona apoyo a las tarjetas inteligentes para fines de autenticación. Cuenta con una protección antiphishing, antimalware e integración con el antivirus; como medida prudencial que ha causado controversia, Firefox no incluye compatibilidad con los sistemas ActiveX, debido a la decisión de la fundación Mozilla de no incluirlo por tener vulnerabilidades de seguridad. (31)

1.8. Estándares

El término estándar, de origen inglés, tiene varios significados: originalmente, en inglés, significaba bandera; color; pancarta; especialmente nacional u otra enseña. En tecnología y otros campos, un estándar es: una especificación que regula la realización de ciertos procesos o la fabricación de componentes para garantizar la interoperabilidad [11].(33) Para el desarrollo de este trabajo de diploma se estudiaron los estándares más beneficiosos para al Sistema de Gestión de Entidades Cedrux, estos se describen a continuación.

[11] *Interoperabilidad: Capacidad de los sistemas de tecnologías de la información y las comunicaciones (TIC), y de los procesos empresariales a los que apoyan, de intercambiar datos y posibilitar la puesta en común de información y conocimientos.* (32)

1.8.1. SAML

En el contexto de los servicios web, el estándar SAML (Security Assertion Markup Language, en español afirmación de lenguaje de marcado de seguridad) estandariza el intercambio de los datos de autenticación y autorización generados por el proveedor de identidad e interpretados por las aplicaciones. SAML define la sintaxis y la semántica de la transformación de afirmaciones que se hacen acerca de un tema por una entidad del sistema. (34) El estándar SAML define cuatro conceptos fundamentales:

- **Aseveraciones o afirmaciones:** definen las afirmaciones de seguridad de una entidad dentro de un sistema. (35)
- **Protocolos:** para que SAML sea útil se requiere que además de definir un formato estándar para las afirmaciones, se defina un mecanismo para el intercambio automático de estas. Para esto el estándar define un conjunto de protocolos de pedido/respuesta que permite la obtención de las afirmaciones de un usuario. (35)
- **Vinculaciones :** los dos conceptos anteriores están definidos como un esquema XML. Para ser utilizados en la práctica se requiere de un mapeo a los distintos protocolos de transporte utilizados (HTTP-GET, HTTP-POST, SOAP). (35)
- **Perfiles:** los protocolos y los enlaces por sí solos no permiten la interoperabilidad de SAML. Los perfiles definen la secuencia específica de mensajes y los enlaces requeridos en cada caso, completan cada uno de los casos de uso definidos en el estándar (por ejemplo SSO (Single sign-on) web).Por cada combinación de caso de uso y binding se pueden obtener variaciones de un mismo profile (no todas con posibilidad de utilizarse en la práctica). En el caso del profile SSO Web, teniendo en cuenta también desde donde se inicia el proceso, se obtienen tres variaciones: (35)
    - SSO iniciado por el SP con el binding Redirect/POST. (35)
    - SSO iniciado por el SP con el binding POST/Artifact. (35)
    - SSO iniciado por el IDP con el binding POST. (35)

1.8.2. RBAC

RBAC (Role Based Access Control) es un modelo de control de acceso basado en roles. Una de las tecnologías que más ha llegado a satisfacer las principales necesidades en cuanto a control de acceso. El principal objetivo de RBAC es prevenir que los usuarios tengan libre acceso a la información de la organización. El modelo introduce el concepto de rol y asocia a los usuarios con los roles por los que va pasando durante la vida del sistema. (36)

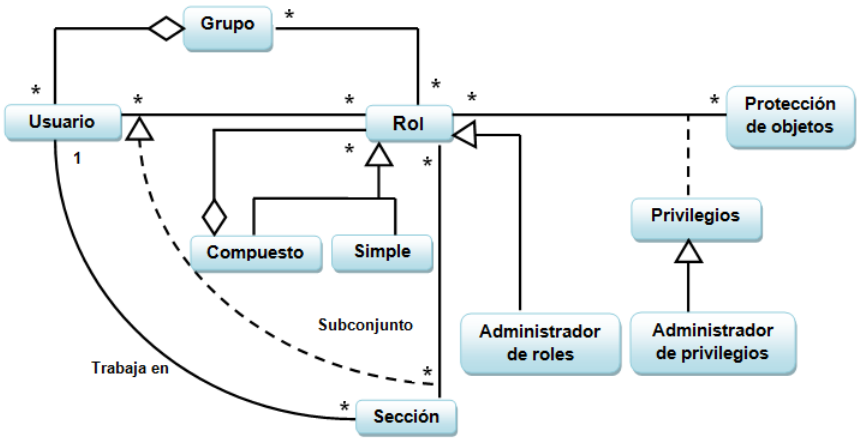


Figura 6: Modelo y diseño del sistema de seguridad RBAC. (37)

Características:

- El propietario o administrador del sistema es quien maneja los datos, para así satisfacer las necesidades de la empresa u organización. (37)
- Permite la construcción jerárquica de roles. Una jerarquía define roles que tiene propiedades en común y que pueden contener otros roles. A través de un análisis de generalización/especialización se forma esta jerarquía en la que se refleja autoridades, responsabilidades y capacidades. (37)
- Separación de responsabilidades, previniendo fraudes y errores. Existe la separación de responsabilidades estática y dinámica. (37)
- Menor privilegio, utiliza la política de que un usuario debe tener los privilegios suficientes y necesarios para realizar su tarea. (37)

Ventajas de RBAC: habilita a los usuarios a llevar a cabo un amplio rango de operaciones autorizadas y provee una gran flexibilidad y un amplio espectro de aplicación. Los administradores de sistema pueden controlar el acceso a un nivel de abstracción que es natural La administración de esta manera es más sencilla. La productividad de la organización también puede llegar a ser mejorada con RBAC, gracias a la flexibilidad que este modelo tiene, además de que éste puede adaptarse para reflejar la jerarquía y estructura general. Existirá también un incremento en la producción de los empleados. (37)

1.9. Sistemas de gestión de seguridad utilizados a nivel mundial y en la UCI

Actualmente en el mundo existen soluciones para el control de la seguridad de varias aplicaciones de manera centralizada. Alguna de las soluciones que proponen una seguridad centralizada es: AccessMaster IAM (Gestión de Identidades y Acceso), SSO (Single Sign-On) y la plataforma v-Go Single ON.

La investigación en la UCI sobre software de gestión de seguridad arrojó como resultado la existencia del Sistema de Gestión de Sesiones basado en la arquitectura SSO, que es 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[12]. También se encontró el uso del sistema Acaxia, dentro del Centro de Informatización de la Gestión de Entidades, sistema creado por el propio proyecto, sobre este y alguno de los mencionados se abordará a continuación.

1.9.1. SSO

Single sign-on (SSO) es un procedimiento de autenticación que habilita al usuario para acceder a varios sistemas con una sola instancia de identificación. (39)

Hay cinco tipos principales de SSO, también se les llama sistemas de autenticación reducida. (39)

- Enterprise single sign-on (E-SSO). (39)
- Web single sign-on (Web-SSO). (39)
- Kerberos. (39)
- Identidad federada. (39)
- OpenID. (39)

El tipo de arquitectura SSO a implementar en una determinada organización deberá determinarse con base en varios factores como son:

- Su capacidad de personalización y flexibilidad. (40)
- La complejidad deseada en la infraestructura. (40)
- Los recursos de tecnología informática disponibles. (40)
- Los recursos económicos disponibles. (40)

La implementación de la estrategia de Single Sign-On sugiere algunos beneficios adicionales para las organizaciones como son:

- Reducción de costos de administración de la seguridad. (40)
- Disminución de la operatividad asociada con la administración de contraseñas. (40)
- Incremento en los niveles de seguridad existentes. (40)
- Control centralizado de autenticación para las aplicaciones corporativas. (40)

[12] *Petróleos de Venezuela S.A: es la corporación estatal de la República Bolivariana de Venezuela que se encarga de la exploración, producción, manufactura, transporte y mercadeo de los hidrocarburos.* (38)

- Mayor comodidad y facilidad de uso de las aplicaciones corporativas para los usuarios finales. (40)

1.9.2. Acces Master IAM

Un sistema de gestión de identidades y accesos, es el centro de toda solución de seguridad. Identifica a los usuarios que trabajan en los sistemas, controla sus derechos de acceso para procesar datos y utilizar los recursos, evita los accesos no autorizados, documenta con precisión qué derechos de acceso tiene cada usuario y quién le ha dado los permisos, además de hacer un seguimiento de lo que hace cada usuario, cuándo y con qué resultado. (41)

IAM aporta tres bases funcionales:

- Una nueva oferta para unificar y controlar la gestión de identidad y la gestión de accesos. (42)
- Una solución completa y eficaz para la gestión de las identidades. (42)
- La mejor solución de gestión de autentificación, control de acceso y de SSO. (42)

Sus ventajas:

- Reducción de la complejidad. (42)
- Puesta en marcha óptima. (42)
- Retorno rápido de la inversión. (42)
- Nivel de seguridad mejorado. (42)

1.9.3. v- GO SSO

El v-GO SSO (v-GO Single Sign-On) almacena de forma segura los nombres de los usuarios y contraseñas de aplicación. Al iniciar cada una de sus aplicaciones, v-GO Single Sign-On automáticamente inicia la sesión con el nombre de usuario correcto y la contraseña para esa aplicación. Se tiene acceso rápido y seguro a sus aplicaciones para que puedan hacer su trabajo con eficacia. Este se puede instalar en equipos cliente, propiedad de la organización y puesta a disposición de la demanda de un portal de internet a cualquier equipo cliente. Como resultado, los usuarios tienen acceso a las aplicaciones desde cualquier lugar (escritorios corporativos, las máquinas de casa o las oficinas remotas). (43)

Seguridad

- v-GO SSO el motor de autenticación se basa en dos factores independientes que sólo están presentes en tiempo de ejecución y al azar cifrado durante la sesión en la memoria. (44)
- Credenciales fijados en todo momento en el directorio, en tránsito, en el cliente, y en la memoria. (44)
- Credenciales individuales sólo se descifra en la marcha en el momento de inicio de sesión. (44)

- Lleno FIPS 140-2 compatible con MS CAPI apoyo para 3DES, AES y RC4, así como para la generación de claves de hashing y servicios. (44)
- Defensas protección contra incumplimiento o inspección por parte de otros procesos. (44)
- Todos los componentes firmados digitalmente y en tiempo de ejecución validados. (44)

Beneficios

- Administrar contraseñas. (45)
    - Se autentican una vez, todos los demás complementos de inicio de sesión automático.
    - Reduce los costes mediante la eliminación de llamadas de los usuarios al servicio de asistencia para el restablecimiento de contraseñas.
    - Aumenta la seguridad al permitir el cumplimiento automático de toda la empresa con las directivas de contraseñas más fuertes.
- Acceso desde cualquier lugar. (45)
    - Los usuarios tienen acceso desde cualquier lugar a sus aplicaciones.
    - Evita revelar las contraseñas a los usuarios en lugares remotos.
- Gestión. (45)
    - Los usuarios ahorran tiempo al evitar un largo proceso de inicio de sesión de Windows.
    - Asegura el cumplimiento de HIPAA para reglas de seguridad mediante la suspensión de sesiones desatendida.
    - Elimina el uso de identificadores de genéricos y la incapacidad de seguimiento de las acciones específicas del usuario.
- Auditoría de cuentas de uso. (45)
    - Rápida conciliación de cuentas exactas.
    - Define acceso basado en roles a las aplicaciones.
    - Baja los costos de mantenimiento de software.

1.9.4. Acaxia

La seguridad de las aplicaciones del Sistema de Gestión de Entidades Cedrux, perteneciente Centro de Informatización de la Gestión de Entidades, se gestiona mediante el sistema Acaxia, éste debe su nombre a un acuerdo tomado por el propio proyecto de nombrar los sistemas que se realicen con nombres de árboles, en este caso el nombre del árbol es Acacia, la x es parte de la identidad del proyecto, por el modelo unix o sea es el detalle que lo caracteriza. Acaxia está concebido para garantizar la seguridad en un entorno de varias aplicaciones, desarrollado sobre el marco de trabajo Sauxe. (1)

Es un sistema de seguridad el cual se divide en 4 módulos: configurar nomencladores, configurar sistemas, configurar servidores y configurar usuarios. Gestiona la seguridad en un entorno de varias aplicaciones y garantiza temas tan importantes como: (1)

- Autenticación. (1)
- Autorización. (1)
- Auditoría. (1)
- Administración de perfiles. (1)
- Administración de conexiones. (1)
- Entornos multi-entidad. (1)
- Entornos multi-lenguaje. (1)
- Entornos multi-temas. (1)
- Compartimentación de la información. (1)

Brinda tres tipos de integración para reutilizar todas las ventajas que ofrece:

- **Integración a nivel de interfaz.** Esta forma permite mostrar diferentes aplicaciones desarrolladas en distintas plataformas web en el portal principal. (1)
- **Integración a nivel de servicios internos.** Para la integración mediante esta forma las aplicaciones deben estar desarrolladas sobre el mismo marco de trabajo, pudiendo así consumir sus servicios mediante el componente IoC[13]. (1)
- **Integración a nivel de servicios web.** Este tipo de integración permite a sistemas de arquitecturas diferentes comunicarse con Acaxia mediante servicios web. (1)

Brinda sus servicios a todos los sistemas que se suscriban a él. Para ello se gestionan las conexiones a la base de datos, las funcionalidades asociadas y las acciones que realizan las mismas.

1.10. Soluciones existentes

Actualmente en el proyecto ERP-Cuba no existe un documento que rija la seguridad de los sistemas de gestión de forma centralizada en un entorno de varias aplicaciones. Por lo que es muy importante crear la "Vista de arquitectura de seguridad", que es un documento rector, que brinda la posibilidad de tener una guía que estandarice la seguridad del Sistema de Gestión de Entidades Cedrux, permitiendo el avance más rápido

[13] *IOC del inglés Inversion of Control, es un concepto junto a unas técnicas de programación en las que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales, en los que la interacción se expresa de forma imperativa haciendo llamadas a procedimientos (procedure calls) o funciones.(1)*

del trabajo, disminuyendo así el tiempo de desarrollo de las aplicaciones y los riesgos de seguridad. Brindando a su vez una futura comodidad a los usuarios y desarrolladores que interaccionarán con el sistema.

1.11. Resultados esperados

Como resultados esperados se obtiene un documento de seguridad en el cual se cumple con los estándares internacionales existentes para la protección de la información y que cuenta con la descripción de:

- Las configuraciones seguras de software diferentes tanto en el entorno de desarrollo como en el entorno de despliegue.
- Una política de codificación segura en las capas de presentación, negocio y acceso a datos.
- Una estrategia de comunicación segura entre los diferentes componentes.
- Las validaciones necesarias a los diferentes niveles para evitar que se introduzcan datos maliciosos.
- Los componentes necesarios que deben formar parte de un sistema de gestión integral de seguridad, garantizando la administración centralizada de la seguridad en un entorno de varias aplicaciones.
- Un mecanismo de salvas de seguridad a utilizar por el proyecto ERP-Cuba.
- Una lista de chequeo que regule y controle el desarrollo y el despliegue del software.

1.12. Conclusiones

En el presente capítulo se han abordado aspectos referidos a la seguridad informática, las tecnologías, herramientas y estándares utilizados en el proyecto ERP-Cuba y el mundo, así como el hallazgo de la no existencia de un documento que rija la seguridad de sistemas de gestión.

