UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 5

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

“Componente de seguridad para el proyecto Sistema Integral de Confiabilidad Operacional”

Autor

Ernesto Betancourt Fonte

Tutor

Ing. Reinier Chávez La Rosa

La Habana, Cuba

Mayo del 2012

Declaración de autoría

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

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

_____	_____
Ernesto Betancourt Fonte	**Ing. Reinier Chávez La Rosa**
Autor	Tutor

Agradecimientos

A la UCI.

A los pocos amigos que quedan, a los que se fueron y no volverán, a los que creyeron en mí y a esos que todavía no lo hacen.

A todos los que de una forma u otra me ayudaron y supieron dejar su marca en mi formación como profesional, todo lo que aprendí de ustedes viajará por siempre a mi lado como parte indisoluble de mi conciencia.

A todos los que intervinieron directa e indirectamente en la realización de este documento.

En fin a todos, mis más sinceros agradecimientos.

Dedicatoria

A mis padres por concebirme, por saber entregarme sus días y permitirme ser el desvelo de sus noches, por ser los perfectos extremos opuestos, lo que me hizo a mí ser el centro y siempre estar en equilibrio.

A mis hermanas y a mi hermano, que les sirva de inspiración y nunca dejen de superarse en la vida, no teman el dar siempre ese paso más allá en el conocimiento.

A mi familia que me vio crecer y me acompañó en el largo viaje de convertirme en lo que soy.

Y a ti, que te interesas hoy por leer el fruto de mis modestos esfuerzos.

Nulla dies sine línea…

Resumen

Con el desarrollo acelerado de la informática y las telecomunicaciones y debido al exponencial nivel de crecimiento de Internet en los últimos años, el medio de acceso a las aplicaciones informáticas está siendo enfocado hacia la Web, exponiendo servicios e informaciones sensibles a un amplio ecosistema de amenazas siempre en constante cambio que atenta contra la confidencialidad, integridad y disponibilidad de los datos. Por lo que paralelo a este desarrollo es necesario llevar a cabo métodos cada vez más eficaces para implementar y dar seguimiento a la seguridad en una aplicación. El siguiente trabajo tiene como objetivo desarrollar un componente para los procesos de autenticación y control de acceso para el Proyecto Sistema Integral de Confiabilidad Operacional (SIC).

Dicho componente pretende ser una solución potente y eficiente, englobando la mayoría de los mecanismos de seguridad necesarios para garantizar estos procesos. Este documento recoge los resultados obtenidos de la investigación realizada describiéndose los principales patrones, metodologías y tecnologías existentes en la elaboración de componentes de seguridad en aplicaciones Web. Exponiéndose además la modelación del sistema propuesto partiendo de diferentes diagramas y se analiza la implementación del mismo para la mejor comprensión del lector.

Palabras clave: confidencialidad, disponibilidad, integridad, seguridad.

Índice

Introducción

Creada en septiembre del 2002 en la Ciudad de la Habana como uno de los mayores proyectos de la Revolución, La Universidad de las Ciencias Informáticas (UCI) tiene como misión fundamental ser una "universidad innovadora de excelencia científica, académica y productiva que forme de manera continua profesionales integrales comprometidos con la patria, (…) soporte de la informatización del país y la competitividad internacional de la industria cubana del software. (…) Sobre la base de la fusión en los procesos de formación-producción-investigación como modelo en la gestión, y generalización de los resultados de los proyectos de investigación, a través de una red de centros de investigación, desarrollo e innovación a nivel nacional" (UCI, 2012). La UCI se encuentra dividida en varias facultades orientadas a determinados perfiles, en la Facultad 5 se encuentra el Centro de Consultoría y Desarrollo de Arquitecturas Empresariales (CDAE) y el Centro Productivo de Informática Industrial (CEDIN) que tiene la misión de "generar soluciones integrales para la Industria, tecnologías, productos y servicios informáticos, que cumplan con las necesidades y expectativas de los clientes (...) garantizando la formación y superación de los estudiantes y especialistas de las diferentes áreas, altamente comprometidos con la Revolución" (CEDIN, 2012).

Debido a la insuficiente disponibilidad de modelos, métodos y técnicas que permitían estimar el impacto sobre la confiabilidad operacional existente, en dicho centro se decide crear el proyecto Sistema Integral de Confiabilidad Operacional, cuyo objetivo principal es generar nuevos conocimientos en el campo de la confiabilidad operacional, así como herramientas e instrumentos que permitan mejorar la estimación del impacto de las acciones realizadas sobre los sistemas técnicos complejos durante las fases del ciclo de vida relacionadas con la operación y el mantenimiento. La Confiabilidad Operacional es calculada o valorada a través de metodologías de análisis y avanzadas herramientas de diagnóstico enmarcadas en las tecnologías específicas de una empresa.

"La Confiabilidad se percibe comúnmente como la capacidad de un activo para suministrar largos períodos de operación satisfactoria sin fallas durante su uso. En términos cuantitativos, una gestión eficiente de la Confiabilidad, permitirá disminuir la incertidumbre en el proceso de control de las fallas, ayudando a

incrementar de forma eficiente, la Disponibilidad de los activos industriales dentro de un sistema de producción" (IngeCon, 2012).

El proyecto SIC está formado por elementos autónomos interconectados mediante una estructura simple y coherente, dividiendo así el trabajo por módulos y componentes a desarrollar por estudiantes y profesionales de la universidad, dentro de dichos componentes se encuentra el de Seguridad.

Debido al exponencial nivel de crecimiento de Internet, los sistemas basados en la Web están siendo cada vez más accedidos a través de las redes por usuarios y sistemas, exponiendo servicios e informaciones sensibles a una amplia variedad de amenazas informáticas, por lo que una correcta implementación de políticas, herramientas y métodos de seguridad juegan un papel decisivo para minimizar las probabilidades de riesgos ante estos ataques cibernéticos.

El Sistema Integral de Confiabilidad Operacional (SIC) requiere de un componente que garantice la seguridad de sus funciones, al no contar con un servicio confiable en la autenticación de los usuarios, diferenciación en cuanto a la capacidad de acceder a los distintos recursos, ni un historial de seguridad que registre los diferentes eventos en el Sistema. Ante esta ***situación problemática*** se puede definir el siguiente ***problema científico***: ¿Cómo controlar y registrar la actividad de los usuarios sobre los recursos del SIC según su correspondiente perfil?

A partir del problema a resolver se define como el ***objeto de estudio***: El proceso de gestión de la seguridad en aplicaciones Web.

Para dar solución al problema se realiza el siguiente trabajo de diploma que tiene como ***objetivo general***: Desarrollar un componente para controlar y registrar la actividad de los usuarios sobre los recursos del Sistema Integral de Confiabilidad Operacional según sus correspondientes perfiles.

Siendo el ***campo de acción*** que abarcara esta investigación el proceso de gestión de la seguridad en aplicaciones Web, enmarcado en los sistemas desarrollados bajo la plataforma Java.

Para darle cumplimiento al objetivo general se han trazado las siguientes ***Tareas de investigación*:**

- Recopilación de información acerca de seguridad informática, específicamente en sistemas Web.
- Investigación de los métodos y operaciones existentes para la identificación y autenticación de usuarios.
- Investigación de los métodos y operaciones existentes para el control de acceso de usuarios a recursos.
- Selección de las tecnologías y herramientas a utilizar en el desarrollo de la aplicación.
- Identificación y especificación de los requerimientos del componente de seguridad.
- Implementación de un componente que dé solución del Problema Científico.
- Validación y prueba al componente.

La investigación estará estructurada de la siguiente forma:

- Capítulo 1: Describe el estado del arte, haciendo referencia a los conceptos inherentes a la seguridad Informática, se explican las tecnologías y herramientas actuales a considerar en la realización del componente, se seleccionaran aquellas que serán utilizadas en el desarrollo del componente teniendo en cuenta la arquitectura propuesta.

- Capítulo 2: Se describe el desarrollo de la solución propuesta a partir de la modelación de los requisitos en casos de uso, la descripción de los Actores y su interacción con el sistema a través de diagramas de actividades.

- Capítulo 3: Se define la arquitectura del componente transformando los requerimientos en especificaciones de implementación mediante diagramas de interacción y diagramas de clases. Además se especificarán los diferentes patrones utilizados para enriquecer la arquitectura del sistema.

- Capítulo 4: Se realizará la implementación de la aplicación definiendo los diagramas de componentes y de despliegue, para luego aplicar distintos tipos de pruebas a la aplicación.

Capítulo 1: Fundamentación Teórica

A lo largo de este capítulo se tomarán conceptos y características importantes referentes a la seguridad informática en aplicaciones Web, se mostrarán los sistemas gestores de seguridad más usados a nivel mundial, así como sus ventajas y desventajas. Se expondrán las tecnologías y herramientas a utilizar en la implementación del componente, así como la metodología a usar en la modelación del sistema.

1.1. Conceptos y definiciones básicas para la investigación

Se entiende por seguridad informática a todas aquellas medidas preventivas y reactivas del hombre, de las organizaciones y de los sistemas tecnológicos que permitan garantizar a través de métodos y herramientas, la confidencialidad, integridad y disponibilidad de los bienes informáticos. El concepto de seguridad en la informática es utópico porque no existe un sistema 100% seguro. Para que un sistema se pueda definir como seguro, este debe al menos velar por mantener las siguientes características o patrones de diseño seguro[1]:

1.1.1. Confidencialidad

La confidencialidad es la propiedad de prevenir el acceso a los activos informáticos o la divulgación de información a personas o sistemas no autorizados.

1.1.2. Integridad

La integridad es la propiedad que busca mantener los datos libres de modificaciones no autorizadas y que de realizarse, estas sean de la forma prevista.

[1] *Nota del autor:* Algunos de estos patrones de diseño seguro pueden encontrarse en el libro *"Security Patterns Integrating security and system engineering"* (Schumacher, 2006).

1.1.3. Disponibilidad

La disponibilidad es la característica, cualidad o condición de la información de encontrarse a disposición de quienes deben acceder a ella, ya sean personas, procesos o aplicaciones y que esta sea en el momento requerido.

1.1.4. Seguridad en aplicaciones

Las primeras aplicaciones Web fueron usadas principalmente por investigadores con fines educativos sin una gran masificación en su uso, generalmente perseguían la unión de pequeños nodos y redes locales donde preocuparse por la seguridad de las mismas no despertaba mucha atención. Con el nacimiento del comercio electrónico, las redes sociales y el uso cada vez más acelerado del ordenador para la gestión de las necesidades básicas del ser humano, hoy en día, en la llamada red de redes Internet, millones de ciudadanos comunes intercambian productos, servicios e información hacia todas las latitudes del mundo que requieren mecanismos de protección.

Aunque existen varias formas de gestionar la misma, cabe resaltar algunas de las más comunes:

- Política de Seguridad.
- Autenticación.
- Control de acceso.
- Monitoreo y registro de eventos.

El primer paso en orden de gestionar la seguridad en una aplicación informática, es la puesta en marcha de una Política de seguridad.

1.1.5. Política de seguridad

Generalmente se ocupa de asegurar los derechos de acceso a los datos y recursos con herramientas de control y mecanismos de identificación, define la lista de requisitos de seguridad de la aplicación y normas que regulen el acceso de los usuarios a los activos de la organización. Debe ser estudiada para que solo conceda el acceso necesario y no impida el trabajo de los operadores, de esta manera el sistema informático puede ser utilizado con libertad y confianza.

Dentro de las pautas referentes a elaborar una política de seguridad están el elaborar reglas y procedimientos para cada servicio de la organización y no menos importante el sensibilizar a los operadores con los problemas inherentes a la seguridad.

1.1.6. Autenticación

Es el proceso de identificación y autenticación en una aplicación, se fundamenta en darle a conocer al sistema nuestra identidad (¿Quiénes somos?) donde lo usual es tener una base de datos de los sistemas, dispositivos, clientes o usuarios que operan en la misma, usando un elementos que los distingan de los demás (identificadores) y que permitan reconocer su identidad (saber si es en realidad quien dice ser). Establece además que el usuario identificado previamente es responsable de las acciones que realizará dentro del sistema (no-repudio).

Existen cuatro tipos de técnicas para la identificación y autenticación de los usuarios:

- Algo que solamente el individuo conoce (ejemplo una contraseña).
- Algo que la persona posee (ejemplo una tarjeta magnética).
- Algo que el individuo es y que lo identifica unívocamente (ejemplo huellas digitales).
- Algo que solo el individuo es capaz de hacer (ejemplo patrones de escritura).

La más básica y ampliamente usada es mediante el uso de un ID de usuario y contraseña donde el usuario es dirigido a una pantalla de bienvenida al autenticarse correctamente o gentilmente dirigido a otras opciones en caso de error. Evidentemente mientras más factores se usen en la identificación más sólida será la misma (autenticación de varios factores), la cantidad utilizada en sistemas de gran envergadura suele ser de hasta tres factores.

1.1.7. Control de Acceso

Una vez autenticado en el sistema (comprobada su identidad), se deben saber qué acciones están permitidas para dicho usuario o proceso (¿Qué podemos hacer?) o sea a qué recursos del sistema estará autorizado a acceder. “La autorización es el proceso de determinar, a quién y a qué se le debe permitir el acceso a un recurso particular: control de acceso es el mecanismo para cumplir la autorización” (Rubinos Carvajal, 2009).

Los controles de acceso pueden ser:

- Control de acceso obligatorio (MAC).
- Control de acceso discrecional (DAC).
- Control de acceso basado en roles (RBAC).

El control de acceso de más amplio uso es el basado en roles (Role Based Access Control, RBAC), donde los usuarios son asignados a uno o varios roles mientras que los permisos y privilegios son asignados a estos roles. Se le considera una buena práctica al implementar el menor privilegio y a la separación de responsabilidades, donde se restringe el acceso a todos los recursos a cada usuario limitando solo el acceso necesario para que este opere, e ir escalando en la cadena de responsabilidades hasta eventualmente llegar a los usuarios avanzados con operaciones más complejas como el manejo de los activos informáticos sensibles para la empresa.

Los roles más usados suelen ser el de administrador y el de usuario regular, donde el primero tendrá los privilegios administrativos y el segundo solo los necesarios para cumplir sus funciones. En dependencia de los requisitos del sistema, se pueden asignar varios usuarios a cada rol o varios roles a un mismo usuario. Dentro de este proceso se deben considerar incluidos los procesos de validación de sesiones, gestión de Usuarios, Roles, Perfiles y Permisos, además de la configuración y el manejo de Reglas.

1.1.8. Monitoreo y Registro de Eventos

Consiste en llevar registros de las acciones realizadas sobre la aplicación, se les llama historiales o logs de seguridad y son sumamente importantes para reconstruir los eventos dentro del sistema y poder rastrearlos hasta el usuario o proceso que las generó. Los logs de seguridad contienen información muy útil para la detección y recuperación ante incidentes de seguridad y son usados como evidencia legal y punto de partida del análisis forense.

Ejemplo de la información contenida en un log de seguridad:

- Fecha y hora.
- Direcciones IP de origen y destino.

