Universidad de las Ciencias Informáticas

Facultad 5

Diseño e implementación de un módulo de seguridad para el sistema de negocio EBMS DSerp

Trabajo de diploma para optar por el título de:

Ingeniero en Ciencias Informáticas

Autor: Yoan Hernández Méndez

Tutor: Ing. Elieser García Ramos

Co-Tutor: Ing. Luanner Kerton Martínez

La Habana, 2013

Declaración de autoría

Me declaro como único autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas como entidad con los derechos patrimoniales exclusivos sobre la misma.

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

_____

Yoan Hernández Méndez

_____ _____

Elieser García Ramos. Luanner Kerton Martínez

Tutor Co-tutor.

Dedicatoria

A mi padre por ser mi guía y ejemplo en mis decisiones, en lo profesional y cotidiano.

A mi madre por alentarme a seguir adelante y pensar positivo ante las cosas.

A mi novia por darme la fuerza que necesito para hacer bien las cosas.

Agradecimiento

A mi padre por aconsejarme en cada decisión difícil a la que me enfrento en la vida.

A mi madre por no dejar que caiga mi autoestima y tener la capacidad de tener paciencia ante las cosas.

A mi novia por hacerme cambiar la forma en las que veía las cosas e intentar ver siempre lo mejor de uno.

A mis amigos, que han estado presentes y me han apoyado: Ramcés, Arturo, Deibi, Yamil, Raúl, Denis, Justo, Leonardo, la loca de Amanda y compañía, la delgada Isabel, y otros más que me han acompañado cuando lo necesitaba.

A viejas amigas graduadas que confiaron en mí: Lourdes, Yissel y Ariadna.

A mis amistades de la universidad que me acompañaron siempre: Rosa, Adonis, Livan, Lisandra, Elvis, Yoilen, el papa Ortiz.

A todo mi equipo de trabajo del EBMS DSerp.

A mis tutores y en especial a Luanner Kerton que nunca dejó de guiarme.

A todos mis profesores por darme la formación que necesitaba para convertirme en ingeniero.

A todos los que en alguna forma me apoyaron en hacer realidad mis sueños de profesional.

Resumen

La seguridad de información en los sistemas de gestión empresarial ha estado en constantes cambios a causa del desarrollo evolutivo de las tecnologías y el uso de las redes. Lo que ha traído una constante competitividad por satisfacer las necesidades del cliente, siendo necesario crear mecanismos de seguridad sobre los recursos que estos manejan.

Cuba, por las condiciones económicas y las peculiaridades de su sistema empresarial, desarrolla soluciones para el manejo y control de las empresas basándose en las ventajas de los sistemas de información y las tecnologías. El *EBMS* DSerp es un sistema para la gestión de recursos y procesos que permite administrar varias empresas que comparten dependencias funcionales sustanciales. Este sistema carece de un módulo que permite salvaguardar la información manipulada, poniendo en riesgo los recursos más valiosos de la empresa. En la presente investigación se detalla el proceso de desarrollo del sistema de seguridad del *EBMS* DSerp, puntualizando los aspectos relacionados al negocio, sus fases de desarrollo y los resultados alcanzados tras su integración al sistema. Además, se describen los artefactos de la ingeniería y se definen las principales técnicas que se utilizan para asegurar la confidencialidad, integridad y disponibilidad de la información empresarial gestionada por la plataforma.

Índice

Introducción

En los últimos años ha habido un aumento considerable en el número de empresas producto del auge del internet e inversiones en tecnologías de comercio electrónico, provocando un entorno competitivo y productivo de la sociedad empresarial, que busca optimizar y automatizar sus procesos con el fin de elevar su rendimiento operativo y financiero apoyado en los sistemas de información como estructura fundamental del negocio. En la actualidad los grandes sistemas dedicados a la gestión empresarial, generan información que es dirigida a la gerencia para la toma de decisiones, por lo cual la seguridad informática es factor clave e indispensable para la administración de sus recursos, garantizando la integridad, disponibilidad y confidencialidad de la información que se maneja.

A raíz de la competencia generada de las necesidades y demandas cada vez más exigentes por parte de los consumidores, existen varios sistemas para la gestión y administración de empresas, destacándose *SAP*[1], *OpenBravo 3* y *OpenERP* como los principales sistemas de planificación de recursos empresariales; además de varios modelos y sistemas de seguridad que integrado a los anteriores, proporcionan un sólido fundamento para la agilidad en el negocio, garantizando que la información manipulada se encuentre salvaguardada íntegramente.

Cuba, como país subdesarrollado, con condiciones excepcionales de bloqueo comercial, carente de una gran economía y con poco respaldo tecnológico, no puede abogar por la compra de patentes ni licencias de software en un mercado externo dominado por países industrializados. Por tanto, el país debe encaminarse al desarrollo de sus propias tecnologías enfocadas a la gestión de sus datos empresariales, desarrollando productos que permitan la automatización del modelo empresarial cubano y den como resultado un aumento en el control, eficiencia y ganancia necesaria para nuestro desarrollo económico.

EBMS[2] DSerp es un sistema estratégico para la gestión avanzada de procesos y recursos que se desarrolla con el objetivo de dar soporte a la gestión de información empresarial. Su arquitectura de *EBMS* le ofrece adaptabilidad para desplegarse en cualquier ambiente corporativo, con un modelo centralizado, flexible e interactivo, que le permite integrar y administrar procesos claves de una

[1] *SAP:* Siglas del inglés *Systems, Applications, and Products in Data Processing* o Sistemas, aplicaciones y productos para el procesamiento de datos.
[2] *EBMS*: Siglas del inglés *Enterprise Business Management System* o Sistema de Administración de Negocios Empresariales en español (Kerton, 2012).

organización, garantizando la automatización de las actividades generadas por un conjunto de empresas, creando un dominio de negocio para la gestión de información en un entorno corporativo. Está diseñado para operar sobre diferentes sectores de la sociedad, y se ha visto aplicado principalmente en empresas agrícolas y en algunos procesos del sector médico, siendo proveedor de un conjunto de herramientas, que permiten elevar la eficacia y eficiencia del proceso empresarial a partir de soluciones tecnológicas.

Actualmente el sistema *EBMS* DSerp no cuenta con un mecanismo que le permita establecer políticas de seguridad que garanticen la protección de la información manipulada, no cumpliendo con los aspectos mínimos estipulados para los sistemas dirigidos a la gran empresa. Esto causa que la información en el sistema quede vulnerable ante la manipulación indebida y el sistema no logra un correcto desempeño en el desarrollo de sus objetivos; no permite contar con una arquitectura segura y centralizada en la disponibilidad de sus servicios y operaciones, trayendo consigo que sea muy difícil limitar violaciones de seguridad o no detectarlas y tratarlas en caso de que ocurran, pudiéndose provocar daños irreparables para las empresas, asociados a gastos innecesarios y pérdida de tiempo. Dichas deficiencias, afectan todo el bloque de procesos de la plataforma *EBMS* DSerp, tanto los operativos como los financieros, haciendo que esta sea vulnerable a ataques externos y manipulación indebida de datos, incidiendo directamente en los diez valores fundamentales sobre los cuales se desarrolló el sistema: accesibilidad, integridad, confidencialidad, información, sencillez, organización, fiabilidad, rapidez, eficiencia y eficacia.

Teniendo en cuenta la situación problémica planteada se define como **problema a resolver:** ¿Cómo garantizar la seguridad de la información en el sistema *EBMS* DSerp?

Lo que determina como **objeto de estudio:** Seguridad en sistemas de información.

Por lo expresado, se define como **campo de acción:** Seguridad en sistemas de información en entornos empresariales.

Para darle solución al problema planteado se propone como **objetivo general:** Desarrollar un módulo que permita detallar y estructurar los niveles y derechos de acceso de las empresas y usuarios sobre la información manejada en el sistema *EBMS* DSerp.

Teniéndose como **objetivos específicos:**

- ✓ Asegurar una política de acceso al sistema basada en roles y usuarios.

- ✓ Permitir la configuración de seguridad para las distintas empresas y a distintos niveles, de forma dinámica y personalizada.
- ✓ Garantizar la protección basada en tiempos y direcciones de acceso.
- ✓ Brindar un sistema de trazas que garantice el seguimiento de acciones dentro de la plataforma.

Como **posible resultado** se tiene: un módulo de seguridad que permita el acceso y control de la información manejada en el sistema *EBMS* DSerp, garantizando la confidencialidad, integridad y accesibilidad de la información.

Para dar cumplimiento a los objetivos específicos se definen las siguientes **tareas de investigación**:

- ✓ Definición de los diferentes conceptos, modelos y técnicas asociados a las políticas de seguridad en sistemas dirigidos a la gran empresa.
- ✓ Diseño del sistema de seguridad del *EBMS* DSerp.
- ✓ Integración del módulo de seguridad a los diferentes ambientes y escenarios del sistema *EBMS* DSerp.
- ✓ Realización de pruebas a la seguridad del sistema.

Desde el punto de vista metodológico se emplean los siguientes **métodos científicos**:

Métodos teóricos:

Analítico – Sintético: Utilizado para analizar elementos bibliográficos y definiciones sobre la seguridad en sistemas de información, con el objetivo de arribar a conclusiones que sustenten la necesidad de la investigación.

Análisis histórico – lógico: Es utilizado para analizar la evolución histórica de soluciones similares, las tendencias más recientes en cuanto a la seguridad de los sistemas de información de las empresas, complementar las características necesarias y deseables para la solución que se propone.

Modelación: Se empleará para representar mediante gráficas, modelos y esquemas la solución propuesta.

Métodos empíricos:

Observación: Para realizar el estudio de las características y comportamientos que se emplean para dar soluciones permitiendo la formulación global de la investigación.

Entrevista: Se aplica a especialistas familiarizados con la seguridad en los sistemas de información, para determinar requerimientos e impresiones que aporten valores y bases para la investigación.

Estructura de capítulos

Para dar cumplimiento a los objetivos propuestos se establece la siguiente estructura de capítulos:

Capítulo 1. Fundamentación teórica: En este capítulo se definirán los principales conceptos asociados al negocio realizados durante todo el proceso de investigación.

Capítulo 2. Descripción del sistema: Se argumentarán las necesidades del sistema *EBMS* DSerp, estructuración del diseño de seguridad y se realiza una descripción de los requerimientos y la planificación a tener en cuenta durante el desarrollo.

Capítulo 3. Descripción de la solución: Se describe el diseño propuesto para la solución, se define el módulo de seguridad atendiendo a los diferentes ambientes y sistemas.

Capítulo 4. Implementación y validación de resultados: En el desarrollo de este capítulo se describirá la implementación de la aplicación, además de dar una descripción de las pruebas realizadas al sistema con el despliegue de la solución propuesta.

Capítulo 1. Fundamentación teórica

1.1. Introducción

En este capítulo se describen una serie de conceptos y definiciones relacionados con las políticas de seguridad que se establecen para el negocio, describiendo los principales sistemas que se destacan en el mercado actual para la gerencia empresarial así como los principales modelos y esquemas de seguridad que se implementan para la gestión de información y las nuevas tendencias que más sobresalen en el ámbito internacional. Se definen las herramientas y tecnologías a emplear durante el desarrollo del sistema que satisfagan las necesidades del nuevo modelo empresarial de la sociedad.

1.2. Seguridad de software

Con el crecimiento del internet y la inmensa propagación de herramientas como tendencias tecnológicas para el uso de las redes, tales como el comercio electrónico, se hace necesaria un esquema que le permita al actual modelo empresarial tener dominio de la seguridad sobre la información que se maneja, permitiéndole mediante una configuración adecuada poder evitar el acceso y configuración de sus recursos, por parte de personas no autorizadas, ya sea localmente o a nivel de red. Puede entonces definirse como la seguridad de la información, como un conjunto de mecanismo que mediante una serie de procedimientos garantiza la integridad, la disponibilidad y confidencialidad de la información que se maneja en un sistema informático. Cumpliendo este concepto, se dice que un software es seguro, para esto, se debe de establecer una serie de parámetros y requisitos que permitan satisfacer las necesidades del sistema, definiéndose la seguridad de software como un concepto multidimensional (Shreyas, 2001). Las múltiples dimensiones de la seguridad son:

- ✓ **Autenticación:** El proceso de verificar la identidad de una entidad.
- ✓ **Control de acceso:** El proceso de regular las clases de acceso que una entidad tiene sobre los recursos.
- ✓ **Auditoría:** Un registro cronológico de los eventos relevantes a la seguridad de un sistema. Este registro puede luego examinarse para reconstruir un escenario en particular.
- ✓ **Confidencialidad:** La propiedad de que cierta información no esté disponible a ciertas entidades.
- ✓ **Integridad:** La propiedad de que la información no sea modificada en el trayecto fuente-destino.
- ✓ **Disponibilidad:** La propiedad de que el sistema sea accesible a las entidades autorizadas.
- ✓ **No repudio:** La propiedad que ubica la confianza respecto al desenvolvimiento de una entidad en una comunicación.

Todo sistema que utilice sistemas informáticos requiere información que es dirigida a la gerencia para la toma de decisiones, para esto el sistema debe de estar preparado ante futuras amenazas, ya sea mediante la violación de su seguridad lógica, como de su seguridad física. Para esto se determinan una serie de objetivos con el fin de alcanzar que el software sea seguro (Asteasuain, y otros, 2007):

- ✓ **Independencia de la seguridad:** La seguridad debe construirse y utilizarse de manera independiente de la aplicación.
- ✓ **Independencia de la aplicación:** La aplicación no debe depender del sistema de seguridad usado, debe ser desarrollada y mantenida en forma separada.
- ✓ **Uniformidad:** La seguridad debe aplicarse de manera correcta y consistente a través de toda la aplicación y del proceso que desarrolla la misma.
- ✓ **Modularidad:** Mantener la seguridad separada. Entre otras ventajas, esto nos brindará mayor flexibilidad y menor costo de mantenimiento.
- ✓ **Ambiente seguro:** Se debe partir de un entorno confiable. Es decir, las herramientas de desarrollo y lenguajes de programación no deben contener agujeros de seguridad.
- ✓ **Seguridad desde el comienzo:** La seguridad debe ser considerada como un requerimiento desde el inicio del diseño.

1.3. Mecanismos de seguridad

Para poder crear un sistema informático, hay que llevar a cabo un análisis profundo y detallado de cuáles podrían ser las principales amenazas que pudiera sufrir dicho sistema. A partir de este análisis, habrá que diseñar una política de seguridad capaz de integrarse a las estrategias del negocio, a su misión y visión, estableciendo un grupo de responsabilidades y reglas a seguir para evitar esas amenazas o minimizar los efectos si se llegan a producir (Corletti, 2011). Se puede definir una política de seguridad como un conjunto de reglas llevado a cabo por alguna entidad o empresa que mantengan los niveles de seguridad estables, que sea capaz de minimizar los daños producidos ante alguna amenaza y en caso de que ocurra tratar de reponerse. Para esto se implementan un grupo de mecanismos para la seguridad del sistema, que garanticen la protección del mismo o de la propia red. Estos tipos de mecanismos pueden ser clasificados en tres grupos (Gerner, y otros, 2006):

- ✓ Mecanismo de prevención.
- ✓ Mecanismos de detección.
- ✓ Mecanismos de recuperación.

De estos mecanismos, los sistemas de información empresarial se centran en los mecanismos de prevención, dado a que se espera garantizar un comportamiento seguro desde el inicio de los procesos, en cuyo caso, la detección y la recuperación constituyen procesos de segundo grado.

1.3.1. Mecanismo de prevención

Son aquellos que aumentan la seguridad de un sistema durante su funcionamiento. Los mecanismos de prevención más habituales en redes son los siguientes:

- ✓ **Mecanismos de autenticación e identificación.** Este tipo de mecanismo permite identificar la entrada al sistema de una determinada entidad y una vez identificada realiza el proceso de autentificación, de esta forma se garantiza que el usuario que quiere entrar a la aplicación es quien dice ser.
- ✓ **Mecanismos de control de acceso.** Permite tener un control sobre los servicios u objetos que ofrece el sistema.
- ✓ **Mecanismos de separación.** Cualquier sistema con diferentes niveles de seguridad ha de implementar mecanismos que permitan separar los objetos dentro de cada nivel, evitando el flujo de información entre objetos y entidades de diferentes niveles siempre que no exista una autorización expresa del mecanismo de control de acceso. Los mecanismos de separación se dividen en cinco grandes grupos, en función de cómo separan a los objetos: separación física, temporal, lógica, criptográfica y fragmentación.
- ✓ **Mecanismos de seguridad en las comunicaciones.** En un sistema es muy importante proteger la integridad y privacidad de los datos cuando se transmite a través de la red, la mayoría de estos mecanismos se basan en la criptografía que no es más que un conjunto de protocolos y sistemas que se utilizan para proteger la información y dotar de seguridad a las comunicaciones y a las entidades que se comunican (Salazar Riaño, y otros, 1998).

1.3.2. Mecanismo de detección

Son aquellos que se utilizan para detectar violaciones de la seguridad o intentos de violación; ejemplos de estos mecanismos son los programas de auditoría como *Tripwire*[3].

1.3.3. Mecanismo de recuperación

Son aquellos que se aplica cuando ha habido una violación en el sistema y poder retornar al mismo a un funcionamiento correcto.

[3] *Tripwire*: Herramienta de seguridad para la realización de auditorías.

1.4. Sistemas empresariales

La empresa es una unidad económico-social en la que el capital, el trabajo y la dirección se coordinan para lograr una producción que responda a los requerimientos del medio humano en el que la propia empresa actúa (Power, 2002). Asumiéndose que la empresa es una organización cuyo objetivo principal va ser la de obtener un beneficio mediante la satisfacción de una necesidad de mercado. Se define entonces, que un sistema empresarial está formado por un conjunto de grupos, en el que cada grupo va a realizar una determinada actividad dentro del sistema, por lo cual la interacción que va a existir dentro de estos grupos tiene que ser lo menor posible, debido a que mientras más pequeño sea este acoplamiento, más regular será el flujo de señales y materiales de trabajo. Estos grupos en conjunto, deben dirigirse a satisfacer las necesidades de la empresa, generalmente en la gestión y automatización de sus procesos.