La investigación realizada arrojó como resultado que los estándares, tecnologías y herramientas aportan muchas ventajas y que los sistemas de seguridad existentes estudiados están basados en la arquitectura SSO, solo garantizan la gestión de sesiones y los desarrollados fuera del país son muy costosos. Por lo que es necesario crear la guía de seguridad para beneficiar a todos los trabajadores, estudiantes y usuarios que interactúan con el Sistema de Gestión de Entidades Cedrux.

2

CAPÍTULO II Seguridad en el entorno de desarrollo y despliegue

2. Introducción

Actualmente en el proyecto ERP-Cuba no se cuenta con un documento rector de la seguridad de los sistemas de gestión, por lo que este capítulo está enmarcado en describir algunos de los aspectos que son necesarios para constituir la guía de la seguridad del mismo. En la presente sección se ofrece una explicación de cómo estandarizar y configurar el entorno de desarrollo y despliegue de las aplicaciones pertenecientes al Sistema de Gestión de Entidades Cedrux.

2.1.1. Configuración del servidor web

El servidor web que se utiliza es el Apache. Para evitar ataques y reducir vulnerabilidades, se deben realizar algunas configuraciones como: realizar parches de seguridad, desactivar módulos innecesarios, ver las cuentas del servidor web, desactivar las opciones para explorar directorios, algunos ejemplos son:

- **Apache debe funcionar bajo su propia cuenta y grupo de usuario:** algunas versiones de Apache corren bajo el usuario nobody, esto compromete mucho su seguridad por lo tanto se debe poner *apache* como nombre de usuario y grupo (User apache, Group apache).
- **Asegurar que los archivos a los que se accede son los deseados:** no se desea que se pueda acceder a los directorios que no tengan permisos para ello.
- **Desactivar las opciones para explorar directorios:** esto se realiza con las opciones de directiva dentro de la etiqueta directorio, tiene dos posibles valores none o indexes.
- **Restringir acceso por IP:** si se tiene un recurso al que deba solamente tener acceso alguna red, o IP en concreto, puede configurarlo en Apache.
- **Desactivar los includes del lado servidor:** esto también se hace con las opciones de directiva dentro de la etiqueta directorio, tiene dos posibles valores none o include.

- **No permitir que apache siga enlaces simbólicos:** se configura con las opciones de directiva dentro de la etiqueta directorio, tiene dos posibles valores none o FollowSymLinks.
- **Desactivar todas las opciones:** si desea desactivar el uso de todas las opciones, simplemente: Options None. Si solamente desea desactivar algunas en concreto, sepárelas con un espacio.
- **Desactivar la ayuda para los archivos .htaccess:** esto está ya hecho, pero, con la directiva AllowOverride cámbielo a none, AllowOverride None. Otra opción interesante sería bloquear la descarga de todos los archivos que comienzen con .ht.
- **Disminuir el valor máximo de tiempo de espera:** por defecto, el tiempo de espera es de 300 segundos. Puede disminuirlo por seguridad, para prevenir ataques.
- **Limitar el tamaño máximo de peticiones:** Apache tiene varias directivas que permiten que limite el tamaño de una petición, esto puede ser muy útil.
- **Ocultar la versión y otra información delicada:** por defecto muchas instalaciones de Apache muestran el número de versión que está funcionando, el sistema operativo y un informe de módulos de Apache están instalados en el servidor. Los usuarios maliciosos pueden utilizar esta información para atacar el servidor. Hay dos directivas que necesita agregar, o corregir en su archivo de httpd.conf, estas son: ServerSignature Off y ServerTokens Prod.

Para mayor información remitirse al epígrafe 1.1 titulado: Configuración del servidor web, del documento "Vista de arquitectura de seguridad del proyecto ERP-Cuba", el cual puede encontrar en la dirección: http://10.12.179.3:3389/svn/erp/ERP/Ingenieria/Arquitectura/Desarrollo/ArquitecturaSeguridad/Expediente de Arquitectura de Seguridad/Vista de seguridad/Vista de la arquitectura de seguridad del proyecto ERP -Cuba V-2.3.doc.

2.1.2. Configuración del servidor de base de datos

El sistema gestor de base de datos a utilizar es el Postgres a continuación se hace un breve análisis acerca de la arquitectura del mismo y de los principales elementos que se deben configurar para garantizar la seguridad de los datos.

Arquitectura de Postgres. Estructura de ficheros

- **postgressql.conf**: fichero de configuración principal contiene la asignación a los parámetros que configuran el funcionamiento del servidor.
- **pg_hba.conf:** fichero de configuración de la autenticación de los clientes, usuarios y del acceso a las bases de datos del clúster.

- **pg_ident.conf:** fichero accesorio al anterior, determina como se realiza la autenticación ident que contiene la correspondencia entre usuarios del sistema operativo y de Postgresql.
- **PG_VERSION:** fichero de texto con la versión de software Postgres que crea el clúster.

Seguridad de las bases de datos en los servidores: En servidores de bases de datos se pueden mencionar 4 niveles básicos de seguridad.

- **Seguridad de acceso al sistema:** Se implementa de dos formas posibles, a nivel de sistema operativo, en cuyo caso el Sistema Gestor de Bases de Datos se apoya en la seguridad de entrada al sistema operativo para comprobar la validez del acceso a los datos almacenados, o bien lo que se llamará modo mixto, en el cual la seguridad de entrada a la información la llevará a cabo el propio servidor de datos a partir de la definición de cuentas de usuarios.
- **La seguridad a nivel de objetos:** Detalla el acceso a nivel de creación y administración de objetos de datos: tablas, vistas, índices, relaciones, reglas. Es decir las responsabilidades o acciones que el usuario puede hacer en el esquema de base de datos.
- **La seguridad a nivel de datos:** Se realiza en la capa de información, donde se indicará quién puede acceder a qué información para su consulta, actualización, inserción o borrado.
- **Seguridad a nivel de protección de los almacenamientos físicos de la información:** Es tarea del sistema operativo, de los archivos de datos del sistema y las políticas de copias de seguridad y restauración de los datos.

Principios básicos de la seguridad de las bases de datos.

- **Disgregación de responsabilidades:** plantea que el usuario administrador de bases de datos (ABD) debe ser de total confianza y tiene la responsabilidad de mantener la integridad de los datos.
    - El ABD y el administrador del sistema operativo (ASO) no deben ser la misma persona.
    - Los privilegios del usuario operador y los del ABD no deben ser los mismos.
    - Los usuarios no deben compartir sus cuentas ni sus contraseñas.
- **Mínimo de privilegios:**
    - Sólo se debe instalar en los servidores de software requeridos.
    - Habilitar sólo los servicios y puertos requeridos.
    - La cuenta de root y de administrador sólo la debe tener el personal requerido.
    - Restringir las conexiones remotas usando el fichero pg_hba.conf.
- **Estándares de seguridad y contraseñas:**
    - El número mínimo de caracteres de las contraseñas de los usuarios será de 7 caracteres.

- La contraseña del usuario no debe ser igual al nombre del usuario.
- La contraseña debe tener fortaleza requerida con la utilización de letras, números y caracteres especiales.
- La contraseña debe cambiarse como máximo cada tres meses.

- **Acceso restringido desde la red:** Es recomendable usar siempre un firewall que mantenga protegido al servidor para contrarrestar los ataques. El firewall debe ser configurado para que sólo tenga acceso al servidor la red o subred requerida, con esto se protegen los datos. El primer aspecto que se debe tener en cuenta después de haber instalado PostgresSQL, es el de seguridad:
  - **Seguridad en la manipulación de los ficheros de postgres:** El directorio más crítico que presenta PostgresSQL es el PGDATA en el que todos los ficheros que se encuentran dentro de él pertenecen al usuario postgres. Este usuario es el único que tiene permisos de lectura y escritura sobre los directorios y ficheros dentro del PGDATA.
  - **Seguridad en el acceso a los clientes:** Este punto es importante, aquí se definen quienes se pueden conectar al servidor, a que base de datos y con qué usuario (la configuración de este punto se realiza en el fichero pg_hba.conf), en éste fichero se editan una serie de reglas que se procesan en forma descendente.
  - **base_datos:**
    - **ALL:** Permite la conexión a cualquier base de datos.
    - **SOMEUSER:** Permite la conexión solo a bases de datos que su nombre sea el mismo que el usuario que se conecta.
    - **SOMEROL:** Permite la conexión solo a bases de datos que su nombre sea el mismo que el del rol que se conectó.
  - **Usuario:**
    - **ALL:** Permite la conexión de cualquier rol.
    - **role1, [+] role2,...:** Permite la conexión de los roles de la lista y además se permite la conexión de cualquier rol que sea miembro de role2.
    - **@fichero:** Permite la conexión de los roles incluidos en el fichero, que debe estar en el mismo directorio que pg_hba.conf.
  - **Método-autenticación:**
    - **TRUST:** Conexión aceptada sin condiciones.
    - **REJECT:** Conexión rechazada sin condiciones.

- **PASSWORD:** Se solicita palabra de paso sin encriptar, las palabras de paso se almacenan en la tabla pg_authid y pueden estar cifradas o no según como se crea el rol.
- **CRYPT:** Palabra de paso encriptada.
- **MD5:** Palabra de paso con el método de encriptación md5, y se almacena también con este método. Es el método recomendado por PostgresSQL.
- **KRB5:** Usa Kerberos v5 para autenticar al cliente, se habilita en la instalación del servidor.

Para la configuración del fichero pg_hab.conf se debe tener en cuenta las políticas de cómo se podrá acceder al servidor, si permite una conexión local o una remota o ambas inclusive.

Antes de configurar dicho fichero existe otro llamado postgresql.conf, éste es el fichero de configuración de PostgreSQL, en él existe una sección llamada CONECCTION AND AUTHENTICATION, en la misma es donde se configuran las conexiones y cómo va a ser la seguridad.
La configuración que trae por defecto el fichero pg_hba.conf no es muy segura, pues permite que cualquiera se conecte al servidor tanto local como de forma remota sin contraseñas.

Tabla 3: Configuración que trae por defecto el fichero pq_hba.conf.

Tipo	**BD**	**Usuarios**	**Dirección**	**Método**
# " local" es sólo para conexiones socket de dominio unix				
local	todos	todos		seguro
# IP v 4 conexiones locales				
host	todos	todos	127.0.0.1/32	seguro
# IP v 6 conexiones locales				
host	todos	todos	::1/128	seguro

Para tener un control más detallado se usará la siguiente configuración, solo se puede conectar el usuario administrador de PostgreSQL, a todas las bases de datos de forma local, y de forma remota nada más los ip que se especifiquen, los dos utilizando el método md5.

Tabla 4: Configuración utilizando el MD5.

Tipo	**BD**	**Usuarios**	**Dirección**	**Método**
# " local" es sólo para conexiones socket de dominio unix				
local	todos	todos		seguro

local	todos	postgres		md5
# IP v 4 conexiones locales				
host	todos	todos	127.0.0.1/32	seguro
host	SIGLA	postgres	10.32.19.12/32	md5
# IP v 6 conexiones locales				
host	todos	todos	::1/128	seguro

En el ejemplo anterior se utiliza el usuario postgres como super usuario pero se puede utilizar cualquier usuario previamente creado. Para permitir que varios clientes se conecten al servidor usando conexión TCP/IP y los mismos están en una misma red, se especifican los permisos de la siguiente forma:

- 10.32.19.0/24 ó 10.32.19.0 255.255.255.0: Se pueden conectar todas los IPs de la red 10.32.19.
- 10.32.0.0/16 ó 10.32.0.0 255.255.255.0: Se pueden conectar todas los IPs de la red 10.32.
- 10.32.19.12/.32: Sólo se puede conectar ese IP de la red.
- 0.0.0.0/0 ó 0.0.0.0.0.0.0.0: Se puede conectar cualquier IP.

Es importante ponerle una contraseña fuerte al usuario administrador, para así de esta forma proteger la conexión de autenticación desde conexiones remotas utilizando el método md5 pero las consultas realizadas dentro de psql viajarán de forma plana.

2.1.3. Configuración del lenguaje de programación

PHP viene por defecto configurado para desarrollo, pero en producción hay opciones que no son recomendables sobre todo por temas de seguridad. Se debe cambiar en el *php.ini* las siguientes opciones para mejorar la seguridad en el entorno de producción:

- **Desactivar el acceso a ficheros remotos**: las funciones *fopen*, *file_get_contents*, y *include* permiten el acceso a ficheros remotos (*http://host/...*), lo cual puede dar problema en temas de seguridad. Si se necesita acceder a ficheros remotos se puede usar *fsockopen* o funciones de CURL.
- **Restringir a qué ficheros puede acceder PHP**: normalmente PHP sólo necesita acceder a ficheros situados en cierto path, por lo que para evitar que se acceda a otros paths, es conveniente restringir su acceso.
- **Modo seguro**: La posibilidad de escritura en variables de entorno usando la función puntenv () está restringida con el uso del modo seguro.

Otras restricciones del modo seguro

- Acceso permitido a ficheros binarios.
- Acceso a variables de entorno.
- Controlar límites.
- Control de acceso a ficheros mediante Apache.
- Evitar el acceso a la Shell.
- Carga dinámica de módulos.
- Información sobre php en las cabeceras del servidor.
- Desactivando funciones y clases.

Si ninguno de los scripts de las aplicaciones que se ejecutan en php necesita estas funciones, es recomendable deshabilitarlas.

2.1.4. Configuración de la herramienta de control de versiones

Como herramienta para el control de versiones se propone el uso del Subversion. Como clientes svn se propone el uso de TortoiseSVN para Windows y RapidSVN para Linux. TortoiseSVN es un cliente Subversion implementado como una extensión al shell. Es software libre liberado bajo la licencia GNU GPL.

RapidSVN es un cliente gráfico para Subversion. Es fácil de usar, tanto por quienes ya conocen Subversion como para quien empieza, pudiendo acceder a direcciones SVN, subir y descargar contenido y sincronizarlo con el servidor original, comprobar su estado, crear y fusionar direcciones.

Para asignar los permisos en la herramienta para el control de versiones Subversion se crean roles o grupos a los cuales se le asignan usuarios a los cuales se les da permiso a partir del grupo al que pertenezca.
Se crean los siguientes grupos:

- Jefes de línea.
- Analista.
- Gestor de calidad.
- Planificador.
- Gerente.
- Gestor de configuración.
- Arquitecto principal.
- Líder de proyecto.
- Consultoría.
- Implantación.

- Presentación.

Además de estos grupos se crea uno para cada línea de desarrollo y a partir de estos se les dará permisos de lectura y/o escritura en correspondencia al rol que desarrolle cada persona en el proyecto.

2.1.5. Mecanismos de salva de seguridad

Realizar salvas de seguridad de los datos contenidos en el servidor de control de versiones resulta de gran importancia, ya que de esta forma se asegura que toda la información del proyecto incluyendo el código fuente se encuentre resguardada en una copia diferente.

- Se deben realizar salvas de la BD todos los días en el servidor destinado para este fin y 3 veces al día en una PC de la línea.

Además de esto se proponen las siguientes medidas:

- Realizar salva del repositorio y de la base de datos en un servidor distinto al servidor de desarrollo dentro del nodo.
- Realizar salva del repositorio y de la base de datos en un disco duro externo.
- Las salvas se realizarán 2 veces al día a las 5 pm y a las 3 am.
- Las salvas se mantendrán por un período de 7 días.
- Las versiones estables se guardarán en un directorio con acceso restringido.