- Puertos de origen y destino.
- Usuarios.
- Tipos de solicitud.
- Activos solicitados.
- Errores emitidos.

1.2. Amenazas y vulnerabilidades de las aplicaciones Web

Open Web Application Security Project (OWASP) es una de las principales organizaciones en el manejo de la seguridad de las aplicaciones Web (Foundation, 2011) que propone su famoso "OWASP Top Ten" (Primeros diez), que cubre las vulnerabilidades de Internet más comunes:

1. Inyección.
2. Cross Site Scripting (XSS).
3. Autenticaciones quebradas y Gestión de Sesiones.
4. Referencias directas a objetos inseguros.
5. Cross Site Request Forgery (CSRF).
6. Seguridad mal configurada.
7. Almacenamiento de cifrado inseguro.
8. Fracaso en la restricción de acceso a URLs.
9. Insuficiente protección en la capa de transporte.
10. Redireccionamiento no validado y Forwards (envíos adelante).

1.2.1. Inyección de código

Ejemplos de fallos de inyección son: SQL, Header (encabezado) HTTP (cookies, consultas), e inyecciones de comandos OS (Sistema Operativo). Los ataques ocurren cuando datos no confiables, como consultas, comandos o argumentos, son enviados hacia un intérprete. Las aplicaciones vulnerables pueden ser engañadas para ejecutar comandos no deseados o dejando acceder al atacante a modificar datos (Foundation, 2010).

1.2.2. Cross Site Scripting (XSS)

Existen tres tipos de ataques XSS: guardados, reflejados y basados en Dom. Los ataques XSS ocurren cuando una aplicación permite datos que no están validados ni correctamente enviados al navegador Web, donde el código malicioso es ejecutado en el navegador de la víctima permitiéndole al atacante apropiarse de la sesión del usuario, robar cookies, desfasar sitios Web, redireccionar usuarios a sitios Web maliciosos, y tomar el control remoto del navegador (Foundation, 2010).

1.2.3. Autenticaciones quebradas y Gestión de Sesiones

Los usuarios son suplantados para crear huecos o fallas en el proceso de autenticación. El ataque ocurre cuando el ID de una sesión es visible para otros, los tiempos de salida no están correctamente implementados, o alguna otra falla en el sistema de autenticación es detectado (Foundation, 2010).

1.2.4. Referencias a objetos directos inseguros

El ataque ocurre cuando un usuario no autorizado puede cambiar el valor de un parámetro que hace referencia a un objeto del sistema al cual no está autorizado. Puede ocurrir para casi cualquier referencia que pueda ser alcanzada por el URL para adjuntar o incluir referencias a archivos, caminos, llaves de la base de datos, nombres de clases reflejadas, entre otros (Foundation, 2010).

1.2.5. Seguridad mal configurada

El atacante aprovecha las páginas no seguras, las cuentas predeterminadas, fallas no atendidas o alguna otra vulnerabilidad que pudiera haber sido olvidada por una apropiada configuración. Estos ataques pueden resultar en un completo compromiso del sistema (Foundation, 2010).

1.2.6. Almacenamiento de cifrado inseguro

La razón más común para este ataque es que los datos que deberían estar encriptados son almacenados en texto claro. Son el resultado del uso de algoritmos de encriptación pobres, inseguros o débiles. El uso de hashes débiles o no salteados para proteger contraseñas es una falla común que permite que la información a proteger sea vista por intrusos (Foundation, 2010).

1.2.7. Fracaso en la restricción de acceso a URLs

Este ataque toma lugar cuando un usuario no autorizado puede cambiar simplemente un URL para acceder a una página privilegiada. Los atacantes suelen buscar funciones administrativas para emplear este tipo de ataque. Los Links pueden ser obtenidos desde campos ocultos, código del lado del cliente, robots.txt, archivos de configuración, ficheros estáticos de XML, acceso a directorios, entre otros (Foundation, 2010).

1.3. Estudio de las Tecnologías y Herramientas actuales

1.3.1. La tecnología Java

La tecnología Java es una tecnología madura y eficaz, que permite a los desarrolladores crear software en una plataforma y ejecutarlo después en otra, resulta ideal para ser aplicada a redes por su versatilidad y eficiencia. La misma está compuesta por dos partes: El lenguaje de programación orientado a objetos y la plataforma basada en la filosofía "write once, run anywhere" (escríbelo una vez, córrelo dondequiera).

1.3.1.1. Lenguaje de programación Java

“Orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico” (García de Jalón, 1999). Java es un lenguaje de programación de alto nivel diseñado para que un programa escrito en él, sea ejecutado independientemente de la plataforma de hardware, software o sistema operativo en la que se esté ejecutando.

Características más importantes:

- Independiente de la plataforma.
- Ideal para Internet por lo integrado que tiene el protocolo TCP/IP.
- Los programas escritos en Java no pueden ser atacados por virus, pues para que estos tengan efecto deben utilizar rutinas de acceso directo a memoria, que Java no tiene.
- Capacidad de realizar varias tareas a la vez; esto significa que un programa puede dividirse en varios fragmentos para ser más eficiente en realizar su tarea.

1.3.1.2. La plataforma Java

Es una plataforma sólo de software que es ejecutada sobre otras plataformas de hardware dicha plataforma posee dos componentes: La máquina virtual de Java (JVM) y las APIs de Java (Application Programming Interface):

Máquina Virtual Java (Java VM o JVM): Es un entorno de ejecución que no es ni interpretado, ni compilado; es un sistema híbrido que se compila en un formato independiente al de la máquina denominado “bytecode” (código de bytes) para luego ser interpretado por la máquina virtual.

Interfaz de Programación de Aplicaciones (API): Es una colección de componentes software, agrupados en bibliotecas o paquetes de componentes que proporcionan una multitud de características útiles para el diseño de interfaces gráficas, acceso a redes, entre otros. El API de Java proporciona paquetes referentes a la seguridad que implementan o dan acceso a herramientas de alto y bajo nivel.

1.3.1.3. La seguridad en el entorno de desarrollo Java

La tecnología de seguridad de Java (Java Security o JDK) incluye un amplio conjunto de APIs, herramientas e implementaciones de algoritmos de seguridad de uso común, así como mecanismos y protocolos abarcando una amplia gama de áreas que incluyen criptografía, comunicaciones seguras, autenticación y control de acceso. Algunas de ellas son:

- Java Generic Security Services (Java GSS-API).
- Java Secure Socket Extension (JSSE).
- Simple Authentication and Security Layer (SASL).
- SSL/TLS-based RMI Socket Factories.

Estos componentes no son explicados con detalle debido a que no forma parte del objetivo principal de este trabajo, pero puede ser encontrada una amplia documentación disponible (Oracle, 2011).

1.3.2. Frameworks usados bajo la tecnología Java.

"En el desarrollo de software, un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otros softwares para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Un framework representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio" (UCIPedia, 2007).

1.3.2.1. JAAS

(Java Authentication and Authorization Service) es una API que permite extender a las aplicaciones su arquitectura de seguridad para los módulos de autenticación y autorización. El mecanismo usual para autenticarse contiene información de dónde se origina y quién emite el fragmento de código, JAAS implementa una capa de verificación extra adicionando una marca acerca de quién ejecuta el código, extendiendo así el vector de verificación.

El sistema de administración JAAS consiste en dos ficheros:

- *.login.conf: Específica cómo conectar módulos de autenticación dentro la aplicación.
- *.policy: Específica cuáles Identidades (Usuarios o programas) son los que poseen los permisos.

1.3.2.2. OWASP Java Enterprise Security API

La ESAPI (Enterprise Security API) está diseñada para hacerse cargo automáticamente de muchos de los aspectos de la seguridad en las aplicaciones, portando una colección gratis y de código abierto de métodos de seguridad. Aunque está programada para Java, ESAPI es independiente del lenguaje donde se implementa (Foundation, 2011).

Características principales que posee:

- Criptografía.
- Filtros.
- Reglas de validación.
- Etiquetas JSP.
- Rutinas seguridad.

1.3.2.3. Jasypt

Jasypt (Java Simplified Encryption), es una biblioteca Java que permite realizar encriptación de datos.

Algunas de sus características son:

- Sigue el estándar RSA para criptografía basada en contraseña y proporciona técnicas de cifrado unidireccional y bidireccional.
- API abierta que puede ser usada con cualquier proveedor JCE.
- Seguridad elevada para contraseñas de usuarios.
- Varios tipos de cifrados (texto, binario, numérico).
- Proporciona herramientas para cifrado sin ninguna configuración previa así como una alta configuración para usuarios más experimentados.
- Integración con Hibernate 3 y Spring Security.
- Estándares avanzados de seguridad.
- Criptografía de alto rendimiento.

1.3.2.4. JGuard

JGuard es una biblioteca que provee autenticación y autorización para aplicaciones Web Java. Está construida sobre el framework JAAS. Es una herramienta flexible que permite diferentes modos de configuración permitiéndole al sistema de seguridad ser independiente de la forma en que son almacenadas las credenciales del usuario, así las mismas pueden ser guardadas en una base de datos relacional, en ficheros XML, o servidores LDAP, manteniendo abierta la posibilidad de cambiar de una fuente de datos otra con facilidad.

Características principales:

- Autenticación y autorización basada en roles.
- Construido totalmente sobre Java Security y compatible con la especificación JSE.
- Autenticación externa (LDAP, JDBC, Kerberos, AD) o configurable vía XML.
- Integración con Struts, DWR, JEE.
- Configuración basada en XML o bases de datos (JDBC).

- Módulo de autenticación con Captcha y administración básica de certificados.

1.3.2.5. HDIV

HDIV (HTTP Data Integrity Validation) validación de la integridad de datos HTTP, es un marco de seguridad que implementa el principio de "Security By Design" (Seguridad mediante el diseño) o "Security By Default" (seguridad por defecto), esto quiere decir que sin alterar el modelo de programación del marco, implementa de forma automática para los programadores validaciones de seguridad que solucionan o mitigan las principales vulnerabilidades Web. Está disponible para aplicaciones desarrolladas en Struts, Spring MVC y JSTL, si se requiere de su uso en otras aplicaciones desarrolladas en otros marcos es necesario crear modificaciones extras en las JSP (HDIV, 2011).

Funcionalidades:

Garantiza la integridad de los datos generados por el servidor que no deben ser modificados por el cliente (ejemplo: links, campos ocultos, valores combinados, radioButtons, páginas de destino, entre otros) por lo que elimina la mayoría de las vulnerabilidades basadas en la manipulación de parámetros. Elimina el riesgo originado por los ataques del tipo XSS e Inyección de SQL mediante validaciones genéricas de los datos editables (ejemplo: texto y textarea).

Posee un token anti-cross-site para evitar la falsificación de peticiones (CSRF), al usar una cadena denominada token que se coloca en cada formulario y enlace de respuesta HTML. Esta cadena es generada al azar y cambiada por cada página visitada, ofreciendo una excelente protección debido a que no sólo el sitio comprometido debe saber la dirección URL y el formato de solicitud válido, sino que también debe conocer la cadena aleatoria generada para el sitio de destino.

1.3.2.6. ApacheShiro

Shiro (que en japonés significa castillo) es un framework de seguridad Java que provee un modelo de dominio por defecto para modelar Usuarios, Roles y Permisos similar a Spring Security. Desarrolla un controlador base llamado “JsecAuthBase” para cada controlador que se desee asegurar y provee un bloqueo al control de acceso para el manejo de Roles.

Características principales:

Almacenamiento de sesiones: Los objetos de las sesiones son basados en POJO, “esto permite configurar dónde estarán los datos de las sesiones, ejemplo: el archivo system (sistema), la caché de una empresa, una base de datos relacional, entre otros” (Foundation, 2010).

Retención de la dirección del Host (Huésped o anfitrión): “Shiro es capaz de retener la dirección IP del host donde fue iniciada la sesión, permitiendo determinar la ubicación del usuario” (Foundation, 2010).

Soporte para vencimiento por inactividad: “Las sesiones expiran por inactividad y pueden ser prolongadas mediante un método llamado touch()” (Foundation, 2010).

Uso Web transparente: “El soporte Web de Shiro implementa interfaces Http y todas las APIs asociadas. Permitiendo el uso de sesiones en aplicaciones Web ya existentes sin necesidad de cambiar el código Web existente” (Foundation, 2010).

1.3.2.7. Spring Security

Spring Security es un framework de seguridad que ofrece herramientas y características para implementar la seguridad en aplicaciones Web para Java. Anteriormente conocido como Acegi Spring Security, es el resultado devenido de una evolución de Acegi presentando una autenticación potente, altamente personalizable y con un marco de control de acceso, posee un estándar para asegurar las aplicaciones basadas en Spring, es un producto maduro y ampliamente utilizado y probado. Es también fácil de aprender, implementar y administrar, con un espacio de nombres dedicada a la seguridad, proporciona directrices para la mayoría de las operaciones más comunes, lo que permite conformar aplicaciones de seguridad completas en tan sólo unas pocas líneas de código.

Principales características de autenticación:

- Soporte para LDAP.
- Single Sing-On (Servicio de Autenticación Central).

- Módulo de autenticación JAAS.
- Soporte para HTTP o HTTPS.
- Soporte CAPTCHA para la detección de los usuarios humanos.
- Central de Servicio de autenticación (CAS).
- Autenticación anónima, lo que significa que a las sesiones no autenticadas se le asignan una identidad de seguridad.
- "Remember me" (recuérdame) a través de cookies HTTP.
- Soporte para sesiones simultáneas, lo que limita el número de inicios de sesión simultáneos autorizados.
- Soporte para la personalización y adición de complementos de autenticación personalizados.

1.4. Valoración del estudio del Estado del arte

En el caso específico del Sistema Integral de Confiabilidad Operacional las principales herramientas adoptadas para desarrollar la aplicación fueron el entorno de Grails (Groovy on Rails) para gestionar la lógica de negocio y Google Web Toolkit (GWT) para la capa de presentación y visualización, por lo que todo análisis hecho fue siempre en aras de facilitar una integración entre estas dos tecnologías que inicialmente no están concebidas para operar juntas en una aplicación, lo que supuso un reto para los desarrolladores de la misma.

Los marcos de trabajo expuestos integran un conjunto de componentes que resuelven muchos de los escenarios en los que nos podríamos encontrar a la hora de gestionar la seguridad, tales como: encriptación de datos, Autenticación, Autorización, entre otros, esto permite agilizar el proceso de implementación y obtener buenos resultados en breves períodos de tiempos. Aunque suelen ser soluciones genéricas, la mayoría de los casos estudiados abarcan las operaciones propuestas por las bibliotecas de seguridad o incluso las implementan dentro de sí mismos, como es el ejemplo de Spring Security que posee un método de autenticación basado en JAAS, y que además abarca la mayoría de las APIs de seguridad de Java. Por lo que se implementará el módulo basado en las funcionalidades propias que nos brindan estos frameworks de desarrollo.