1.5. Consideraciones necesarias para la seguridad en sistemas empresariales

Producto de las demandas cada vez más exigentes por parte de los consumidores y la competencia que se genera dadas estas necesidades, cada empresa se traza una estrategia de mantener segura tanto la información que se maneja en el sistema, como la del personal dentro de la entidad. Para esto, en dependencia de la envergadura que tenga la empresa se trazan los siguientes aspectos con el objetivo de alcanzar una fiabilidad de la organización (Corletti, 2011):

- ✓ Definir elementos administrativos.
- ✓ Definir políticas de seguridad a nivel departamental y a nivel institucional.
- ✓ Organizar y dividir las responsabilidades.
- ✓ Contemplar la seguridad física contra catástrofes (incendios, terremotos, inundaciones, u otros).
- ✓ Definir prácticas de seguridad para el personal.
- ✓ Plan de emergencia (plan de evacuación, uso de recursos de emergencia como extinguidores).
- ✓ Números telefónicos de emergencia.
- ✓ Definir el tipo de pólizas de seguros.
- ✓ Definir elementos técnicos de procedimientos.
- ✓ Definir las necesidades de sistemas de seguridad.

1.6. Estándares internacionales de seguridad informática

En la sociedad moderna actual, es importante para una organización que utiliza sistemas de información, tener una gestión competente y efectiva de la seguridad de los recursos y datos que gestionan, por lo cual entonces se acogen a un grupo de normas y estándares internacionales como la

manera de demostrar que tienen implementado un buen uso de las prácticas de su seguridad. Este grupo de estándares durante muchos años ha sido regido por dos grandes instituciones de la normalización a nivel internacional: la *ISO*[4] y por otra parte a *NIST*[5] las cuales proponen un marco de trabajo en cuanto a la seguridad informática definidas en las siguientes normas (Mártiles, 2008):

- ✓ ***NIST* serie 800.** La serie 800 del *NIST* contiene una serie de documentos de gran interés sobre Seguridad de la Información, refiriéndose a los requerimientos mínimos de cualquier sistema.

Se encuentran también la familia de normas ***ISO/IEC* 2700** como la familia de estándares ISO que proporcionan un marco para la gestión de la seguridad:

- ✓ ***ISO/IEC* 27000.** Define el vocabulario estándar empleado en la familia 27000 (Definición de términos y conceptos).
- ✓ ***ISO/IEC* 27001.** Especifica los requerimientos a cumplir para implementar un Sistema de Gestión de la Seguridad de la Información (SGSI) certificable conforme con las normas 27000.
- ✓ ***ISO/IEC* 27002.** Código de buenas prácticas para la gestión de la seguridad.
- ✓ ***ISO/IEC* 27003.** Guía de implementación de SGSI e información acerca del uso del modelo *PDCA*[6] y de los requerimientos de sus diferentes fases (en desarrollo, pendiente de publicación).
- ✓ ***ISO/IEC* 27004.** Especifica las métricas y las técnicas de medida aplicables para determinar la eficacia de un SGSI y de los controles relacionados (en desarrollo, pendiente de publicación).
- ✓ ***ISO/IEC* 27005.** Gestión de riesgos de seguridad de la información (recomendaciones, métodos y técnicas para evaluación de riesgos de seguridad).
- ✓ ***ISO/IEC* 27006.** Requisitos a cumplir por las organizaciones encargadas de emitir certificaciones *ISO/IEC* 270001.
- ✓ ***ISO/IEC* 27007.** Guía de actuación para auditar los SGSI conforme a las normas 27000.
- ✓ ***ISO/IEC* 27011.** Guía de gestión de seguridad de la información específica para telecomunicaciones (en desarrollo) elaborada conjuntamente con la *ITU*[7].
- ✓ ***ISO/IEC* 27031.** Guía de continuidad de negocio en lo relativo a tecnologías de la información y comunicaciones (en desarrollo).
- ✓ ***ISO/IEC* 27032.** Guía relativa a la ciber-seguridad (en desarrollo).
- ✓ ***ISO/IEC* 27032.** Guía de seguridad en aplicaciones (en desarrollo).

[4] *ISO: International Organization for Standardization* del ingles.
[5] *NIST: National Institute of Standards and Technology* del ingles.
[6] *PDCA: Plan do Check Act* del inglés o plan para chequear actividades del español.
[7] *ITU*: Unión Internacional de Telecomunicaciones del español.

1.7. Estructura para garantizar la seguridad en sistemas empresariales cubanos

Nuestro país, con el pasar de los años, debido al descontrol del uso de las nuevas tecnologías demandadas por el mundo contemporáneo actual, hace todo lo posible para que nuestro sistema empresarial en busca del perfeccionamiento en toda la infraestructura de su sistema, pueda estar al alcance de la competitividad requerida en la sociedad moderna, en un mundo que cada vez va siendo más globalizado del cual juega un papel importante la seguridad informática.

En este contexto, han de tomarse como punto de partida para la implementación de un sistema de seguridad informática en nuestro sistema empresarial, a partir de las disposiciones legales vigentes para tales fines. El mismo deberá contener (Meléndez Carballido, y otros, 2010):

1. Alcance. El mismo se hará extensivo a cada una de las personas con acceso a los medios informáticos que se relacionen en dicho plan.
2. Caracterización del Sistema Informático. Donde se definirán las Características de las Computadoras haciendo énfasis en las que se conectarán a internet así como el software y el hardware de cada una de ellas.
3. Resultado del análisis de Riesgo Para realizar este Plan de Seguridad Informática se deberá llevar a cabo el estudio y análisis de los riesgos que implicaría la conexión a internet con el objetivo de identificar los recursos que se afectarían por las violaciones de seguridad y las amenazas a las que están expuestos dichos recursos.
4. Las políticas de Seguridad. Resultará indispensable, establecer claramente una política de seguridad que garantice.
5. Sistema de seguridad informática, En este aspecto, se definirán las áreas a proteger y dentro de estas una especial atención a aquella destinada a la navegación de Internet tales como:

- ✓ Medios Humanos.
- ✓ Medios Técnicos.
- ✓ Medidas y Procesamientos de Seguridad Informática.
- ✓ Medidas de Protección Física.
- ✓ A las Áreas con Tecnología Instaladas
- ✓ A las Tecnologías de Información.
- ✓ A los Soportes de Información.
- ✓ Medidas Técnicas o Lógicas.
- ✓ Identificación de Usuarios.

- ✓ Autenticación de Usuarios.
- ✓ Control de Acceso a los Activos y Recursos.
- ✓ Integridad de los Ficheros y Datos.
- ✓ Auditorías y Alarmas.
- ✓ Medidas de Seguridad de las Operaciones.
- ✓ Medidas de Recuperación ante Contingencias.

1.8. Principales tipos de ataques a la seguridad de sistemas

Cuando hablamos de los diferentes tipos de ataques a los cuales un sistema se enfrenta, podemos referirnos a los elementos que potencialmente resulta una amenaza, dividiéndolos en un grupo **personas** y otro grupo **amenazas lógicas**.

1.8.1. Personas

Los ataques que puede sufrir un sistema está dado mediante personas ya sea de forma intencional o no intencional, pudiendo causar grandes pérdidas, pero por lo general este tipo de ataques van a ser provocados por piratas informáticos aprovechando las debilidades o agujeros de software que pudiera tener nuestro sistema. Pudiéndose clasificar a estos tipos de personas como atacantes pasivos, aquellos que se adentran en el sistema pero que no lo modifica o destruyen, y los activos, aquellos que si destruyen el objetivo que se trazan cuando invaden al sistema.

- ✓ **Personal.** Las amenazas a la seguridad proveniente del personal de la propia organización rara vez son tomadas en cuenta, ya se crea un entorno de confianza que a veces no existe. Aunque los ataques pueden ser realizados intencionalmente, pudiera ser causados por un error o por desconocimientos de las normas de seguridad de la entidad.
- ✓ **Ex-empleados.** Se pudiera tratar de antiguos empeñados de la entidad descontenta por la entidad o bien pudiera ser en el caso de redes de empresa, que pasaron a la competencia, que aprovechan las debilidades de un sistema que conocen perfectamente para dañarlo como venganza o por algún hecho que no consideran justo.
- ✓ **Curiosos.** Aunque en la mayoría de situaciones se trata de ataques no destructivos, parece claro que no benefician en absoluto al entorno de fiabilidad que se pueda generar en un determinado sistema.
- ✓ ***Crackers*.** Los entornos de seguridad media son un objetivo típico de los intrusos, ya sea para fisgonear, para utilizarlas como enlace hacia otras redes o simplemente por diversión. Las redes son generalmente abiertas, y la seguridad no es un factor tenido muy en cuenta en ellas.

- ✓ **Terroristas.** Nos referimos a terroristas informáticos, bajo esta definición se engloba a cualquier persona que ataca al sistema simplemente por causar algún tipo de daño en él.
- ✓ **Intrusos remunerados.** Se trata de piratas con gran experiencia en problemas de seguridad y un amplio conocimiento del sistema, que son pagados para robar o simplemente para dañar la imagen de la entidad afectada.

1.8.2. Amenazas lógicas

Dentro del grupo de amenazas lógicas encontramos todo tipo de programas que de una forma u otra pueden dañar a nuestro sistema, creados de forma intencionalmente tales como software malicioso, también conocidos como *malware*[8] o simplemente por error llamados *bugs* o agujeros dentro los que se encuentran (Stephen, 2000):

- ✓ ***Software* incorrecto.** Las amenazas más habituales a un sistema provienen de errores cometidos de forma involuntaria por los programadores de sistemas o de aplicaciones. A estos errores de programación se les denomina *bugs*, y a los programas utilizados para aprovechar uno de estos fallos y atacar al sistema se denominan *exploits*.
- ✓ **Herramientas de seguridad.** Cualquier herramienta de seguridad representa un punto vulnerable: de la misma forma que un administrador las utiliza para detectar y solucionar fallos en sus sistemas o en la subred completa, un potencial intruso las puede utilizar para detectar esos mismos fallos y aprovecharlos para atacar los equipos. Herramientas como *nessus* o *saint*, que pasan de ser útiles a ser peligrosas cuando las utilizan *crackers*[9] que buscan información sobre las vulnerabilidades de un *host* o de una red completa.
- ✓ **Puertas traseras.** Durante el desarrollo de aplicaciones grandes o de sistemas operativos es habitual entre los programadores insertar atajos en los sistemas habituales de autenticación del programa o del núcleo que se está diseñando. A estos atajos se los denomina puertas traseras, y con ellos se consigue mayor velocidad a la hora de detectar y depurar fallos.
- ✓ **Bombas lógicas.** Las bombas lógicas son partes de código de ciertos programas que permanecen sin realizar ninguna función hasta que son activadas; en ese punto, la función que realizan no es la original del programa, sino que generalmente se trata de una acción perjudicial.
- ✓ **Canales cubiertos.** Los canales cubiertos o canales ocultos, son canales de comunicación que permiten a un proceso transferir información de forma que viole la política de seguridad del

[8] *Malware:* Código maligno. El término *malware* es muy utilizado por profesionales de la informática para referirse a una variedad de software hostil, intrusivo o molesto.
[9] *Crakers*: Personas que rompen algún sistema de seguridad.

sistema. No constituyen una amenaza demasiado habitual en redes de I+D[10], sin embargo, es posible su existencia, y en este caso su detección suele ser difícil.

- ✓ **Virus.** Es un *malware* que tiene por objeto alterar el normal funcionamiento de la computadora, sin el permiso o el conocimiento del usuario. Los virus, habitualmente, reemplazan archivos ejecutables por otros infectados con el código de este.
- ✓ **Gusanos.** Un gusano es un programa capaz de ejecutarse y propagarse por sí mismo a través de redes, en ocasiones portando virus o aprovechando *bugs* de los sistemas a los que conecta para dañarlos. Al ser difíciles de programar su número no es muy elevado, pero el daño que pueden causar es muy grande. Un gusano puede automatizar y ejecutar en unos segundos todos los pasos que seguiría un atacante humano para acceder al sistema, pero en un tiempo muchísimo menor. De ahí su enorme peligro y sus devastadores efectos.
- ✓ **Caballos de Troya.** Los troyanos o caballos de Troya son instrucciones escondidas en un programa de forma que éste parezca realizar las tareas que un usuario espera de él, pero que realmente ejecute funciones ocultas, es decir que ocultan su función real bajo la apariencia de un programa inofensivo que a primera vista funciona correctamente.
- ✓ **Programas conejo o bacterias.** Son los programas que no hacen nada útil, sino que simplemente se dedican a reproducirse hasta que el número de copias acaba con los recursos del sistema ya sea memoria, procesador, disco, u otros. produciendo una negación de servicio.
- ✓ **Técnicas salami.** Por técnica salami se conoce al robo automatizado de pequeñas cantidades de bienes, generalmente dinero de una gran cantidad origen. El hecho de que la cantidad inicial sea grande y la robada pequeña hace extremadamente difícil su detección. Las técnicas salami no se suelen utilizar para atacar sistemas normales, sino que su uso más habitual es en sistemas bancarios; sin embargo, en una red con requerimientos de seguridad medios es posible que haya ordenadores dedicados a contabilidad, facturación de un departamento o gestión de nóminas del personal.

1.9. Sistemas para el control de seguridad

En la actualidad, con el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC), las empresas gestionan su información a través de sistemas informáticos, por lo cual se hace necesario que los sistemas tengan estructurado una política de seguridad adecuada que permita la protección de sus datos. Existen sistemas que en menor o mayor medida brinda seguridad en cuanto a

[10] I+D: Investigación y desarrollo.

la gestión de información, destacándose los sistemas *single sign-on* como un fuerte marco de trabajo dentro de la empresa.

1.9.1. Sistemas single sign-on

Single sign-on (SSO) es un procedimiento de autenticación y autorización que habilita al usuario para acceder a varios sistemas con una sola instancia de identificación. Con la implementación de un agente *SSO* el sistema se encarga de almacenar, en una base datos o directorios protegidos, las credenciales que permiten al usuario acceder a cada una de las aplicaciones o servicios en el momento que lo desee. Se puede decir que con dicha implementación el sistema simplificaría y centralizaría el control de accesos a todas las aplicaciones de la empresa o corporación, reduciendo el costo en la administración de seguridad, lo que logra un mejor rendimiento y velocidad de los procesos de autenticación y acceso que facilita a los usuarios la interacción con los sistemas de la empresa, simplificando el manejo de claves y lo fundamental es que aumenta los niveles de seguridad, ya que contará con una plataforma central para el manejo de la seguridad en todos los procesos de autenticación y acceso a sus aplicaciones (Mártiles, 2008). Existen cinco tipos principales de *SSO*, también se les llama *reduced sign on systems* (en español, sistemas de autenticación reducida), los cuales son:

- ✓ **Enterprise single sign-on (E-SSO)**

  También llamado *Legacy Single Sign*-On, el cual funciona luego de una autenticación primaria, interceptando los requerimientos de autenticación presentados por las aplicaciones secundarias para completarlos con el usuario y la contraseña. Los sistemas *E-SSO* permiten interactuar con sistemas que pueden deshabilitar la presentación de la pantalla de autenticación.

- ✓ **Web single sign-on (Web-SSO)**

  Conocido como *Web Access Management (Web-AM),* trabaja sólo con aplicaciones y recursos que se acceden vía web. Los accesos son interceptados con la ayuda de un servidor proxy o de un componente instalado en el servidor web destino. Los usuarios no autenticados que tratan de acceder son redirigidos a un servidor de autenticación y regresan sólo después de haber logrado un acceso exitoso. Se utilizan *cookies*, para reconocer aquellos usuarios que acceden y su estado de autenticación.

- ✓ **Kerberos**

  Es un método popular de externalizar la autenticación de los usuarios. Los usuarios se registran en el servidor Kerberos y reciben un *ticket*, que luego utilizan para obtener acceso. Kerberos mantiene una base de datos de claves secretas; cada entidad en la red sea cliente o servidor comparte una clave secreta conocida únicamente por él y Kerberos. El conocimiento de esta clave sirve para

probar la identidad de la entidad. Para una comunicación entre dos entidades, Kerberos genera una clave de sesión, la cual pueden usar para asegurar sus problemas.

- ✓ **Identidad federada**

  Es una nueva manera de concebir este tema, también para aplicaciones web. Utiliza protocolos basados en estándares para habilitar que las aplicaciones puedan identificar los clientes sin necesidad de autenticación redundante. Mediante soluciones de Identidad Federada los individuos pueden emplear la misma identificación personal (típicamente usuario y contraseña) para identificarse en redes de diferentes departamentos o incluso empresas. De este modo las empresas comparten información sin compartir tecnologías de directorio, seguridad y autenticación.

- ✓ **OpenID**

  OpenID es un estándar de identificación digital descentralizado, con el que un usuario puede identificarse en una página web a través de una *URL* (o un *XRI* en la versión actual) y puede ser verificado por cualquier servidor que soporte el protocolo. A diferencia de arquitecturas *Single Sign-On*, *OpenID* no especifica el mecanismo de autenticación. Por lo tanto, la seguridad de una conexión *OpenID* depende de la confianza que tenga el cliente *OpenID* en el proveedor de identidad. Si no existe confianza en el proveedor, la autenticación no será adecuada para servicios bancarios o transacciones de comercio electrónico, sin embargo el proveedor de identidad puede usar autenticación fuerte pudiendo ser usada para dichos fines.

1.10. Seguridad en sistemas de información cubana

1.10.1. Versat Sarasola

Es un Sistema integrado de gestión económica, diseñado para ser utilizado por el sector empresarial Cubano, que se adecua a las características de cada entidad, ya que es configurable por cada una de ellas en el momento de su instalación y tiene como objetivo fundamental ofrecerle a los usuarios la posibilidad de contar con un instrumento seguro, rápido, eficaz y de fácil manejo para la Planificación, Control y el Análisis de la Gestión Económica (Cabrera González, y otros, 2012).

Para tener control sobre la información del sistema, se agrupan en diferentes funciones las que se pueden resumir en:

- ✓ Definir los usuarios que trabajarán en el Sistema.
- ✓ Asignar a los usuarios los permisos de trabajo en las Unidades Contables y dentro de estas a que Subsistemas tienen acceso.
- ✓ Definirle a cada usuario que opciones de trabajo tiene dentro de cada subsistema.

Estas a su vez, tienen un grupo opciones para garantizar la configuración de cada una de ellas así como las tareas que pueden realizar los usuarios dentro de cada Subsistema, lo que permite cumplir mediante el sistema, importantes principios de control interno, definiéndose los usuarios, permisos de los mismos, niveles de acceso y las operaciones de trabajo.