Igualmente se propone que el sistema tenga implementado un módulo para la realización de salvas automáticas (configurables) o manuales.

- También se le deben realizar salvas al servidor LDAP.
- Es recomendable que durante la explotación del software se realicen salvas al menos una vez al día.

2.2. Configuración del entorno de despliegue

Es necesaria la seguridad durante la explotación de la aplicación, para garantizar la integridad y la protección de los datos ante cualquier ataque o contingencia. Para ello se debe realizar una configuración segura tanto en el servidor de aplicaciones como en el servidor de base de datos. Se debe tener implementado algún mecanismo para el control de la integridad de los datos y del código fuente, se debe realizar la ofuscación del código para protegerse de ataques de ingeniería inversa y escoger correctamente la tecnología que se necesita para el despliegue del software.

Para mayor información remitirse al epígrafe 2 titulado: Configuración del entorno de despliegue, del documento “Vista de arquitectura de seguridad del proyecto ERP-Cuba”, el cual encontrará en la dirección: http://10.12.179.3:3389/svn/erp/ERP/Ingenieria/Arquitectura/Desarrollo/ArquitecturaSeguridad/Expediente de

Arquitectura de Seguridad/Vista de seguridad/Vista de la arquitectura de seguridad del proyecto ERP -Cuba V-2.3.doc.

2.3. Soluciones de apoyo a la seguridad

Una solución es la respuesta a un problema; una herramienta, es un subprograma o módulo encargado de funciones específicas y afines entre sí, que aumenta la capacidad de hacer ciertas tareas; relacionando estos dos conceptos, una solución a un problema, puede ser una herramienta que sea capaz de cumplir con las necesidades que se requieran. Debido a la necesidad de apoyar la seguridad dentro del Centro de Informatización de la Gestión de Entidades, se diseñaron y fabricaron algunas herramientas como soluciones, para cumplir uno o más propósitos específicos. Este epígrafe engloba las herramientas realizadas por el proyecto, para apoyar la seguridad, así como, los elementos a tener en cuenta en la utilización del marco de trabajo Sauxe, el cual es también una solución.

2.3.1. Estructura y configuración del marco de trabajo Sauxe

La estructura y configuración de Sauxe resuelve un grupo de escenarios que pueden traer consigo violaciones de seguridad y errores en la integración de sistemas, a continuación se describen un grupo de elementos a tener en cuenta en la utilización de esta tecnología:

Seguridad:

- Acceso a los ficheros de desde el cliente sin pasar por el controlador frontal (configuración, cache, session, clases y plantillas).
- Cruzamiento de sesiones.

Integración con otros sistemas:

- Estructura física de los frameworks (symfony) más usados en la UCI que están integrados con el ERP (alfaomega, aduana, patdsi, etc).

Portabilidad:

- Separación física de las capas del marco de trabajo (presentación, negocio y dominio).

Estructura:

- Permite separar la capa de presentación de la capa de negocio y del dominio.
- Permite que aplicaciones desarrolladas sobre distintos frameworks mantengan una misma estructura física.
- Creación de un VirtualHost para que solo queden públicos los ficheros de presentación (controlador frontal, js, css, imágenes) y para que otras aplicaciones no puedan acceder a la sesión de Cedrux.

Aclaraciones importantes

- La carpeta de publicación del marco de trabajo Sauxe es web.
- Es recomendable que la carpeta que se cree para descargar el repositorio este fuera del www que crea apache por defecto para evitar conflictos entre los VirtualHost.

2.3.2. Auditor

CRC (cyclic redundancy check, en español comprobación de redundancia cíclica): es un código de detección de error cuyo cálculo es una larga división de computación en el que se descarta el cociente y el resto se convierte en el resultado, con la importante diferencia de que la aritmética que se usa conforma que el cálculo utilizado es el arrastre de un campo finito, en este caso los bits. El tamaño del resto es siempre menor que la longitud del divisor, que por lo tanto, determina el tamaño del resultado. La definición de un CRC especifica el divisor que se utilizará, entre otras cosas, aunque CRC se puede construir utilizando cualquier tipo de regla finita, todos los CRC de uso común emplean una base finita binaria, esta base consta de dos elementos, generalmente el 0 y 1.
En el proyecto ERP-Cuba se implementó el componente CRC que se muestra en la figura 15, para detectar violaciones de modificación de código fuente.

El proceso de control de integridad comienza cuando el sistema se va a desplegar en alguna entidad, el departamento de tecnología del proyecto ERP-Cuba le hace un cálculo CRC al sistema, el mismo firma todos los archivos y devuelve el resultado del cálculo.

Cuando se desea ejecutar una auditoría al código de la aplicación, se carga por un lado, el cálculo protegido por el departamento de tecnología y por el otro, la del código de la aplicación desplegada y se efectúa la comparación. En este momento el componente es capaz de detectar si existe alguna modificación del código desplegado.

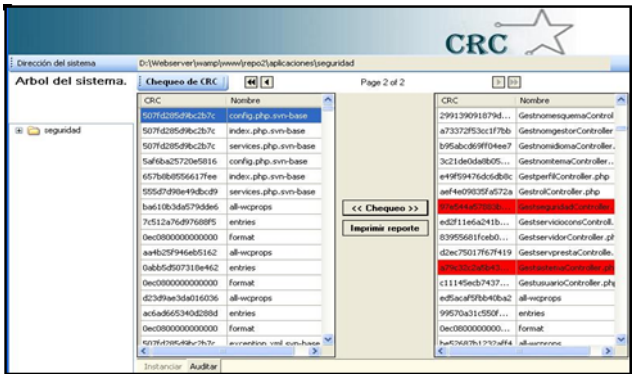


Figura 7: Componente CRC.

2.3.3. Ofuscador de código

Ofuscar significa, deslumbrar, oscurecer, turbar la vista, hacer sombra, trastornar, confundir las ideas o alucinar. Ahora bien, un ofuscador se define como un algoritmo O tal que, para cualquier programa P, O (P). Es también un programa con las características siguientes:

- **Funcionalidad**: el programa ofuscado debe tener la misma funcionalidad como el programa original.
- **Eficacia**: El programa ofuscado no debe ser mucho menos eficiente que el programa original.
- **Ofuscación**: Esto significa que el código del programa ofuscado debe ser difícil de entender.

En el entorno web la mayoría de programas que se ejecutan, lo hacen interpretando el código fuente, como Perl, Python, ASP, JavaScript, PHP. Lo que plantea un problema con respecto a la protección del trabajo intelectual invertido en el desarrollo de software, que les supone un lastre a la hora de competir comercialmente con el resto de los lenguajes.
Para protegerse de ataques haciendo frente a la ingeniería inversa, se debe ofuscar el código oscureciéndolo de forma tal que resulte prácticamente imposible interpretarlo correctamente, incluyendo bucles irrelevantes, cálculos innecesarios, funciones que no sirven para nada, nombres de funciones y de variables que no indiquen su cometido, generando confusión que finalmente desanime al interesado.

En este sentido el término ofuscación se refiere a la realización de cambios no destructivos ya sea en el código fuente intermedio o código máquina, con el fin de que no sea fácil de entender o leer. Como efectos colaterales de la ofuscación normalmente se consiguen programas más pequeños, al reducirse o eliminarse información innecesaria, que serán más rápidos en la carga y en la ejecución.

Para garantizar la protección del código durante la explotación del Sistema de Gestión de Entidades Cedrux se propone el uso de la herramienta Encoding PHP (PHP Encoder), hasta que se termine la solución que se está desarrollando en el centro de gestión localizado en la provincia de Holguín.

Figura 8: PHP Encoder.

Ventajas:

- Codificación de scripts PHP con bytecodes compilados para el rendimiento en tiempo de ejecución acelerada y la máxima seguridad.
- Cifrar archivos arbitrarios, por ejemplo, archivos XML, archivos de plantilla de Smarty, o imágenes.
- Generar los archivos de licencia para restringir el acceso a los archivos codificados.
- Prevención de la manipulación de archivos mediante el uso de firmas digitales.
- No necesita pagar licencia por servidores distribuidos.

Un archivo ofuscado se verá así:

Figura 9: Ejemplo de un archivo ofuscado.

2.4. Políticas

Las políticas de seguridad informática o PSI como se suelen llamar, surgen como una herramienta organizacional para concientizar a cada uno de los miembros de una organización sobre la importancia y sensibilidad de la información y servicios críticos. Por lo que es un recurso valioso que amerita la dedicación de tiempo y esfuerzo que brindará una base sólida para respaldar el plan general de seguridad y para resguardar un sistema sólido. En este epígrafe se norman todas las políticas a tener en cuenta en el desarrollo de las aplicaciones del Sistema de Gestión de Entidades Cedrux.

2.4.1. Política de codificación segura

2.4.1.1. Capa de presentación

Política de codificación segura a tener en cuenta en la capa de presentación:

- Prohibir el uso de EVAL[14].

[14] *Es una función que evalúa el contenido pasado como parámetro como si fuera una expresión.* (46)

- Realizar peticiones POST[15].
- No se deben realizar peticiones a URL [16] externas.

2.4.1.2. Capa de negocio

Política de codificación segura a tener en cuenta en la capa de negocio:

- Para las validaciones se recomienda el uso del componente validación.
- Acceso a ficheros.
- Se recomienda el uso de cache para resolver el problema del abrazo fatal.
- Se recomienda el uso de sesiones.
- Se recomienda el uso de variables globales.

2.4.1.3. Capa de acceso a datos

Política de codificación segura a tener en cuenta en la capa de acceso a datos:

- SQL Inyección.
- Utilizar el Doctrine validator.
- Utilizar la validación de conexiones.

2.4.2. Política para la base de datos

- Se debe llevar a 3 niveles:
  - Nivel de gestor: la no autenticación de los usuarios.
  - Nivel de objetos: Los usuarios no pueden ver ni tablas, ni esquemas.
  - Nivel de datos: Los usuarios no pueden ver los datos.

2.5. Estándares

Los estándares aportan muchas ventajas pues proporcionan un modelo arquitectónico coherente, en el cual se pueden integrar distintas soluciones y se pueden realizar evoluciones y actualizaciones de forma controlada, brindándoles a los usuarios la posibilidad de desear cambios que aportarían un mínimo riesgo. Se

---

[15] *Es un método, consiste, básicamente, en enviar las variables de forma "oculta" para que nadie pueda ver los datos que ha enviado al pulsar el botón de envío de formulario.* (47)

[16] *Un localizador uniforme de recursos, más comúnmente denominado URL (sigla en inglés de uniform resource locator), es una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en una red para su localización o identificación.* (48)

puede decir que: sin estándares no hay un punto de partida para realizar mejoras en un proceso, operación o actividad. En éste epígrafe se norman los estándares a tener en cuenta.

2.5.1. Estándar de validación

2.5.1.1. Capa de presentación

La validación en la capa de presentación se basa en interceptar el momento en que se realiza el envío de los datos. Se validarán cada uno de los campos del navegador. Si existe algún campo no relleno o con información errónea, se mostrará el campo que está incorrecto y se solicitará al usuario que lo cambie. Algunas de las validaciones a tener en cuenta son:

- Comprobar que se han suministrado todos los campos obligatorios.
- Comprobar que el formato de un campo es el esperado (fechas, teléfonos).
- Comprobar la validez (sintáctica) de las direcciones de correo y URL.
- Comprobar que no se sobrepasa la longitud, número de líneas o tamaño de la entrada.

2.5.1.2. Capa de negocio

Se utilizará la clase ZendExt_Validation para buscar acciones, verificar las precondiciones de la acción y validar los datos según el tipo de dato declarado en el xml, para la acción específica. Se usarán varias expresiones regulares para las validaciones, estas se pueden ver en el epígrafe 4.1.2 titulado: Capa de negocio, en el documento “Vista de arquitectura de seguridad del proyecto ERP-Cuba” el cual encontrará en la dirección: http://10.12.179.3:3389/svn/erp/ERP/Ingenieria/Arquitectura/Desarrollo/Arquitectura Seguridad/Expediente de Arquitectura de Seguridad/Vista de seguridad/Vista de la arquitectura de seguridad del proyecto ERP -Cuba V-2.3.doc.

2.5.1.3. Capa de acceso a datos

- Utilizar el Sistema Gestor de Base de Datos (SGBD): PostgreSql. El servicio de base de datos se brindará mediante el SGBD PostgreSql, mínimo la versión 8.3.
- Como cliente de administración en Linux se recomienda utilizar pgadmin3, como mínimo la versión 1.8.
- Utilizar servidores de base de datos centrales para desarrollo y pruebas. No se deben utilizar servidores de base de datos local para el desarrollo.
- Para el desarrollo se debe utilizar el servidor central:
- Respaldos de la base de datos central.
- Para los nomencladores se establece que los identificadores pertenecerán a:

  - *NUMERIC (1,0)* si el crecimiento es 0 < X < 10 (X ∈ R).
  - *NUMERIC (3,0)* si el crecimiento es 9 < X < 1000 (X ∈ R).

- Para los valores enteros se establecen:
  - *NUMERIC (5,0)* admite enteros de 5 cifras (2 bytes).
  - *NUMERIC (10,0)* admite enteros de 10 cifras (4 bytes).
  - *NUMERIC (19,0)* admite enteros de 19 cifras (8 bytes).
- Para los valores de coma flotantes, se define:
  - *NUMERIC* con grado de precisión después de la coma es de 2 cifras.
  - *NUMERIC* si el grado de precisión después de la coma es de 8 cifras.
- Para los campos autoincrementables se definirá una secuencia. De manera que no se utilice otro tipo de dato para los valores numéricos.
- Para todos los campos que sean fecha se define:
  - *DATE* representa los tipos de datos fecha.
- Para los valores condicionales (BOOLEAN), se define:
  - *NUMERIC* (1,0) para los valores condicionales.
- Para el trabajo con imágenes, se establece:
  - *BYTEA* representado en Visual Paradigm por el tipo de dato BLOB.
- Las cadenas serán representadas por el tipo de dato VARCHAR, con diferentes longitudes de cadena.
- **Tamaño de la base de datos:**
  - Compruebe el número de usuarios registrados en la base de datos y elimine usuarios en caso necesario. El número aceptable de usuarios varía en función del rendimiento del servidor y de la red.
  - Quite cualquier archivo temporal.
  - Si los proyectos disponen de información de historial obsoleta, quítela.
  - Elimine todos los archivos y proyectos o esquemas (*shemas*) que no se utilicen en la base de datos.
  - En caso de que haya proyectos o esquemas (*shemas*) que no tienen interrelación, muévalos según sea necesario con el fin de separar las bases de datos.
  - También puede utilizar la utilidad *VACUUM* y *ANALYZE* para darle mantenimiento a su base de datos y además explorar su integridad.

De manera general y en el caso particular del rendimiento de PostgreSQL, este se puede ver atacando desde dos frentes:

- **Hardware**: configurando y optimizando los recursos hardware de que va a disponer el servidor, como por ejemplo la memoria.
- **Software**: utilizando técnicas basadas en software como el uso de índices, optimización de consultas, VACUUM, etc.

Es importante lograr un rendimiento del 0% al 100% modificando tan sólo algunos parámetros del fichero de configuración postgresql.conf.