HDIV aunque es un framework relativamente nuevo ya es un referente mundial dentro del mundo de la seguridad Web Java que ha sido incluido dentro de los 10 primeros lugares como solución de vulnerabilidades publicado por OWASP para entornos Java (OWASP, 2007). Posee un mecanismo de seguridad basado en Spring Security, pero no posee la totalidad de sus funciones, además actualmente no tiene soporte para Grails por lo que sería complicado buscar una integración dentro del framework de desarrollo escogido.

Apache Shiro a pesar de su solidez y su semejanza con Spring Security, uno de los framework de seguridad más maduros, no soporta funcionalidades extras como la jerarquía de roles, ni etiquetas y servicios para Grails como lo hace su homólogo. Su método de seguridad a pesar de brindar muchas otras facilidades, está basado básicamente en crear un controlador base por cada controlador a asegurar, como antes se mencionaba la fusión de Grails y GWT trae sus costos, donde la funcionalidad de los mismos como únicos gestores de la lógica de control es combinada con otro mecanismo enfocado en el patrón Modelo Vista Presentador (MVP) dentro de GWT para el manejo de las vistas, por lo que no se optará por el uso de este framework en el desarrollo del componente.

Spring Security es actualmente la solución de seguridad de mayor uso para la plataforma Java, es soportado por la compañía Spring que a su vez desarrolla Grails y a pesar de implementar mecanismos de seguridad para la mayoría de las operaciones existentes, y de mantener un plugin llamado Spring Security Core precisamente dado a facilitar la integración con Grails, presenta el mismo inconveniente que Apache Shiro, posee una gestión del control de acceso orientado a controladores que no sería compatible con la arquitectura definida para el desarrollo del sistema y al buscar una integración se perderían muchas de sus funcionalidades por lo que no sería factible su uso en el componente.

Por lo que surge la necesidad de crear nuevos conocimientos en este ámbito e investigar los métodos posibles de implementación del control de acceso y el manejo de recursos bajo las operaciones existentes en Grails y GWT.

1.5. Metodologías y herramientas para el desarrollo del software

1.5.1. Framework de Desarrollo

Como antes se mencionaba el Framework de desarrollo escogido para desarrollar la aplicación fue el entorno de Grails.

Grails es un Framework de código abierto para el desarrollo de aplicaciones Web que utiliza el lenguaje de programación Groovy, que a su vez está basado en la plataforma Java. Pretende ser un marco de alta productividad usando paradigmas como "codificación por convención" (convention over configuration) o “no te repitas a ti mismo” o (Don’t Repeat Yourself, DRY) proporcionando un entorno de desarrollo independiente y ocultando muchos de los detalles de configuración al desarrollador (Rocher, 2012).

Proporciona un marco de alta productividad mediante el uso de tecnologías Java como Hibernate y Spring en una interfaz simple y consistente. Posee un entorno de desarrollo completo que incluye un servidor Web, donde todas las bibliotecas requeridas son parte de la distribución propiciando un despliegue de forma automática, además cuenta con métodos dinámicos en varias clases a través de mixins.

Un mixin es un método que se añade a una clase dinámicamente como si la funcionalidad se hubiese compilado en el programa. Estos métodos dinámicos permiten a los desarrolladores realizar las operaciones sin tener que implementar interfaces o extender de las clases base. Esto quiere decir que Grails proporciona métodos dinámicos basados en el tipo de clase (ejemplo: para las clases de dominio existen métodos para automatizar las operaciones de persistencia, como guardar, eliminar y buscar “get()”, “set()”, “FindBy()”, entre otros.

¿Qué hace Grails automáticamente por la seguridad?

Es extremadamente seguro e inmune a la mayoría y más comunes formas de saturación de buffer y al uso de URL malformados. Todo el acceso estándar a la base de datos vía objetos de dominio automáticamente escapa a los ataques por inyección de código SQL, gracias a Hibernate que es la tecnología subyacente a los objetos de dominio GORM. Los links de Grails para crear etiquetas (link, form, createLink, createLinkTo, entre otros) utilizan mecanismos apropiados para prevenir la inyección de código.

1.5.2. Lenguaje de Programación

Groovy es un lenguaje de programación orientado a objetos implementado sobre la plataforma Java, se basa en los puntos fuertes de Java, pero tiene características adicionales que le aderezan flexibilidad y potencia ya que está inspirado en lenguajes como Python, Ruby, Perl y Smalltalk (SpringSource, 2011). Aumenta la productividad de los desarrolladores de código mediante la reducción de "scaffolding" (andamios) en el desarrollo Web, interfaz gráfica de usuario, bases de datos o aplicaciones de consola, simplifica las pruebas mediante el apoyo a las pruebas de unidad y bocetos fuera de la caja. Groovy usa una sintaxis muy parecida a Java y comparte el mismo modelo de objetos, de hilos y de seguridad. Desde Groovy se puede acceder directamente a todas las API existentes en Java y el bytecode generado en el proceso de compilación es totalmente compatible con el generado para la Java Virtual Machine (JVM), por tanto puede usarse directamente en cualquier aplicación Java.

1.5.3. Interfaces de Usuario

Para el desarrollo de las Interfaces de Usuario se usó Google Web Toolkit, específicamente el complemento de Google para Eclipse, que posee un conjunto de herramientas de código abierto que permite a los desarrolladores Web crear y mantener aplicaciones complejas JavaScript en Java.

Facilita las tareas de creación, reutilización y el mantenimiento de una gran cantidad de componentes AJAX y bases de código JavaScript al ofrecer a los desarrolladores la posibilidad de crear y mantener interfaces dinámicas y complejas, pero de gran rendimiento para el lenguaje de programación Java. Aparte de unas pocas bibliotecas nativas, todo el código es en este lenguaje y puede ser construido en cualquier plataforma compatible con los ficheros de creación GWT Ant (Google, 2011).

Los principales componentes de GWT son:

- El compilador de Java a JavaScript, encargado de traducir del lenguaje de programación Java para el lenguaje de programación JavaScript.
- El Modo de desarrollo, que permite correr y ejecutar aplicaciones en modo de desarrollo (la aplicación se ejecuta como Java en la JVM sin compilar a JavaScript).

- Biblioteca de emulación JRE, con las implementaciones JavaScript de las clases de uso común para la biblioteca estándar de Java (por ejemplo la mayor parte de las clases del paquete java.lang y un subconjunto de las clases del paquete java.util).
- Biblioteca de clases GWT Web UI, que posee un conjunto de interfaces personalizadas y clases para la creación de Widgets y su gestión.

Características:

- Componentes de interfaz de usuario dinámicos y reutilizables permitiendo a los programadores implementar aplicaciones a partir de clases previamente diseñadas.
- Posee un mecanismo simple de RPC (Llamada a procedimiento remoto).
- Brinda la posibilidad de gestionar la historia del navegador.
- Posee soporte para el manejo de algunas cuestiones cross-browser.
- Soporte para JUnit.
- Soporte para Internacionalización y localización.
- Es posible mezclar JavaScript en el código fuente de Java utilizando la interfaz nativa de JavaScript.

1.5.4. Metodología de Desarrollo de Software

Se utiliza la metodología RUP para el desarrollo del sistema y UML como lenguaje de modelación para describir su estructura.

El Proceso Unificado de Desarrollo RUP, permite transformar los diferentes requerimientos de los usuarios en un sistema de software. “El proceso unificado o RUP es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos” (Jacobson, 2000).

Principales características que definen a RUP:

Proceso dirigido por Casos de Uso: Con esto se refiere a la utilización de los Casos de Uso para el desenvolvimiento y desarrollo de las disciplinas con los artefactos, roles y actividades necesarias. Los Casos de Uso son la base para la implementación de las fases y disciplinas del RUP. Un Caso de Uso es una secuencia de pasos a seguir para la realización de un fin o propósito, y se relaciona directamente con los requerimientos, ya que un Caso de Uso es la secuencia de pasos que conlleva la realización e implementación de un Requerimiento planteado por el Cliente (Rueda Chacón, 2006).

Proceso centrado en la arquitectura: Define la Arquitectura de un sistema, y una arquitectura ejecutable construida como un prototipo evolutivo. Arquitectura de un sistema es la organización o estructura de sus partes más relevantes. Una arquitectura ejecutable es una implementación parcial del sistema, construida para demostrar algunas funciones y propiedades. RUP establece refinamientos sucesivos de una arquitectura ejecutable, construida como un prototipo evolutivo (Rueda Chacón, 2006).

Proceso iterativo e incremental: Es el modelo utilizado por RUP para el desarrollo de un proyecto de software. Este modelo plantea la implementación del proyecto a realizar en iteraciones, con lo cual se pueden definir objetivos por cumplir en cada iteración y así poder ir completando todo el proyecto iteración por iteración, con lo cual se tienen varias ventajas, entre ellas se puede mencionar la de tener pequeños avances del proyectos que son entregables al cliente el cual puede probar mientras se está desarrollando otra iteración del proyecto, con lo cual el proyecto va creciendo hasta completarlo en su totalidad (Rueda Chacón, 2006).

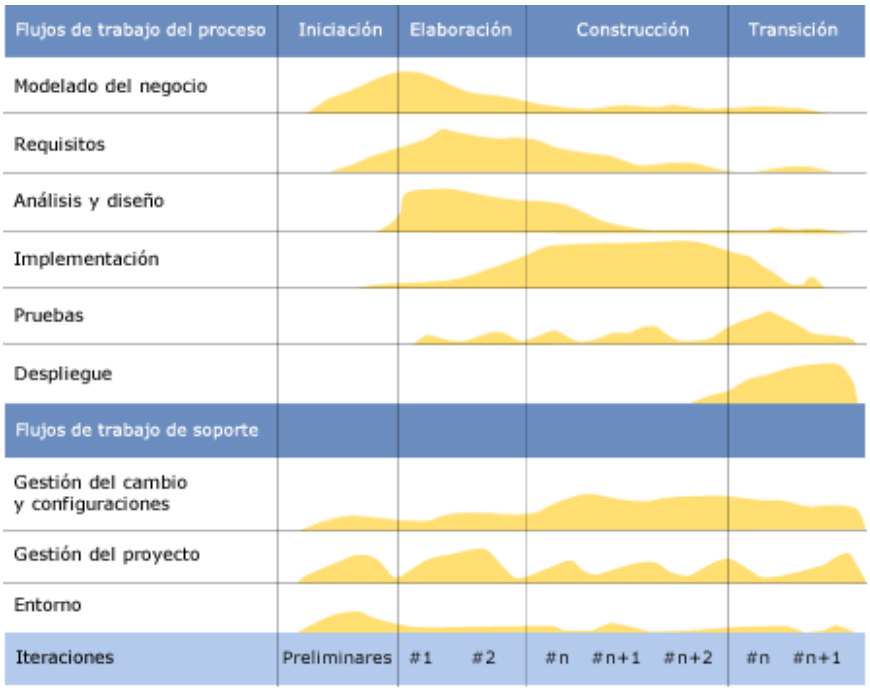


Figura 01. Flujos de trabajo del RUP.

"RUP utiliza el Lenguaje Unificado de Modelado (UML), para preparar todos los esquemas de un sistema de software. UML es una parte esencial del Proceso Unificado, pues fueron desarrollados de forma paralela" (Jacobson, 2000).

1.5.5. Lenguaje Unificado de Modelado UML

Lenguaje Unificado de Modelado permite la representación conceptual y física de un sistema; es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un plano del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. Básicamente, UML permite a los desarrolladores visualizar los resultados de su trabajo en esquemas o diagramas estandarizados siendo un lenguaje de propósito general para el modelado orientado a objetos, que

combina notaciones provenientes desde el Modelado: Orientado a Objetos, de Datos, de Componentes y de Flujos de Trabajo.

1.5.6. IDE de Desarrollo

El IDE de desarrollo escogido fue el STS (SpringSource Tool Suite) desarrollado por el proyecto Spring. Ya que ofrece la mejor potencia de desarrollo del entorno Eclipse para la construcción de las aplicaciones ofreciendo herramientas para todas las tecnologías basadas en Java, Spring, Groovy y Grails. Proporciona una interfaz gráfica en tiempo real, así como la vista de las métricas de rendimiento de las aplicaciones que permiten a los desarrolladores identificar y diagnosticar problemas. Eclipse es un entorno de desarrollo integrado de código abierto y multiplataforma para desarrollar entornos de desarrollo integrados (ECURED, 2011).

1.5.7. Herramienta CASE

Visual Paradigm for UML, es una herramienta concebida para soportar el ciclo de vida completo de los procesos de desarrollo de software a través de la representación de todo tipo de diagramas, propicia un conjunto de ayudas para el desarrollo, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de la aplicación y su documentación.

Fue diseñado para la construcción de sistemas de software a través de la utilización de un enfoque Orientado a Objetos. Esta herramienta permite aumentar la calidad del software, a través de la mejoría de la productividad en el desarrollo y mantenimiento del software. También permite la reutilización del software, portabilidad y estandarización de la documentación, además del uso de las distintas metodologías propias de la Ingeniería del Software.

Principales características:

- Es una herramienta de software libre.
- Posee un soporte completo para UML.
- Editor de Detalles de Casos de Uso.
- Administración de requerimientos.
- Modelado del proceso de negocio.

- Modelado de la base de datos.
- Soporte ORM, para la generación de objetos Java desde la base de datos.
- Genera documentación en variados formatos y reportes HTML.
- Integra a varios IDE como el STS.
- Ingeniería directa e inversa.
- Editor de formas.
- Posibilidad de presentación automática.
- Generación de código para Java y exportación como HTML.

1.5.8. Sistema gestor de Base de Datos

Para la persistencia de los datos se usó el gestor de Base de Datos PostgreSQL. Un servidor de base de datos orientado a objetos, altamente sofisticado, de alto rendimiento, estable y capacitado para lidiar con grandes volúmenes de datos. Está liberado bajo la licencia BSD y como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales que trabajan en su desarrollo.

Principales funcionalidades implementadas por PostgreSQL:

- Base de datos objeto-relacional (ORDBMS).
- Soporte a transacciones.
- Bloqueo a nivel de registro.
- Integridad referencial.
- Número ilimitado de registros e índices en tablas.
- Interfaz de administración gráfica.
- Uso optimizado de recursos del sistema operativo.
- “Triggers” (Disparadores), Vistas y procedimientos almacenados.
- Consultas y sub-consultas definidas en las cláusulas FROM.
- Backup online (Respaldos, soportes en línea).
- Sofisticado mecanismo de tuning (Ajuste).
- Soporte a conexiones de base de datos seguras (criptografía).
- Modelo de seguridad para acceso a objetos de base de datos por usuarios y grupos de usuarios.

1.6. Conclusiones parciales

En este capítulo se realizó un estudio de los principales conceptos referentes a la seguridad informática en las aplicaciones Web necesarios para la base teórica del desarrollo del componente. Se arrojó como resultado del análisis de los sistemas gestores de seguridad, la necesidad de la investigación teniendo en cuenta las características de las metodologías de desarrollo, herramientas y tecnologías escogidas pudiéndose resumir:

Se implementará un componente de seguridad basado en las funciones que brindan los siguientes marcos: Grails que usará el lenguaje de programación Groovy y será el encargado de la de la lógica de negocio. GWT que usará el lenguaje Java para el diseño de la interfaz de usuario y la visualización de los datos.