1.10.2. Acaxia

CEDRUX es un sistema informático para la gestión empresarial, como parte del proyecto ERP-Cuba. La seguridad de este sistema está basada en la solución Acaxia desarrollada sobre software libre, incorpora la gestión dinámica de multi-tema, multi-idioma, multi-entidad, multi-escritorio y multi-sistema. De esta forma el usuario tendrá acceso a un conjunto de sistemas en una o varias entidades, garantizándose siempre la compartimentación de la información, cada usuario solo podrá gestionar la información a la que tenga acceso. Acaxia implementa cinco procesos parta la seguridad de su sistema (Gómez Baryolo, y otros, 2011):

- ✓ Autenticación.
- ✓ Administración de conexiones.
- ✓ Autorización.
- ✓ Auditoría.
- ✓ Administración de perfil.

El sistema de gestión integral de seguridad Acaxia es capaz de gestionar la seguridad de otros sistemas que se suscriban a él de forma centralizada. Para ello se controla la información de cada uno de ellos que serán provistos de seguridad o auditados en un momento determinado. De cada uno de estos sistemas se controlan las funcionalidades que brindan y de cada una de ellas las acciones que se realizan. Una vez registrada la estructura de los sistemas se procede a la creación de roles, donde se le asignarán los permisos a los diferentes niveles de la misma, estos roles serán asignados a los usuarios dentro de una o varias entidades, de esta forma si la seguridad se maneja de forma centralizada el usuario podrá acceder a dichas entidades desde cualquier punto del país. Los usuarios cuentan con un perfil configurable donde puede seleccionar el tema, el idioma, el tipo de escritorio para su entorno de trabajo, además de estos datos se recoge el dominio de entidades al cual tiene acceso, servidor de autenticación si se cuenta con uno, rango de direcciones IP desde donde puede conectarse, usuario y contraseña.

En muchos casos, el problema de estas soluciones es su tecnología y adaptación a procesos específicos. En el caso de Acaxia, aunque su principio se basa en criterios mutil-objetivos, la

integración a aplicaciones que difiere de su tecnología se complejiza cuando se detallan los niveles de seguridad y se determinan las responsabilidades sobre los recursos que se manejan.

1.11. Herramientas y soluciones técnicas

1.11.1. Metodología de desarrollo

Como metodología de desarrollo de software se elige *Extreme Programming (XP)*[11]. La Programación Extrema es una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación, la realimentación o reutilización del código desarrollado y colaboración del cliente, permitiendo el desarrollo incremental del software con iteraciones muy cortas (Gerardo, 2002).

1.11.2. Elección sobre otras metodologías

XP es una metodología ágil capaz de obtener un software de calidad mediante iteraciones cortas y rápidas. Le permiten al cliente participar en conjunto con el equipo de trabajo durante el levantamiento de requisitos del negocio, con la particularidad de que estos puedan ser eliminado o modificados sin afectar el desarrollo del software. En generar la metodología *XP* crea un ambiente favorable dentro del grupo de trabajo manteniendo una buena comunicación interna que permiten la agilidad, eficacia y optimización del producto que se desarrolla, englobando las características más importantes entre las metodologías existentes, y no presenta una documentación extensa dentro de su modelo, lo que la convierte como una de las metodologías que más se utilizan en el desarrollo de software.

Con respecto a las metodologías tradicionales, como RUP y Métrica 3, se elige *XP* basándose en los siguientes aspectos:

Metodología *XP*	**Metodologías Tradicionales**
Basada en heurística provenientes de buenas prácticas de producción de código	Basa en heurísticas de estándares seguidos por el entorno de desarrollo
Preparada para cambios constantes	Preparado para la invariabilidad de requisitos
El cliente es parte del equipo de desarrollo	El cliente actúa como salida al proceso de desarrollo
Pocos artefactos de documentación y producción acelerada	Muchos artefactos de documentados y producción normada

Tabla 1.1 Comparación entre metodología *XP* y metodologías Tradicionales

[11] *XP*: Programación extrema.

1.11.3. Lenguaje de modelado

Se escoge el Lenguaje Unificado de Modelado *(UML)*[12] por ser un lenguaje gráfico que permite visualizar, especificar y documentar un sistema. Permite detallar determinados aspectos del sistema tales como la descripción del negocio, la especificación de métodos y procesos, así como el funcionamiento del sistema y los esquemas de bases de datos (Jacobson, y otros, 2000). Es uno de los lenguajes de modelado de sistemas de software más conocido y que más se utilizan por ser una excelente herramienta que brinda un amplio soporte a las metodologías durante el desarrollo de software.

1.11.4. IDE NetBeans

El *IDE*[13] NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Es una plataforma con una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes (Domínguez, 2005). Entre las ventajas que justifican su elección se encuentran:

- ✓ Administración de interfaces de usuario.
- ✓ Integración a múltiples *frameworks*.
- ✓ Administración de almacenamiento.
- ✓ Gran cantidad de módulos y extensiones para múltiples lenguajes y tecnologías.

1.11.5. Herramienta *CASE* Visual Paradigm

Es una herramienta *CASE*[14] para visualizar y diseñar elementos de software, para ello hace uso del lenguaje de modelado *UML*. Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Facilita la interoperabilidad con otras herramientas *CASE* como el *Rational Rose* y puede ser integrada con herramientas Java. Posee además gran facilidad de uso, acelera el desarrollo de aplicaciones, ya que sirve de puente visual entre arquitectos, analistas y diseñadores de sistemas de información, haciendo el trabajo más fácil y dinámico (Giraldo, y otros, 2005).

[12] *UML*: Siglas en inglés *Unified Modeling Language.*
[13] *IDE*: Siglas en inglés *Integration Development Environment* o Entorno de Desarrollo Integrado en español.
[14] *CASE*: de sus siglas en inglés *Computer Aided Software Engineering* o Ingeniería de Software Asistida por Computadora en español.

1.11.6. Framework

Un *framework* simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, un *framework* proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, un *framework* facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas (Gerner, y otros, 2006).

1.11.6.1. Ext JS Framework

Ext JS es una biblioteca Java Script, por lo que funciona sobre los navegadores web del lado del cliente, ligera y de alto rendimiento, compatible con la mayoría de navegadores que nos permite crear páginas e interfaces web dinámicas. Las opciones más comunes para su integración son PHP, Java, *.NET*[15] y muchos más. Ext contiene algunos adaptadores como Ajax, animación, manipulación dinámica del *DOM*[16] y otras características para la interacción con el usuario. Sus valores más destacados son la filosofía basada en *widgets*, lo que le brinda componentes predefinidos de fácil utilizació, y su orientación a *RIA*[17] (Barrios, 2007).

1.11.6.2. Symfony Framework

Symfony es un completo marco de trabajo diseñado para optimizar, gracias a sus características, al desarrollo de las aplicaciones web. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación (Potencier, y otros, 2008).

1.11.7. Servidor web 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, posee diversos módulos que permiten incorporarle nuevas funcionalidades (estos son muy

[15] *NET*: Framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.
[16] *DOM*: Biblioteca o framework de JavaScript que permite simplificar la manera de interactuar con los documentos HTML.
[17] *RIA*: Siglas del inglés *Rich Internet Applications* o Aplicaciones Enriquecidas de Internet en español.

simples de cargar) y es capaz de utilizar lenguajes como PHP, TCL, Python, entre otros (Portela González, y otros, 2010).

1.11.8. Base de datos

Una base de datos (BD) es un conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora. Una base de datos puede considerarse una colección de datos variables en el tiempo. El software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez, se denomina sistema de gestión de bases de datos (SGBD) (García, 1999).

1.11.8.1. PostgreSQL

Es un potente motor de bases de datos, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de bases de datos comerciales (PostgreSQL Development Group, 2009). Es tan completo como MySQL y superior a motores como SQlite y Firebird. PostgreSQL basándose en ventajas asociadas a: 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. Además, entre sus características destacan:

- ✓ Claves ajenas también denominadas Llaves ajenas o Llaves Foráneas.
- ✓ Disparadores.
- ✓ Implementación del estándar SQL92/SQL99.
- ✓ Vistas.
- ✓ Integridad transaccional.
- ✓ Acceso concurrente multi-versión (no se bloquean las tablas, ni siquiera las filas, cuando un proceso escribe).
- ✓ Capacidad de albergar programas en el servidor en varios lenguajes.
- ✓ Herencia de tablas.
- ✓ Licencia libre de pago.

1.11.8.2. EMS SQL Management

Es una solución completa para la administración de bases de datos y el desarrollo, es una herramienta de alto rendimiento para la administración de bases de datos, une a todos los componentes que hay que tener que se centran en las tareas de gestión de bases de datos más críticos en un entorno potente y fácil de usar que puede trabajar todo el día. *EMS*[18] SQL[19] *Management* ofrece todas las herramientas esenciales para hacer más productivo. Las versiones más comercializadas son para PostgreSQL y MySQL.

1.12. Conclusiones parciales

En el presente capítulo se ha realizado un estudio conciso sobre los diferentes modelos y esquemas respecto a la seguridad de la información de la sociedad moderna, haciendo énfasis en los principales conceptos y definiciones que se trazan las empresas en el presente con el fin de contar con herramientas que le permitan automatizar todas sus actividades para agilizar el negocio, detallándose normas y procedimientos para la creación de ambientes seguro, lo que permitió definir el marco teórico que fundamenta la presente investigación. Se realizó un estudio de las herramientas orientadas a la gran empresa y sus sistemas de seguridad, sirviendo de base para el análisis e identificación de propiedades y requisitos a considerar. Por último, se definieron las herramientas a utilizar, justificándose los valores que estas aportan al desarrollo de la solución propuesta.

[18] EMS: Del inglés *Environmental Management System* o Sistema de Gestión Ambiental.
[19] SQL: Del inglés *structured query language* o lenguaje de consulta estructurado.

Capítulo 2. Descripción del sistema

2.1. Introducción al capítulo

En este capítulo se realizará una descripción de las principales características del sistema para la gestión de los procesos y recursos del *EBMS* DSerp, así como las consideraciones necesarias que se plantean dentro de sus objetivos en la automatización de sus procesos. Se definirán los requisitos del proyecto, determinando las iteraciones necesarias para completarlo, así como la fecha de entrega de dichas iteraciones.

2.1. Consideraciones del negocio

Un sistema *EBMS* permite a partir de una arquitectura centralizada, flexible e interactiva, integrar y administrar procesos empresariales que garanticen la automatización de actividades en todas las áreas claves de un conjunto de empresas que tengan dependencias funcionales sustanciales, creando un dominio de negocios común para la gestión de información de un entorno corporativo (Kerton, 2012).

2.1.1. EBMS DSerp

Cuando se habla de un *EBMS* DSerp, se refiere a un proyecto pionero en la clase de sistemas *EBMS*, definidos como un dominio de sistemas estratégicos para la gestión avanzada de recursos y procesos. Un *EBMS* DSerp provee una plataforma con herramientas integradas para automatizar procesos de negocios, y su arquitectura de *EBMS* le ofrece adaptabilidad para desplegarse en cualquier ambiente empresarial, pudiéndose especificar sus procesos a actividades comerciales específicas. El sistema incluye herramientas para la gestión de contabilidad y finanzas, facturaciones, activos fijos, productos y almacenes, clientes y proveedores, recursos humanos, flujos productivos, auditorías y organización empresarial. Todos asistidos por bloques inteligentes para la consultoría estratégica, la toma de decisiones, análisis de datos, con mecanismos que incluyen sistemas de notificación y alarmas según las necesidades del cliente (Kerton, 2012).

2.1.2. Objeto de automatización

El sistema *EBMS* DSerp presenta un ambiente diseñado para la automatización de procesos y recursos que se gestionan en las empresas. Como parte de su estructura, contiene el módulo DSerp Rector, herramienta para la gestión sobre los valores constantes del sistema, configuración de módulos, personalización del sistema de comunicación integrada, normas de notificación y grado de automatización.

Producto del manejo constante de la información existente que se manipula dentro del sistema, que influyen directamente con la administración de las empresas en la gestión de sus procesos y recursos imprescindibles para el negocio, se hace necesario la protección de los datos así como el acceso de los mismos mediantes diferentes mecanismos o niveles personalizables.

Por estos motivos, se tiene como objeto de automatización, la creación de un ambiente de gestión de seguridad que controle el acceso al sistema basados en una políticas definidas por los administradores del sistema, que su vez cumpla con el concepto de seguridad informática de multidimensional, capaz de asignar diferentes tipos de permisos de acceso según las necesidades propias de la plataforma, así como el seguimiento sistemáticos de las acciones realizadas por los usuarios dentro de la misma.

2.2. Información que se manipula

Es toda la información que se genera en una empresa, los mismos se interpretan de la siguiente forma:

Información contable: Procesos y datos que tienen que ver con los recursos de la empresa.

Información operativa: Información transaccional o el flujo informativo de la empresa.

Usuarios: Agentes o actores que se conectan al sistema.

Roles: Agrupan un conjunto de operaciones o procesos vinculados a los usuarios del sistema.

Restricciones: Limita las actividades de los servicios en las operaciones del sistema.

2.3. Propuesta del sistema

Se propone módulo de seguridad para el subsistema DSerp Rector, que mediante una interfaz permita definir y configurar usuarios y roles, permitiendo delimitar el nivel de acceso a módulos y escenarios de la aplicación, además de realizar el monitoreo de las trazas del sistema, lo que posibilitará controlar el comportamiento de los usuarios dentro del *EBMS* DSerp. Este ambiente servirá como método para garantizar el correcto funcionamiento de la aplicación y que esta a su vez contenga una estructura robusta y segura sobre la información manipulada, además de regular el acceso a los recursos atendiendo a las necesidades de seguridad de la empresa. Se implementa con una arquitectura basada en capas, en la cual se podrá delimitar los diferentes aspectos del negocio, las actividades automatizadas y la forma en que se comunica la información dentro del sistema.

2.4. Exploración y Planificación

Utilizando *XP* como metodología, en esta fase los clientes plantean sus necesidades a través de historias de usuario que son realizadas por el propio cliente las cuales describen las características y

funcionalidad del software que se va a elaborar y se definen cuáles son las prioridades de estas. Los clientes y desarrolladores trabajan juntos para decidir cómo agrupar las mismas. A medida que avanza el trabajo, el cliente puede agregar historias, cambiar el valor de una ya existente, descomponerlas o eliminarlas. Entonces, el equipo de trabajo reconsidera todas las entregas faltantes y modifica sus planes en consecuencia. Al mismo tiempo, el equipo estima el tiempo de desarrollo de las historias de usuario y se familiariza con las herramientas, tecnologías y prácticas que se utilizará en el proyecto. Se prueba la tecnología y se explorar las posibilidades arquitectónicas del sistema. Esta fase debe ser corta, dependiendo del alcance del proyecto y las capacidades del equipo de desarrollo.

2.4.1. Historias de Usuario

Una historia de usuario es una representación de un requisito de software utilizando el lenguaje común del usuario. Las historias de usuario son utilizadas en las metodologías de desarrollo ágiles para la especificación de requisitos. Las historias de usuario son una forma rápida de administrar los requisitos del cliente sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Debe ser lo suficientemente sencilla como para que el cliente pueda comprenderla y además el programador debe saber que implementar y poder llevar una traza de su trabajo sobre esta funcionalidad. Si cuando el cliente escribe la historia de usuario, el programador entiende que no es lo completamente sencilla como para implementarla como una funcionalidad atómica, entonces se divide en dos o más historias. Las historias de usuario del sistema propuesto se escribieron con el mismo formato de las descritas en el proyecto *EBMS* DSerp, asegurando una compatibilidad y estandarización para el equipo de desarrollo y los clientes. Estas historias serán de la siguiente forma:

HISTORIA DE USUARIO			
Código:	Rector-01	**Nombre:**	Gestionar Usuario
Usuario:	Cliente	**Actividad:**	Gestión de datos
Riesgo:	Bajo	**Prioridad:**	Alta
Iteración:	1	**Puntos estimados:**	2
Descripción:	Realiza las operaciones crear, eliminar y cambiar contraseña de los usuarios del sistema.		

Tabla 2.1: Ejemplo de historia de usuario.

Campos de la historia de usuario:

Código. El código estará definido en dos partes. La primera será el módulo o subsistema al que pertenece la historia de usuario. Uno de los siguientes:

- Vitae.
- Gabinete.
- Terra.
- Rector.
- Mensaje.
- Germina.
- Labranza.
- Camarada.
- Hacienda.
- Sumario.
- Caudal.

La segunda parte será el número de la funcionalidad que representa.

Nombre: El nombre será la funcionalidad que se satisface, y también responderá al nombre de la acción que se registra en la base de datos.

Usuario: Define si la funcionalidad es iniciada por un el sistema o un usuario externo.

Actividad: La actividad responderá a uno de los posibles sucesos que se registran en la base de datos:

- Ver/ Listar datos.
- Gestionar datos.
- Proceso de relación.
- Exportar datos.
- Proceso interno.
- Importar datos.
- Mensaje de alerta.
- Mensaje de error.
- Inicio de sesión.
- Cierre de sesión.
- Permiso denegado.

Riesgo: Es el grado de riesgo en el desarrollo que se asocia a la historia de usuario. Determina la posibilidad real de implementarse o no con las condiciones previstas por el equipo de desarrollo (tiempo, recursos, personal). Puede ser Bajo, Medio o Alto.

Prioridad: La prioridad la define el cliente, y es el grado de importancia que le concede a la funcionalidad. Aquellas historias de prioridad alta deberían ser las primeras en implementarse.

Iteración: Es el número de la fase o iteración en la cual se define la historia de usuario.

Puntos estimados: Es un número entero que representa la cantidad de semanas que se supone para el desarrollo de la tarea de usuario. Según las consideraciones del equipo de proyecto, las historias con altos puntos estimados deben ser separadas en varias tareas. Un punto es una semana efectiva de desarrollo.

Descripción: Se escribe una pequeña descripción de lo que hace la funcionalidad.

En la fase de exploración se encontraron las siguientes necesidades funcionales:

1. Gestionar roles.
2. Gestionar usuario.
3. Autenticar usuario.
4. Comprobar permisos.
5. Gestionar restricciones de tiempo.
6. Gestionar direcciones de IP.
7. Generar trazas del Sistema.
8. Visualizar trazas del sistema.