Con la utilización de la herramienta VACUUM se puede liberar espacio en disco debido a que elimina toda la basura que deja el gestor de base de datos, cuando se realizan muchas operaciones de DELETE, UPDATE e INSERT, esto se puede lograr poniendo una tarea programada que realice la operación antes de empezar a usar la base de datos y por la tarde cuando se termine de usar.

2.5.2. Estándares de nomenclatura del código

2.5.2.1. Nomenclatura según el tipo de clases

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing[17]. Ejemplo: GestionarUsuario.

- **Clases controladoras:** Las clases controladoras después del nombre llevan la palabra:"Controller". Ejemplo: GestionarUsuarioController.
- **Clases de los modelos**
  - **Business (Negocio):** Las clases que se encuentran dentro de Business después del nombre llevan la palabra: "Model". Ejemplo: MonedaContModel.
  - **Domain (Dominio):** Las clases que se encuentran dentro de Domain el nombre que reciben es el de la tabla en la Base de Datos.Ejemplo: User.
  - **Generated (Dominio bases):** Las clases que se encuentran dentro de Generated el nombre comienza con la palabra: "Base" y seguido el nombre de la tabla en la Base de Datos.Ejemplo: BaseUser.
- **Clases del framework :** Como parte del marco de desarrollo de Zend existe el Zend_Loader (Cargador) que, amén del cumplimiento de ciertas normas para la nomenclatura de las clases garantiza que, a partir de una ruta de inclusión, este sea el responsable de la inclusión de los recursos requeridos en el proceso. Ejemplo: Los nombres de las clases contienen la dirección donde

[17] *Es como la notación húngara pero sin prefijos. En este caso, los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula.* (49)

se encuentran, de la siguiente forma, si se tiene una clase llamada Condado en la siguiente estructura Cuba/VC/SantaClara/Condado.php entonces un identificador de la misma debe ser Cuba_VC_SantaClara_Condado lo que garantiza que a través de una casuística [18] particular el cargador localice estos recursos.

2.5.2.2. Nomenclatura de las funciones

El nombre a emplear para las funciones se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing[19], y con sólo leerlo se reconoce el propósito de la misma. Ejemplo: insertarMoneda.En caso de ser una acción de la clase controladora se le pone el nombre y seguida la palabra:"Action".Ejemplo: insertarMonedaAction.

2.5.2.3. Nomenclatura de las variables

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

2.5.2.4. Nomenclatura de las constantes

El nombre a emplear para las constantes se escribe con todas las letras en mayúscula. Ejemplo: MONEDA.

2.5.2.5. Nomenclatura de los atributos

El nombre a emplear para los atributos se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing. Además en caso de ser un objeto se comienza con:"_" y después se escribe el nombre. Ejemplo: intMoneda=dinero, objMoneda=_dinero.

2.5.2.6. Nomenclatura en las llaves

Cada vez que se vaya a cerrar una llave se escribe en forma de comentario lo siguiente: Fin y el nombre de lo que se esté cerrando. Ejemplo ver Anexo 2**: Nomenclatura de las llaves. Anexo 3: Nomenclatura de sangría o indexado.**

---

[18] *En informática, cada situación que pueda darse en base a cierta funcionalidad, para la que habrá que proveer de métodos concretos para cada caso particular.(50)*

[19] *Camel Casing es un procedimiento de programación común en el lenguaje Java. Es parecido al Pascal Casing con la excepción que la letra inicial del identificador no debe estar en mayúscula.(49)*

2.5.2.7. Nomenclatura de estilo del código

En la implementación cuando se vaya a escribir una sentencia en php la forma de utilizar los tab del mismo es la siguiente:

```
<?php
// código
?>
```

- **Sangría o indexado:** La política de sangría a utilizar en la implementación es por tab. Declaraciones dentro del cuerpo de la clase. Las clases se comienzan a declarar pegado al margen izquierdo, después de poner el nombre de la clase se pone un espacio y se abre llave en la misma línea. Ejemplo Anexo 2**: Nomenclatura de las llaves.** **Anexo 3: Nomenclatura de sangría o indexado.**.

- **Declaraciones dentro del método/el cuerpo de la función.** Las declaraciones dentro del método/el cuerpo de la función se realizan como se muestra en el
-
- Anexo **4: Declaraciones dentro del método.** **Anexo 5: Brazas o llaves.**.
- **Brazas o llaves:** En la declaración de clases o interfaces, métodos, bloques y switch, la apertura de llaves se hace en la misma línea. Ver ejemplo en el
-
- Anexo **4: Declaraciones dentro del método.** **Anexo 5: Brazas o llaves.**

2.5.3. Estándares de nomenclatura de la base de datos

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

2.5.3.1. Apariencia de los esquemas

El nombre a emplear para los esquemas se escribe con todas las letras en minúscula, comenzando por el prefijo mod, a continuación el símbolo "_", y por último el nombre del módulo. Ejemplo: create schema 'mod_finanza'.

2.5.3.2. Nombres de las tablas

El nombre empleado, debe escribirse con todas las letras en minúscula para evitar problemas con el *Case Sensitive* del gestor y con solo leerlo se reconoce el propósito de la misma. Ejemplo: create table 'nom_producto'.

2.5.3.3. Prefijos a utilizar en la creación de tablas

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

- **dat_**Prefijo utilizado en tablas que almacenan la mayor cantidad de características de una entidad.
- **nom_** Prefijo utilizado en tablas nomencladoras.
- **seg_**Prefijo utilizado en tablas que almacenan control de acceso, usuarios y opciones de acceso de uno o varios sistemas. (Tablas de Seguridad).
- **conf_** Prefijo utilizado en tablas que almacenan parámetros de configuración del sistema**.** (Tablas de Configuración).
- **tmp_** Prefijo utilizado para tablas que almacenan datos transitorios. (Tablas Temporales).
- **his_** Prefijo utilizado para tablas que almacenan datos por largos períodos de tiempo y que solo son utilizados para análisis esporádicos. (Tablas Históricos).
- **res_** Prefijo utilizado para las tablas resúmenes, empleadas en los reportes.

2.5.3.4. Apariencia de los campos

El nombre a emplear para los campos se escribe con todas las letras en minúscula, con solo leerlo se reconoce el propósito del mismo y debe incluir un comentario con su descripción. Si el campo es un identificador debe empezar con id. Ejemplo: add field 'idproducto'. cantemb: cantidad de embalajes.

2.5.3.5. Nombre de las llaves primarias

El nombre de las restricciones se escribe con minúscula. Comienza con el identificador id. Seguido el nombre de la tabla todo junto y en minúscula. Ejemplo: idcuenta (Llave primaria de la tabla dat_cuenta).

2.5.3.6. Nombre de las llaves foráneas

El nombre de las llaves foráneas se escribe con minúscula y el nombre de la llave primaria de la tabla donde pertenece. Ejemplo: idcuenta. (Llave foránea de la tabla "dat_cuenta").

2.5.3.7. Nombres de las funciones, triggers, tipos de datos y vistas

El nombre empleado permite con sólo leerlo reconocer el propósito del mismo. Se utilizan los prefijos siguientes para la denominación de funciones, triggers, tipos de datos y vistas.

- **f** Funciones (Procedimientos Almacenados).
- **ft** Funciones de triggers.
- **t** Triggers.
- **td** Tipo de datos.
- **v** Vistas.

2.6. Conclusiones

En este capítulo se argumentaron algunos de los aspectos necesarios para mantener la seguridad del Sistema de Gestión de Entidades Cedrux, conformándose así la mayor parte de la vista de seguridad.

Dentro de esta sección se abordaron los temas referentes a la configuración del servidor de base de datos, el servidor web y el lenguaje de programación.

También el uso de herramientas, estándares, validaciones y requisitos a tener en cuenta para brindar seguridad tanto en el entorno de desarrollo como en el entorno de despliegue.

3

CAPÍTULO III Seguridad de las aplicaciones

3.1. Introducción

Contrariamente a lo que se suele pensar, cuando se habla de “Seguridad en Aplicaciones”, no se dialoga de defectos o vulnerabilidades en sistemas operativos o servidores HTTP, no se discute de problemas que puedan ser resueltos instalando el último service pack del software de moda que se encuentra corriendo en los equipos, por el contrario, se platica de vulnerabilidades en el propio software, en las aplicaciones que se han desarrollado o pedido a alguien que desarrolle. Este último caso es el del Centro de Informatización de la Gestión de Entidades, específicamente para las aplicaciones pertenecientes al Sistema de Gestión de Entidades Cedrux, para el que es importantísimo protegerlas. El presente capítulo posee las especificaciones necesarias para proteger las aplicaciones.

3.2. SAML–Security Assertion Markup Language

El conjunto especificación SAML cubre lo siguiente:

- **Afirmaciones:** esquema XML y las definiciones para el intercambio de seguridad "afirmaciones" a través de los servicios.
- **Solicitud/protocolo de respuesta:** esquemas XML y las definiciones de la solicitud/modelo de respuesta de la transmisión de información de seguridad.
- **Fijaciones:** llamamientos específicos de SOAP a través de HTTP para la transmisión de solicitudes de SAML y respuestas.
- **Perfiles:** para la implantación y la extracción de las afirmaciones SAML.
- **Consideraciones de seguridad:** durante el uso de SAML.
- **Conformidad con las directrices.**
- **Un conjunto de pruebas:** los casos de uso y los requisitos.

SAML hace uso del concepto de las afirmaciones para intercambiar información de seguridad a través de sistemas dispares y servicios. En un ciclo típico de SAML, la parte que confía, que necesita autenticar una solicitud de cliente específico, envía una solicitud basada en SAML SOAP a la autoridad de emisión. La

autoridad responde con una afirmación SAML, que afirma la parte que confía con la seguridad de la información solicitada.

Independientemente de su tipo, todas las aserciones SAML incluyen la siguiente información:

- La autoridad emisora y su marca de tiempo.
- La afirmación de ID.
- Asunto (Nombre + dominio de seguridad).
- Términos y condiciones contra las que la afirmación es válida en curso (período de validez la afirmación, la restricción de la audiencia, la restricción de destino, y así sucesivamente).
- Adicionales "consejos" (información sobre cómo se hizo la afirmación, y así sucesivamente).

Ventajas

Antes	Después
No hay un estándar implantado.	Se regirá por el estándar SAML.
No hay control de las sesiones.	Servidor de sesiones independiente.
No existe el cifrado de los datos.	Se obtiene una firma cifrada de la autenticación.
Solo se maneja la solicitud por servicio web.	Se utilizará además la solicitud por HTTP Post.
No hay integración en las aplicaciones.	Se obtendrá la integración.
El servidor de aplicaciones estaba unido al de sesiones.	Se tratarán de forma independientes.

Uso en ERP–Cuba

SAMl se va a utilizar para la autenticación de forma que trabaje como un SSO [20], donde todo funciona con el llamado certificado, que no es más que los permisos de acceso que tendrán los usuarios para acceder a un lugar u otro. Esto se realizará utilizando estos dos proveedores:

[20] *Single Sign-On: los usuarios se autentican una única vez, contra el sistema y después este sistema se encarga de forma transparente de las autenticaciones subsiguientes en su lugar, según se van produciendo los accesos correspondientes.* (51)

- **Proveedor de servicios:**
  - Verificar que tenga el certificado.
  - Si no tiene el certificado redireccionar hacia el proveedor de identidades especificando la url de retorno.
  - Si tiene el certificado chequearlo.
    - Guardar en sesión el certificado.
    - Redireccionar hacia el recurso solicitado.
  - Si se mandaron los datos de autenticación el certificado retorna sino mostrar la ventana de autenticación.
  - Si cancela la autenticación mostrar un mensaje de acceso denegado.
- **Proveedor de identidad:**
  - Chequear el certificado.
  - Obtener el protocolo de comunicación.
  - Conformar la url de retorno.
  - Redireccionar hacia el proveedor de servicios especificando la url de retorno.

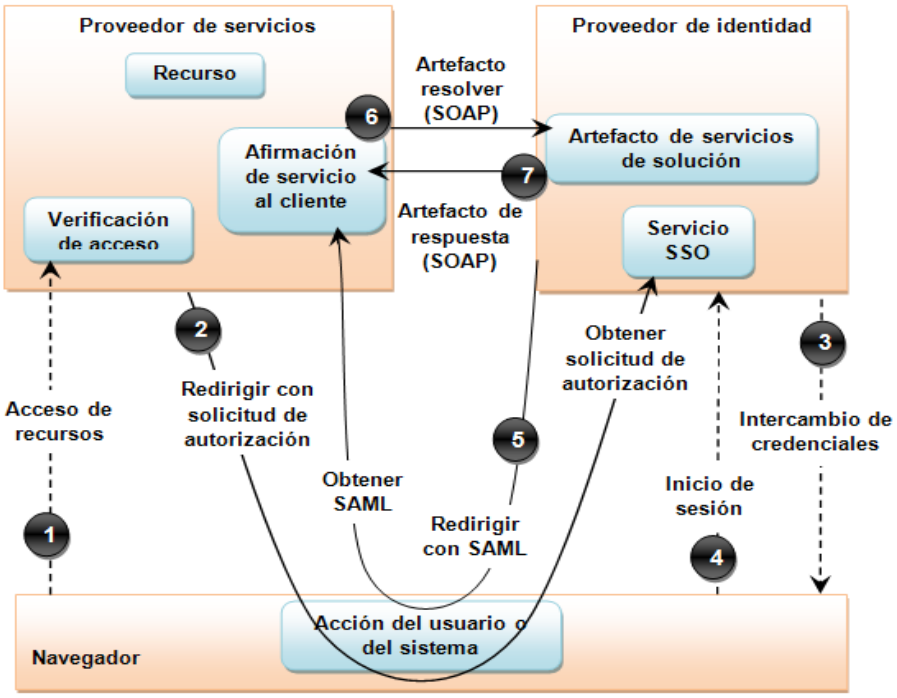


Figura 10: Proveedor de servicios iniciado con SSO.

3.3. RBAC-Role Based Access Control

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, 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.

Hierarchical (jerárquica) RBAC agrega los requerimientos para el soporte de jerarquías de roles. Una jerarquía es un orden parcial en el conjunto de roles que define una relación de herencia entre estos, donde los roles ascendentes adquieren los permios de sus descendentes y los roles descendentes adquieren los usuarios pertenecientes a sus ascendentes.

Las relaciones de Static Separation of Duty (SSD) en español, separación de deberes estáticos son utilizadas para hacer cumplir políticas de conflicto de intereses. Un conflicto de interés en un sistema basado en roles puede darse cuando un usuario obtiene autorización para permisos asociados con roles conflictivos, por ejemplo un usuario que tenga asignado un rol que le permita gestionar una compra y un rol que le permita aprobar dicha gestión.

Es deseable poder especificar la política de conflicto de intereses de manera centralizada y luego imponerla de manera uniforme en el sistema. Una manera de implementar dicha funcionalidad es haciendo cumplir restricciones en las asignaciones usuario-rol. Las relaciones de Dinamic Separation of Duty (DSD) en español, separación de deberes dinámicos, al igual que las relaciones SSD, son utilizadas para limitar los permisos que un usuario puede ejercer simultáneamente. Sin embargo, estas difieren de las relaciones SSD en el contexto en que son impuestas.

DSD limita la disponibilidad de permisos imponiendo restricciones en los roles que pueden ser activados en las sesiones de los usuarios, en contraposición con SSD que impone las restricciones en las asignaciones usuario-rol. 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.