Se escogió como metodología de desarrollo del software RUP y UML para describir su estructura, siendo Visual Paradigm la herramienta escogida para realizar el modelado del sistema. El entorno de desarrollo integrado adoptado fue el STS y como sistema gestor de base de datos PostgreSQL.

Capítulo 2: Descripción de la solución propuesta

En este capítulo se dará solución a la problemática planteada haciendo uso de las herramientas seleccionadas. Se realizará el proceso de modelación del sistema, para ello se especificarán los requerimientos que debe tener la aplicación, se identificarán los actores, se describirán los casos de usos obtenidos a través de los requisitos del sistema y se construirán varios diagramas para su mejor comprensión.

2.1. Modelo del Dominio

El modelo de dominio, es un subconjunto del modelo de negocio, se utiliza cuando los procesos del negocio no pueden ser definidos con claridad, un modelo de dominio recoge los tipos de objetos más importantes que existen en el sistema, y tiene por objetivo ayudar a comprender mejor los conceptos que utilizan los usuarios, con los que se trabaja.

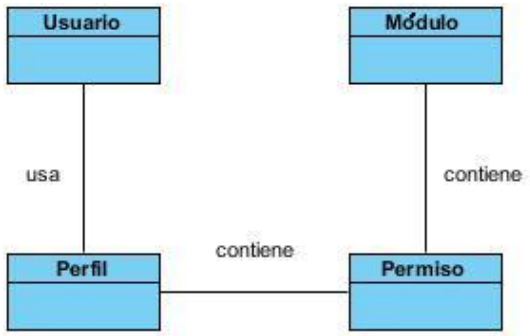


Figura 02: Diagrama del Modelo del Dominio.

La clase Usuario será el referente a todos los usuarios de la aplicación, conteniendo su información y los atributos necesarios para realizar la identificación y autorización. Esta clase usará a la clase Perfil que dictará los privilegios de los Usuarios, pudiéndose asignar varios Perfiles a los mismos. Los Perfiles contendrán Permisos, estos serán los recursos que deberán ser asegurados por el Componente de Seguridad. Para asignar un Permiso a un Perfil, este debe ser seleccionado previamente de un Módulo y

cada módulo deberá ser el encargado de gestionar los Permisos que contiene (ejemplo: creación, eliminación).

La implementación de mecanismos de seguridad trae aparejados costos, por lo que no tiene sentido invertir en algún mecanismo que no se necesita y por ende no se usará; por lo que una previa investigación de los activos a resguardar desde el comienzo del ciclo de vida del proyecto, cuando los requisitos del sistema aún están siendo definidos evita el derroche de recursos y además nos permite la mitigación de muchos riesgos. “Estos resultados, también enriquecerán la lista de requisitos de seguridad del sistema, contribuirán a la reducción del costo del proyecto, por concepto de corrección de problemas de seguridad y permitirán la creación de pruebas de seguridad más eficaces” (Quintas Santiago, 2010).

2.2. Requerimientos a satisfacer

Establecerá lo que el sistema debe o no debe hacer, identificando las funcionalidades requeridas definiendo así los límites y restricciones impuestas.

2.2.1. Requerimientos funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir:

- RF1. Autenticación al sistema:
  - ✓ RF1.1. Permitir la autenticación de los usuarios de la aplicación.
  - ✓ RF1.2. Impedir la autenticación de usuarios no autorizados al sistema.
- RF2. Gestión de los Usuarios:
  - ✓ RF2.1. Permitir la creación de un nuevo Usuario.
  - ✓ RF2.2. Permitir el acceso a los datos de los Usuarios.
  - ✓ RF2.3. Permitir la modificación de los datos de los Usuarios.
  - ✓ RF2.4. Permitir la eliminación de los Usuarios.
- RF3. Gestión de los Perfiles:
  - ✓ RF3.1. Permitir la creación de un nuevo Perfil.
  - ✓ RF3.2. Permitir la asignación de Perfiles a Usuarios.
  - ✓ RF3.3. Permitir la modificación de los Perfiles de los Usuarios.

- ✓ RF3.4. Permitir la eliminación de Perfiles.

- RF4. Gestión de los Permisos a los recursos del sistema:
  - ✓ RF4.1. Permitir la creación de Permisos.
  - ✓ RF4.2. Permitir la asignación de los Permisos a los Perfiles.
  - ✓ RF4.3. Permitir la modificación de los Permisos.
  - ✓ RF4.4. Permitir la eliminación de los Permisos.
- RF5. Gestión de los Módulos y sus recursos al sistema:
  - ✓ RF4.1. Permitir la inserción de Módulos y sus Permisos.
  - ✓ RF4.2. Permitir la asignación de los Permisos pertenecientes a un Módulo.
  - ✓ RF4.3. Permitir la modificación de los Permisos.
  - ✓ RF4.4. Permitir la eliminación de los Permisos.
- RF6. Control de Acceso:
  - ✓ RF6.1. Permitir el acceso de los Usuarios a los recursos cuyos Perfiles posean los Permisos apropiados.
  - ✓ RF6.2. Impedir el acceso de los Usuarios a los recursos cuyos Perfiles carezcan de los Permisos apropiados.
- RF7. Logs de Seguridad:
  - ✓ RF7.1. Permitir la creación de Logs de seguridad al inicio de cada acceso o intento de acceso al Sistema.

2.2.2. Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe cumplir. Estas propiedades son las características que hacen al producto atractivo, usable, rápido y confiable.

- RNF1. Apariencia o interfaz externa:
  - ✓ RNF1.1. La interfaz de usuario debe ser de fácil operación, permitiendo que no sea necesario mucho entrenamiento para utilizar el sistema.
  - ✓ RNF1.2. Permitir una experiencia agradable mediante una interfaz amigable.
- RNF2. Usabilidad:

- ✓ RNF2.1. El sistema podrá ser utilizado por personas que posean conocimientos básicos de computación.

- RNF3. Seguridad:
  - ✓ RNF3.1. Permitir validaciones sobre acciones irreversibles sobre datos sensibles, (ejemplo: eliminación de Usuarios).
  - ✓ RNF3.2. Aplicar técnicas de cifrado de información para impedir que éstas sean interpretadas por intrusos.
  - ✓ RNF3.3. El sistema garantiza el acceso a la modificación solo a usuarios autorizados.
  - ✓ RNF3.4. El sistema debe ser capaz de garantizar la integridad del material almacenado.
- RNF4. Portabilidad:
  - ✓ RNF4.1. El sistema debe ser multiplataforma.
- RNF5. Licencia:
  - ✓ RNF5.1. El sistema debe cumplir con los lineamientos necesarios para la producción de software libre y la comunidad de desarrollo y soporte, manteniendo la confidencialidad y las operaciones de la institución, ya que es la política de la universidad.
- RNF6. Legales:
  - ✓ RNF6.1. Los requisitos legales, de derecho de autor y otros se establecen a través del Centro de Informática Industrial (CEDIN).
- RNF7. Restricciones en el Diseño y la implementación:
  - ✓ RNF8.1. El sistema debe ser multiplataforma.
  - ✓ RNF8.2. El lenguaje de programación a ser usado en la implementación será Groovy.
  - ✓ RNF8.3. El sistema debe utilizar el Framework de desarrollo: Grails.
  - ✓ RNF8.4. El sistema debe utilizar el Framework de desarrollo: GWT.
  - ✓ RNF8.5. El sistema debe utilizar el IDE de desarrollo Spring Source Tool Suite (STS).
- RNF9. Software:
  - ✓ RNF9.1. Nova, Ubuntu 10.04, Debian y Windows XP o superior.
  - ✓ RNF9.2. La Máquina Virtual de Java versión 1.6.
  - ✓ RNF9.3. Grails versión 1.3.5.
  - ✓ RNF9.4. STS versión 2.5.0.

- ✓ RNF9.5. PostgreSQL versión 9.1.
- ✓ RNF9.6. GWT versión 2.2.0.
- ✓ RNF9.7. Requiere que se emplee un navegador de Internet, ya sea Internet Explorer, Mozilla Firefox, entre otros.

- RNF10. Hardware:
  - ✓ RNF10.1. Tener un mínimo PC Pentium 4, 2.4 GHz, 512 MB de memoria RAM.

2.3. Modelado del Sistema

El objetivo de la modelación del sistema es describir los procesos, existentes u observados, con el propósito de comprenderlos. Especificando qué procesos soportará el sistema, además de identificar los objetos del dominio implicado, este modelo establecerá los actores y las operaciones que llevan a cabo dentro de la aplicación.

El Sistema Integral de Confiabilidad Operacional requería de un componente de seguridad sumamente flexible, que se adaptase con total facilidad a cualquier ambiente de trabajo con el mínimo de implementación de código posible, sin perder la calidad ni la estabilidad de los mecanismos de seguridad intrínsecos. Un reto ya que cada entidad presenta un conjunto de características particulares que deben ser configuradas, lo que podría provocar posteriores problemas en la etapa de implementación del software, por lo que el componente de seguridad debió cumplir con requisitos especiales que permitieran diferentes configuraciones para el usuario final del sistema.

No es un secreto que la seguridad no puede ser considerada simplemente como una cuestión técnica que corresponde a los equipos de desarrollo de código que estén encargados de la implementación de la seguridad en las aplicaciones. La seguridad debe ser responsabilidad de todas las partes involucradas desde el inicio del proyecto de la aplicación. Por lo que en el desarrollo siempre se tuvo en cuenta por parte de todos los implicados algunos parámetros definidos como buenas prácticas de seguridad:

- Establecer un proceso de seguridad.
- Definir las metas de seguridad del producto desarrollado.
- El diseño fue teniendo en cuenta el principio del mínimo privilegio.

- Se implementaron registros de las acciones realizadas en la aplicación para poder reconstruir los eventos.
- Se buscó limitar el consumo de recursos.
- Codificación de datos sensibles.
- Construcción de distintas capas de defensa.
- Implementación teniendo en cuenta las amenazas.
- Manejo de errores de forma apropiada.
- Se consideró la seguridad como una funcionalidad del producto.

2.4. Modelo de Objetos

Describe cómo colaboran los trabajadores y entidades del sistema y es utilizado para identificar los roles dentro de la organización.

2.4.1. Actores del Sistema

Los actores interactúan con el negocio poseyendo una relación directa con los Casos de Uso del sistema. Para cumplir un estándar en la configuración del componente de seguridad se pudieron identificar dos actores del sistema (estos actores solo compiten al componente de seguridad, no al proyecto SIC en su totalidad) donde se engloban en el “Usuario Regular” todos los posibles actores que trabajaran con la aplicación, quedando para el “Usuario Administrador” aquellos usuarios con los privilegios administrativos sensibles para la configuración de la seguridad.

A continuación se muestra la descripción de los Actores:

- Usuario Regular: Podrá realizar operaciones con los privilegios mínimos requeridos según su función en el sistema:
  - ✓ Autenticarse a través de su usuario y contraseña.
  - ✓ Cambiar su contraseña.
- Usuario Administrador: Controla la configuración del sistema y podrá realizar operaciones de administración. Las funcionalidades adicionales son:
  - ✓ Gestionar Usuarios.
  - ✓ Gestionar los diferentes Perfiles del sistema.

- ✓ Gestionar los Módulos y sus Permisos al sistema.
- ✓ Gestionar el acceso a los recursos (Permisos).

2.5. Modelo de Casos de Uso

Los casos de uso proporcionan un medio para modelar los requisitos del sistema, representando los procesos que ocurren en el negocio. Modela la funcionalidad que el sistema ofrece agrupándola en descripciones de acciones ejecutadas.

2.5.1. Casos de Uso del Sistema

Se identificaron los siguientes Casos de Uso:

- Autenticar Usuario: Permite la identificación y autenticación de los usuarios de la aplicación, permitiéndoles realizar las operaciones dentro de la misma para las que tengan previa autorización, en esencia creará una sesión dentro de la misma.
- Desactivar Sesión: Permitirá a los usuarios de la aplicación terminar la sesión iniciada dentro de la misma.
- Generar Log de Seguridad: Caso de uso donde los eventos relacionados con la autenticación en el sistema serán registrados.
- Cambio de Contraseña: Utilizado por todos los usuarios de la aplicación para cambiar la contraseña de acceso de forma personal.
- Gestionar Usuario: Caso de Uso reservado para el usuario administrador, el cual le permitirá realizar la gestión de los usuarios que trabajarán dentro de la aplicación. La creación, consulta, modificación y eliminación de los datos de los mismos.
- Gestionar Perfil: Caso de Uso reservado para el usuario administrador, el cual le permitirá realizar la gestión de los Perfiles usados en la aplicación. La creación, consulta, modificación y eliminación de los datos de los mismos, así como su asignación a los usuarios.
- Gestionar Módulo: Caso de Uso reservado para el usuario administrador, el cual le permitirá realizar inclusión de los módulos, la gestión de estos y sus permisos en el sistema. Incluye la creación, consulta, modificación y eliminación.

- Gestionar Permiso: Caso de Uso reservado para el usuario administrador, el cual le permitirá realizar la gestión de los permisos del sistema. La creación, consulta, modificación y eliminación de los mismos.

2.5.2. Diagrama de casos de uso del sistema

Luego de haber definido los actores y requerimientos funcionales del sistema, se modela las relaciones y dependencias que existen entre los actores y los casos de usos definidos a partir de la captura de los requerimientos.

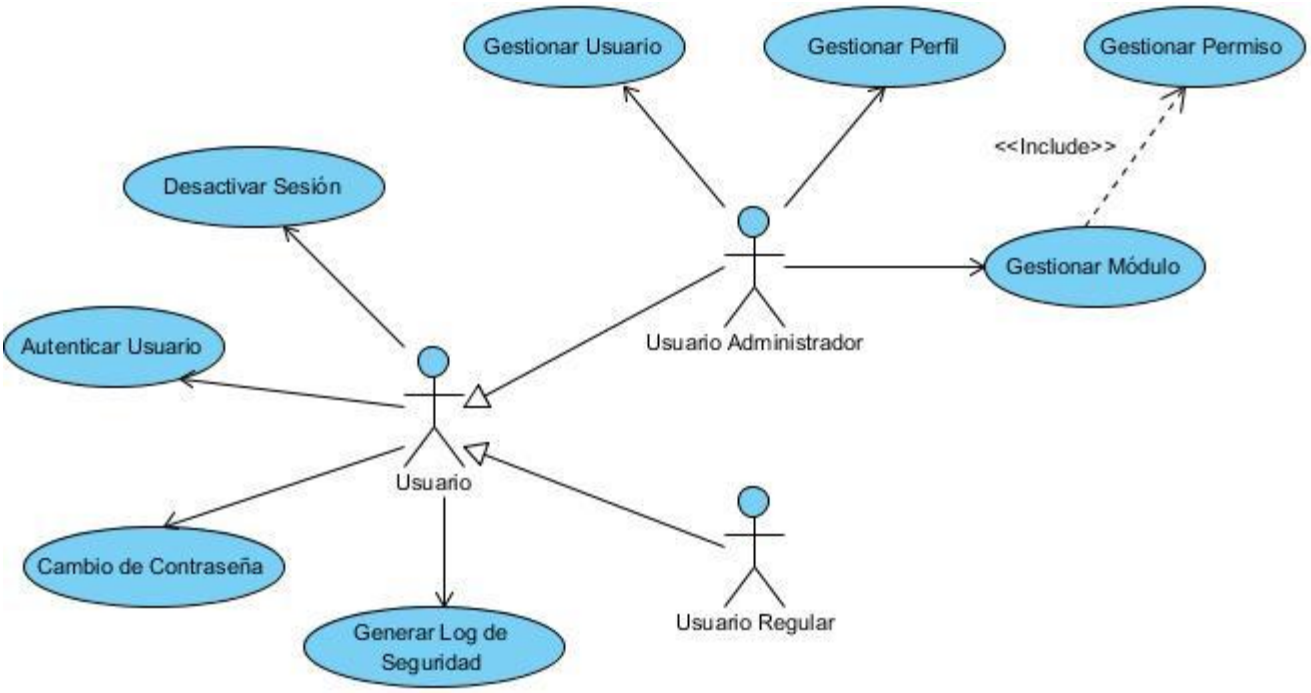


Figura 03: Diagrama de Casos de Uso del Sistema.