Argumentadas en las siguientes historias de usuario:

HISTORIA DE USUARIO			
Código:	Rector-01	**Nombre:**	Gestionar roles
Usuario:	Usuario	**Actividad:**	Gestión de datos
Riesgo:	Bajo	**Prioridad:**	Alta
Iteración:	1	**Puntos estimados:**	2
Descripción:	Realiza las operaciones de crear, modificar y eliminar Roles a los usuarios del sistema.		

Tabla 2.2 Historia de usuario: Gestionar roles.

HISTORIA DE USUARIO			
Código:	Rector-02	**Nombre:**	Gestionar usuario
Usuario:	Usuario	**Actividad:**	Gestión de datos
Riesgo:	Bajo	**Prioridad:**	Alta
Iteración:	1	**Puntos estimados:**	2
Descripción:	Realiza las operaciones crear, eliminar y cambiar contraseña de los usuarios del sistema.		

Tabla 2.3 Historia de usuario: Gestionar usuario.

HISTORIA DE USUARIO			
Código:	Rector-03	**Nombre:**	Autenticar usuario
Usuario:	Usuario	**Actividad:**	Proceso interno
Riesgo:	Bajo	**Prioridad:**	Alta

Iteración:	1	**Puntos estimados:**	1
Descripción:	Realiza la operación de la entrada de un usuario al sistema.		

Tabla 2.4 Historia de usuario: Autenticar usuario.

HISTORIA DE USUARIO			
Código:	Rector-04	**Nombre:**	Gestionar restricciones de tiempo
Usuario:	Usuario	**Actividad:**	Gestión de datos
Riesgo:	Bajo	**Prioridad:**	Alta
Iteración:	2	**Puntos estimados:**	3
Descripción:	Realiza la operación de restricción por lotes de tiempo a los usuarios respecto a la entrada al sistema.		

Tabla 2.5 Historia de usuario: Gestionar restricciones de tiempo.

HISTORIA DE USUARIO			
Código:	Rector-05	**Nombre:**	Comprobar permisos
Usuario:	Sistema	**Actividad:**	Proceso interno
Riesgo:	Bajo	**Prioridad:**	Alta
Iteración:	1	**Puntos estimados:**	2
Descripción:	Realiza la operación de verificación de permisos que tenga el usuario en el sistema.		

Tabla 2.6 Historia de usuario: Comprobar permisos.

HISTORIA DE USUARIO			
Código:	Rector-06	**Nombre:**	Gestionar restricciones de IP
Usuario:	Usuario	**Actividad:**	Gestión de datos
Riesgo:	Bajo	**Prioridad:**	Alta
Iteración:	2	**Puntos estimados:**	2
Descripción:	Realiza la operación de restricciones de la entrada al sistema por acceso a redes.		

Tabla 2.7 Historia de usuario: Gestionar restricciones de IP.

HISTORIA DE USUARIO			
Código:	Rector-07	**Nombre:**	Generar Trazas del Sistema
Usuario:	Sistema	**Actividad:**	Proceso interno
Riesgo:	Bajo	**Prioridad:**	Alta
Iteración:	2	**Puntos estimados:**	1
Descripción:	Realiza la operación de seguimiento de las acciones realizadas por los usuarios		

	en el sistema.

Tabla 2.8 Historia de usuario: Generar Trazas del Sistema.

HISTORIA DE USUARIO			
Código:	Rector-08	**Nombre:**	Visualizar trazas del sistema
Usuario:	Usuario	**Actividad:**	Proceso interno
Riesgo:	Bajo	**Prioridad:**	Alta
Iteración:	2	**Puntos estimados:**	1
Descripción:	Visualiza el seguimiento de las acciones realizadas por los usuarios en el sistema.		

Tabla 2.9 Historia de usuario: Visualizar trazas del sistema.

2.4.2. Requerimientos no funcionales

Los requerimientos funcionales son propiedades o cualidades que el producto debe tener, los cuales garantiza que un producto tenga una alta calidad en cuanto a eficacia y eficiencia. Aunque estos requisitos no definen el éxito del producto, influyen considerablemente en la evaluación del mismo. Teniendo en cuenta las exigencias del cliente, las características de la plataforma *EBMS* DSerp y las capacidades tecnológicas que debe el sistema se definieron los siguientes requerimientos no funcionales:

Usabilidad:

El sistema será utilizado por todas las personas que manipulen la información, y que tengan permisos para operar sobre los diferentes escenarios de gestión de los módulos de la plataforma. Estos usuarios deben poseer conocimientos informáticos elementales. Las opciones principales para la manipulación de roles, usuarios y restricciones serán visibles y accesibles.

Seguridad:

El ambiente de seguridad debe estar basado en los derechos de los usuarios sobre la administración del sistema, basándose en los principios de:

- ✓ **Integridad.** La información generada debe ser consistente y protegida contra alteraciones de cualquier tipo. Durante la manipulación de informaciones de los procesos, éstas no pueden ser alteradas.
- ✓ **Disponibilidad.** El sistema deberá estar disponible durante todo el tiempo laboral, y permitir el acceso desde todas las áreas de la empresa para dar soporte a los procesos de decisión especializado y en grupo al personal autorizado para ello.

- ✓ **Confidencialidad.** La información debe responder a los permisos laborales de los usuarios que utilizan el sistema dejando constancia de las operaciones y funciones de dichos usuarios sobre los recursos y procesos que se manipulan.

Interfaces externas:

La interfaz debe ser amigable, con las funciones elementales visibles en todo momento, mínimas distracciones, utilizando bloques organizados para mostrar la información según su relevancia, interactiva y con elementos gráficos que ayuden a identificar las funcionalidades rápidamente.

Rendimiento:

La aplicación debe ejecutarse utilizando eficientemente los recursos de software y hardware, y además debe asegurarse que los tiempos de respuesta a las diferentes peticiones de los usuarios sea el menor posible.

Software:

- Servidor Web Apache 2.2 o superior.
- Servidor de Base de Datos PostgreSQL 8.4.2 o superior.
- Navegador Web Mozilla Firefox , Internet Explorer, Google Chrome, Opera
- PHP versión 5.x

Hardware:

- Máquina Servidor:
  - Procesador Intel/AMD, 2.4 HGz o superior.
  - Memoria RAM: 2 Gb.
  - Disco duro: 500 Gb o superior.
  - Tarjeta de Red o módem.
- Cliente:
  - Procesador Intel/AMD, 1.0 HGz o superior.
  - Memoria RAM: 64 Mb o superior.
  - Disco duro: 3 GB o superior.
  - Tarjeta de red o módem.

2.4.3. Estimación de esfuerzo por historias de usuario

La estimación de esfuerzo de usuario en relación con las historias de usuarios que se definieron, permite una planificación concreta de los objetivos propuestos en el trabajo para un control cronológico de la culminación del proyecto basándose en la suma de puntos correspondientes a las historias de usuario, donde un punto estimado equivale a una semana de desarrollo.

Historia de usuario	Estimación de esfuerzo
Gestionar roles	2
Gestionar usuario	2
Autenticar usuario	1
Comprobar permisos	2
Gestionar restricciones de tiempo	3
Gestionar direcciones de IP	2
Generar trazas del sistema	1
Visualizar trazas del sistema	1

Tabla 2.11 Estimación de esfuerzo por historias de usuario.

2.4.4. Plan de iteraciones

Para proveer un desarrollo iterativo e incremental y asegurar la organización del trabajo, se crea el plan de iteraciones donde se planifica el orden de desarrollo de las historias de usuario. Se definió realizar dos iteraciones, clasificando las historias de usuario según su prioridad y grado de afectación al sistema. La duración total de cada iteración dependerá de los puntos estimados de las historias que en él se desarrollan. Este plan de iteraciones será de obligatorio cumplimiento, y está normalizado según el modelo de desarrollo e implementación del proyecto *EBMS* DSerp.

Iteración # 1:

Se implementarán los procesos de configuración y gestión que sirven de base para el correcto procedimiento de la seguridad dentro del sistema.

Iteración # 2

Se implementarán los procesos que permitirán la verificación y visualización de los procedimientos de la seguridad en el sistema.

Iteración	Historia de usuario	Duración	
Iteración #1	Gestionar roles	2	7
	Gestionar usuario	2	
	Autenticar usuario	1	
	Comprobar permisos	2	
Iteración #2	Gestionar restricciones de tiempo	3	7
	Gestionar direcciones de IP	2	
	Generar trazas del sistema	1	
	Visualizar trazas del sistema	1	

Tabla 2.12 Plan de iteraciones.

2.4.5. Plan de entregas

En la siguiente tabla se representa el plan de entrega donde se incluyen las fechas de culminación de las iteraciones y sus correspondientes historias de usuario.

Iteración	Iteración #1	Iteración #2
Cantidad de historias de usuario	4	5
Fecha de entrega	20/12/2012	25/02/2013

Tabla 2.13 Plan de entrega

2.5. Conclusiones parciales

En el presente capítulo se profundizó en la descripción del sistema, lo que contribuyó a identificar las necesidades que se planteaban permitiendo automatizar la seguridad del sistema. Se pudieron identificar los requerimientos para el sistema, argumentados en 9 historias usuarios, lo que permitió definir las necesidades funcionales dentro de la misma, contribuyendo así al control del desarrollo de las actividades dentro del equipo de trabajo. La creación de los planes de iteración y entrega, permitieron normar el ciclo de vida del proyecto y controlar el proceso incremental e iterativo propuesto para su desarrollo.

Capítulo 3. Descripción de la solución

3.1. Introducción al capítulo

En este capítulo se describirá la solución mediantes técnicas y procesos que permitirán dar solución a la problemática dada. Se argumentará cuáles son los patrones de diseño que sirvieron de apoyo a la realización de un software funcional. Se explica el modelo de dato del sistema y el diseño del mismo como soporte en la seguridad del *EBMS* Dserp.

3.2. Elección de patrones

Un patrón de diseño es una solución a un problema de diseño cuyo objetivo fundamental es:

- ✓ Reutilizar diseños abstractos que no incluyan detalles de la implementación.
- ✓ Un patrón es una descripción del problema y la esencia de su solución, que se puede reutilizar en casos distintos.
- ✓ Es una solución adecuada a un problema común.
- ✓ Asociado a orientación a objetos, pero el principio general es aplicable a todos los enfoques de diseño software.

Dentro del marco de trabajo seguido durante el desarrollo del software por el equipo de trabajo, se utiliza el *framework* Symfony, que por sus características, traen consigo una serie de patrones que proponen la solución al problema de diseño de la presente investigación, estos son:

3.2.1.1. Patrones GOF

***Singleton*:** Este patrón se implementa mediante la clase *sfRouting* y la variable global dserp. El *sfRouting* contiene el método *getInstance()* que ejecuta todas las peticiones que se realizan a la aplicación a través del controlador *sfWebFrontController*. Mientras que la variable global dserp, se comporta como instancia única del sistema que se encarga de construir todas las clases, las funciones y acciones dentro de las vistas generadas por *Ext JS*.

Comando: Este patrón se manifiesta a través de la clase *sfWebController*, encargada de establecer el módulo y la acción que se usará por cada petición del usuario.

Decorador: Utilizado en la clase *sfView*, padre de todas las vistas y que permite agregar funcionalidades dinámicamente.

Registro: Este patrón se aplica en la clase *sfConfig*, que es la encargada de acumular todas las variables de uso global en el sistema, es una manera fácil y rápida de compartir datos y objetos en la aplicación, permitiendo que no sea necesario conservar muchos parámetros y reduce el uso de variables globales.

3.2.1.2. Patrones GRASP

Experto: Este patrón se introduce mediante el uso de la librería Propel para mapear la Base de Datos. Esta librería, es usada en la capa del modelo, maneja toda la lógica de los datos, generando clases a partir del modelo racional con todas las funcionalidades necesarias para el desarrollo orientado a objetos.

Creador: En la clase *Actions*, se evidencia dicho patrón, en la misma se encuentran las acciones que crean los objetos de las clases que representan las entidades.

Alta cohesión: El uso del *framework* Symfony provee mediante una alta cohesión para la asignación de responsabilidades y el empaquetamiento de archivos.

Bajo acoplamiento: Este patrón se evidencia durante la creación de objetos de las entidades con la clase *Actions*, que hereda únicamente del controlador *sfActions*, alcanzando un bajo acoplamiento de las clases. El modelo de tres capas además abstrae la vista y el controlador del modelo, proporcionado una baja dependencia.

Controlador: Todas las peticiones desde la web son manipuladas por un solo controlador: *sfActions*, que es el punto de entrada único para la aplicación dentro de un determinado entorno. Este patrón se evidencia en las clases *sfFrontController*, *sfWebFrontController* y *sfContext*.

3.2.1.3. Patrón Modelo - Vista - Controlador

El patrón Modelo-Vista-Controlador tiene como idea básica separar al sistema en tres capas: el modelo, la vista y el controlador.

El **modelo:** se encarga de la representación específica de la información con la que trabaja el sistema, de las operaciones y persistencia sobre los datos. Guarda y recupera información del medio persistente, ya sean base de datos o ficheros de textos.

La **vista:** presenta la información obtenida con el modelo, en un formato adecuado, que pueda interactuar y ser visible al usuario.

El **controlador:** es la capa intermedia, que responde a múltiples eventos dentro del sistema, se encarga de implementar toda la capa de negocio.

3.3. Diseño de bases de datos

Para el desarrollo del sistema se propuso el siguiente diseño de base de datos, el mismo contiene toda la información necesaria para realizar todo el proceso de la seguridad del sistema *EBMS* DSerp.

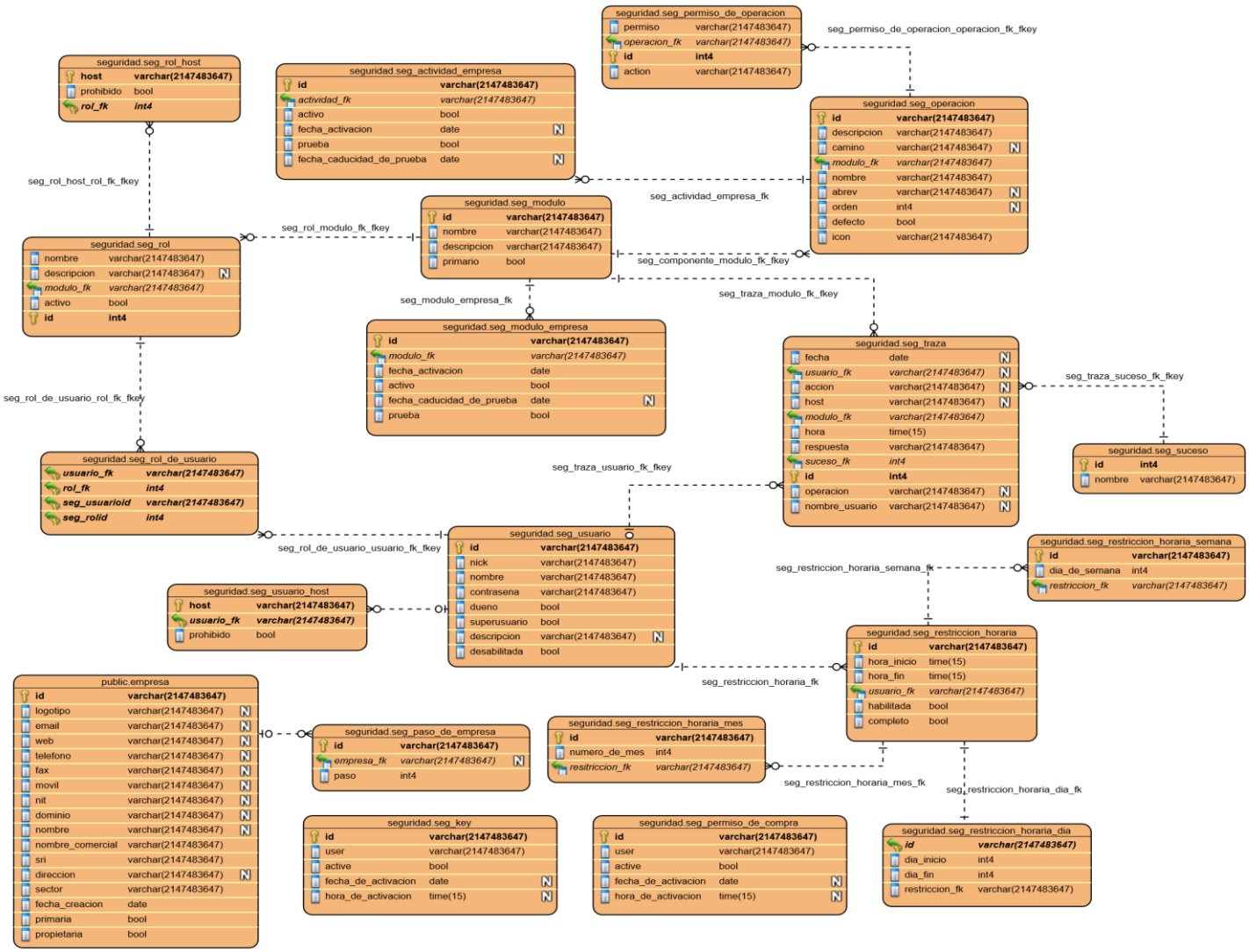

Figura 3.1 Diagrama Entidad-Relación

Descripción de las tablas:

Tabla	Módulo	Descripción
seg_usuario	DSerp Rector	Registra los datos generales de todos los usuarios que pueden interactuar con el sistema.
seg_usuario_host	DSerp Rector	Registra las direcciones de redes desde donde los usuarios pueden acceder al sistema.
seg_rol_de_usuario	DSerp Rector	Relaciona al usuario con uno o más roles definido en el sistema.
seg_restricion_horaria_semana	DSerp Rector	Define los días de la semana en los que no se puede acceder al sistema.
seg_restricion_horaria_mes	DSerp Rector	Define los meses en los que no se puede acceder al sistema.
seg_restricion_horaria	DSerp Rector	Define las horas del día en los que no se puede acceder al sistema.
seg_restricion_horaria_dia	DSerp Rector	Define los períodos de tiempo en los que

		no se puede acceder al sistema.
seg_operacion	DSerp Rector	Define las operaciones que se realizan y la ubicación que tendrán dentro del sistema.
seg_permiso_de_operacion	DSerp Rector	Define el permiso que tendrá cada operación para ser ejecutada.
seg_permiso_de_rol	DSerp Rector	Define los permisos que puede tener un rol dentro del sistema.
seg_modulo	DSerp Rector	Registra los datos de los módulos que tendrá el sistema.
seg_usuario_modulo	DSerp Rector	Define a que módulo puede acceder cada usuario.
seg_traza	DSerp Rector	Registra todas las actividades que se realizan en el sistema.
seg_actividad_empresa	DSerp Rector	Define la actividad que realiza la empresa en el sistema (prueba o activación por compra)
seg_modulo_empresa	DSerp Rector	Define la actividad que realiza la empresa en cada módulo del sistema (prueba o activación por compra)
seg_suceso	DSerp Rector	Registra los tipos de acciones que se pueden realizar en el sistema.
seg_key	DSerp Rector	Registra las llaves de los usuarios para poder inicializar empresas.
seg_paso_de_empresa	DSerp Rector	Registra los datos correspondientes a la configuración de una empresa. En caso de que dicha configuración no se termine, esta entidad registra el *paso* por donde se quedó el usuario para luego continuar desde este.
seg_permiso_de_compra	DSerp Rector	Registra los datos correspondientes a la compra del sistema por una empresa.