Sesión: 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 un 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 está dividido en dos partes. La primera es un conjunto de identificadores validos de sesión llamado - Session - introducido en la sección.

Separación de deberes estáticos. Esta política define que un usuario al cual es miembro de un rol, este no puede ser miembro de un segundo rol. Con el uso de las restricciones en los roles, y el rol (padre) tiene una relación SSD con un recurso, esto implica que los hijos de este también presentan una relación SSD con este recurso.

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. Con el uso de las restricciones en los roles, y el rol (padre) tiene una relación DSD con un recurso, esto implica que los hijos de este también presentan una relación DSD con este recurso.

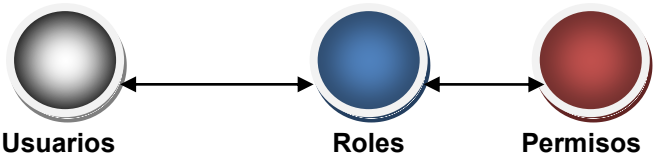
3.3.1 Niveles

RBAC presenta varios niveles los cuales fueron abordados en el capítulo I de forma general, a continuación una mejor explicación de los mismos.

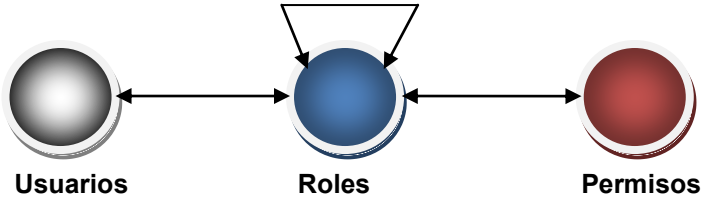
Nivel 0 "Básico": Este nivel es lo fundamental de RBAC, donde están los usuarios, roles y los permisos, con sus relaciones. Donde los usuarios adquieren permisos a través de los roles.

Relación de:

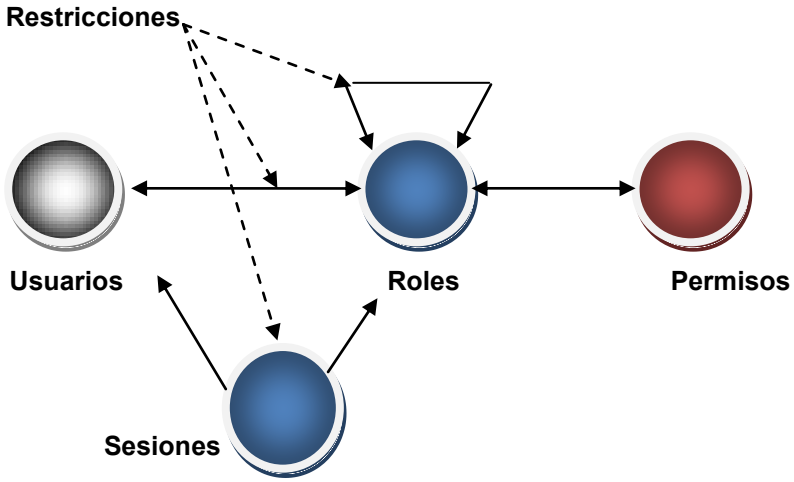
- Usuario a Rol: mucho a mucho (viceversa).
- Rol a Permisos: mucho a mucho (viceversa).



Nivel 1 "Jerarquía": En este está la jerarquía entre los roles, es decir que un rol puede tener dentro del subconjunto de roles y están presentes las del nivel 0.



Nivel 2 "Restricciones": En este está presente las características de los niveles 0 y 1, y en él se establecen restricciones a las relaciones y a la jerarquía.



Esta representación seria Restricciones-SOD Dinámico, para el caso de Restricciones-SOD Estático no aparecen las secciones.

Nivel 3 "Simétrico": Quedaría con las características de los demás niveles, además de la restricción a la relación entre los roles y los permisos. Quedaría estructurado el estándar RBAC como se muestra en el Anexo 6**:**

Estructura del estándar RBAC.

Teniendo en cuenta a que un usuario puede tener diferentes roles, pero solo puede tener un rol en una entidad, se necesita incluir una estructura que es la entidad, entonces como una segunda versión del tema de autorización se tiene el siguiente esquema. La entidad está relacionada con los roles, donde ella establece una relación de mucho a mucho con el rol.

3.3.2 Solución propuesta

La solución propuesta para el proceso de autorización es la que se muestra a continuación:

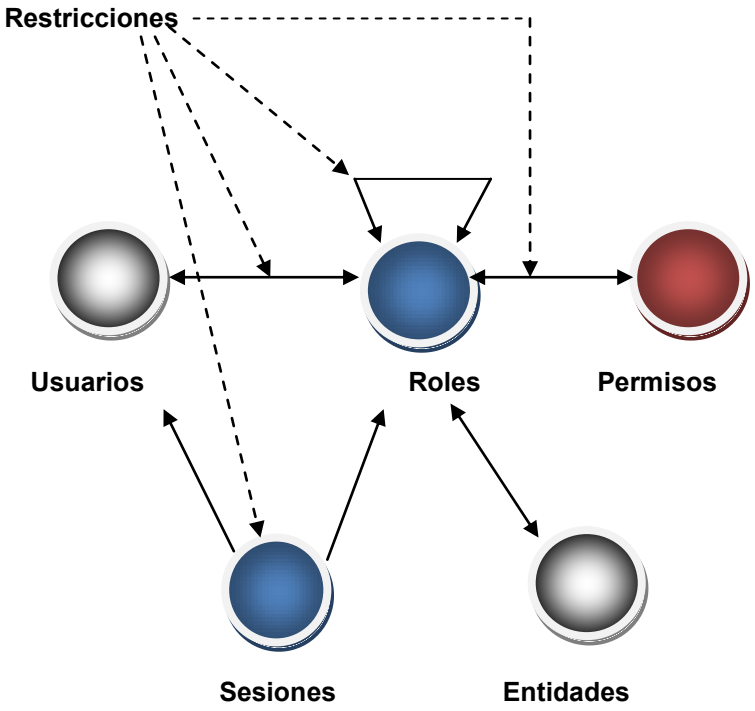


Figura 14: Solución propuesta para el proceso de autorización.

3.4. PKI-Public Key Infrastructure

PKI por sus siglas en inglés, en español significa, infraestructura de clave pública, es la forma común de referirse a un sistema complejo, necesario para la gestión de certificados digitales y aplicaciones de la firma digital. (52)

Una PKI bien construida debe proporcionar:

- Autenticidad. La firma digital tendrá la misma validez que la manuscrita. (52)
- Confidencialidad de la información transmitida entre las partes. (52)
- Integridad. Debe asegurarse la capacidad de detectar si un documento firmado ha sido manipulado. (52)
- No repudio de un documento firmado digitalmente. (52)

Los sistemas:

- **Criptografía asimétrica.** Los algoritmos asimétricos basan su funcionamiento en un par de claves (matemáticas dependientes) para cada usuario, con la característica de que la información cifrada con una clave, sólo puede descifrarse con la otra del mismo par. (52)
- **Firma digital.** Es un paquete de información de tamaño fijo, dependiente del documento original y sólo puede generarse por el poseedor de la clave privada. (52)
- **Certificados digitales.** Vincula la clave pública de un usuario con su identidad. (52)
- **Autoridad certificadora.** Es la entidad que asegura la identidad de los usuarios de los certificados digitales. Posee su propio par de claves y firma digitalmente los certificados con su clave privada. (52)
- **Autoridad de registro.** En toda PKI deben establecerse los mecanismos para que los usuarios soliciten su propio certificado, de tal forma que se asegure la identidad de dicho usuario. A este procedimiento se le denomina "Proceso de Registro" y se realiza a través de la denominada "Autoridad de Registro".(52)

Deben englobarse en un único sistema, este es la PKI, cuyos elementos se pueden ilustrar en la siguiente figura.

Figura 11: Componentes de una PKI. (52)

Existen multitud de componentes adicionales, y cada elemento es un sistema complejo en sí mismo. Los componentes básicos pueden resumirse en:

- **La Autoridad de certificación.** La pieza central del "puzle" y la que proporciona la base de confianza en la PKI. Constituido por elementos hardware, software y evidentemente, humanos. (52)
- **Publicación de certificados.** El repositorio de certificados permite a los usuarios operar entre ellos (por ejemplo para la validación de una firma digital), y es un requisito legal que cuente con una total disponibilidad de acceso. (52)
- **Soporte de la clave privada.** La elección de un buen soporte para que los usuarios custodien su clave privada es un punto esencial y complejo en sí mismo (p.e. si la clave está en una SmartCard, es

necesario diseñar el sistema de gestión de SmartCards que permita la emisión y distribución de las tarjetas a los usuarios). (52)

- **Aplicaciones "PKI-Enabled**". Se denomina así a las aplicaciones de software capaces de operar con certificados digitales. Estas aplicaciones son las que dan el valor real de la PKI de cara al usuario. (52)
- **Políticas de certificación.** Deben diseñarse una serie de políticas, o procedimientos operativos, que rigen el funcionamiento de la PKI y establecen los compromisos entre la Autoridad Certificadora y los Usuarios Finales. Estos documentos tendrán un carácter tanto técnico como legal. (52)

El Proceso de construcción de una PKI deberá siempre partir de la definición de las políticas operativas y contemplar como requerimiento esencial el asegurar la calidad y seguridad de las operaciones, que los usuarios finales realizan con sus claves privadas (ejemplo la firma digital de documentos). (52)

En el proyecto ERP-Cuba no se tiene acceso a ninguna de las existentes y no se permite iniciar el desarrollo de una propia; debido a ello solo se ha abordado el tema globalmente.

3.5. Encriptación

3.5.1 RSA

Es un sistema criptográfico de clave pública desarrollado en 1977. En la actualidad, RSA es el primer y más utilizado algoritmo de este tipo y es válido tanto para cifrar como para firmar digitalmente. La seguridad de este algoritmo radica en el problema de la factorización de números enteros. Los mensajes enviados se representan mediante números, y el funcionamiento se basa en el producto conocido de dos números primos grandes elegidos al azar y mantenidos en secreto. Actualmente estos primos son del orden de 10200, y se prevé que su tamaño aumente con el aumento de la capacidad de cálculo de los ordenadores. (53)

Como en todo sistema de clave pública, cada usuario posee dos claves de cifrado: una pública y otra privada. Cuando se quiere enviar un mensaje, el emisor busca la clave pública del receptor, cifra su mensaje con esa clave, y una vez que el mensaje cifrado llega al receptor, este se ocupa de descifrarlo usando su clave privada. (53)

En el proyecto ERP-Cuba se cuenta con un sistema criptográfico de este tipo, para cifrar datos como: las contraseñas de los usuarios de base de datos en el fichero de configuración de las conexiones y para encriptar otros datos que ponen en riesgo la seguridad de la información.

3.5.2 MD5

Hoy en día la mayoría de las páginas web utilizan bases de datos para poder desarrollar portales dinámicos y así hacerlos más atractivos a la vez que útiles. Pero esta información que se guarda en la base de datos tiene que tener algún tipo de protección. Es por ello que algunos campos se guardan encriptados, principalmente cuando una página requiere el nombre de usuario y contraseña, esta última se encripta y se guarda en la BD. (54)

En PHP se utiliza la función MD5 (abreviatura de Message-Digest Algorithm 5, Algoritmo de Resumen del Mensaje 5) que es una función hash irreversible (de un sólo sentido), es decir, encripta el password tecleado por el usuario y es imposible que partiendo desde la cadena encriptada se vuelva a la contraseña origen. Por esto mismo no hay problema de que alguien pueda acceder al campo encriptado de la base de datos. Como en la BD se guarda la contraseña encriptada, cuando un usuario quiere acceder, habrá que realizar una comparación entre el password que introduce encriptado en MD5, y lo que está en la base de datos, (que es la contraseña encriptada en MD5), si coincide se le permite el acceso, si no, se rechaza. (54)

MD5 se utiliza también para que cuando un usuario olvida su password, si quiere recuperar la contraseña se le pide que introduzca por ejemplo el correo, y se le envía un mail con una url tal que si entra en ella genere una nueva contraseña que se le indica al usuario y se reescribe en md5 en la base de datos (borrando la anterior contraseña). (54)

Hay que tener en cuenta que esto no es 100% seguro, puesto que la contraseña se encripta en el servidor, entonces al enviar la contraseña desde el cliente al servidor podría ser interceptada. Para que se tenga una idea, el algoritmo MD5 convierte el mensaje en un bloque múltiplo de 512 bits, (si hace falta añadirá bits por el final). Luego coge el primer bloque de 512 bits del mensaje y realiza diversas operaciones lógicas con los 128 bits de cuatro vectores iniciales ABCD de 32 bits cada uno. (Dichos vectores tendrán el valor inicial que se quiera). Como resultado obtiene una salida de 128 bits que se convierte en el nuevo conjunto de los 4 vectores ABCD. Se repite el algoritmo hasta procesar el último bloque del mensaje. Al terminar, el algoritmo devuelve los últimos 128 bits de estas operaciones.

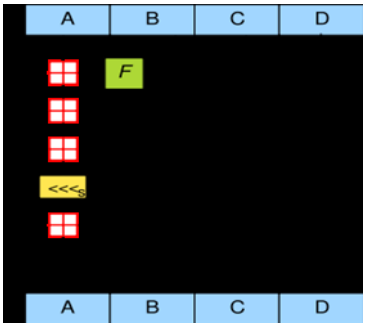


Figura 12. Una operación en MD5. (54)

Como se muestra en la figura 23, MD5 consta de 64 de estas operaciones, agrupadas en cuatro rondas de 16 operaciones. F es una función no lineal, una función se utiliza en cada ronda. **Mi** denota un bloque de 32-bit de la entrada de mensajes, y Ki denota una constante de 32-bit, diferente para cada operación.

En el proyecto ERP-Cuba se utiliza para encriptar datos, las claves de los usuarios en la base de datos y otros datos de interés.

3.6. Acaxia

3.6.1 Funcionalidades del sistema

El subsistema de seguridad está dividido en 4 módulos:

- Configurar nomencladores.
- Configurar sistemas.
- Configurar servidores.
- Configurar usuarios.

El módulo configurar nomencladores permite el manejo de los dominios, base de datos, gestores de base de datos, esquemas, idiomas, temas, escritorios, expresiones y claves, para el manejo posterior de los servidores, sistemas y usuarios.

El módulo configurar servidores permite manejar los datos de los servidores, gestores de base de datos, base de datos y esquemas de base de datos.

El módulo configurar sistemas posee las funcionalidades correspondientes al manejo de los sistemas, las funcionalidades, acciones, servicios que brinda ó consume sus funciones y parámetros de las mismas.

El módulo configurar usuarios permite la gestión de los usuarios, roles, perfiles de usuario y los campos del perfil de usuario.

Una vez que el sistema ha sido instalado usted tendrá la posibilidad de escoger el ambiente que le resulte más cómodo para trabajar. Cedrux brinda esta facilidad, en el Anexo 7**: Ambiente inicial de Cedrux.** se muestra el primer ambiente con una estructura de árbol en la parte izquierda, la que aparecerá inmediatamente al hacer clic en **(1**.

Haga clic sobre estructura y composición **(1** y aparecerán los módulos, y al hacer clic sobre estos aparecerá en el lado derecho **(2** las funcionalidades que puede ejecutar dentro del mismo (ver).