2.5.3. Descripción de Casos de Uso del Sistema

A continuación se realizará la descripción de los casos de usos identificados para satisfacer los requerimientos funcionales del sistema:

Caso de uso	Autenticar Usuario.

Actores	Usuario Regular, Usuario Administrador
Resumen	Este caso de uso permite la identificación del Usuario en el Sistema y, basado en la información del Perfil del usuario se le otorgarán los privilegios correspondientes que le permitirán trabajar en la aplicación.
Precondición	El usuario no puede estar autenticado.
Flujo normal de eventos	
Acción del Actor	Respuesta del Sistema
	1. El caso de uso comienza cuando el sistema solicita al usuario su autenticación mediante su usuario y contraseña.
2. Introduce su Nombre de Usuario y Contraseña y pulsa la opción "Aceptar".	3. Valida que los datos introducidos están correctos y completos. 4. Verifica el Nombre de Usuario y Contraseña. 5. Realiza la autenticación con los datos suministrados por el usuario.
Flujo alterno de eventos	
	3.1. Muestra un mensaje especificando el error cometido al introducir los datos.
	3.2. El Sistema retorna al paso 1 del flujo normal de eventos para que intente nuevamente el inicio de sesión.
	4.1. Muestra un mensaje de error "Usuario y/o contraseña incorrectos". 4.2. El Sistema retorna al paso 1 del flujo normal de eventos para que el Usuario intente nuevamente el inicio de sesión.
Post-condiciones	El Usuario se autentica en el sistema.
Prototipo de Interfaz	
Véase Anexo5.	

Caso de uso	Cambio de Contraseña.

Actores	Usuario Regular, Usuario Administrador
Resumen	Este caso de uso le permite al usuario cambiar su contraseña.
Precondición	El usuario debe estar autenticado.
Flujo normal de eventos	
Acción del Actor	Respuesta del Sistema
1. El Usuario accede a la opción Cambiar Contraseña.	2. Verifica si el usuario esta Autenticado. 3. Muestra la interfaz “Cambiar Contraseña”.
4. Introduce la Contraseña nueva, su confirmación y pulsa la opción “Aceptar”.	5. Valida que los datos introducidos están correctos y completos. 6. Registra la contraseña del Usuario. 7. Muestra el mensaje “Contraseña cambiada con éxito”.
Flujo alterno de eventos	
	5.1. Muestra un mensaje especificando el error cometido al introducir los datos. 5.2. El Sistema retorna al paso 3 del flujo normal de eventos para que el Usuario suministre los datos requeridos.
Post-condiciones	El Usuario Cambia su Contraseña.
Prototipo de Interfaz	
Véase Anexo6.	

2.6. Diagrama de actividades

Los diagramas de actividades representan qué es lo que ocurre durante un proceso del negocio, simbolizado por los actores y las actividades que los mismos realizan. Se utiliza para modelar el comportamiento del sistema, ya sea un caso de uso, una clase, o un método complicado.

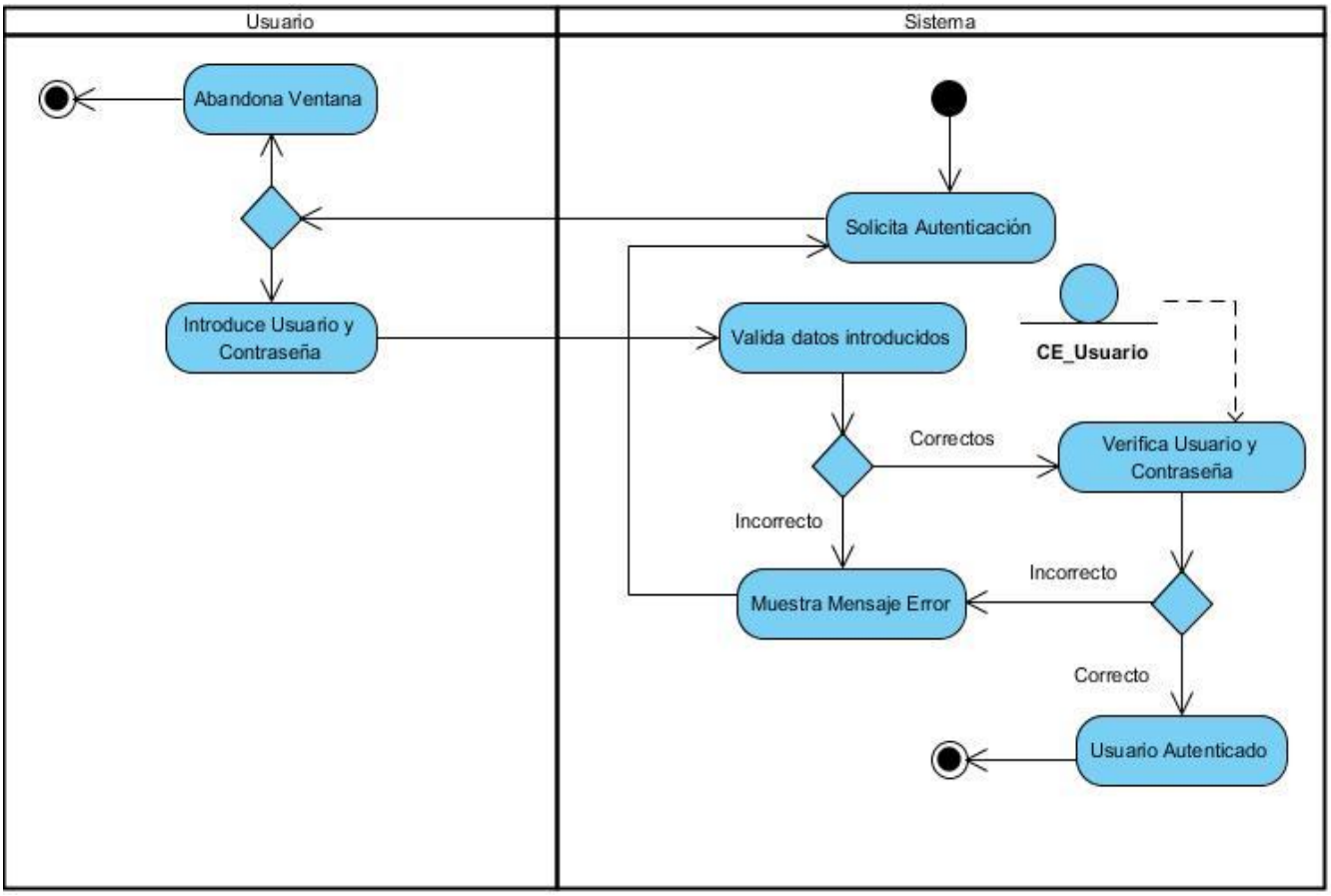


Figura 03: Diagrama de Actividad Caso de Uso Autenticar Usuario.

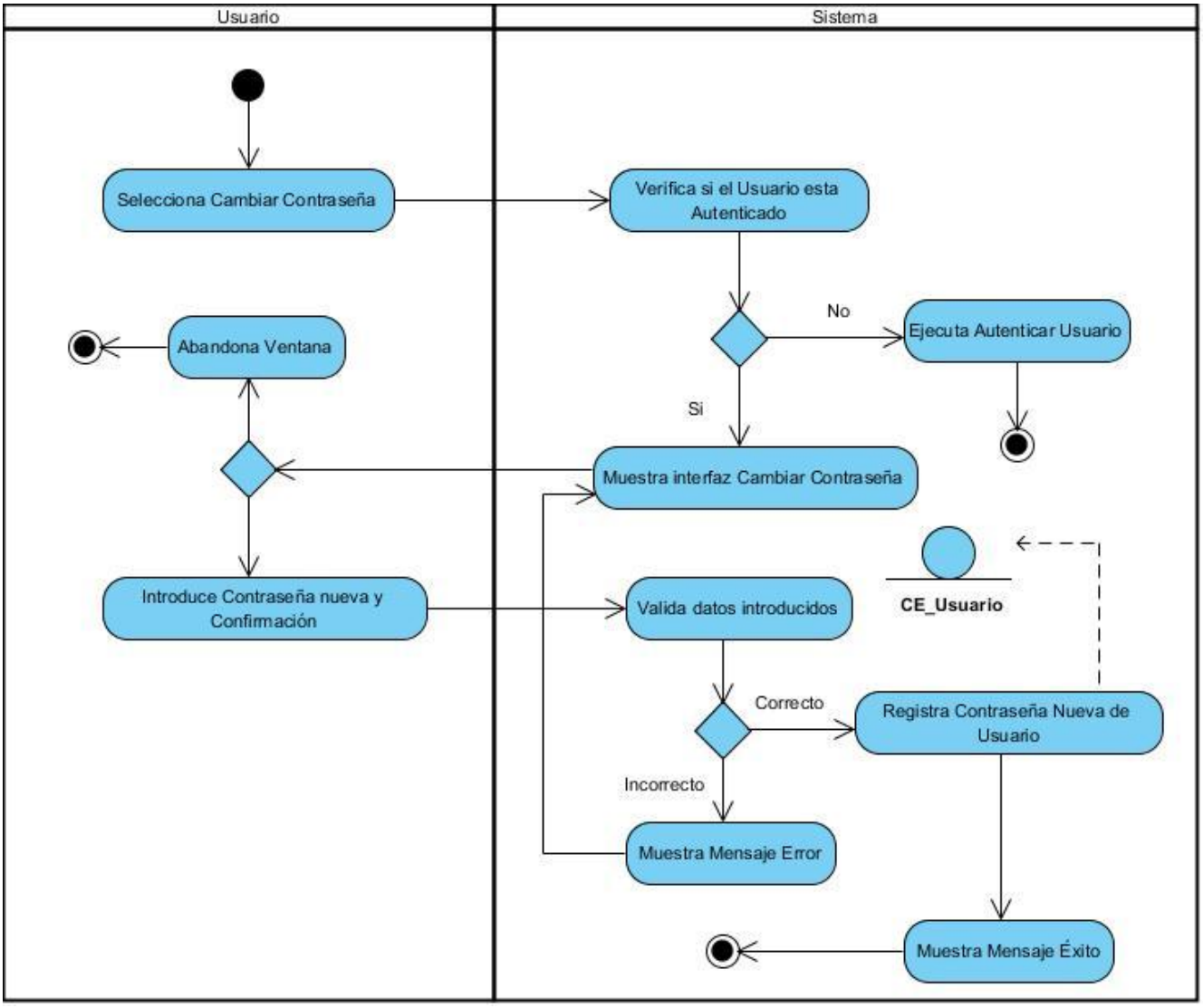


Figura 04: Diagrama de Actividad Caso de Uso Cambio de Contraseña.

2.7. Conclusiones parciales

En este capítulo se realizó la modelación del dominio, exponiéndose los conceptos de Usuario, Perfil, Módulo y Permiso. Se analizaron y valoraron los requisitos necesarios para la implantación del componente, construyéndose diferentes artefactos como los diagramas de caso de uso y diagramas de actividades que formarán los cimientos de las próximas fases de desarrollo.

Capítulo 3: Análisis y Diseño del sistema

En este capítulo se define la arquitectura del sistema y tiene como objetivo trasladar los requisitos en especificaciones de implementación. En el análisis se transformarán los casos de uso en clases para llevar a cabo las funcionalidades contenidas en los mismos. Y en el diseño se refinará el análisis para poder implementar los diagramas de interacción y de clases del diseño, logrando una descripción del diseño de la solución propuesta, para más tarde en el Capítulo 4 conformar el Modelo de Implementación y Despliegue de la arquitectura.

3.1. Modelo de Diseño

El diseño debe definir una solución que satisfaga de modo efectivo y eficiente los requisitos especificados en el análisis, incorpora nuevos artefactos (nuevas clases, atributos y operaciones) y se tiene en cuenta la plataforma tecnológica concreta sobre la que se construirá el sistema informático.

3.1.1. Diagramas de interacción

Expresan cómo pueden desarrollarse las relaciones entre las clases que intervienen en un caso de uso, se hace por cada escenario del caso de uso, se puede optar entre diagrama de secuencia y de colaboración para flujo principal y flujos alternativos.

3.1.1.1. Diagramas de secuencia

El Diagrama de Secuencia del Sistema refleja la interacción que tienen los usuarios con el sistema, enfatizando la existente entre los objetos y los mensajes que intercambian entre sí, junto con el orden temporal de los mismos.

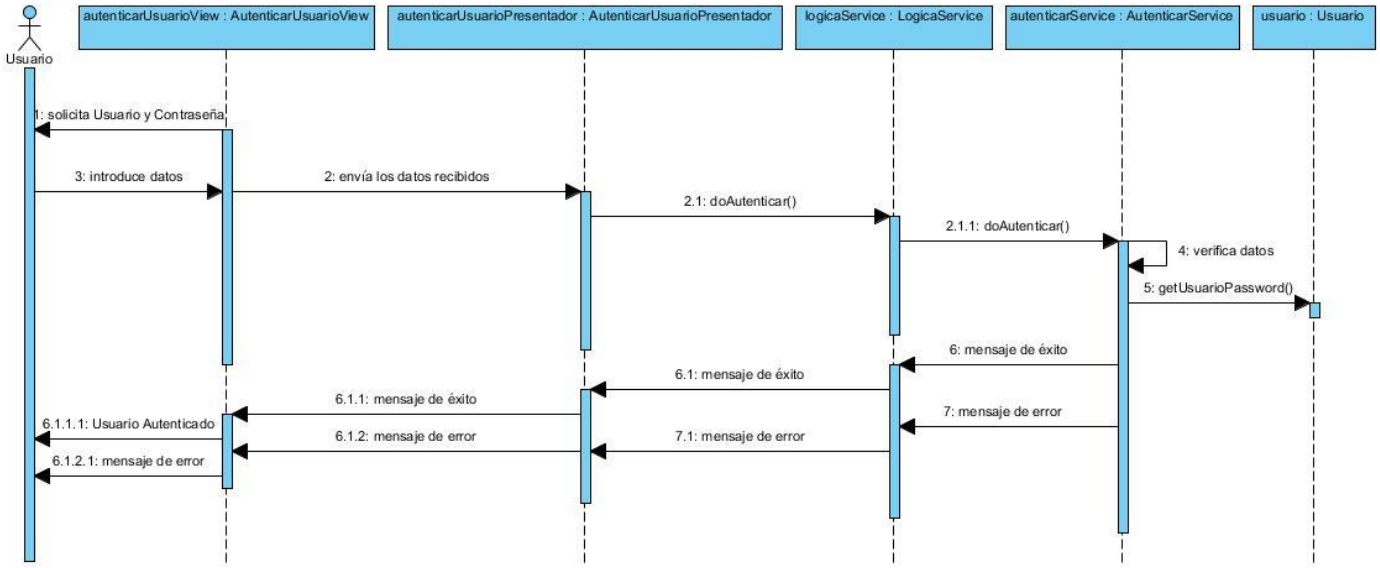


Figura 04: Diagrama de Secuencia CU Autenticar Usuario.