Tabla 3.1 Descripción de las tablas del diagrama Entidad-Relación

3.4. Diagrama de clases

A partir del modelo de datos establecido se define el siguiente diagrama de clases:

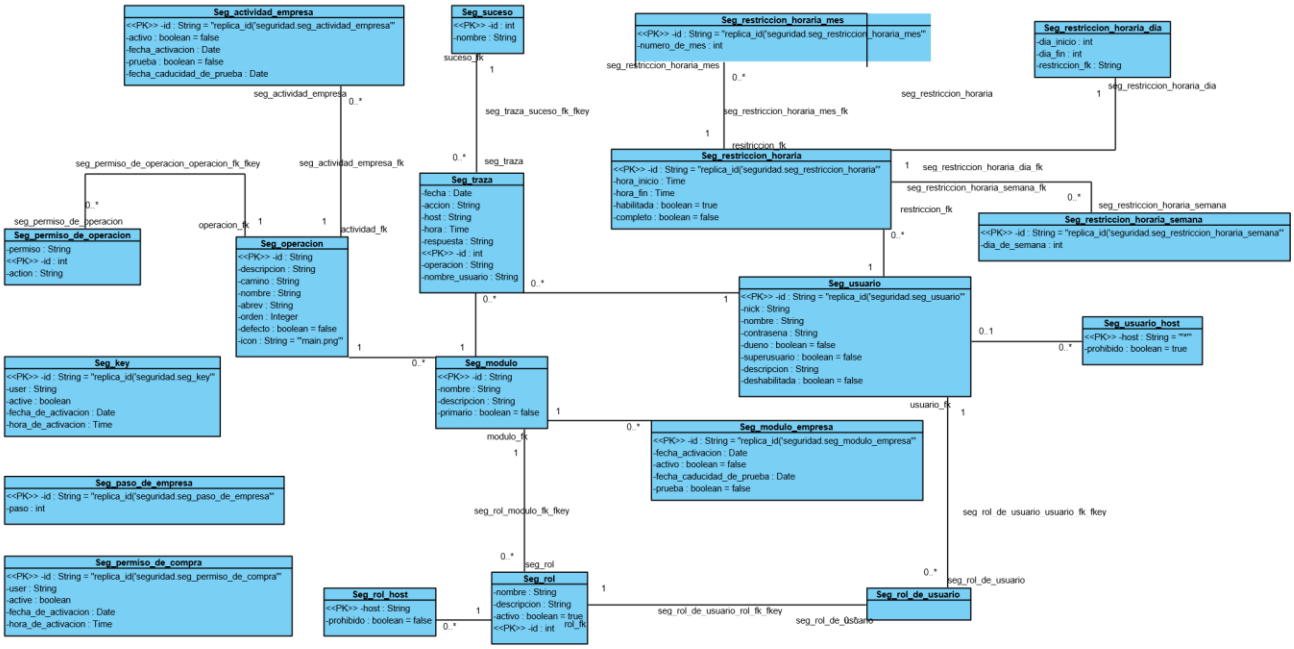

Figura 3.2 Diagrama de clases.

3.5. Diagrama de componente

En el siguiente diagrama se expresa como quedará desplegada la solución:

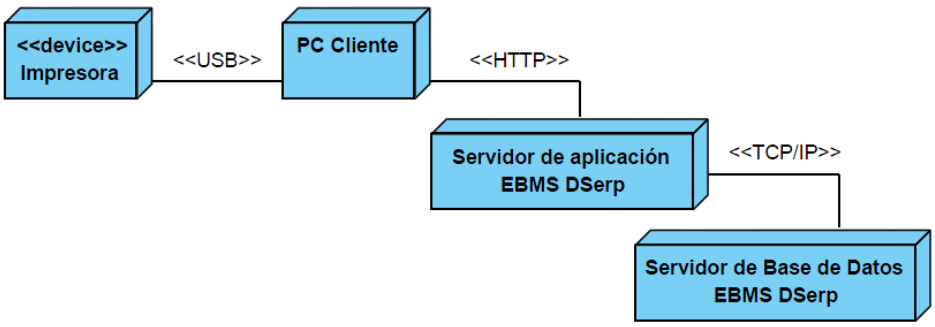

Figura 3.3 Diagrama de despliegue del *EBMS* DSerp.

3.6. Arquitectura del sistema

La arquitectura del módulo de seguridad del *EBMS* DSerp responde a la arquitectura del modelo *EBMS*, y se describe de la siguiente forma.

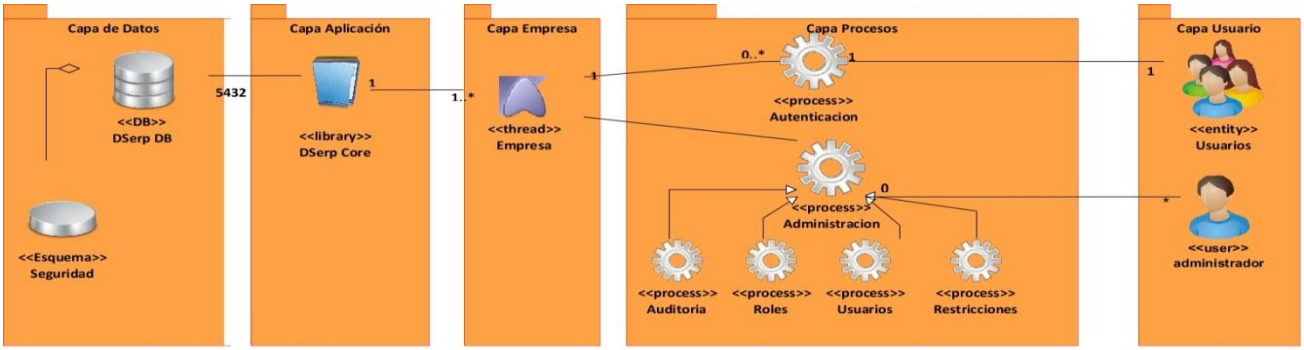

Figura 3.3 Arquitectura del sistema

Capa de datos: En la base de datos del *EBMS* DSerp, se encuentra el esquema seguridad. Este esquema tiene todas las entidades necesarias para garantizar la salvaguarda de la información. Estas entidades están enlazadas a la entidad empresa y persona del esquema *public*, lo que permite establecer la relación usuario-rol-persona-empresa.

Capa de aplicación: El *EBMS* DSerp es una aplicación evolutiva a partir su núcleo, por lo que todas las políticas de seguridad están implementadas dentro de este. El DSerp *Core* contiene todos los mecanismos de seguridad de la plataforma, a partir de las cuales el sistema puede especificarse en otras aplicaciones, heredando aquellas políticas que considere convenientes. En esta capa se construyen las operaciones para realizar la seguridad del *EBMS* DSerp.

Capa de empresa: Una empresa no es más que un canal o hilo de conexión de los procesos a la aplicación. Cuando un usuario se conecta, se abre una conexión de la empresa asignada, mediante la cual el sistema conoce que empresa está conectada y determina los recursos y procesos que se debe disponer para la empresa conectada; de esta forma los usuarios se conectan a los procesos utilizando sus credenciales mediante el canal habilitado por la empresa, permitiendo que el sistema atienda múltiples solicitudes de forma simultánea.

Capa de procesos. El módulo de seguridad del *EBMS* DSerp tiene dos procesos fundamentales, la autenticación y la administración.

La autenticación inicia desde que el usuario se conecta al sistema, determinando sus permisos sobre la aplicación. Durante cada acción del usuario dentro de la plataforma, sus credenciales son autenticadas para verificar sus derechos sobre dichas peticiones, convirtiendo la autenticación en un proceso constante mientras el usuario permanece conectado.

La administración es el proceso realizado por los encargados de garantizar la seguridad de la empresa y se divide en cuatro subprocesos:

- ✓ Auditoría: consiste en el control y análisis de acciones sobre el sistema. Este ambiente debe contener alto nivel de detalle y permitir a los administradores detectar todas las acciones de los usuarios, determinar violaciones de seguridad, realizar gestión de tiempo, filtrado según estados y condiciones y generar reportes.
- ✓ Roles: gestión de los roles de usuario. En este ambiente se determinar sobre que módulos actúan los roles y los derechos sobre estos.
- ✓ Usuarios. Gestión de los usuarios del sistema y asignación a los roles correspondientes.
- ✓ Restricciones. Las restricciones son los parámetros de uso según direcciones de acceso (*host*) y tiempo que tienen los usuarios.

Todos estos procesos serán detallados en el siguiente epígrafe.

Capa de usuarios: Finalmente, los usuarios son los agentes que se conectan a la aplicación. Aquellos con el rol de administradores son los únicos que evidentemente tienen acceso al proceso de administración del sistema.

3.7. Mecanismos de seguridad del EBMS DSerp.

Para el *EBMS* DSerp se propone una defensa escalonada, a partir de criterios de modularidad del sistema y la distribución de funcionalidades de los módulos. De esta forma, se establecen varias capas que dan seguridad a la información en varios niveles, creando un sistema robusto, para garantizar el cumplimiento de los principios de seguridad que deben estar presente en sistemas de información empresarial.

3.7.1. Creación de empresas

Para comprender en sistema de seguridad de la plataforma DSerp, es necesario conocer el proceso de creación de empresas de éste, ya que es una aplicación multiempresa, que da soporte a que varias empresas accedan a sus recursos.

Cuando una empresa solicita adicionarse al dominio, se valida la solicitud analizando si el usuario que la emite previamente se ha registrado como un interesado en su adición al dominio. Las solicitudes son almacenadas en la entidad seg_key, que contiene una llave alfanumérica y el correo del usuario. Esta entidad es llenada mediante agentes externos (página web promocional de la empresa, intranet, servicio móvil, canal telefónico, u otros.). Cuando el usuario valida su permiso a adicionar la empresa al sistema, pasa al proceso de definición de su negocio. Dicho proceso puede ser abortado en cualquier momento, el sistema guardará el estado en que se quedó el usuario, utilizando la entidad seg_paso_de_empresa, lo que le permite al sistema conocer en qué punto del proceso de creación está la nueva empresa. Durante este proceso, se crea un usuario con permisos plenos de administración, que será el administrador de seguridad de la empresa que se crea, detallándolo en la entidad seg_usuario, y asignándole la empresa correspondiente.

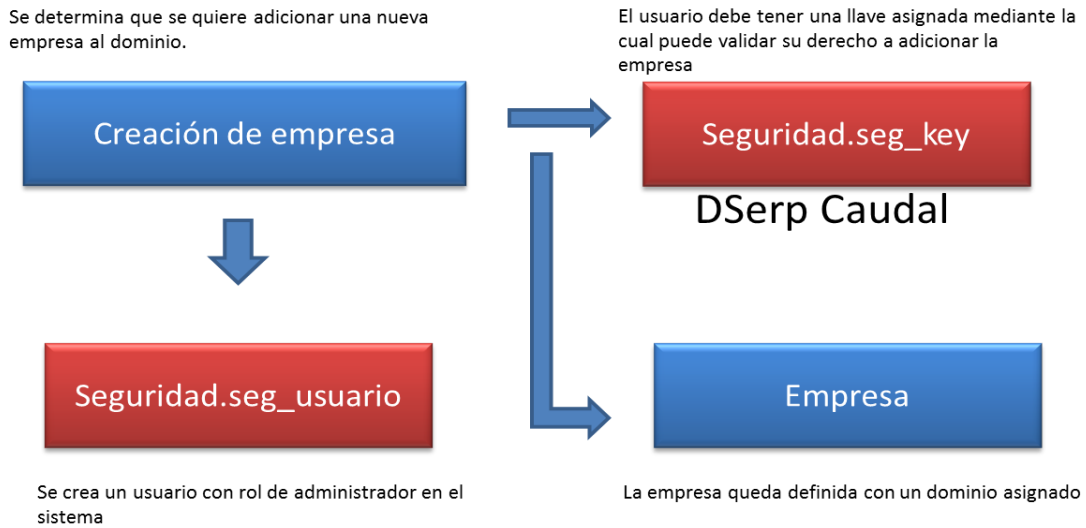

Figura 3.4 Creación de empresas

3.7.2. Autenticación

El proceso de autenticación definido para el sistema *EBMS* DSerp es el siguiente:

Cuando un usuario va autenticarse, ingresará el Nick del usuario y el dominio al que pertenece, con el formato user@dominio. El dominio es un identificador único de cada empresa, dando al sistema la posibilidad de saber qué empresa está abriendo un nuevo canal de conexión. En caso de que la empresa exista mediante el dominio insertado, se procede a acceder al módulo de seguridad, y se verifica el *Nick* y la contraseña introducidas, comparándolas con las de la entidad seg_usuario. Todas las contraseñas son guardadas utilizando un procedimiento irreversible MD5, eliminando las posibilidades de obtener directamente la contraseña desde la base de datos. Si los datos son correctos, se verifican las restricciones del usuario, haciendo uso del método verificarPermiso. Un usuario puede tener restricciones de tiempo o de host, o puede tener deshabilitada su cuenta de acceso.

Una restricción de tiempo es una limitación de uso condicional que establece los horarios o días que el usuario no puede acceder al sistema, Esta condición puede estar determinada por un rango de tiempo (horas y minutos), días de la semana (Lunes, Martes, u otros.) o meses del año (Enero, Marzo, u otros.) o la combinación de cualquiera de estos elementos (De 9.00 AM a 11.45 AM, los Lunes y Jueves de los meses Abril y Junio). Esto permite que los administradores puedan relacionar directamente los horarios laborales, bajas, vacaciones, viajes u otras condiciones con el acceso a la plataforma, elevando su nivel de seguridad.

Las restricciones por *host* determinan cuales son las direcciones IP desde las cuales no se puede acceder al sistema. Los administradores también pueden establecer rangos de IP o direcciones unitarias, tanto de acceso como de bloqueo, garantizando que cada usuario pueda tener direcciones específicas de acceso.

Todas estas operaciones están definidas en la clase del modelo SegUsuario, manejada a través del patrón Experto del componente Propel de Synfomy.

Cuando las restricciones son validadas y en caso de ser afirmativos los permisos, utilizando las funciones de SegUsuario definidas en el DSerp *Core*, se determinan los módulos a los que el usuario tiene acceso, y dentro de estos, los procesos correspondientes, mediante una política de permisos heredados.

Figura 3.5 Proceso de autenticación

3.7.3. Determinación de permisos sobre operaciones

Un usuario podrá acceder a un módulo mediante el proceso de autenticación, sin embargo, es necesario determinar dentro de dicho módulo, las operaciones (procesos) correspondientes.

Cuando un usuario accede a un módulo, el sistema hace una petición para obtener todas las operaciones a las que el usuario debe acceder. En este procedimiento, se verifican en primera instancia cuales son las operaciones adquiridas por la empresa. Como un módulo puede tener múltiples operaciones, especificadas en diversos sectores, la empresa puede determinar cuáles de estos procesos son de su interés, esto se realiza mediante el módulo de adquisición de funcionalidades del *EBMS* DSerp. Las operaciones se validan teniendo en cuenta aquellas que son primarias (comunes para todas las empresas). Esta función, definida utilizando el patrón controlador y experto del sistema, devuelve una lista de funciones que son comparadas con las establecidas para el rol que representa al usuario, aquellas cuyos resultados son afirmativos son devueltas al usuario y se les permite su acceso. Dentro de una operación, también pueden existir varios permisos. Un permiso es una acción específica y unitaria en la mayoría de los casos (Adicionar, Eliminar, Modificar, Asentar, Exportar, u otros.). Todos los usuarios que acceden a una operación puede que no compartan los mismo derechos, y estos se restringen con la función getOperacionesPermisos del componente SegUsuario.

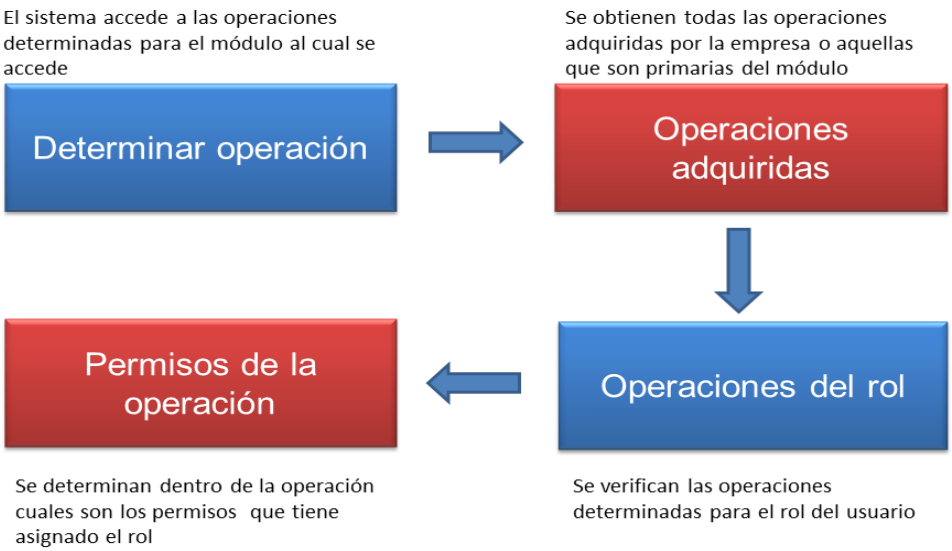

Figura 3.6 Proceso de asignar permisos sobre operaciones

Con este modelo, el sistema de seguridad adquiere una profundidad de 4 capas.