El

Anexo 9: Segundo ambiente de trabajo.muestra el segundo ambiente de trabajo. Haga clic en el botón Inicio y se desplegarán un menú en el cual debe colocar el puntero en Sub-sistemas. Una vez que se hayan desplegado todos los subsistemas escogerá: seguridad.

El tercer ambiente de trabajo. Es similar al anterior con la diferencia de que a través de éste usted podrá llegar directamente hasta la funcionalidad que desea a través del menú inicio (ver Anexo 10**: Sub-sistema: seguridad.**).

3.6.2 Auditoría del sistema

Las trazas permiten en el desarrollo de software crear un mecanismo de registro oficial de eventos durante un período de tiempo en particular, además de registrar datos o información sobre quién, qué, cuándo, dónde y por qué un evento ocurre para un dispositivo en particular o aplicación. Todo esto permite monitorear las actividades de la aplicación o dispositivo, donde se puede obtener una buena oportunidad para determinar eventos y tomar la acción necesaria para corregir el problema o iniciar una investigación en caso de un incidente de seguridad.

La auditoría de las aplicaciones se realiza mediante la utilización del componente traza (ver

Anexo **11: Componente gestionar traza.**), implementado por el equipo de desarrollo de la línea de arquitectura. Este componente permite el registro de los eventos que ocurren en el sistema permitiendo de esta forma poder corregir problemas que se presenten, el mismo puede ser instalado en un servidor independiente para garantizar la confidencialidad de los datos. Se podrían controlar eventos como:

- **Inicio de una acción:** Se dispara un evento al comienzo de una acción.
- **Terminación de una acción:** Se dispara un evento al finalizar una acción.
- **Error en una acción:** Se dispara un evento cuando en una acción ocurre un error.

- **Autenticar usuario:** Se dispara un evento por cada URL visitada.
- **IoC Externo:** Se dispara un evento cuando ocurre integración entre diferentes subsistemas.
- **IoC Interno:** Se dispara un evento cuando ocurre integración entre diferentes componentes de un subsistema.
- **Excepciones:** Se dispara un evento cuando ocurre una excepción en el sistema.
- **Rendimiento:** Es el tiempo de ejecución de una acción.
- **Error IoC Externo:** Se dispara un evento cuando ocurre un error en la integración entre diferentes subsistemas.
- **Error IoC Interno:** Se dispara un evento cuando ocurre un error en la integración entre diferentes componentes de un subsistema.

3.7. Listas de chequeo

Entidad	Parámetros	
Entidad 1	**Sistema operativo.**	Utilización del SO Linux en los servidores.
	Tecnologías en los servidores.	Servidor web Apache.
		Servidor de base de datos Postgres 8.3.
		Navegador Web Mozilla Firefox 2.0.17 ó superior.
		No tener instaladas extensiones de Mozilla Firefox.
		Lenguaje de programación PHP 5.2.4.
	Configuración del servidor web.	Actualización de los parches de seguridad.
		Opciones de explorar directorio, desactivadas.
		Restringir acceso por direcciones ip.
		Apache debe funcionar bajo su propia cuenta y grupo de usuario.
		Módulos innecesarios deshabilitados: mod_imap, mod_include, mod_info, mod_status, mod_cgi, mod_autoindex.
		Los includes del lado servidor desactivados.
		Ejecución de CGI desactivada.

		Desactivar error, notice, alert, warn y demás opciones que brinden información sensible.
		No permitir que apache siga enlaces simbólicos.Desactivar todas las opciones.
		Desactivar la ayuda para los archivos .htaccess
		Disminuye el valor máximo de tiempo de espera. Timeout 60
		Limitar el tamaño máximo de peticiones. LimitRequestBody 2097152.
		Ejecuta Apache en un entorno Chroot.
	Configuración de PHP.	Desactivar el acceso a ficheros remotos.
		Register globals desactivado.
		Acceso restringido a los ficheros puede acceder PHP.
		Modo seguro activado.
		Límites controlados: max_execution_time = 30 ; max_input_time = 60 ; memory_limit = 16M ; upload_max_filesize = 2M ; post_max_size = 8M ;
		Control de acceso a ficheros mediante Apache.
		Carga dinámica de módulos desactivada.
		Evitar el acceso a la Shell.
		Deshabilitar información sobre php en las cabeceras del servidor.
		Funciones de integración de php/Apache desactivadas.

		Creación de un nuevo directorio para almacenamiento de las sesiones de usuario.
		Al directorio donde se almacenan las sesiones solo puede tener acceso el usuario Apache.
	Principios básicos de seguridad en la BD.	El administrador de la BD y el administrador del sistema operativo (ASO) no pueden ser la misma persona.
		Los privilegios del usuario operador y los del ABD (administrador de BD) no deben ser los mismos.
		El número mínimo de caracteres de las contraseñas de los usuarios será de 7 caracteres.
		Los usuarios no deben compartir sus cuentas ni sus contraseñas.
		Habilitar sólo los servicios y puertos requeridos.
		La cuenta de root y de administrador sólo la debe tener el personal requerido.
		Restringir las conexiones remotas usando el fichero pg_hba.conf.
		El número mínimo de caracteres de las contraseñas de los usuarios será de 7 caracteres.
		La contraseña del usuario no debe ser igual al nombre del usuario.
		La contraseña debe tener fortaleza requerida con la utilización de letras, números y caracteres especiales.
		La contraseña debe cambiarse como máximo cada tres meses.
		Uso de un firewall que mantenga protegido al servidor.
		El firewall configurado para que sólo tenga acceso al

		servidor la red o subred requerida.
		Habilitar en el postgresql.conf que postgres escuche todas las peticiones, tanto locales como remotas. listen_addresses = 'ALL'
		Activar la encriptación de contraseñas de la base de datos a la hora de conectarse en el postgresql.conf. password_encryption = on
		Uso de SSL activado en el postgresql.conf. ssl = on
		No se debe permitir en la configuración de las conexiones en el fichero pg_hba.conf: DATABASE ALL USER ALL METHOD TRUST
		Crear un usuario de base de datos por cada sistema.
		Para cada usuario de base de datos especificar en el pg_hba.confç: BD, rango de ip, y método de encriptación MD5.
	Mecanismos de control de integridad.	Uso del auditor de código fuente (CRC) para el control de la integridad de las aplicaciones.
	Ofuscador de código.	Ofuscación de código.
	CRC.	Cálculo de CRC.
	Seguridad de las aplicaciones.	Deben crearse dominios que contengan exactamente las estructuras a las que tiene acceso uno o varios usuarios.
		Deben registrarse la estructura de los sistemas hasta el nivel de acción para garantizar la seguridad hasta la base.

		Deben crearse roles que contengan exactamente los sistemas, funcionalidades y acciones a las cuales tiene acceso uno o varios usuarios.
		La configuración de las claves de usuarios debe contener signos, letras y números, contar con un número mayor de 7 caracteres y caducar en un tiempo de 60 días.
	Seguridad informática.	Existencia de un plan de contingencia.
		Existencia de un plan de seguridad informática.
		Control de acceso a los locales y las PCs.
		Existencia de una política de salva de seguridad.

3.8. Conclusiones

Durante el transcurso del capítulo se argumentaron los fundamentales aspectos que se llevan a cabo para brindar seguridad a las aplicaciones, comenzando por los procesos de autenticación, autorización y encriptación. También se abordó sobre la infraestructura de clave pública, el sistema Acaxia con sus ambientes de trabajo y las listas de chequeo que se deben tener en cuenta para tener un control y seguimiento tanto en el entorno de desarrollo, como en el entorno de despliegue. Dando fin a los aspectos que conforman la "Vista de arquitectura de seguridad del proyecto ERP-Cuba".

3.9. Validación

En el Anexo 12**: Avales** está la constancia de que el documento "Vista de arquitectura de seguridad del proyecto ERP-Cuba", ha sido reutilizado en varios proyectos de la universidad en los cuales ha tenido una gran aceptación, ya que estudiantes y profesionales se han logrado familiarizar con el mismo en un corto período de tiempo.

Conclusiones generales

Al realizar un estudio del estado del arte sobre la seguridad de los sistemas de gestión tanto dentro como fuera del país, arrojó como resultado que no existía una guía para estandarizar el proceso de seguridad en el proyecto ERP-

Cuba, trayendo consigo que el Sistema de Gestión de Entidades Cedrux cuente con un sin número de vulnerabilidades que pueden ser utilizadas para perpetrar ataques que puedan ocasionar pérdidas irreparables para la entidad o el país de forma general.

Para disminuir los riesgos de seguridad existentes surge como parte del departamento de tecnología la idea de crear una vista de seguridad que establezca cómo desarrollar sistemas de gestión de forma segura. Como parte de la investigación se realizó un estudio de las soluciones existentes en el mundo para dar solución a la problemática planteada, se obtuvo como resultado que no existe en Cuba ninguna que pueda ser utilizada por el proyecto para ejecutar un proceso de desarrollo de forma segura. Se realizó un estudio de los principales temas relacionados a la seguridad en software de gestión. Se hizo un estudio de las principales herramientas y tecnologías utilizadas en el centro para identificar las debilidades y cómo prevenirlas.

Se descubrió que los desarrolladores existentes en el proyecto no cuentan con la cultura de programación segura, aspecto que trae consigo que existan huecos de seguridad en el código implementado, por esta razón se realizó una propuesta sobre los aspectos que se deben tener en cuenta para realizar una programación de forma segura.
La existencia de ataques que pueden causar daños irreparables como resultado de una validación ineficiente a todos los niveles permite que se puedan explotar vulnerabilidades, para evitarlo se conformó un estándar de validación en el cual se especifica qué se debe tener en cuenta en las validaciones para las capas de presentación, negocio y acceso a datos.
Uno de los eslabones fundamentales de la seguridad en aplicaciones de gestión es la protección de la información por medios de permisos o restricciones asignados o no a los usuarios. La mayoría de los sistemas existentes en el mundo proponen soluciones especificas que resuelven escenarios relacionados al negocio de sus aplicaciones, por esta razón no cubren todos los aspectos necesarios para lograr la protección de la información. Para dar solución a esta problemática se decidió utilizar el estándar SAML que recoge todos los aspectos a tener en cuenta para el proceso de autenticación de usuario y el estándar RBAC para el proceso de autorización, permitiendo obtener soluciones integrales con altos niveles de integración y seguridad.

La ¨Vista de arquitectura de seguridad del proyecto ERP-Cuba¨, brindará a los usuarios y desarrolladores del Sistema de Gestión de Entidades Cedrux, los aspectos necesarios para protegerlo, entenderlo, y hacerlo más fuerte en cuanto a seguridad se refiere.

Recomendaciones

Se recomienda para la validación y utilización futura de este trabajo:

- ✓ Trabajar en nuevas versiones del documento para fortalecer la estandarización de la seguridad brindada por este.

Referencias bibliográficas

1. **Rivero Pino, Noel Jesús y Massó Machandi, Aquiles.** *Sistema de Gestión Integral de Seguridad.* Ciudad de La Habana : s.n., 2009.

2. scribd. [En línea] [Citado el: 5 de diciembre de 2009.] http://www.scribd.com/doc/2023909/manual-de-politicas-y-normas-de-seguridad-informatica.

3. **Borghello, Lic. Cristian.** *SEGU-INFO Seguridad de la Información.* [En línea] http://www.segu-info.com.ar/tesis/.

4. eva.uci.cu. . [En línea] [Citado el: 10 de diciembre de 2009.] http://eva.uci.cu/mod/resource/view.php?inpopup=true&id=7875.

5. definicion.de. [En línea] [Citado el: 10 de diciembre de 2009.] http://definicion.de/seguridad-informatica/.

6. SeguridadInformatica. [En línea] [Citado el: 15 de diciembre de 2009.] http://www.mitecnologico.com/Main/SeguridadInformatica.

7. eva.uci.cu. [En línea] [Citado el: 4 de febrero de 2010.] http://eva.uci.cu/mod/resource/view.php?inpopup=true&id=7890.

8. kioskea.net. [En línea] [Citado el: 12 de diciembre de 2009.] http://es.kioskea.net/contents/secu/secuintro.php3.

9. segu-Info. [En línea] [Citado el: 12 de diciembre de 2009.] http://www.segu-info.com.ar/logica/seguridadlogica.htm.

10. ariesgo.pdf. [En línea] [Citado el: 12 de febrero de 2010.] http://seguridad.internet2.ulsa.mx/congresos/2003/esime/ariesgo.pdf.

11. segu-info.com.ar. [En línea] [Citado el: 8 de enero de 2010.] http://www.segu-info.com.ar/logica/seguridadlogica.htm.

12. librosweb.es. [En línea] [Citado el: 20 de enero de 2010.] http://librosweb.es/javascript/capitulo1.html.

13. wikipedia, la enciclopedia libre. [En línea] [Citado el: 20 de enero de 2009.] http://es.wikipedia.org/wiki/PHP.

14. ciberaula. [En línea] [Citado el: 20 de enero de 2010.] http://www.ciberaula.com/curso/lamp/que_es/.

15. Ventajas de ExtJs. [En línea] [Citado el: 21 de enero de 2010.] http://www.nohaylimites.com/?p=162.

16. Glossary. [En línea] [Citado el: 22 de enero de 2010.] http://www.joomla-gnu.com/glosario.html.

17. masadelante. [En línea] [Citado el: 21 de enero de 2010.] http://www.masadelante.com/faqs/plug-in.

18. Rich_Internet_Applications. [En línea] [Citado el: 21 de enero de 2010.] http://es.wikipedia.org/wiki/Rich_Internet_Applications.

19. Renderización. [En línea] [Citado el: 22 de enero de 2010.] http://es.wikipedia.org/wiki/Renderizaci%C3%B3n.

20. La Zona linux. [En línea] [Citado el: 22 de enero de 2010.] http://lazonalinux.com.ar/?topic=639.

21. **Bauta Camejo, Rene Rodrigo.** *Desarrollo de una herramienta generadora de ficheros de mapeo, para la persistencia de objetos en esquemas relacionales basada en Doctrine.* Ciudad de La Habana : s.n., 2009.

22. ¿Que es Framework ? [En línea] [Citado el: 10 de febrero de 2010.] http://buenmaster.com/?a=554.

23. Zend Framework. [En línea] [Citado el: 25 de enero de 2010.] http://es.wikipedia.org/wiki/Zend_Framework.

24. ZOOsocial-Alternativa Open source. [En línea] [Citado el: 1 de febrero de 2010.] http://www.zoosocial.com.ar/?topic=1226.

25. Lucene. [En línea] [Citado el: 1 de febrero de 2010.] http://solobuscadores.blogspot.es/general.phtml?title~=lucene.

26. **Baryolo, Oiner Gómez.** *fundamentación.* Ciudad de La Habana : s.n.

27. ciberaula. [En línea] [Citado el: 3 de febrero de 2010.] http://www.ciberaula.com/curso/lamp/que_es/.

28. postgre. [En línea] [Citado el: 10 de febrero de 2010.] http://www.guia-ubuntu.org/index.php?title=PostgreSQL.

29. ventajas postgre. [En línea] [Citado el: 5 de febrero de 2010.] http://soporte.tiendalinux.com/portal/Portfolio/postgresql_ventajas_html.

30. Subversion. [En línea] [Citado el: 8 de febrero de 2010.] http://es.wikipedia.org/wiki/Subversion.

31. Mozilla_Firefox - EcuRed. [En línea] [Citado el: 16 de febrero de 2010.] http://www.ecured.cu/index.php/Mozilla_Firefox.