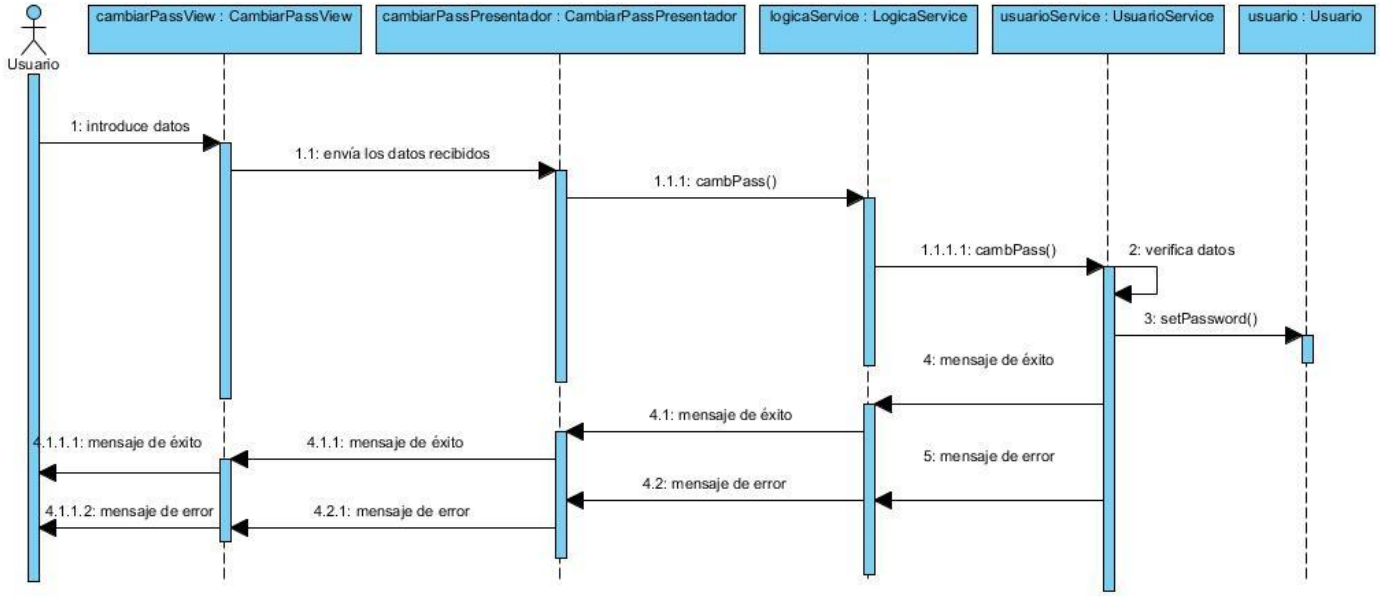


Figura 05: Diagrama de Secuencia CU Cambiar Contraseña.

3.1.2. Diagramas de clases del Diseño

Un diagrama de clases muestra un conjunto de clases (descripciones de objetos que comparten características comunes) que componen el sistema y cómo se relacionan estructuralmente entre sí desde

un punto de vista lógico, son los diagramas principales para el flujo de trabajo, análisis y diseño y se basan fundamentalmente en los diagramas de interacción.

3.1.2.1. Diagrama de clases en paquetes

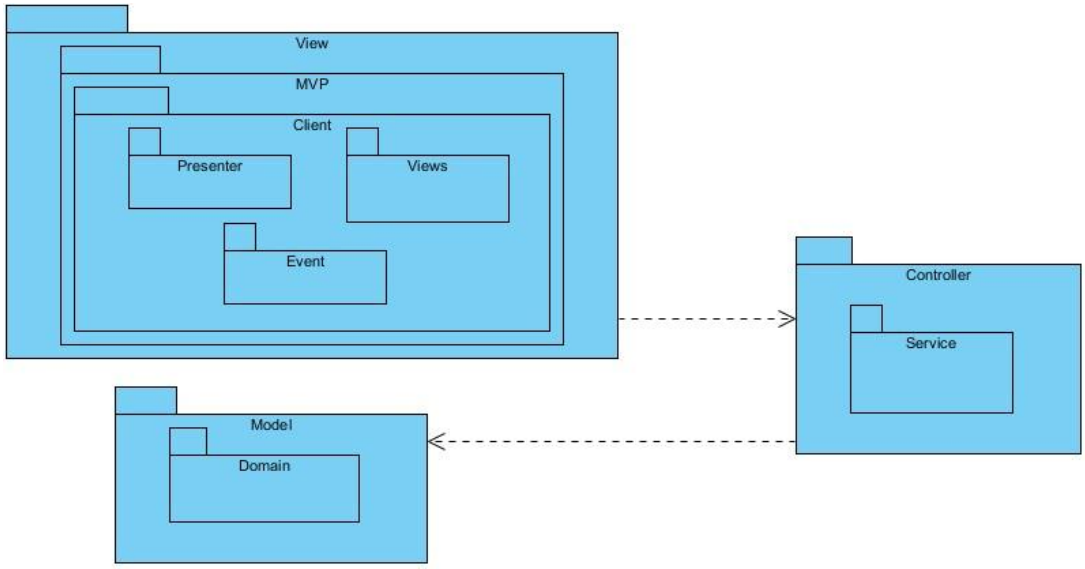


Figura 06: Diagrama de clases en paquetes.

Explicación de los diferentes paquetes:

Controller: “Este paquete cuenta con un subsistema que agrupa las clases encargadas de la selección de los eventos para cada Caso de Uso, darle interpretación a estos eventos y darle a las otras dos partes que conforman el sistema los mensajes de la acción a realizar, previamente procesada por la clase correspondiente dentro de este paquete. Estas acciones a realizar van desde informar a las interfaces que deben mostrar y en qué momento hasta informarle a las partes del modelo las acciones a realizar” (Socorro Borges, 2012).

View: “Este paquete cuenta con la agrupación de tres subsistemas que responden a la aplicación del patrón MVP en la capa de presentación. Estos subsistemas van a agrupar las clases que se utilizan para mostrar la información que desea ver el usuario y los recursos que necesitan para la actualización de las

vistas. Estos tres subsistemas que se agrupan en este paquete son los encargados de manejar la interacción del usuario con la aplicación" (Socorro Borges, 2012).

Model: "Este paquete cuenta con un subsistema que es el encargado de representar detalladamente la información con la que el sistema debe operar" (Socorro Borges, 2012).

3.1.2.2. Diagramas de clases Persistentes

Se presenta el diseño de las clases persistentes del sistema, que propician una aproximación al diseño del modelo de datos. Para el desarrollo del sistema se utilizaron cuatro clases entidades llamadas: Usuario, Perfil, Módulo y Permiso conteniendo todos los tipos de datos necesarios para el manejo de la información.

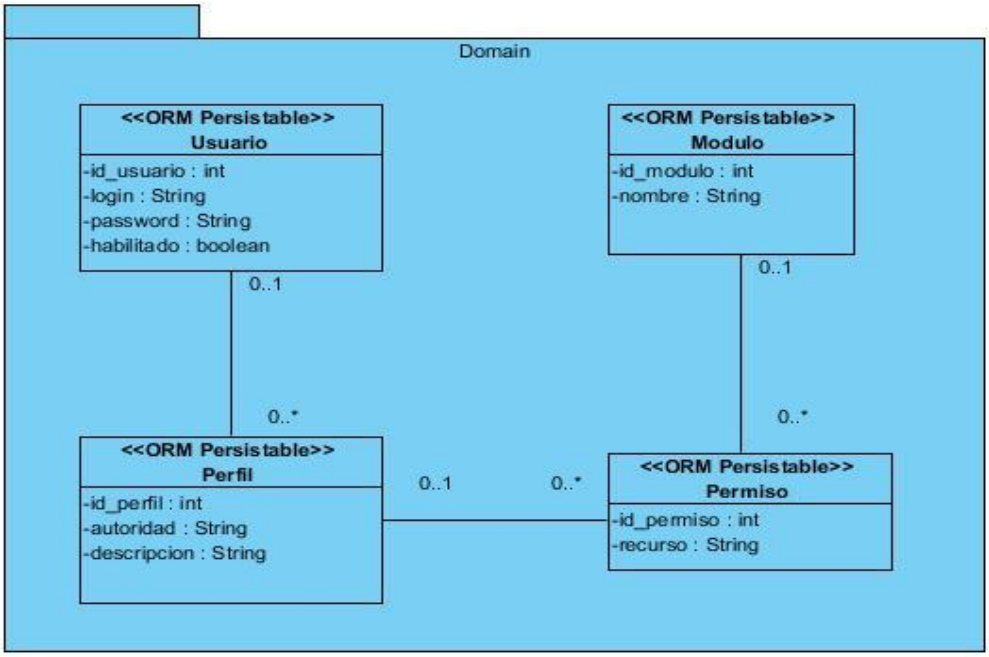


Figura 07: Diagrama de Clases Persistentes.

3.1.2.3. Descripción de la funcionalidad de las clases

Nombre: Usuario	
Tipo de clase: <<ORM Persistable>>	
Atributo	Tipo
id_usuario	Integer
login	String
password	String
Para cada responsabilidad:	
Nombre:	
Descripción:	

Nombre: Perfil	
Tipo de clase: <<ORM Persistable>>	
Atributo	Tipo
id_perfil	Integer
autoridad	String
descripción	String
Para cada responsabilidad:	
Nombre:	
Descripción:	

Nombre: Modulo	
Tipo de clase: <<ORM Persistable>>	
Atributo	Tipo
id_modulo	Integer
nombre	String
Para cada responsabilidad:	

Nombre:	
Descripción:	

Nombre: Permiso	
Tipo de clase: <<ORM Persistable>>	
Atributo	Tipo
id_permiso	Integer
recurso	String
Para cada responsabilidad:	
Nombre:	
Descripción:	

3.1.2.4. Diagramas de clases

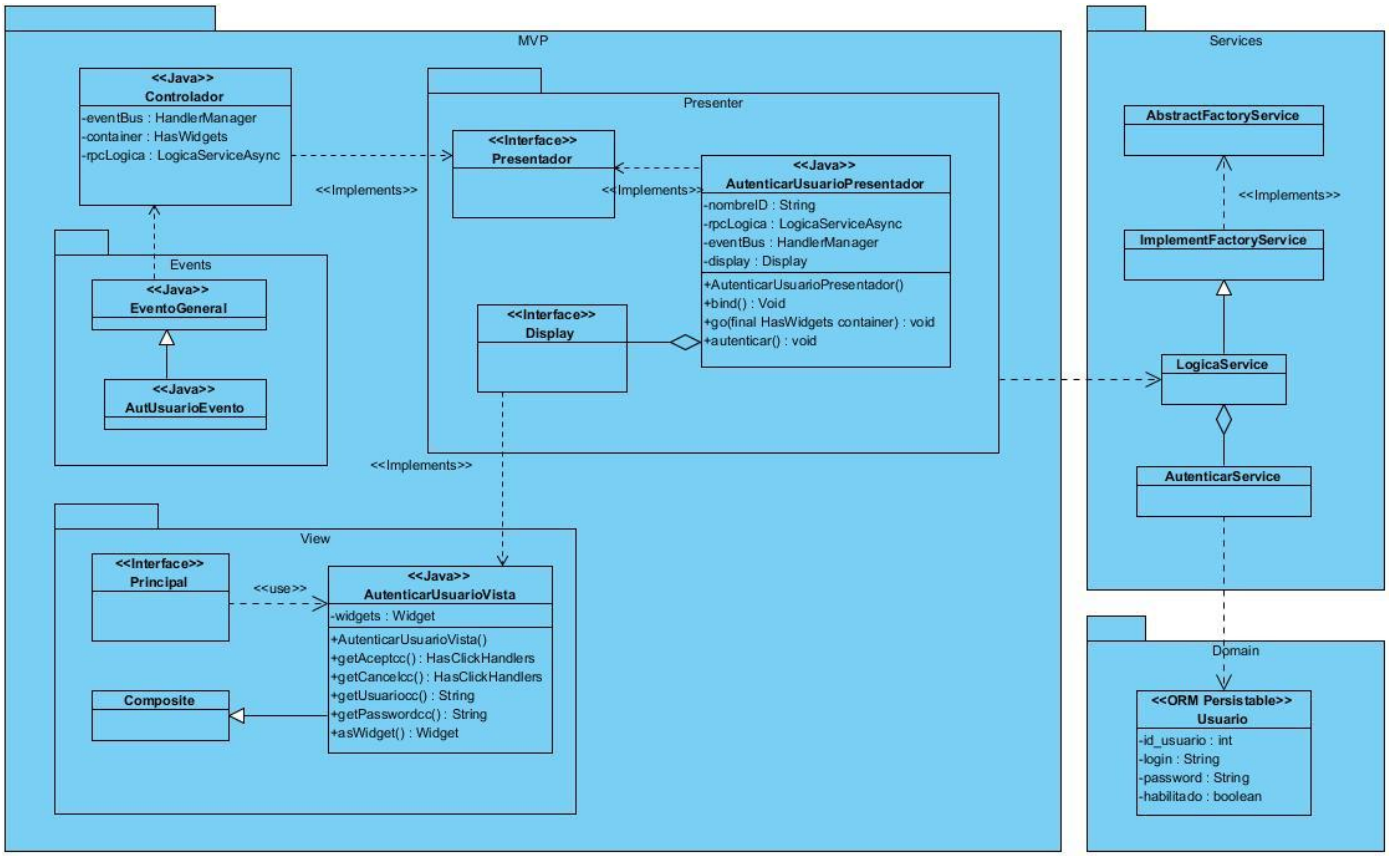


Figura 08: Diagrama de Clases Caso de Uso Autenticar Usuario.