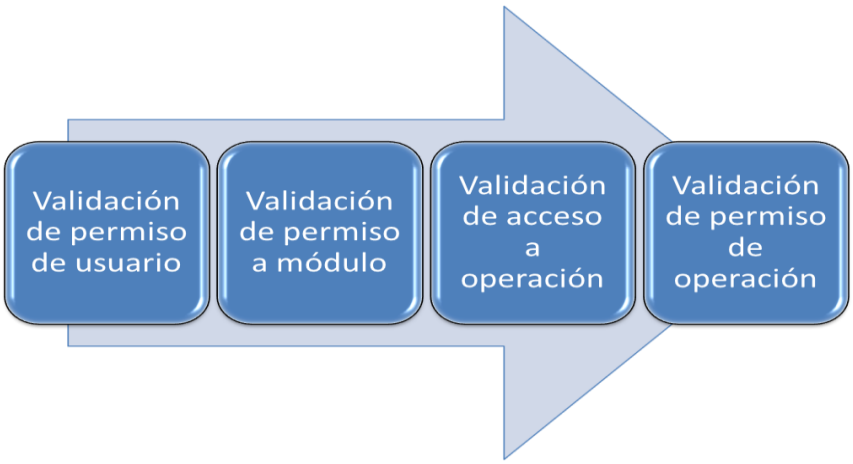

Figura 3.7 Capas de la seguridad del sistema

3.7.4. Gestión de roles de usuario

Cuando se define un rol se establece su nombre y descripción; a un rol se le asigna un módulo, y dentro del módulo se determinan las operaciones correspondientes, y por cada operación los permisos deseados. Un usuario puede tener varios roles, haciendo flexible el sistema. Todas estas definiciones son guardadas en sus entidades correspondientes dentro del modelo de datos y son las que permiten realizar la gestión de la seguridad.

3.8. Componentes base de la seguridad.

Para poder realizar toda esta gestión de seguridad hay varios componentes que regulan las actividades dentro del sistema.

3.8.1. Componentes expertos

Los componentes expertos son dos clases definidas dentro del modelo, que a través del patrón experto que hace uso de la librería Propel. Se utilizan para regular todas las operaciones dentro del sistema. Los componentes son:

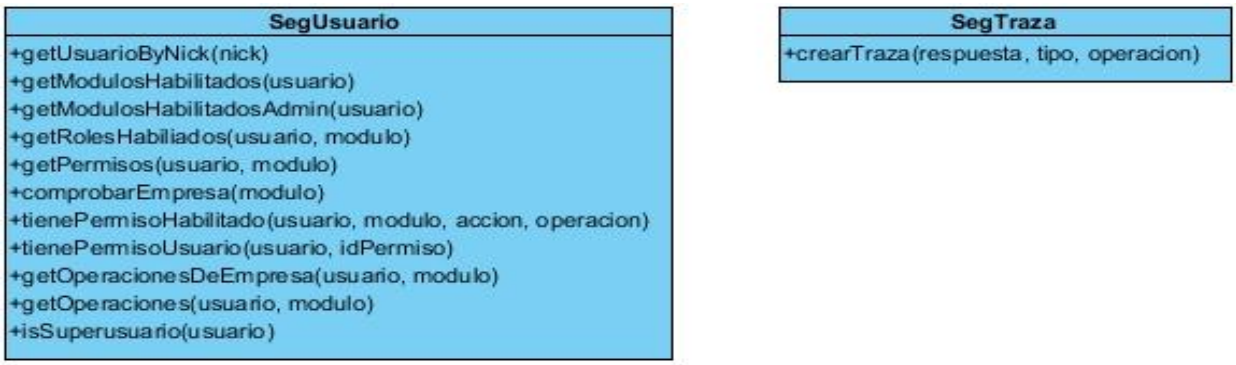

Figura 3.8 Tablas de los componentes externos

En la clase SegUsuario se encuentran los métodos necesarios para establecer la seguridad general del sistema. Dichos procedimientos son llamados en dependencia de las necesidades de seguridad, y pueden ser en cada petición, acciones de autenticación de usuario, o acciones determinadas. La clase SegTraza contiene un procedimiento encargado de crear las trazas del sistema, que luego pueden ser analizadas mediante el proceso de auditoría. Esta acción es llamada durante la ejecución de cada procedimiento del lado del servidor, pasando los parámetros correspondientes a la acción que lo genera, a través de la determinación del suceso asignado. En el siguiente ejemplo se puede apreciar la generación de una traza durante el proceso de creación de una empresa, además de los datos definidos por parámetro, la clase define hora, fecha, host, modulo, usuario y empresa que realiza la acción.

```
SegTraza::crearTraza('Intento satisfactorio de adición de la empresa ' . $nombre . '', 2, 'Adicion de
empresa');
return $this->renderText("{success: true, msg:'$dev'}");
```

Figura 3.9 Ejemplo de crear trazas

3.8.2. Sucesos

Una acción puede clasificarse según un suceso determinado. Todas las acciones del sistema están asociadas a un suceso definido en la entidad seg_suceso del esquema seguridad. El suceso es el tipo de proceso que se realiza. Los sucesos definidos por defecto son (Iniciar sesión, cerrar sesión, crear/adicionar datos, eliminar datos, modificar datos, mensaje de alerta, mensaje de error, permiso denegado, graficar datos, exportar datos y generar datos). Este mecanismo contribuye a mejorar el proceso de auditoría mediante la clasificación de acciones.

3.8.3. Identificadores de permisos

Los identificadores de permisos son nomenclaturas que se les asignan a los objetos de las vistas con el objetivo de saber si se tiene permiso o no a utilizarlo. Cuando se crea un elemento en la capa de la vista, este de define según la operación a la que pertenece, el tipo de elemento y el permiso asignado, mediante la operación tienePermisoUsuario y tienePermisoHabilitado del componente SegUsuario, se puede determinar si el objeto puede ser utilizado o no, restringiendo su uso en caso de ser necesario. Los identificadores de permiso son vitales para garantizar las cuatro capas de seguridad del sistema. En el siguiente ejemplo se puede ver la creación de un objeto de tipo botón, en la vista, con su identificador de permiso asignado.

```
text: 'Modificar',
id: 'configTipoDeActivo-1502-Btn',
icon: 'images/main/modificar.png',
tooltip : {
    text:'Modificar Tipo de activo'
},
```

Figura 3.10 Ejemplo de crear objeto de tipo botón

Donde el id está compuesto por la operación a la que corresponde el objeto, su identificador de permiso y tipo de objeto de que se crea.

3.8.4. Filtros de Symfony

El mecanismo de seguridad puede ser entendido como un filtro, por el que debe pasar cada petición antes de ejecutar la acción. Según las comprobaciones realizadas en el filtro, se puede modificar el procesamiento de la petición, por ejemplo, cambiando la acción ejecutada (*default/secure* en lugar de la acción solicitada en el caso del filtro de seguridad). Symfony extiende esta idea a clases de filtros. Se puede especificar cualquier número de clases de filtros a ser ejecutadas antes de que se procese la respuesta, y además hacerlo de forma sistemática para todas las peticiones. Se pueden entender los filtros como una forma de empaquetar cierto código de forma similar a *preExecute ()* y *postExecute ()*, pero a un nivel superior (para toda una aplicación en lugar de para todo un módulo) (Potencier, y otros, 2008).

3.8.5. Cadena de filtros

Symfony de hecho procesa cada petición como una cadena de filtros ejecutados de forma sucesiva. Cuando el *framework* recibe una petición, se ejecuta el primer filtro (que siempre es *sfRenderingFilter*). En un punto determinado por el filtro, llama al siguiente filtro en la cadena, luego el siguiente, y así sucesivamente. Cuando se ejecuta el último filtro (que siempre es *sfExecutionFilter*), los filtros anteriores pueden finalizar, y así hasta el filtro de *sfRenderingFilter* (Potencier, y otros, 2008).

Para garantizar que cada petición responda a los requisitos de seguridad. Se le agregó al sistema el filtro dserpPermisoFilter, que utiliza las operaciones brindadas por SegUsuario. Este filtro por cada petición verifica los permisos y garantiza que no existan violaciones de seguridad. Este filtro se ejecuta antes de llenar la caché del sistema y realizar la ejecución de peticiones, dando mayor robustez al sistema.

3.8.6. Generadores

Para asegurar que todo el equipo de desarrollo cumpla con las normas de seguridad y las políticas del sistema, se crearon generadores que permiten mediante la ejecución de comandos (robots) obtener el código base del sistema. Los generadores, crean el modelo a partir de la base de datos,

según el tipo de acción, crean las clases correspondientes, las vistas y los escenarios de configuración. Todo esto contribuye a un desarrollo organizado, rápido, y que siga los patrones propuestos y las políticas de seguridad en todos los componentes del *EBMS* DSerp.

3.9. Conclusiones parciales

En este capítulo se analizaron los métodos y procedimiento realizados en el sistema que se utilizaron para dar solución a la investigación. Se argumentaron las tablas existentes en la aplicación así como la estructuración de la seguridad ejercidas mediante un grupo de patrones de diseño como parte de soporte en la seguridad del sistema *EBMS* DSerp. Con la descripción realizada, se justificaron los mecanismos utilizados y se representaron las técnicas propuestas para el modelo de seguridad, lo que contribuye a un mejor entendimiento de la presente investigación.

Capítulo 4. Implementación y validación de resultados

4.1. Introducción al capítulo

En este capítulo se definirán las tarjetas de ingenierías propuestas para cada historia de usuario. Se realizaran las pruebas al sistema mediante las tarjetas de aceptación que permitan dar cumplimiento a las funcionalidades descritas respondiendo a los requisitos de la misma, así como el resultado de las pruebas, concluyendo así, la presente investigación que validen al sistema para el actual modelo empresarial cubano.

4.2. Tareas de ingeniería

Las tareas de ingeniería son actividades que los programadores conocen que el sistema debe hacer. Deben ser estimables, su tiempo de implementación debe ser corto, aproximadamente entre uno y tres días, y su objetivo es resolver las historias de usuario. Una historia de usuario puede tener una o varias tareas de ingeniería, en dependencia de la funcionalidad a desarrollar. Pueden existir también tareas de ingeniería técnicas, que son aquellas que aunque no derivan directamente de una historia de usuario, es necesaria su consideración para que el sistema funcione (Wallace, y otros, 2002).

Para lograr una estandarización en cuanto a la documentación y siguiendo un grupo de normas atendiendo al modelo de artefactos del sistema *EBMS* DSerp, se plantea el mismo formato seguido a lo largo de la investigación por equipo de desarrollo. El formato para las tareas de ingeniería es el siguiente:

Tarea de Ingeniería			
Código	Rector t-01	**Historia de Usuario**	Rector-01
Nombre de Tarea	Definir roles de sistema.		
Responsable	Yoan Hernández Méndez		
Tipo de Tarea	Desarrollo	**Horas Estimadas**	8
Fecha Inicio	27/10/2012	**Fecha Fin**	2/11/2012
Descripción	Se definen roles en el sistema para especificar el modo de acceso dentro de la misma.		

Tabla 4.1 Formato de la tarea de ingeniería.

Campos de la tarjeta de ingeniería:

Código. El código estará definido en dos partes. La primera será el módulo o subsistema al que pertenece la tarea de ingeniería, uno de los siguientes:

- Vitae.
- Gabinete.
- Terra.
- Rector.
- Mensaje.
- Germina.
- Labranza.
- Camarada.
- Hacienda.
- Sumario.
- Caudal

La segunda parte será el número de la funcionalidad que representa con el prefijo t- que define que el artefacto es una tarea de ingeniería.

Historia de usuario: Es el código de la historia de usuario a la cual pertenece la tarea de ingeniería.

Nombre: Define el nombre o funcionalidad concreta a la que se dedica la tarea, debe estar expresado un forma infinitiva.

Responsable: Programador responsable de la realización de la tarea.

Tipo de tarea: Información del tipo de tarea a realizar, la misma puede ser:

- ✓ **Desarrollo.** Tarea que se realizará por primera vez.
- ✓ **Corrección.** Tarea que se realiza a partir de una anterior que no se realizó correctamente, es decir, que no pasó los casos de prueba satisfactoriamente.
- ✓ **Mejora.** Tarea que se realiza a partir de una anterior incorporándole nuevos requerimientos.
- ✓ **Otra.** Tarea que no corresponde con una de las anteriores, en este caso es necesario especificar el tipo de tarea o realizar una descripción más profunda de esta.

Horas estimadas: Las horas estimadas se definen para el desarrollo del proyecto *EBMS* DSerp. Dado a que una historia de usuario, tiene como tiempo mínimo de desarrollo una semana hábil (40 horas), las horas estimadas constituyen la cantidad de tiempo en horas en desarrollar la tarea de ingeniería. La suma de horas estimadas de las tareas de ingeniería de una historia de usuario no puede superar la cantidad de horas hábiles definidas para la historia.

Fecha inicial: Fecha en la que se inicia el desarrollo de la tarea de ingeniería.

Fecha final: Fecha en la que se concluye el desarrollo de la tarea de ingeniería.

Descripción: Es una breve descripción sobre lo que la tarea debe hacer o resolver.

En el sistema propuesto se identificaron las siguientes tareas de ingeniería:

Historia de usuario	No. De tarea	Tarea de ingeniería
Gestionar roles	Rector t-01	Definir roles del sistema.
	Rector t-02	Asignar operaciones a roles.
	Rector t-03	Asignar permiso a las operaciones.
Gestionar usuario	Rector t-04	Adicionar usuarios.
	Rector t-05	Modificar usuarios.
	Rector t-06	Eliminar usuarios.
	Rector t-07	Asignar roles a usuarios.
Autenticar usuario	Rector t-08	Comprobar datos de usuarios.
	Rector t-09	Validar reglas de acceso.
Comprobar permisos	Rector t-10	Comprobar permisos de usuarios.
	Rector t-11	Restringir permisos de usuarios.
Gestionar restricciones de tiempo	Rector t-12	Definir restricción mensual.
	Rector t-13	Definir restricción diaria.
	Rector t-14	Definir restricción por rangos.
Gestionar direcciones de IP	Rector t-15	Definir dirección ip.
Generar trazas del sistema	Rector t-16	Capturar detalles de trazas.
Visualizar trazas del sistema	Rector t-17	Ordenar trazas.
	Rector t-18	Filtrar trazas.
	Rector t-19	Exportar trazas.

Tabla 4.2 Tareas de ingeniería por historias de usuario.

Las siguientes tablas argumentadas en las siguientes tareas de ingeniería:

Tarea de Ingeniería			
Código	Rector t-01	**Historia de Usuario**	Rector-01

Nombre de Tarea	Definir roles de sistema.		
Responsable	Yoan Hernández Méndez		
Tipo de Tarea	Desarrollo	**Horas Estimadas**	32
Fecha Inicio	27/10/2012	**Fecha Fin**	2/11/2012
Descripción	Se definen roles en el sistema para especificar el modo de acceso dentro de la misma.		

Tabla 4.3 Tarea de ingeniería: Definir roles del sistema.

Tarea de Ingeniería			
Código	Rector t-02	**Historia de Usuario**	Rector-01
Nombre de Tarea	Asignar operaciones a roles.		
Responsable	Yoan Hernández Méndez		
Tipo de Tarea	Desarrollo	**Horas Estimadas**	24
Fecha Inicio	3/11/2012	**Fecha Fin**	7/11/2012
Descripción	Se definen las operaciones que se pueden realizar en cada rol sobre los módulos del sistema.		

Tabla 4.4 Tarea de ingeniería: Asignar operaciones a roles.

Tarea de Ingeniería			
Código	Rector t-03	**Historia de Usuario**	Rector-01
Nombre de Tarea	Asignar permiso a las operaciones.		
Responsable	Yoan Hernández Méndez		
Tipo de Tarea	Desarrollo	**Horas Estimadas**	16
Fecha Inicio	8/11/2012	**Fecha Fin**	11/12/2012
Descripción	Se definen los permisos que tiene el usuario sobre alguna operación en el sistema.		

Tabla 4.5 Tarea de ingeniería: Asignar permisos de operaciones.

Tarea de Ingeniería			
Código	Rector t-04	**Historia de Usuario**	Rector-02
Nombre de Tarea	Adicionar usuarios.		
Responsable	Yoan Hernández Méndez		
Tipo de Tarea	Desarrollo	**Horas Estimadas**	24
Fecha Inicio	12/11/2012	**Fecha Fin**	15/11/2012
Descripción	Se definen los usuarios que tienen acceso a la aplicación.		

Tabla 4.6 Tarea de ingeniería: Asignar usuarios.

Tarea de Ingeniería			
Código	Rector t-05	**Historia de Usuario**	Rector-02
Nombre de Tarea	Modificar usuarios.		
Responsable	Yoan Hernández Méndez		
Tipo de Tarea	Mejora	**Horas Estimadas**	12
Fecha Inicio	16/11/2012	**Fecha Fin**	18/11/2012
Descripción	Se cambia alguna propiedad de algún usuario que tiene acceso al sistema.		

Tabla 4.7 Tarea de ingeniería: Modificar usuarios.

Tarea de Ingeniería			
Código	Rector t-06	**Historia de Usuario**	Rector-02
Nombre de Tarea	Eliminar usuarios.		
Responsable	Yoan Hernández Méndez		
Tipo de Tarea	Otra	**Horas Estimadas**	24
Fecha Inicio	19/11/2012	**Fecha Fin**	22/11/2012
Descripción	Se elimina por completo un usuario en el sistema, ya sea por motivo de baja en sus actividades laborales o por alguna medida que establece la empresa dentro del control de la		

	misma.

Tabla 4.8 Tarea de ingeniería: Eliminar usuarios.

Tarea de Ingeniería			
Código	Rector t-07	**Historia de Usuario**	Rector-02
Nombre de Tarea	Asignar roles a usuarios.		
Responsable	Yoan Hernández Méndez		
Tipo de Tarea	Desarrollo	**Horas Estimadas**	28
Fecha Inicio	23/11/2012	**Fecha Fin**	26/11/2012
Descripción	Se define que roles van a tener cada usuario dentro del sistema.		

Tabla 4.9 Tarea de ingeniería: Asignar roles a usuarios.