32. El problema de la interoperabilidad I parte: Las definiciones. [En línea] [Citado el: 16 de febrero de 2010.] http://fernando-acero.livejournal.com/61831.html.

33. GUIA-ESTANDARES-DE-SCE. [En línea] [Citado el: 16 de febrero de 2010.] http://www.scribd.com/doc/18112674/GUIA-ESTANDARES-DE-SCE.

34. **Nadalin, Tony, y otros.** *sstc-saml-exec-overview-2.0-cd-01.*

35. Federacion-de-Identidades-SAML. [En línea] [Citado el: 17 de febrero de 2010.] http://www.scribd.com/doc/13364049/Federacion-de-Identidades-SAML.

36. **Baryolo, Oiner Gómez.** *Role Based Access Control.* Ciudad de La Habana : s.n.

37. **Cruz, Marcos.** *Control de acceso basado en roles - RBAC.* Cidad de La Habana : s.n., 2009.

38. Sitio Web PDVsA. [En línea] [Citado el: 17 de febrero de 2010.] http://www.pdvsa.com/index.php?tpl=interface.sp/design/readmenuprinc.tpl.html&newsid_temas=11.

39. Single sing- on. [En línea] [Citado el: 18 de febrero de 2010.] https://www.ccn-cert.cni.es/publico/seriesCCN-STIC/series/400-Guias_Generales/401-Glosario_y_abreviaturas/401/es/s/sso.htm.

40. consideraciones para implementar una arquitectura Single sign-on. [En línea] [Citado el: 19 de febrero de 2010.] http://www.criptored.upm.es/guiateoria/gt_m142j.htm.

41. Hacia una solución IAM paso a paso. [En línea] [Citado el: 19 de febrero de 2010.] http://www.datati.es/hacia-una-solucion-iam-paso-a-paso/.

42. Bull. [En línea] [Citado el: 19 de febrero de 2010.] http://www.bull.com/es/news/iam.htm.

43. Passlogix-SSO. [En línea] [Citado el: 9 de marzo de 2010.] http://www.passlogix.com/products/v-GO_universalauthenticationmanager/.

44. passloxic. [En línea] [Citado el: 19 de marzo de 2010.] http://www.passlogix.com/products/v-GO_universalauthenticationmanager/.

45. passlogix. [En línea] [Citado el: 19 de marzo de 2010.] http://www.passlogix.com/products/v-GO_universalauthenticationmanager/.

46. Eval. [En línea] [Citado el: 10 de febrero de 2010.] http://es.wikipedia.org/wiki/Eval.

47. Pos y Get. [En línea] [Citado el: 6 de marzo de 2010.] http://codigoaldescubierto.wordpress.com/2007/12/20/post-y-get/.

48. SISTEMAS OPERATIVOS. [En línea] [Citado el: 8 de marzo de 2010.] http://www.scribd.com/doc/30609591/SISTEMAS-OPERATIVOS.

49. Dime cómo programas y te diré quién eres. [En línea] http://www.informatizate.net/articulos/dime_como_programas_y_te_dire_quien_eres_23082004.html.

50. Casuistica/Spanish. [En línea] [Citado el: 5 de marzo de 2010.] http://www.babylon.com/definition/Casu%C3%ADstica/Spanish.

51. IAM & SSO. [En línea] [Citado el: 5 de marzo de 2010.] http://www.bull.com/es/security/folletos/IAM%20&%20SSO.pdf.

52. ¿Qué es una PKI? [En línea] [Citado el: 6 de marzo de 2010.] http://www.eurologic.es/soluciones/que-es-pki.htm.

53. SSH y los host conocidos en MAC OS X. [En línea] [Citado el: 1 de abril de 2010.] http://www.macdiario.com/ssh-y-los-host-conocidos-en-mac-os-x/.

54. Encriptación com MD5 en PHP. [En línea] [Citado el: 1 de abril de 2010.] http://php.ciberaula.com/articulo/encriptacion_md5_php/.

Glosario de términos

Análisis de riesgo: proceso por el cual se identifican las amenazas y vulnerabilidades de una organización Con el fin de generar controles que minimicen los efectos de los riesgos.

Apache: servidor web más utilizado mundialmente. Por defecto lo traen instalado en todas las distribuciones Linux. También existe para otras plataformas incluso Windows. Su funcionamiento básico es ejecutando un proceso padre y tantos procesos hijos como peticiones reciba para atender a cada cliente.

Archivo Log: ficheros de registro o bitácoras de sistemas, en los que se recoge o anota los pasos que dan (lo que hace un usuario, como transcurre una conexión, horarios de conexión, terminales o IP involucradas en el proceso).

Administración remota: forma de administrar los equipos informáticos o servicios de la Universidad de Oriente, a través de terminales o equipos remotos, físicamente separados de la institución.

Administración de perfil: es la forma de personalización de las aplicaciones de este dominio a nivel de cada usuario, se define como perfil los datos únicos de cada recurso dentro del sistema que define el comportamiento del mismo ante las entradas emitidas por este recurso y las salidas entregadas por el (los) subsistema(s), esto garantizaría un sin número de bondades tanto de usabilidad como de configurabilidad a la solución en cuestión.

Administración de conexiones: consiste en un grupo de procesos dedicados a la gestión de las conexiones a la base de datos de un sistema determinado ubicado en un servidor de bases de datos definido también como un parámetro configurable, así como el gestor en uso.

Afirmación: es una declaración de ciertos hechos (las declaraciones) sobre uno principal (sujeto).

Clientes: Constituyen las PC que usan los operarios o clientes de la solución en cada uno de los puestos de trabajo. Tiene como características que no necesitan alto procesamiento, y al ser una solución de software basada en la web, prácticamente cualquier ordenador con las mínimas prestaciones deben garantizan una explotación correcta de la solución de software.

Cliente web: Solicita información al servidor web. Se les suele llamar navegadores (ejemplo Mozilla).

Credenciales: Contienen información que le permite al usuario acceder a las diferentes aplicaciones (nombre de usuario, contraseña, entre otros).

Cuenta: Mecanismo de identificación de un usuario, llámese de otra manera, al método de acreditación o autenticación del usuario mediante procesos lógicos dentro de un sistema informático.

Dirección IP: número de cuatro dígitos separados por puntos (x.x.x.x) para identificar a un equipo en una red TCP/IP lo cual permite su localización.

DNS (Domain Name Server): máquina servidora de traducción de nombres.

DQL (Doctrine Query Language): es el lenguaje que utiliza Doctrine para ejecutar sus consultas.

Desastre o Contingencia: interrupción de la capacidad de acceso a información y procesamiento de la misma a través de computadoras necesarias para la operación normal de un negocio.

Encriptación: es el proceso mediante el cual cierta información o "texto plano" es cifrado de forma que el resultado sea ilegible a menos que se conozcan los datos necesarios para su interpretación.

Gateway: es un punto de acceso que relaciona dos o más puntos de un sistema.

HQL (Hibernate Query Language): es un lenguaje de interrogación. En el mundo relacional se dispone del SQL que nos permite obtener información haciendo preguntas basadas en las tablas y sus columnas. El equivalente en el mundo objetual es el HQL, que nos permite hacer preguntas basadas en los objetos y sus propiedades.

IEC (Comisión Electrotécnica Internacional): junto a la ISO, desarrolla estándares que son aceptados a nivel internacional.

ISO (Organización Internacional de Estándares): institución mundialmente reconocida y acreditada para normar en temas de estándares en una diversidad de áreas, aceptadas y legalmente reconocidas.

MVC (Model-View-Controller): es un paradigma utilizado en diversos desarrollos de software, a través de este "Framework" se logra una división de las diferentes partes que conforman una aplicación, siendo su principal razón de ser: manutención del código fuente.

Outsourcing: contrato por servicios a terceros, tipo de servicio prestado por personal ajeno a la institución.

Protocolo: lenguaje por el cual se comunican el cliente y servidor http.

Puerto: en un ordenador, es el lugar por donde entra información, sale información, o ambos. En internet se refiere a un número que se muestra en una URL, después de una coma justo después del nombre de dominio. Finalmente, el término puerto también se utiliza para referirse al acto de traducir un trozo de software de un tipo de sistema informático a otro, por ejemplo de un programa de Windows a uno en un Macintosh.

PDF de Portable Document Format (formato de documento portable): es el formato de archivos desarrollado por Adobe Systems y creado con los programas Adobe Acrobat Reader, Acrobat Capture, Adobe Distiller, Adobe Exchange, y el plugin Amber de Adobe Acrobat.

Redundancia: información replicada múltiples veces y actualizada en tiempo real con el fin de prevenir su pérdida por falla en el sistema o en los equipos.

Resolución de nombres: traduce direcciones IP a un nombre de dominio o viceversa (ya que para los usuarios es más fácil de recordar nombres).

Responsabilidad: en términos de seguridad, significa determinar qué individuo en la institución, es responsable directo de mantener seguros los activos de cómputo e información.

Rol: es el conjunto de actividades, funciones, ítems, que un usuario puede realizar dentro de una organización.

RSS: son las siglas de RDF Site Summary or Rich Site Summary, un formato XML para sindicar o compartir contenido en la web.

Single Sign-On: unifica el proceso de autenticación, administración de credenciales y perfiles de los usuarios.

Servidor: programa ejecutado en una máquina que responde a solicitudes de otros programas llamados clientes.

Servidor web: recibe solicitudes html de los clientes web (ejemplo Apache).

Socket: Elemento de programación que permite a dos máquinas comunicarse a través de una red, mediante el uso de la IP de origen, la IP de destino y el número de puerto. Se crea un socket cuando hay una conexión entre el cliente y el servidor.

Servicio: conjunto de aplicativos o programas informáticos, que apoyan la labor educativa, académica y administrativa, sobre los procesos diarios que demanden información o comunicación de la institución.

SGSI: Sistema de Gestión de Seguridad de la Información**.**

Soporte Técnico: (personal en Outsourcing), personal designado o encargado de velar por el correcto funcionamiento de las estaciones de trabajo, servidores, o equipo de oficina dentro de la institución.

Servicios de no-repudio: ofrecen una prueba al emisor de que la información fue entregada y una prueba al receptor del origen de la información recibida.

Servidor WEB: Constituyen las PC servidoras en las que se publica o instala la solución WEB de software, responsables de la gestión y procesamiento aritmético de los flujos de información generados por las interacciones de las peticiones de los terminales clientes de la capa de presentación. Tiene como características que necesitan recursos de hardware medios o altos, a partir del fuerte procesamiento que en ellos se ejecutan.

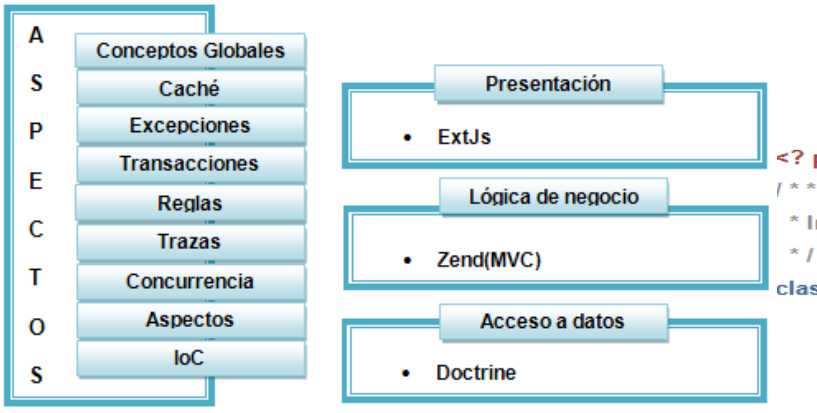
Servidor de bases de datos: Es un sistema bajo arquitectura cliente servidor que proporciona servicios de gestión, administración y protección de la información a través de conexiones de red, gobernadas por protocolos definidos y a los que acceden los usuarios de modo concurrente a través de aplicaciones clientes.

Servicios telemáticos: Constituyen la infraestructura tecnológica para la interconexión entre todos los elementos de hardware involucrados en la solución de software, responsable de la trasmisión de información entre los nodos, en un entorno web donde existe un intercambio continuo entre los tres nodos fundamentales de la arquitectura de despliegue, nodos clientes, nodo servidor y nodo de datos

Usuario: defínase a cualquier persona jurídica o natural, que utilice los servicios informáticos de la red institucional y tenga una especie de vinculación académica o laboral con la institución.

Anexos

Anexo 1: Aspectos de Sauxe.



Anexo 2: Nomenclatura de las llaves.

Anexo 3: Nomenclatura de sangría o indexado.

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

```
class Example {
    var $theInt = 1;
    fuction foo ( $a , $b) {
        switch ( $a ) {
            case 0 :
                $Other ->doFoo ();
            break;
            default :
                $Other->doBaz ();
        } // fin switch
    } // fin de la función foo
    fuction bar ( $v ) {
        for ( $i = 0 , $i < 10 , $i + + ) {
            $v-> add ( $i ) ;
        } // fin del for
    } // fin de la función bar
} // fin de la clase Example
?>
```

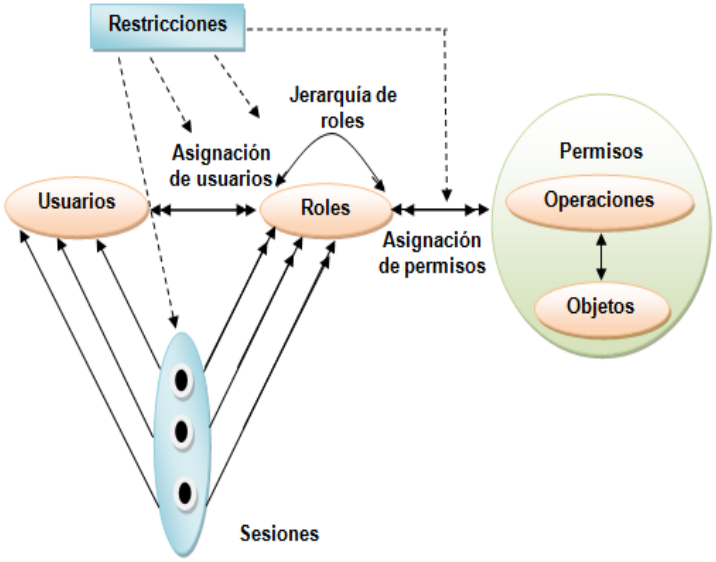
Anexo 4: Declaraciones dentro del método.

Anexo 5: Brazas o llaves.