Descripción de la funcionalidad de las principales clases

Nombre: AutenticarService	
Tipo de clase: Servicio	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	doAutenticar(login,password)

Descripción:	Autentica al usuario a partir de un login y password dado.
Nombre:	doDesAutenticar()
Descripción:	Desautentica al usuario del sistema.

Nombre: AutenticarUsuarioView	
Tipo de clase: Interfaz	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	AutenticarUsuarioVista()
Descripción:	Constructor de la vista.
Nombre:	getAceptar()
Descripción:	Envía el evento aceptar del botón al presentador.
Nombre:	getCancelar()
Descripción:	Envía el evento cancelar del botón al presentador.
Nombre:	getLogin()
Descripción:	Obtiene el login proporcionado y lo envía al presentador de la vista.
Nombre:	getPassword()
Descripción:	Obtiene el password proporcionado y lo envía al presentador de la vista.

Nombre: AutenticarUsuarioPresentador	
Tipo de clase: Presentador	
Atributo	Tipo
nombreID	String
Para cada responsabilidad:	
Nombre:	autenticar()
Descripción:	Gestiona los correspondientes eventos y envía la información para ser procesada por el servicio AutenticarService.

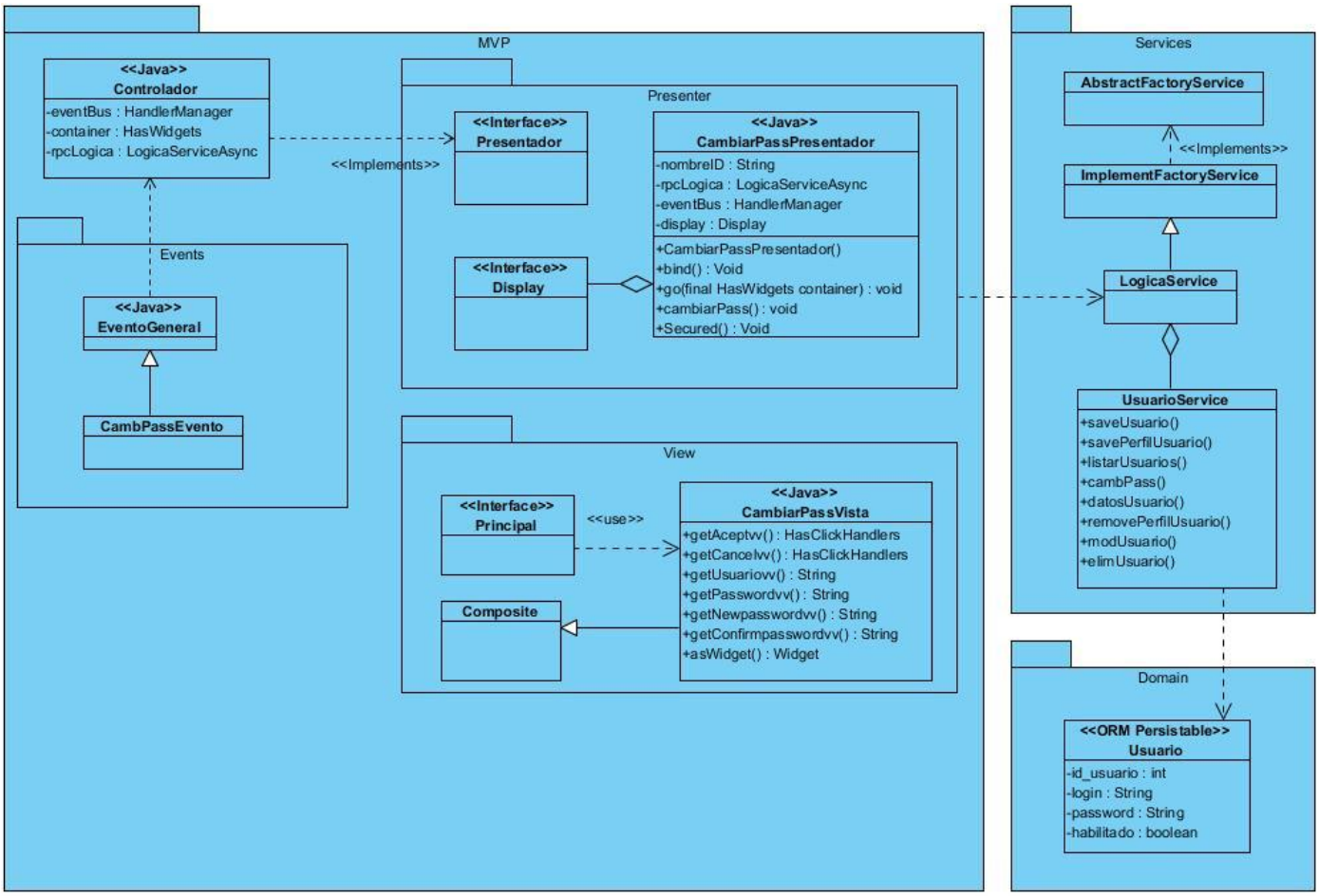


Figura 09: Diagrama de Clases Caso de Uso Cambiar Contraseña.

Descripción de la funcionalidad de las principales clases

Nombre: UsuarioService	
Tipo de clase: Servicio	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	SaveUsuario (login, password, habilitado)
Descripción:	Guarda la información del nuevo usuario en la base de datos.

Nombre:	savePerfilUsuario (login, listaPerfiles)
Descripción:	Guarda en la BD dado un login los Permisos asociados.
Nombre:	listarUsuarios ()
Descripción:	Retorna un listado de todos los usuarios de la base de datos.
Nombre:	cambPass (login, password, newPassword, ConfirmNewPassword)
Descripción:	Cambia la contraseña del usuario seleccionado.
Nombre:	datosUsuario (login)
Descripción:	Retorna los datos del usuario dado su login.
Nombre:	removePerfilUsuario (login, listaPerfiles)
Descripción:	Elimina los Perfiles proporcionados de un usuario dado su login.
Nombre:	modUsuario (login, newLogin, newPassword, newHabilitado)
Descripción:	Modifica los datos del usuario seleccionado.
Nombre:	elimUsuario(login)
Descripción:	Elimina al usuario seleccionado por un login dado.

Nombre: CambiarPassView	
Tipo de clase: Interfaz	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	CambiarPassVista()
Descripción:	Constructor de la vista.
Nombre:	getAceptar ()
Descripción:	Envía el evento aceptar del botón al presentador.
Nombre:	getCancelar ()
Descripción:	Envía el evento cancelar del botón al presentador.
Nombre:	getLogin ()
Descripción:	Obtiene el login proporcionado y lo envía al presentador de la vista.
Nombre:	getPassword ()

Descripción:	Obtiene el password proporcionado y lo envía al presentador de la vista.
Nombre:	getNewPassword ()
Descripción:	Obtiene el nuevo password proporcionado y lo envía al presentador de la vista.
Nombre:	getConfirmNewPassword ()
Descripción:	Obtiene la confirmación del password nuevo proporcionado y lo envía al presentador de la vista.

Nombre: CambiarPassPresentador	
Tipo de clase: Presentador	
Atributo	Tipo
nombreID	String
Para cada responsabilidad:	
Nombre:	cambiarPass ()
Descripción:	Gestiona los correspondientes eventos y envía la información para ser procesada por el servicio UsuarioService.

3.2. Arquitectura del Sistema

"La arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad" (Kruchten, 1995).

El sistema está basado en la arquitectura Cliente-Servidor. Donde el software reparte su carga de cómputo en dos partes independientes, una del lado del Cliente y otra del lado del Servidor.

3.2.1. Arquitectura Cliente-Servidor

La Arquitectura Cliente-Servidor es un modelo para el desarrollo de sistemas informáticos en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar

información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor al proceso que responde a las solicitudes. En este modelo las aplicaciones se dividen de forma que el servidor contiene la parte que debe ser compartida por varios usuarios, y en el cliente permanece sólo el referente particular a cada usuario.

Características de la arquitectura Cliente-Servidor:
Por parte del lado del cliente se proporciona la interfaz entre el usuario y el resto del sistema mientras que por la parte del servidor se actuará como motor de software manejando los recursos compartidos (ejemplo: bases de datos, impresoras, entre otros). Estas tareas tendrán diferentes requerimientos en cuanto a recursos de cómputo, memoria, velocidad, entre otras (Encyclopædia Britannica Inc, 2012).

Los clientes serán los iniciadores del diálogo (peticiones de servicios) con los servidores. Estos últimos tienen un carácter pasivo ya que esperan las peticiones de los clientes. Una de las bondades de usar esta arquitectura es la posibilidad de mantener un ambiente heterogéneo, donde la plataforma de hardware y el sistema operativo del cliente y la del servidor no deben ser necesariamente la misma.

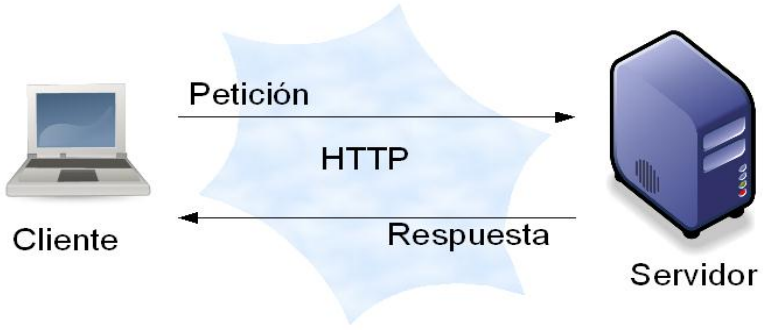


Figura 10. Arquitectura Cliente-Servidor

Esta arquitectura seguirá un estilo lógico en distribuido en tres capas:

Capa de Presentación: Donde se generará la interfaz de usuario en función de las acciones llevadas a cabo por el mismo.

Capa de Negocio: Contiene la lógica para modelar los procesos de negocio y se da solución a las peticiones hechas por el usuario.

Capa de acceso a Datos: Suministra y almacena información para el nivel de negocio.

Figura 11. Arquitectura Cliente-Servidor en tres capas.

3.2.2. Patrones de arquitectura

Los patrones de arquitectura expresan un esquema organizativo estructural fundamental para sistemas de software.

3.2.2.1. Modelo-Vista-Controlador

Grails utiliza el patrón Modelo-Vista-Controlador (MVC) este es un patrón arquitectónico que proporciona una programación multicapa, cuyo objetivo es separar los datos de la aplicación, su interfaz de usuario y la lógica de control en distintos componentes.

Donde el Modelo encapsula los datos y sus funcionalidades controlando todas sus transformaciones, este no tiene conocimiento específico de los Controladores o de las Vistas, ni contiene referencias a ellos, es responsabilidad del sistema mantener enlaces entre el Modelo y sus Vistas notificando a las Vistas cuando cambia el Modelo. Representa la información con la que trabaja la aplicación, o sea, su lógica de negocio.

La Vista muestra la información al usuario, pudiendo existir múltiples vistas del modelo. Cada vista tiene asociado un componente Controlador y es el objeto que genera una representación visual del Modelo mostrando los datos al usuario (esto quiere decir que convierte al modelo en una página Web que facilita al usuario interactuar con ella).

El Controlador recibe las entradas como eventos, estos son traducidos a solicitudes de servicio (service requests) para el modelo o la vista. Interactúa con el Modelo a través de una referencia al propio Modelo, es el encargado de procesar las interacciones del usuario y ejecuta los cambios adecuados en el modelo o en la vista.

Ventajas del Patrón Arquitectónico MVC:
Posee una separación total entre lógica de negocio y la presentación, a esto se le pueden aplicar opciones como el multilenguaje y distintos diseños de presentación sin alterar la lógica de negocio. La separación de capas como la presentación, lógica de negocio y acceso a datos es fundamental para el desarrollo de arquitecturas consistentes, reutilizables y de fácil mantenimiento, lo que desemboca en un ahorro de tiempo de desarrollo en posteriores proyectos.

3.2.2.2. Modelo-Vista-Presentador

En el paquete de las vistas se utilizará como antes se mencionaba GWT que utilizará el patrón Modelo Vista Presentador (MVP) como variante del MVC donde:

La Vista proveerá de una interfaz dinámica al usuario recibiendo todos los comandos del usuario a través de eventos enviándolos al Presentador delegando toda la responsabilidad del manejo de los mismos, este contendrá toda la lógica para responder a los eventos y manipular el estado de la Vista, en la arquitectura definida para desarrollar el SIC la Vista y el Modelo estarán físicamente separados sin conocimiento íntimo uno del otro por lo que el Presentador estará encargado de actualizar a los otros componentes obteniendo datos desde la base de datos y transformándolos para que estos sean visualizados y viceversa, el Modelo no conoce la existencia del Presentador; así que si el modelo cambia por acción de algún otro componente que no sea el Presentador se dispara un evento para que este se entere.

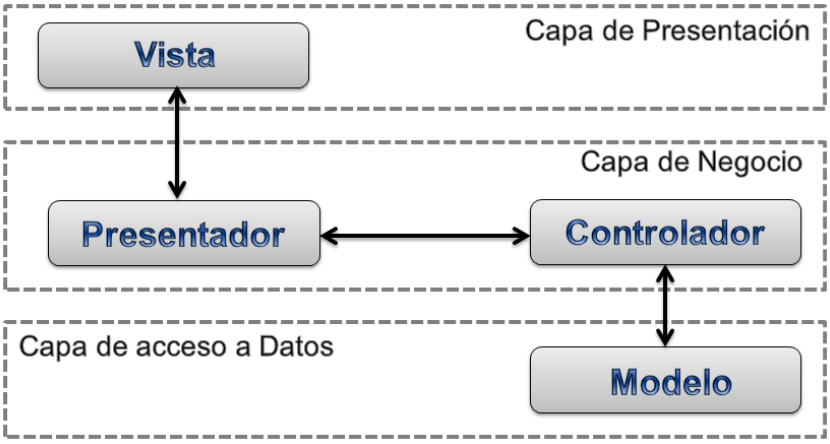


Figura 12. Patrones Arquitectónicos MVC y MVP.

3.2.3. Patrones de diseño

Los patrones de diseño expresan esquemas para definir estructuras de diseño con las cuales construir sistemas de software, estos patrones contribuyen a la reutilización del diseño, identificando aspectos claves de la estructura que pueden ser aplicados en otras situaciones. Brindan una estructura de código común a todos los proyectos que implementan una funcionalidad genérica lo que permite ahorrar grandes cantidades de tiempo en la construcción de software, siendo este más fácil de comprender, mantener y extender.

3.2.3.1. Patrones de creación

Estos muestran una guía de cómo crear objetos cuando se requiere tomar decisiones sobre su creación, estas son resueltas dinámicamente decidiendo que clases instanciar o sobre que objetos se delegarán responsabilidades, en nuestro caso se utilizaron:

Abstract Factory (fábrica abstracta): “Proporciona una interfaz para crear familias de objetos que dependen entre sí, sin especificar sus clases concretas. Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando” (ECURED, 2012).

Builder (constructor virtual): “Separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones. Abstrae el proceso de creación de un objeto complejo, centralizando dicho proceso en un único punto” (ECURED, 2012).