Tarea de Ingeniería			
Código	Rector t-08	**Historia de Usuario**	Rector-03
Nombre de Tarea	Comprobar datos de usuarios.		
Responsable	Yoan Hernández Méndez		
Tipo de Tarea	Desarrollo	**Horas Estimadas**	24
Fecha Inicio	27/11/2012	**Fecha Fin**	30/11/2012
Descripción	Se verifica los datos de los usuarios para que no haya conflicto al acceder al sistema		

Tabla 4.10 Tarea de ingeniería: Comprobar datos de usuarios.

Tarea de Ingeniería			
Código	Rector t-09	**Historia de Usuario**	Rector-03
Nombre de Tarea	Validar reglas de acceso.		
Responsable	Yoan Hernández Méndez		
Tipo de Tarea	Desarrollo	**Horas Estimadas**	24

Fecha Inicio	1/12/2012	**Fecha Fin**	4/12/2012
Descripción	Se validan las reglas de acceso que se definieron por cada usuario.		

Tabla 4.11 Tarea de ingeniería: Validar reglas de acceso.

Tarea de Ingeniería			
Código	Rector t-10	**Historia de Usuario**	Rector-04
Nombre de Tarea	Comprobar permisos de usuarios.		
Responsable	Yoan Hernández Méndez		
Tipo de Tarea	Desarrollo	**Horas Estimadas**	36
Fecha Inicio	5/12/2012	**Fecha Fin**	12/12/2012
Descripción	Se comprueba los permisos definidos a cada usuario sobre las operaciones en el sistema.		

Tabla 4.12 Tarea de ingeniería: Comprobar permisos de usuarios.

Tarea de Ingeniería			
Código	Rector t-11	**Historia de Usuario**	Rector-04
Nombre de Tarea	Restringir permisos de usuarios.		
Responsable	Yoan Hernández Méndez		
Tipo de Tarea	Desarrollo	**Horas Estimadas**	40
Fecha Inicio	13/12/2012	**Fecha Fin**	20/12/2012
Descripción	Se definen operaciones que un usuario no puede realizar en el sistema		

Tabla 4.13 Tarea de ingeniería: Restringir permisos de usuarios.

Tarea de Ingeniería			
Código	Rector t-12	**Historia de Usuario**	Rector-05
Nombre de Tarea	Definir restricción mensual.		

Responsable	Yoan Hernández Méndez		
Tipo de Tarea	Desarrollo	**Horas Estimadas**	40
Fecha Inicio	2/01/2013	**Fecha Fin**	9/01/2013
Descripción	Se definen las restricciones mensuales que pueden tener los usuarios sobre la accesibilidad al sistema.		

Tabla 4.14 Tarea de ingeniería: Definir restricción mensual.

Tarea de Ingeniería			
Código	Rector t-13	**Historia de Usuario**	Rector-05
Nombre de Tarea	Definir restricción diaria.		
Responsable	Yoan Hernández Méndez		
Tipo de Tarea	Desarrollo	**Horas Estimadas**	36
Fecha Inicio	10/01/2013	**Fecha Fin**	16/01/2013
Descripción	Se definen las restricciones diarias que pueden tener los usuarios sobre la accesibilidad al sistema.		

Tabla 4.15 Tarea de ingeniería: Definir restricción diaria.

Tarea de Ingeniería			
Código	Rector t-14	**Historia de Usuario**	Rector-05
Nombre de Tarea	Definir restricción por rangos.		
Responsable	Yoan Hernández Méndez		
Tipo de Tarea	Desarrollo	**Horas Estimadas**	32
Fecha Inicio	17/01/2013	**Fecha Fin**	23/01/2013
Descripción	Se definen las restricciones por un rango de tiempo que pueden tener los usuarios sobre la accesibilidad al sistema.		

Tabla 4.16 Tarea de ingeniería: Definir restricción por rangos.

Tarea de Ingeniería

Código	Rector t-15	Historia de Usuario	Rector-06
Nombre de Tarea	Definir dirección ip.		
Responsable	Yoan Hernández Méndez		
Tipo de Tarea	Desarrollo	Horas Estimadas	60
Fecha Inicio	24/01/2013	Fecha Fin	6/02/2013
Descripción	Se definen las direcciones de red por la cual el usuario puede acceder al sistema.		

Tabla 4.17 Tarea de ingeniería: Definir dirección ip.

Tarea de Ingeniería			
Código	Rector t-16	Historia de Usuario	Rector-07
Nombre de Tarea	Capturar detalles de trazas.		
Responsable	Yoan Hernández Méndez		
Tipo de Tarea	Desarrollo	Horas Estimadas	36
Fecha Inicio	7/02/2013	Fecha Fin	13/02/2013
Descripción	Se recogen todas las actividades realizadas por los usuarios en el sistema.		

Tabla 4.19 Tarea de ingeniería: Capturar detalles de trazas.

Tarea de Ingeniería			
Código	Rector t-17	Historia de Usuario	Rector-08
Nombre de Tarea	Ordenar trazas.		
Responsable	Yoan Hernández Méndez		
Tipo de Tarea	Desarrollo	Horas Estimadas	28
Fecha Inicio	14/02/2013	Fecha Fin	19/02/2013
Descripción	Permite ordenar las trazas para una mejor búsqueda sobre una acción realizada por un usuario en el sistema.		

Tabla 4.20 Tarea de ingeniería: Ordenar trazas.

Tarea de Ingeniería			
Código	Rector t-18	**Historia de Usuario**	Rector-08
Nombre de Tarea	Filtrar trazas.		
Responsable	Yoan Hernández Méndez		
Tipo de Tarea	Desarrollo	**Horas Estimadas**	16
Fecha Inicio	20/02/2013	**Fecha Fin**	22/02/2013
Descripción	Permite una búsqueda avanzada sobre una acción realizada por un usuario en el sistema.		

Tabla 4.21 Tarea de ingeniería: Filtrar trazas.

Tarea de Ingeniería			
Código	Rector t-19	**Historia de Usuario**	Rector-08
Nombre de Tarea	Exportar trazas.		
Responsable	Yoan Hernández Méndez		
Tipo de Tarea	Desarrollo	**Horas Estimadas**	16
Fecha Inicio	23/02/2013	**Fecha Fin**	25/02/2013
Descripción	Permite exportar las trazas realizadas por los usuarios en el sistema para un futuro análisis.		

Tabla 4.21 Tarea de ingeniería: Exportar trazas.

4.3. Pruebas

Los casos de prueba son pruebas funcionales o unitarias que se realizan al sistema para comprobar su funcionamiento. Las pruebas funcionales son validaciones escritas desde la perspectiva del cliente, y las pruebas unitarias son validaciones desde la perspectiva del programador. Mientras un código no haya sido probado no existe. El objetivo general es tener una forma para decirle al cliente que la historia de usuario está lista (Jeffries, y otros, 2000).

Las pruebas de aceptación que se definieron para la seguridad del sistema *EBMS* DSerp y atendiendo a la estandarización de la documentación definida por el equipo de trabajo, quedan estructurado de la siguiente forma:

Caso de prueba de aceptación			
Código:	Rector p-01	**Historia de usuario:**	Rector-01
Nombre:	Comprobar gestionar roles		
Descripción:	Prueba para comprobar la gestión de roles en el sistema		
Condiciones de ejecución:	El usuario tiene que tener acceso a la funcionalidad de gestión de roles.		
Entrada / Pasos de ejecución:	Adicción: Se define un nuevo rol en el sistema. Modificación y Eliminación: Se selecciona un rol del sistema.		
Resultado esperado:	Adicción: Queda registrado un nuevo rol en el sistema. Modificación: Se actualizan los parámetros del rol. Eliminación: Queda eliminado el rol del sistema.		
Evaluación de la prueba:	Prueba satisfactoria.		

Tabla 4.22 Formato de un caso de prueba de aceptación

Los campos de un caso de aceptación de prueba son los siguientes:

Código. El código estará definido en dos partes. La primera será el módulo o subsistema al que pertenece el caso de prueba de aceptación, uno de los siguientes:

- Vitae.
- Gabinete.
- Terra.
- Rector.
- Mensaje.
- Germina.
- Labranza.
- Camarada.
- Hacienda.
- Sumario.
- Caudal.

La segunda parte será el número de la funcionalidad que representa con el prefijo p- que define que el artefacto es un caso de prueba de aceptación.

Historia de usuario: Es el código de la historia de usuario a la cual pertenece el caso de prueba.

Descripción: Es una breve descripción del propósito de la prueba.

Condiciones de ejecución: Condiciones especiales que deben tenerse en cuenta para ejecutar el caso de prueba.

Entradas / pasos de ejecución: Entradas o funciones que deben ejecutarse para realizar el caso de prueba.

Resultado esperado: Salida u objetivo que debe cumplir la funcionalidad a la que se le realiza el caso de prueba.

Evaluación. Evaluación de éxito del caso de prueba. Prueba satisfactoria en caso de éxito o prueba insatisfactoria en caso de fallo.

Al terminar cada fase o iteración del proyecto las pruebas serán agregadas a los artefactos de entrega. Las funcionalidades o historias de usuarios que no cumplan con las expectativas previstas, serán corregidas en la próxima iteración. A continuación se describen los casos de prueba del sistema durante todo su desarrollo de realización.

4.4. Casos de prueba del sistema

Caso de prueba de aceptación			
Código:	Rector p-01	**Historia de usuario:**	Rector-01
Nombre:	Comprobar gestionar roles.		
Descripción:	Prueba para comprobar la gestión de roles en el sistema.		
Condiciones de ejecución:	El usuario tiene que tener acceso a la funcionalidad de gestión de roles.		
Entrada / Pasos de ejecución:	Adicción: Se define un nuevo rol en el sistema. Modificación y Eliminación: Se selecciona un rol del sistema.		
Resultado esperado:	Adicción: Queda registrado un nuevo rol en el sistema. Modificación: Se actualizan los parámetros del rol. Eliminación: Queda eliminado el rol del sistema.		
Evaluación de la prueba:	Prueba satisfactoria.		

Tabla 4.23 Descripción del caso de prueba de aceptación Rector p-01.

Caso de prueba de aceptación			
Código:	Rector p-02	**Historia de usuario:**	Rector-02
Nombre:	Comprobar gestionar usuario.		
Descripción:	Prueba para comprobar la gestión de usuarios en el sistema.		
Condiciones de ejecución:	El usuario tiene que tener acceso a la funcionalidad de gestión de usuario.		
Entrada / Pasos de ejecución:	Adicción: Se define un nuevo usuario en el sistema. Modificación y Eliminación: Se selecciona un usuario del sistema.		
Resultado esperado:	Adicción: Queda registrado un nuevo usuario en el sistema. Modificación: Se actualizan los parámetros del usuario.		

	Eliminación: Queda eliminado el usuario del sistema.
Evaluación de la prueba:	Prueba satisfactoria.

Tabla 4.24 Descripción del caso de prueba de aceptación Rector p-02.

Caso de prueba de aceptación			
Código:	Rector p-03	**Historia de usuario:**	Rector-03
Nombre:	Comprobar autentificación de usuario.		
Descripción:	Prueba para comprobar la autentificación de los usuarios en el sistema.		
Condiciones de ejecución:	El usuario tiene que loguearse en el sistema.		
Entrada / Pasos de ejecución:	El sistema realiza un proceso interno de verificación de la autentificación del usuario sobre la plataforma.		
Resultado esperado:	Realiza correctamente el proceso de autentificación del usuario.		
Evaluación de la prueba:	Prueba satisfactoria.		

Tabla 4.25 Descripción del caso de prueba de aceptación Rector p-03.

Caso de prueba de aceptación			
Código:	Rector p-04	**Historia de usuario:**	Rector-04
Nombre:	Comprobar permisos.		
Descripción:	Prueba para comprobar permisos de acceso de los usuarios en el sistema.		
Condiciones de ejecución:	El usuario tiene que autenticarse en el sistema.		
Entrada / Pasos de ejecución:	El sistema realiza un proceso interno de verificación y asigna los permisos correspondientes al usuario sobre la plataforma.		
Resultado esperado:	Realiza correctamente el proceso de comprobar permisos de acceso en el sistema		
Evaluación de la prueba:	Prueba satisfactoria.		

Tabla 4.26 Descripción del caso de prueba de aceptación Rector p-04.

Caso de prueba de aceptación			
Código:	Rector p-05	**Historia de usuario:**	Rector-05
Nombre:	Comprobar gestionar restricciones de tiempo.		
Descripción:	Prueba para comprobar las restricciones de tiempo en el sistema.		
Condiciones de ejecución:	El usuario tiene que tener acceso a la funcionalidad de gestionar restricciones de tiempo del sistema.		

Entrada / Pasos de ejecución:	Adicción: Se define una nueva restricción de tiempo en el sistema. Modificación y Eliminación: Se selecciona una restricción de tiempo.
Resultado esperado:	Adicción: Queda registrado una nueva restricción de tiempo en el sistema. Modificación: Se actualizan los parámetros de la restricción de tiempo en el sistema. Eliminación: Queda eliminado la restricción de tiempo en el sistema.
Evaluación de la prueba:	Prueba satisfactoria.

Tabla 4.27 Descripción del caso de prueba de aceptación Rector p-05.

Caso de prueba de aceptación			
Código:	Rector p-06	**Historia de usuario:**	Rector-06
Nombre:	Comprobar gestionar direcciones de IP.		
Descripción:	Prueba para comprobar la gestión de direcciones IP en el sistema.		
Condiciones de ejecución:	El usuario tiene que tener acceso a la funcionalidad de gestionar direcciones de IP.		
Entrada / Pasos de ejecución:	Adicción: Se define una nueva restricción de dirección de IP. Modificación y Eliminación: Se selecciona una restricción de dirección de IP.		
Resultado esperado:	Adicción: Queda registrado una nueva restricción de red. Modificación: Se actualizan los parámetros de una restricción de red. Eliminación: Queda eliminada la restricción de red.		
Evaluación de la prueba:	Prueba satisfactoria.		

Tabla 4.28 Descripción del caso de prueba de aceptación Rector p-06.

Caso de prueba de aceptación			
Código:	Rector p-07	**Historia de usuario:**	Rector-07
Nombre:	Comprobar las reglas de acceso.		
Descripción:	Prueba para la validación de las reglas de acceso en el sistema.		
Condiciones de ejecución:	El usuario tiene que loquearse en el sistema.		
Entrada / Pasos de ejecución:	El sistema realiza un proceso interno de verificación y asigna el acceso correspondiente al usuario sobre la plataforma.		
Resultado esperado:	Realiza correctamente las reglas de acceso sobre el sistema.		
Evaluación de la prueba:	Prueba satisfactoria.		

Tabla 4.29 Descripción del caso de prueba de aceptación Rector p-07.

Caso de prueba de aceptación			
Código:	Rector p-08	**Historia de usuario:**	Rector-08
Nombre:	Comprobar las trazas del sistema.		
Descripción:	Prueba para la validar la ejecución de trazas en el sistema.		
Condiciones de ejecución:	El usuario tiene que loquearse en el sistema.		
Entrada / Pasos de ejecución:	El sistema genera las trazas de todas las actividades que se llevan a cabo en la plataforma.		
Resultado esperado:	Realiza correctamente la generación de trazas del sistema.		
Evaluación de la prueba:	Prueba satisfactoria.		

Tabla 4.30 Descripción del caso de prueba de aceptación Rector p-08.

Caso de prueba de aceptación			
Código:	Rector p-09	**Historia de usuario:**	Rector-09
Nombre:	Comprobar la visualización de trazas del sistema.		
Descripción:	Prueba para la validar la visualización de trazas en el sistema.		
Condiciones de ejecución:	El usuario tiene que loquearse en el sistema.		
Entrada / Pasos de ejecución:	El sistema muestra las trazas de todas las actividades que se llevan a cabo en la plataforma.		
Resultado esperado:	Realiza correctamente la visualización de trazas del sistema.		
Evaluación de la prueba:	Prueba satisfactoria.		

Tabla 4.31 Descripción del caso de prueba de aceptación Rector p-09.

4.5. Resultados de la integración al EBMS DSerp

La integración del módulo de seguridad a la plataforma para la gestión de recursos y procesos, permitió una mejora en cuanto a la gestión de responsabilidades y administración de recursos, validada según los siguientes aspectos.

- ✓ Gestión de roles de usuarios: Con el proceso de gestión de roles de usuarios el *EBMS* DSerp puede definir los permisos sobre las operaciones y determinar los ambientes accesibles para cada grupo de usuarios, detallando varios niveles de seguridad a partir de los módulos del sistema.

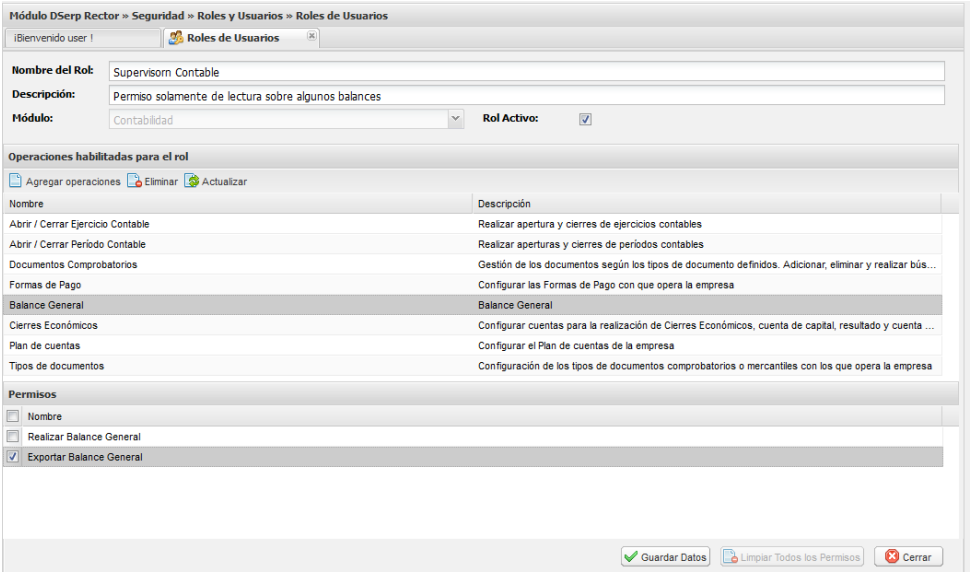

Figura 4.1 Proceso de definición de permisos de un rol

- ✓ Gestión de usuarios: Definir en cada acción que usuario está autenticado y a que empresa pertenece, influyendo en la capacidad de la aplicación de analizar responsabilidades, generar trazas asociadas a los usuarios y relacionar dichos usuarios con los empleados del módulo DSerp Vitae[20], estableciendo dependencias funcionales entre los recursos de la aplicación y las responsabilidades laborales de los usuarios.