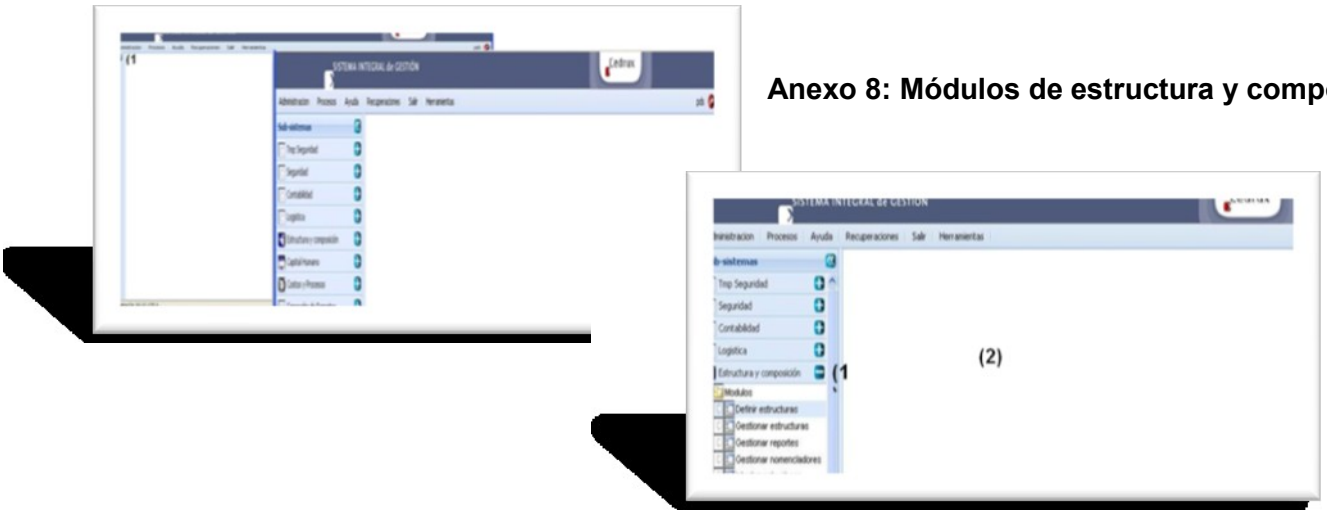
```php
< ?php
/ * *
 * Indentation
 * /
class Example {
      var $theInt = 1;
      fuction foo ( $a , $b) {
          switch ( $a ) {
              case 0 :
                  $Other ->doFoo ();
              break;
              default :
                  $Other->doBaz ();
          }
       }
  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

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

Anexo 6: Estructura del estándar RBAC.

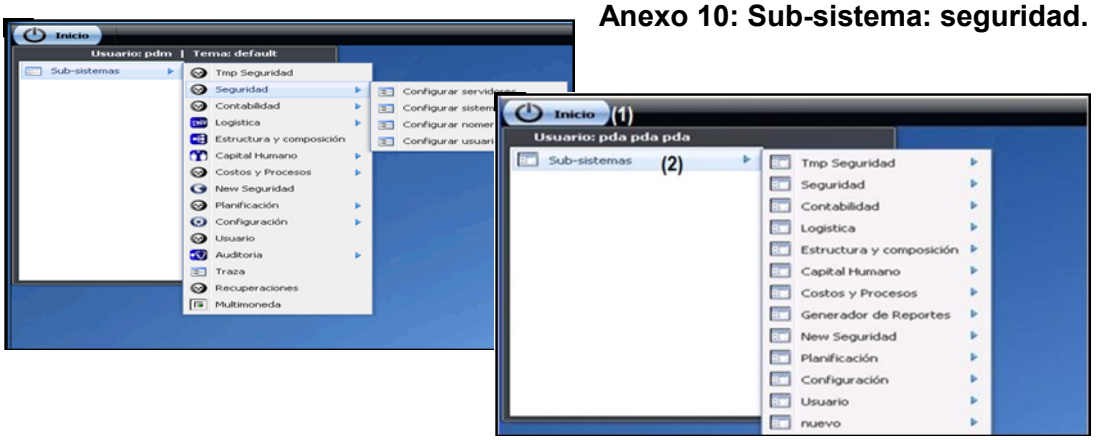


Anexo 7: Ambiente inicial de Cedrux.



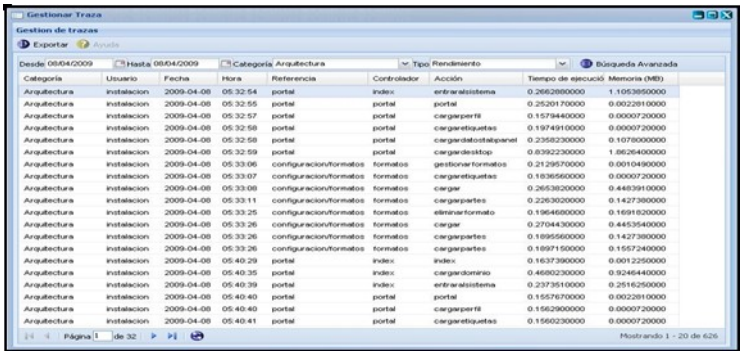
Anexo 8: Módulos de estructura y composición.

Anexo 9: Segundo ambiente de trabajo.



Anexo 10: Sub-sistema: seguridad.

Anexo 11: Componente gestionar traza.



Anexo 12: Avales

UCI Universidad de las Ciencias Informáticas | FACULTAD 4

AVAL

Por la presente damos constancia de que la "Solución para la gestión integral de sistemas en entorno multientidad" se está aplicando en el proyecto Aduana.

El Sistema de Gestión Integral de Seguridad SIGIS está concebido para garantizar la seguridad en un entorno de varias aplicaciones. Se desarrolló sobre el marco de trabajo del Sistema de Gestión de Recursos Empresariales Cedrux, lo que permite que todos los proyectos que se desarrollen sobre este marco de trabajo no tengan que preocuparse por la seguridad de sus aplicaciones ya que SIGIS es un componente más de él.

Con la reutilización de SIGIS se ahorran cuantiosos recurso 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. Este sistema se encuentra debidamente documentado permitiendo que usuarios y desarrolladores dominen a fondo todas las funcionalidades y puedan adaptarlo a sus necesidades.

SIGIS forma parte de una arquitectura de seguridad que, no sólo garantiza la seguridad de las aplicaciones, sino que tiene en cuenta la seguridad durante todas las fases del software. Por lo que utilizar este sistema permitiría la reutilización de esta arquitectura de seguridad que ya se encuentra formalizada y aprobada para su puesta en práctica con su expediente arquitectónico, listas de chequeo, normas y configuraciones seguras.

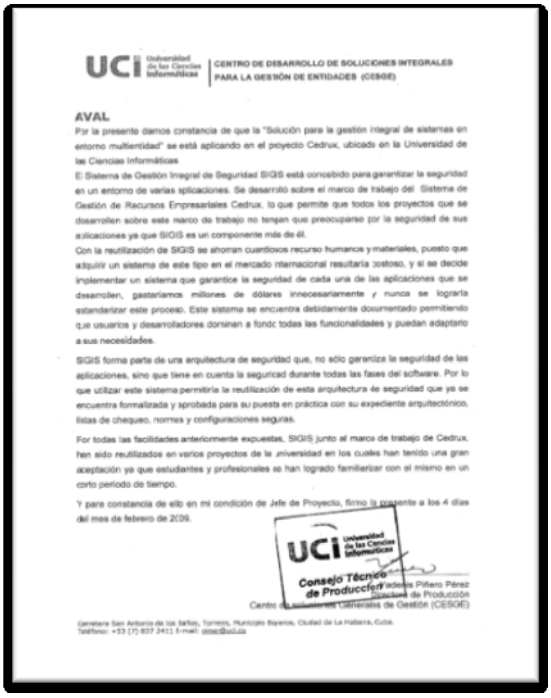
Por todas las facilidades anteriormente expuestas, SIGIS junto al marco de trabajo de Cedrux, han sido reutilizados en varios proyectos de la universidad en los cuales han tenido una gran aceptación ya que estudiantes y profesionales se han logrado familiarizar con el mismo en un corto período de tiempo.

Y para constancia de ello en mi condición de Jefe de Proyecto, firmo la presente a los 8 días del mes de junio de 2009.

Decanato Facultad 4

Alain Eduardo Rodríguez Arias
Jefe de Proyecto
Facultad 4

Carretera San Antonio de los Baños, Torrens, Municipio Boyeros, Ciudad de La Habana, Cuba.
Teléfono: +53 (7) 837 2411 E-mail: aearias@uci.cu



UCI Universidad de las Ciencias Informáticas | CENTRO DE DESARROLLO DE SOLUCIONES INTEGRALES PARA LA GESTIÓN DE ENTIDADES (CESGE)

AVAL

Por la presente damos constancia de que la "Solución para la gestión integral de sistemas en entorno multientidad" se está aplicando en el proyecto Cedrux, ubicado en la Universidad de las Ciencias Informáticas

El Sistema de Gestión Integral de Seguridad SIGIS está concebido para garantizar la seguridad en un entorno de varias aplicaciones. Se desarrolló sobre el marco de trabajo del Sistema de Gestión de Recursos Empresariales Cedrux, lo que permite que todos los proyectos que se desarrollen sobre este marco de trabajo no tengan que preocuparse por la seguridad de sus aplicaciones ya que SIGIS es un componente más de él.

Con la reutilización de SIGIS se ahorran cuantiosos recurso 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. Este sistema se encuentra debidamente documentado permitiendo que usuarios y desarrolladores dominen a fondo todas las funcionalidades y puedan adaptarlo a sus necesidades.

SIGIS forma parte de una arquitectura de seguridad que, no sólo garantiza la seguridad de las aplicaciones, sino que tiene en cuenta la seguridad durante todas las fases del software. Por lo que utilizar este sistema permitiría la reutilización de esta arquitectura de seguridad que ya se encuentra formalizada y aprobada para su puesta en práctica con su expediente arquitectónico, listas de chequeo, normas y configuraciones seguras.

For todas las facilidades anteriormente expuestas, SIGIS junto al marco de trabajo de Cedrux, han sido reutilizados en varios proyectos de la universidad en los cuales han tenido una gran aceptación ya que estudiantes y profesionales se han logrado familiarizar con el mismo en un corto período de tiempo.

Y para constancia de ello en mi condición de Jefe de Proyecto, firmo la presente a los 4 días del mes de febrero de 2009.

Consejo Técnico de Producción

Yadenis Pitiero Pérez
Dirección de Producción
Centro de Soluciones Gerenciales de Gestión (CESGE)

Carretera San Antonio de los Baños, Torrens, Municipio Boyeros, Ciudad de La Habana, Cuba.
Teléfono: +53 (7) 837 2411 E-mail: ypitiero@uci.cu

UCI Universidad de las Ciencias Informáticas | FACULTAD 5

AVAL

Por la presente damos constancia de que la "Solución para la gestión integral de sistemas en entorno multientidad" se está aplicando en el Polo de Hardware y Automática de la facultad 5, específicamente en el proyecto SEUNE.

El Sistema de Gestión Integral de Seguridad SIGIS está concebido para garantizar la seguridad en un entorno de varias aplicaciones. Se desarrolló sobre el marco de trabajo del Sistema de Gestión de Recursos Empresariales Cedrux, lo que permite que todos los proyectos que se desarrollen sobre este marco de trabajo no tengan que preocuparse por la seguridad de sus aplicaciones ya que SIGIS es un componente más de él.

Con la reutilización de SIGIS se ahorran cuantiosos recurso 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. Este sistema se encuentra debidamente documentado permitiendo que usuarios y desarrolladores dominen a fondo todas las funcionalidades y puedan adaptarlo a sus necesidades.

SIGIS forma parte de una arquitectura de seguridad que, no sólo garantiza la seguridad de las aplicaciones, sino que tiene en cuenta la seguridad durante todas las fases del software. Por lo que utilizar este sistema permitiría la reutilización de esta arquitectura de seguridad que ya se encuentra formalizada y aprobada para su puesta en práctica con su expediente arquitectónico, listas de chequeo, normas y configuraciones seguras.

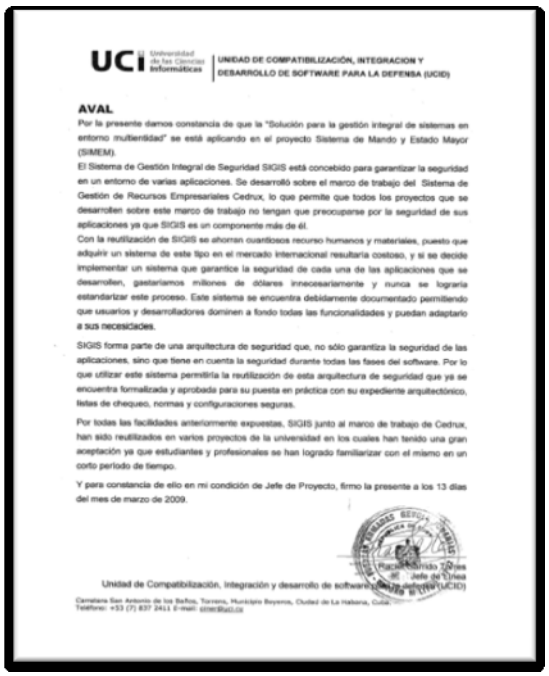
Por todas las facilidades anteriormente expuestas, SIGIS junto al marco de trabajo de Cedrux, han sido reutilizados en varios proyectos de la universidad en los cuales han tenido una gran aceptación ya que estudiantes y profesionales se han logrado familiarizar con el mismo en un corto período de tiempo.

Y para constancia de ello en mi condición de Jefe de Proyecto, firmo la presente a los 8 días del mes de junio de 2009.

Decanato Facultad 5

Raúl Pérez-Alejo Nayra
Jefe de Proyecto
Facultad 5

Carretera San Antonio de los Baños, Torrens, Municipio Boyeros, Ciudad de La Habana, Cuba.
Teléfono: +53 (7) 837 2411 E-mail: rnayra@uci.cu



UCI Universidad de las Ciencias Informáticas | UNIDAD DE COMPATIBILIZACIÓN, INTEGRACION Y DESARROLLO DE SOFTWARE PARA LA DEFENSA (UCID)

AVAL

Por la presente damos constancia de que la "Solución para la gestión integral de sistemas en entorno multientidad" se está aplicando en el proyecto Sistema de Mando y Estado Mayor (SIMEM).

El Sistema de Gestión Integral de Seguridad SIGIS está concebido para garantizar la seguridad en un entorno de varias aplicaciones. Se desarrolló sobre el marco de trabajo del Sistema de Gestión de Recursos Empresariales Cedrux, lo que permite que todos los proyectos que se desarrollen sobre este marco de trabajo no tengan que preocuparse por la seguridad de sus aplicaciones ya que SIGIS es un componente más de él.

Con la reutilización de SIGIS se ahorran cuantiosos recurso 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. Este sistema se encuentra debidamente documentado permitiendo que usuarios y desarrolladores dominen a fondo todas las funcionalidades y puedan adaptarlo a sus necesidades.

SIGIS forma parte de una arquitectura de seguridad que, no sólo garantiza la seguridad de las aplicaciones, sino que tiene en cuenta la seguridad durante todas las fases del software. Por lo que utilizar este sistema permitiría la reutilización de esta arquitectura de seguridad que ya se encuentra formalizada y aprobada para su puesta en práctica con su expediente arquitectónico, listas de chequeo, normas y configuraciones seguras.

Por todas las facilidades anteriormente expuestas, SIGIS junto al marco de trabajo de Cedrux, han sido reutilizados en varios proyectos de la universidad en los cuales han tenido una gran aceptación ya que estudiantes y profesionales se han logrado familiarizar con el mismo en un corto período de tiempo.

Y para constancia de ello en mi condición de Jefe de Proyecto, firmo la presente a los 13 días del mes de marzo de 2009.

Rachel Morrido Torres
Jefe de Línea
Unidad de Compatibilización, Integración y desarrollo de software para la defensa (UCID)

Carretera San Antonio de los Baños, Torrens, Municipio Boyeros, Ciudad de La Habana, Cuba.
Teléfono: +53 (7) 837 2411 E-mail: rmorrido@uci.cu