Figura 4.2 Autenticación de usuarios.

[20] DSerp Vitae: Módulo para la gestión de Recursos Humanos del *EBMS* DSerp (Kerton, 2012).

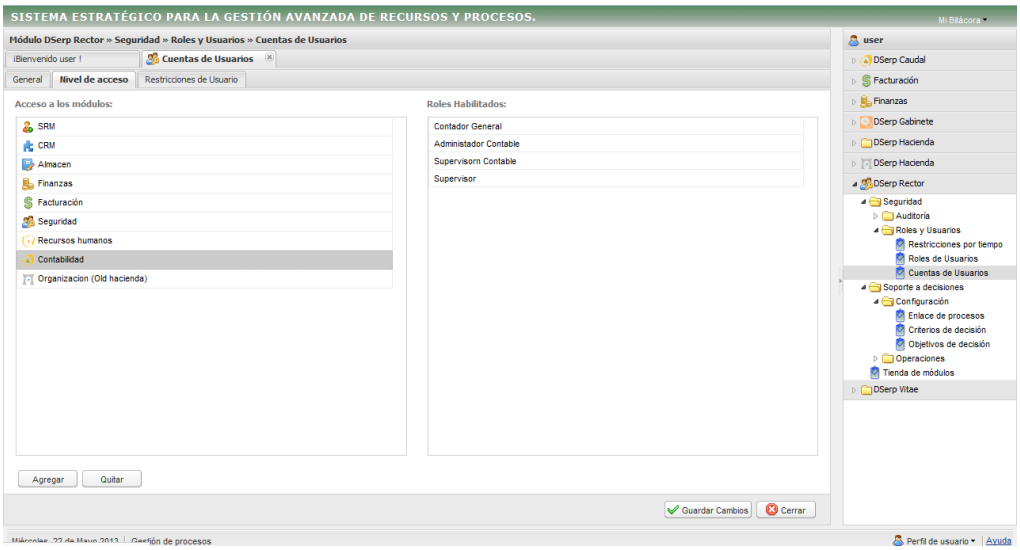

Figura 4.3 Asignación de roles a usuarios

- ✓ Restricciones de usuario: Las restricciones de usuario ofrecieron la capacidad al sistema de determinar los horarios y direcciones de acceso a la aplicación, lo que posibilita evitar violaciones de seguridad.

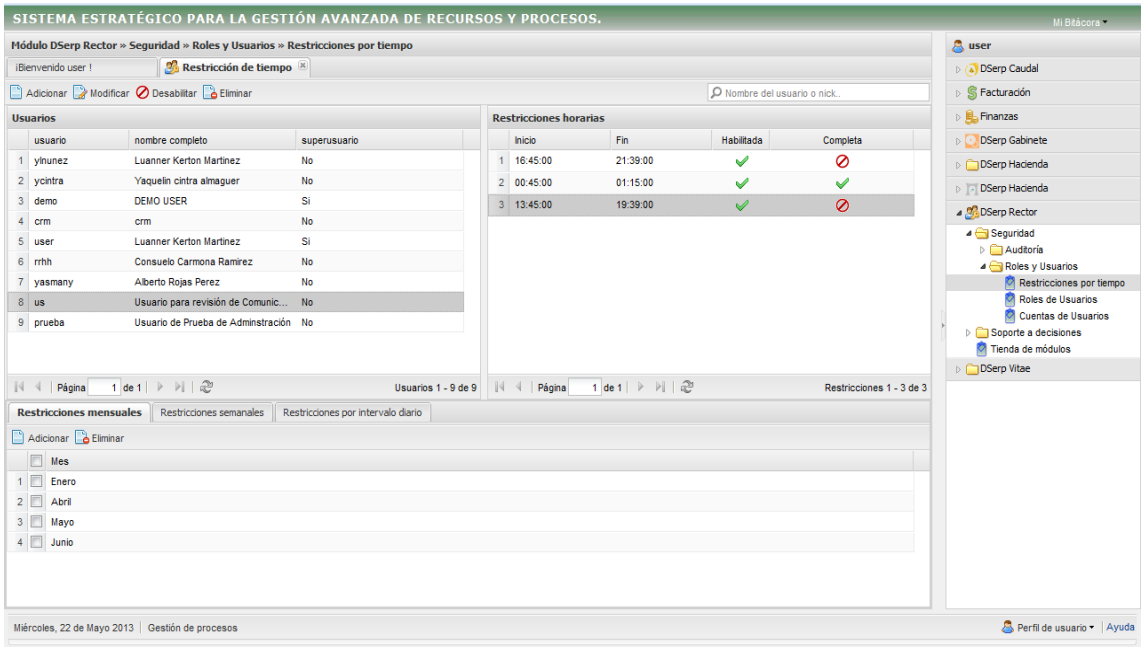

Figura 4.4 Proceso de restricción de horarios de usuarios.

- ✓ La generación y visualización de trazas contribuyeron al análisis de comportamiento de los usuarios en el sistema *EBMS* DServ, además de ofrecer una vía para la evaluación continua de la aplicación. Los sistemas de trazas, pueden ser mejorados hasta alcanzar un nivel que permita la minería de procesos, y su alto nivel de detalle permitió una identificación rápida de anomalías en el sistema.

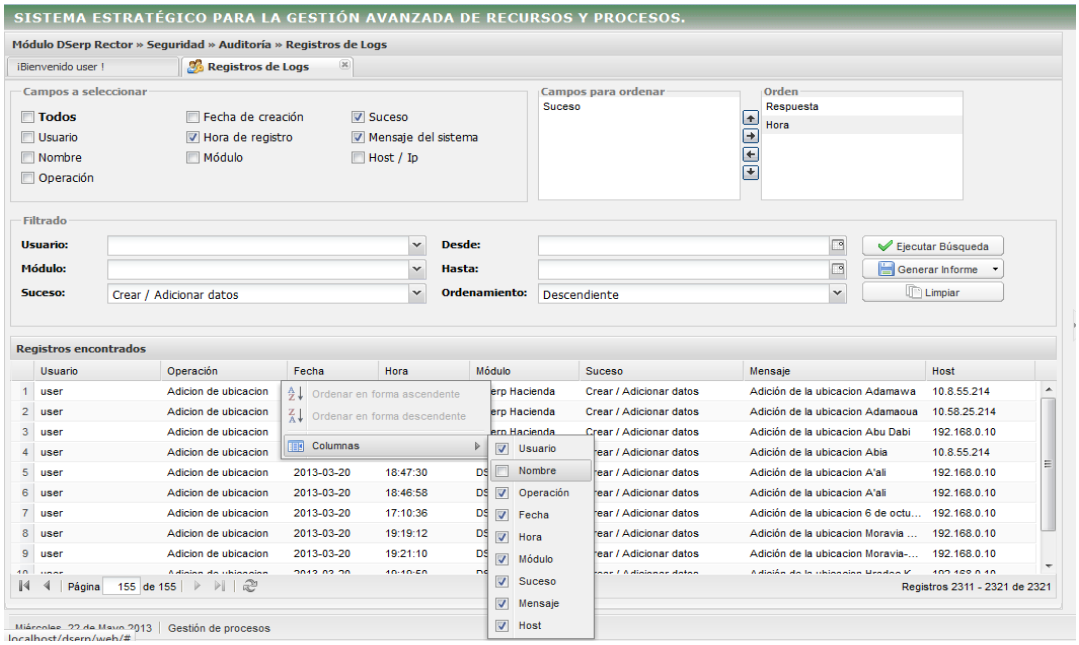

Figura 4.5 Proceso de visualización de trazas.

4.6. Conclusiones parciales

En el presente capítulo se detallaron las tareas de ingenierías de la fase de implementación según la metodología *XP*, que permitieron documentar la construcción de la solución propuesta a partir de las historias de usuario. Se argumentaron las pruebas realizadas al sistema, evidenciando su resultado satisfactorio a través de los casos de prueba de aceptación. Además, se mostraron los resultados alcanzados tras la integración del módulo de seguridad a la plataforma *EBMS* DSerp, lo que validó el cumplimiento de los objetivos trazados para la presente investigación

Conclusiones

Luego del desarrollo del sistema de seguridad de la plataforma *EBMS* DSerp, se arriba a las siguientes conclusiones:

- ✓ La implementación de un modelo basado en roles y usuarios permitió al sistema salvaguardar la información manipulada, asegurando los principios de integridad, confidencialidad y disponibilidad, lo que responde a los intereses de la empresa de proteger sus datos y elevó el índice de aceptación de la plataforma, adecuándola a las reglas cubanas de protección de información empresarial.
- ✓ La capacidad de definir varias capas de seguridad a partir de permisos escalonados, mejoró la adaptabilidad de la aplicación a diferentes entornos y permitió a los administradores definir reglas a distintos niveles, ofreciendo un entorno controlado a los especialistas que utilizan la plataforma.
- ✓ La definición de restricciones basadas en políticas de tiempo y lugares de acceso permitieron la vinculación de usuarios con sus responsabilidades laborares, ofreciendo a la empresa la capacidad de gestionar permisos en correspondencia con la estancia de sus empleados y los puntos terminales desde donde acceden al sistema.
- ✓ La generación y visualización de trazas permitieron elevar la capacidad de análisis dentro de la aplicación, lo que ofreció a los supervisores de seguridad en las empresas un escenario útil para la detección de anomalías y la corrección de violaciones de manipulación de la información.

Recomendaciones

Se emiten las siguientes recomendaciones:

- ✓ Agregar al sistema de seguridad del *EBMS* DSerp métodos de autenticación biométricos, lo que elevará su capacidad de adaptación a escenarios empresariales complejos.
- ✓ Permitir en el ambiente de configuración de la seguridad, la posibilidad de cargar y exportar personalizaciones de restricciones y políticas de acceso para mejorar la distribución y compartición de reglas de acceso.
- ✓ Desarrollar para la plataforma móvil un ambiente de configuración de seguridad, lo que permitirá a los administradores gestionar el acceso desde dispositivos portátiles.

Bibliografía consultada

1. **Alfarotanco, Jose A. y Rábade, Jose A. 2012.** *integración empresa proveedor.* 2012.
2. **Apache Software Foundation.** Apache Software Foundation. [En línea] [Citado el: 10 de Diciembre de 2011.] http://www.apache.org/.
3. **Asteasuain, Fernando y Schmidt, Leandro Ariel. 2007.** *Aplicación de la Programación Orientada a Aspectos como Solución a los.* 2007.
4. **Barrios, Ricardo. 2007.** *Introduction to Ext.* 2007.
5. **Beck, K. 1999.** *Extreme Programming Explained. Embrace Change.* s.l. : Pearson Education, 1999.
6. **Bent, Kent y Fowler, Martin. 2001.** *Planning Extreme Programming.* s.l. : Addison Wesley, 2001. ISBN: 0-201-71091-9.
7. **Benvenuto, Angel. 2006.** *Implantación de Sistemas ERP, su impacto en la gestión de empresa e integración con las TIC.* Santiago de Chile : Universidad de Concepción, 2006. Vol. IV. ISSN: 0718-4662.
8. **Blokdijk, Gerard y Menken, Ivanka. 2008.** *Supplier Management Best Practice Handbook: Evaluating, Sourcing, Managing and Delivering Supplier Excellence in Relationships, Quality and Costs - Ready to Use Bringing Theory Into Action.* s.l. : Emereo Pty Limited, 2008. ISBN: 1921523506.
9. **Cabrera González, Miguel, y otros. 2012.** *Sistema económico integrado Versat Sarasola.* 2012.
10. **Cabrera, Lucy Torres y Urquiaga Rodríguez, Ana Julia . 2009.** *Análisis de la actividad económica como herramienta para la toma de decisiones.* La Habana : Félix Valera, 2009. ISBN: 978-959-07-0942-5.
11. **Corletti, Alejandro. 2011.** *Seguridad por niveles.* Madrid : DarFe, 2011.
12. **Crespo, Ing. Jose Carlos Melo. 2011.** *MANUAL DE COMPRAS GHGC.* 2011.
13. **Cuadrado, Carmen. 2007.** *Protocolo en las relaciones internacionales de la empresa y sus negocios.* s.l. : FUND. CONFEMETAL, 2007. ISBN: 9788496743175.
14. **de los Santos, Sergio. 2009.** *Una al día, 11 años de seguridad informática.* s.l. : Hispasec system, 2009. ISBN: 978-1-4092-4380-9.

15. **Domínguez, M. Dorado. 2005.** *NetBeans IDE 4.1, la alternativa a Eclipse.* Madrid : Editorial Iberprensa, 2005. págs. 32-34.

16. **Ext JS Inc.** Ext JS. [En línea] [Citado el: 11 de noviembre de 2011.] www.extjs.com.

17. **Fowler, Martin, Rice, David y Foemmel, Matthew. 2002.** *Patterns of Enterprise Application Arquitecture.* s.l. : Addison Wesley, 2002. ISBN : 0-321-12742-0 .

18. **García, Rosa María Mato. 1999.** *Diseño de base de datos.* 1999.

19. **Gerardo, Fernández Escribano. 2002.** *Introducción a Extreme Programming, Ingeniería del Software II.* 2002.

20. **Gerner, Jason, y otros. 2006.** *Professional LAMP: Linux, Apache, MySQL and PHP 5 Web Development.* Indianapolis : Wiley Publishing, Inc., 2006. ISBN: 978-0-7645-9723-7.

21. **Giraldo, y otros. 2005.** *HERRAMIENTAS DE DESARROLLO DE INGENIERIA DE SW PARA LINUX.* s.l. : Monitoria de Ingesoft, 2005.

22. **Gómez Baryolo, Oiner , Rivero Pino, Noel Jesús y López Méndez, Daniel . 2011.** *Sistema de gestión integral de seguridad Acaxia.* 2011.

23. **Gutiérrez, Javier. 2006.** *Frameworks de desarrollo.* 2006.

24. **Jacobson, Ivar, Booch, G. y Rumbaugh, J. 2000.** *El proceso unificado de desarrollo de software.* Madrid : Pearson Educación, 2000. ISBN: 84-7829-036-2.

25. **Jeffries, Ron y Anderson, Ann. 2000.** *Extreme Programming Installed.* s.l. : Addison Wesley, 2000. ISBN: 0-201-70842-6.

26. **Kerton, Luanner. 2012.** Nube de servicios para la gestión de recursos y procesos. *XII Semana Tecnológica.* La Habana : Centro Nacional de Formación y Desarrollo, 2012.

27. **—. 2012.** Sistema para la toma de decisiones sobre datos parametrizados. *Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.* La Habana : Universidad de las Ciencias Informáticas, 2012.

28. **Lesester, Timothy. 2000.** *Alianzas Estratégicas con proveedores.* Bogotá, Colombia : Grupo Editorial Norma, 2000. ISBN: 958-04-7756-6.

29. **Malfará, Dayvis, y otros. 2006.** *Gestión de Software. Testing XP.* Argentina : Facultad de Ingeniería, 2006.

30. **Mártiles, Marín. 2008.** *Sistema de autenticación y autorización centralizado.* 2008.

31. **McConnell, Steve. 2002.** *Professional Software Develoment: Shorter Shedules, Higher Quality Products, More Successful Projects, Enhanced Careers [CHM].* s.l. : Adisson Wesley, 2002. pág. 272. ISBN: 0-321-19367-9.

32. **Meléndez Carballido, Rogelio y Pérez Calderón, Manuel José. 2010.** *EL RÉGIMEN JURÍDICO DE LA SEGURIDAD INFORMÁTICA EN EL SISTEMA EMPRESARIAL CUBANO. UNA VISIÓN ACTUAL.* 2010.

33. **Pooler, Victor H. y Pooler, David J.** *Purchasing and Supply Management: Creating the Vision.* s.l. : Springer. ISBN: 0412106019.

34. **—. 1997.** *Purchasing and Supply Management: Creating the Vision.* s.l. : Springer, 1997. ISBN: 0412106019.

35. **Portela González, Miguel, y otros. 2010.** *Estudio de web servers.* 2010.

36. **PostgreSQL Development Group. 2009.** *PostgreSQL 8.4.0 Documentation.* s.l. : PostgreSQL Global Development Group, 2009.

37. **Potencier, Fabien y Zaninotto, Francois. 2008.** *Symfony, la guía definitiva.* 2008.

38. **Potencier, Fabien. 2011.** www.symfony-project.org. [En línea] Symfony Project, 20 de Noviembre de 2011.

39. **Power, D. J. 2002.** *Decision support systems: concepts and resources for managers.* Westport : Quorum Books, 2002.

40. **Power, Daniel J. 2010.** *Decision Support Systems.* s.l. : InTech, 2010. ISBN 978-953-7619-64-0,.

41. **Rodríguez Oromendía, Ainhoa. 2012.** *Marketing, Estrategias y Tendencias.* s.l. : Sanz y Torres, 2012. ISBN: 9788415550037.

42. **Salazar Riaño, José Luis, Sarasa López, Miguel Ángel y Pastor Franco, José . 1998.** *Criptografía digital: fundamentos y aplicaciones.* 1998.

43. **Shreyas, Doshi. 2001.** *Software Engineering for Security: Towards Architecting.* 2001.

44. **Stephen, Thomas A. 2000.** *SSL & TSL Essentials. Securing the Web.* New York : John Wiley & Song. Inc, 2000. ISBN: 0-471-38354-6.

45. **Valdivia, Isaac Guzmán. 1963.** *La sociología de la empresa.* s.l. : Editorial Jus, 1963.

46. **Wahlin, Dan. 2011.** The Prototype Pattern - Techniques, Strategies and Patterns for Structuring JavaScript Code. [En línea] 2011. http://weblogs.asp.net/dwahlin/archive/2011/08/01/techniques-strategies-and-patterns-for-structuring-javascript-code-the-prototype-pattern.aspx.

47. **Wallace, Doug y Ragget, Isobel. 2002.** *Extreme Programming for Web Projects.* s.l. : Addison Wesley, 2002. ISBN : 0-201-79427-6.