Singleton (instancia única): “Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia” (ECURED, 2012).

3.2.3.2. Patrones estructurales

Describen la forma en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros, en nuestro caso se utilizó el Composite.

Composite (Objeto compuesto): “Permite combinar objetos en estructuras de árbol para representar jerarquías. Permite además, agrupar varios objetos como si fueran uno solo” (ECURED, 2012). Favorece la extensibilidad, ya que es muy fácil añadir nuevos tipos de componentes, tanto simples como compuestos.

3.4. Conclusiones parciales

En este capítulo se realizó un análisis del sistema en términos de solución, quedando definida la estructura del sistema mediante:

La implementación y descripción de las principales clases que contienen las funcionalidades que dan solución a los casos de uso. La confección de los diagramas de interacción mediante las interacciones usuario-sistema. La descripción de la arquitectura Cliente-Servidor y su base de programación en tres capas, la definición de los patrones de arquitectura MVC y el MVP como variante en la capa de Presentación. Además se definen los patrones de diseño que ayudaron en la implementación de las funciones genéricas.

Capítulo 4. Implementación y Pruebas

Este capítulo trata sobre los flujos de trabajo Implementación y Prueba, se representan los diagramas de despliegue y de componentes, así como el modelo de implementación como entrada principal a la etapa de pruebas, donde cada construcción generada durante la implementación se someterá a pruebas de integración y de sistema.

4.1. Modelo de Implementación

La implementación comienza con el resultado del diseño y tiene como objetivos implementar las clases y objetos del diseño como componentes, ficheros fuente, binarios y ejecutables. Todos los elementos implementados formarán el Modelo de Implementación, se integran de forma incremental, con el objetivo de que en caso que ocurran fallos sean más fáciles de detectar y los componentes se prueben más a fondo.

El modelo de implementación es uno de los artefactos que se construye durante la fase de implementación y describe como los elementos del modelo de diseño (clases) se implementan en términos de componentes, ficheros de código fuente, ejecutables entre otros. El modelo de implementación describe también como se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen unos de otros (Jacobson, 2000).

4.1.1. Diagrama de Componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas.

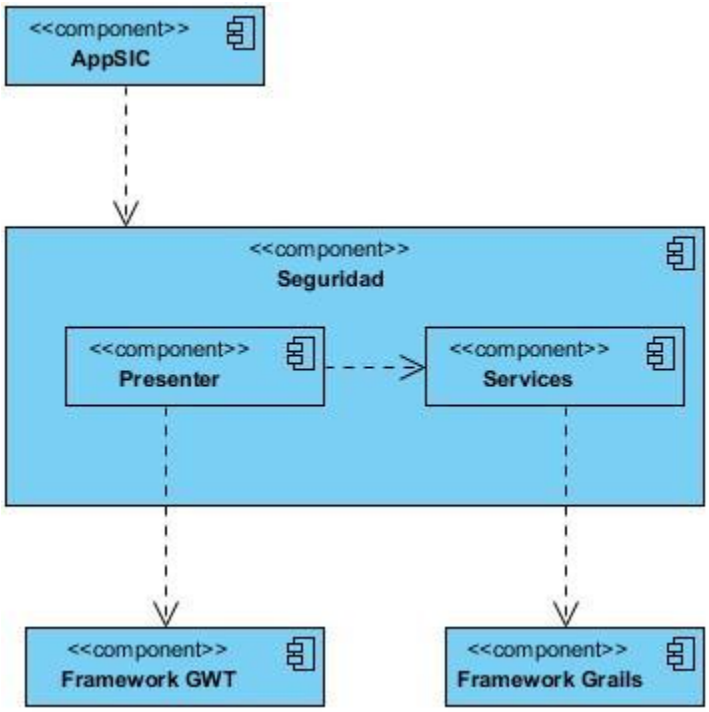


Figura 13. Diagrama de Componentes.

El componente de Seguridad se divide en dos componentes Presenter y Service. Donde el primero hace uso del framework GWT implementando el patrón MVP y donde se contendrá la lógica de presentación usada en la creación de las interfaces de usuario y el manejo de los eventos producidos por el usuario. Services contiene el framework Grails y encierra la lógica del negocio del componente Seguridad.

4.2. Modelo de Despliegue

El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo.

4.2.1. Diagrama de Despliegue

El diagrama de despliegue se utiliza para modelar la vista de despliegue estática de un sistema, muestra las relaciones físicas entre los componentes hardware y software (la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software). El diagrama de despliegue muestra

además los nodos que participan en la ejecución y las relaciones de dependencia y asociación que se establecen entre ellos, estos pueden ser servidores, procesadores o dispositivos.

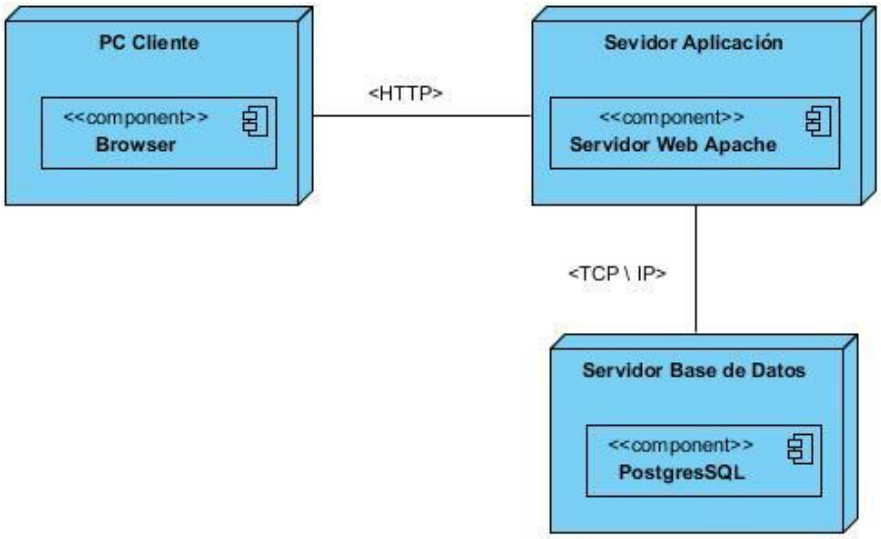


Figura 14. Diagrama de despliegue.

4.3. Modelo de Pruebas

Esta disciplina tiene como objetivos verificar la integración de los componentes y comprobar si el software ha cumplido los requisitos. A continuación se muestra el desarrollo de algunas pruebas de sistema:

4.3.1. Caso de Prueba Autenticar Usuario.

Descripción general: Este caso de uso permite al usuario autenticarse con el sistema a partir de su usuario y su contraseña.

Acciones Válidas	Acciones Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones

Introduce todos los campos correctamente, Ejemplo: usuario: 'ebetancourt' contraseña: 'K0n7r4S3^^4*'		El sistema realiza la validación de los datos correctamente y lleva a cabo la autenticación. El sistema muestra el mensaje de Usuario 'ebetancourt' autenticado con éxito.	Resultado Satisfactorio.
	El usuario deja campos vacíos Ejemplo: usuario: 'ebetancourt' contraseña:' '	El sistema muestra un mensaje de error alertando que existen campos vacíos.	Resultado Satisfactorio.
	El usuario introduce valores incorrectos. Ejemplo: usuario: 'ebetancourt' contraseña: 'abretesessamo'	El sistema muestra un mensaje de error informando al usuario que existen errores en los datos introducidos.	Resultado Satisfactorio.

4.3.2. Caso de Prueba Cambiar Contraseña.

Descripción general: Este caso de uso permite al usuario cambiar su contraseña de forma personal.

Acciones Válidas	Acciones Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Introduce todos los campos correctamente, Ejemplo: usuario: 'ebetancourt' contraseña: 'K0n7r4S3^^4*' nueva contraseña: 'P4zzW00R1)' confirmación: 'P4zzW00R1)'		El sistema realiza la validación de los datos correctamente y actualiza la contraseña. El sistema muestra el mensaje de contraseña cambiada con éxito.	Resultado Satisfactorio.	
	El usuario introduce una contraseña no válida. Ejemplo: contraseña: 'abretesessamo'	El sistema muestra un mensaje de error alertando al usuario que la contraseña no es válida.	Resultado Satisfactorio.	
	El usuario introduce los valores de nueva contraseña y confirmación diferentes. Ejemplo: nueva contraseña: 'P4zzW00R1)'	El sistema muestra un mensaje de error informando al usuario que la nueva contraseña y su confirmación no coinciden.	Resultado Satisfactorio.	

	confirmación: 'P4zz<>0r1)'

4.3.3. Resultados de las pruebas

Las pruebas realizadas demuestran que los casos de prueba analizados realizan un correcto tratamiento de los datos manejados, arrojando además que las funcionalidades brindadas por estos casos de uso no presentan anomalías en su funcionamiento. Por lo que se concluye que los resultados de las pruebas realizadas fueron satisfactorios.

4.4. Conclusiones parciales

En este capítulo se muestran resultados visibles para los clientes y desarrolladores a través de los diagramas de despliegue y de componentes, quedando implementada la aplicación con las principales funcionalidades que se definieron en las primeras fases de elaboración. Además se llevaron a cabo las pruebas al sistema arrojándose resultados satisfactorios.

Conclusiones

El desarrollo del presente documento incluyó un estudio de los principales conceptos referentes a la seguridad informática y un estudio de los principales sistemas gestores de seguridad en las aplicaciones Web, que conformaron la base teórica usada para el desarrollo del componente. Se implementó dicho componente basado en las características de las metodologías de desarrollo, herramientas y tecnologías escogidas, siendo utilizado el marco de Grails para la lógica de negocio y GWT para las interfaces usuario y visualización de datos. Se construyeron diferentes artefactos a través de las diferentes fases de desarrollo del software así como la implementación y descripción de las principales clases contenedoras de las funcionalidades que dan solución a los Casos de Uso que se derivaron de los requerimientos. Quedando definida la arquitectura del sistema del tipo Cliente-Servidor programada en tres capas y el uso de los patrones de arquitectura MVC y MVP. Además se realizó la implementación de las funciones requeridas de seguridad, llevándose a cabo pruebas con resultados satisfactorios.

El componente de seguridad que se propone es capaz de integrarse con total transparencia no solo a las exigencias de la aplicación SIC para la cual fue concebido, sino que también pudiera ser utilizado por su dinámica en la elaboración de módulos o herramientas de seguridad para otros sistemas en desarrollo dentro de nuestra universidad que integren los marcos Grails y GWT. Bajo los principios de la soberanía tecnológica, totalmente basado en herramientas de código abierto, con bajo coste de integración e implementación, flexible, pero potente y eficiente.

Finalmente se puede concluir que con la solución propuesta se materializan los objetivos planteados al inicio de este trabajo de diploma: Desarrollar un componente para controlar y registrar la actividad de los usuarios sobre los recursos del Sistema Integral de Confiabilidad Operacional según sus correspondientes perfiles.

Recomendaciones

Durante el desarrollo de una aplicación, las funciones de seguridad deben ser utilizadas en varios niveles. Cuando se establecen los requisitos, a través del diseño, implementación y prueba de la aplicación. Sin embargo, no termina allí, ya que el entorno de las aplicaciones Web está en constante cambio, por lo que nuevas vulnerabilidades deberán ser descubiertas y con ellas nuestra aplicación deberá evolucionar, por lo que se recomienda:

- La educación continua sobre los temas novísimos de Seguridad, para en las periódicas revisiones en producción, fusionarlas al Componente de Seguridad implementado.
- Continuar con el desarrollo de este trabajo buscando nuevas herramientas con el objetivo de construir un sistema de autenticación aún más potente.
- Añadirle nuevas funcionalidades en dependencia de las necesidades que vayan surgiendo dentro de la organización donde fue implementado.
- Encapsular el modelo del componente para realizar un plugin que pueda ser utilizado bajo cualquier framework de desarrollo Web.

Referencias *Bibliográficas*

CEDIN. 2012. Centro de Informática Industrial. [En línea] 2012. http://gespro.cedin.prod.uci.cu/.

ECURED. 2011. ECURED, IDE de programación. [En línea] 2011. http://www.ecured.cu/index.php/IDE_de_Programaci%C3%B3n.

—. 2012. Ecured, Patrones de diseño. [En línea] 2012. http://www.ecured.cu/index.php?title=Patrones_de_dise%C3%B1o&action=edit&redlink=1.

Encyclopædia Britannica Inc. 2012. client-server architecture. [En línea] 2012. http://www.britannica.com/EBchecked/topic/1366374/client-server-architecture.

Foundation, The Apache Software. 2010. Apache Shiro. [En línea] 2010. http://shiro.apache.org/.

Foundation, The OWASP. 2011. About The Open Web Application Security Project. [En línea] 2011. About. https://www.owasp.org/index.php/About_The_Open_Web_Application_Security_Project/es.

—. 2010. The ten most critical Web application security risks. [En línea] 2010. http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202010.pdf.

García de Jalón, Javier. 1999. *Aprenda Java como si estuviera en primero.* San Sebastián : s.n., 1999.

Google. 2011. Google Web Toolkit. [En línea] Google, 2011. http://code.google.com.es.mk.gd/webtoolkit/.

groovy. [En línea] [Citado el: 28 de Mayo de 2012.] http://docs.codehaus.org/display/GROOVY/Home.

HDIV. 2011. HTTP DATA INTEGRITY VALIDATOR. [En línea] 2011. http://www.hdiv.org.

IngeCon. 2012. Asesoria Integral en Ingenieria de Confiabilidad. [En línea] 2012. http://www.confiabilidadoperacional.com/.

Jacobson, Ivar. 2000. *El Proceso Unificado de Desarrollo Software.* s.l. : Addison Wesley, 2000.

Java. [En línea] [Citado el: 28 de Mayo de 2012.] http://www.java.com/es/about/.

Kruchten, Philippe. 1995. *Architectural Blueprints:The 4+1 View Model of Software Architecture.* 1995.

Oracle. 2011. Java™ Platform, Standard Edition 6. [En línea] 2011. http://docs.oracle.com/javase/6/docs/api/overview-summary.html.

—. Oracle Crystal Ball. [En línea] [Citado el: 10 de Febrero de 2012.] www.oracle.com/us/products/applications/crystalball/index.html.

OWASP. 2007. OWASP top 10 for JEE. [En línea] 2007. https://www.owasp.org/images/8/89/OWASP_Top_10_2007_for_JEE.pdf.

Pressman, Roger. 2010. *Software Engineering, A Practitioner´s Approach.* New York : McGraw-Hill, 2010.

Rocher, Graeme. 2012. The Grails Framework - Reference Documentation. [En línea] 2012. http://grails.org/doc/latest/guide.

Rubinos Carvajal, Alejandro Manuel. 2009. *Subsistema de Seguridad para el SCADA Nacional Guardián del ALBA.* La Habana : UCI, 2009.

Rueda Chacón, Julio César. 2006. *Aplicación de la metodología RUP para el desarrollo rápido de aplicaciones basado en el.* Guatemala : s.n., 2006.

Schumacher, Markus. 2006. *Security Patterns: Integrating Security and Systems Engineering.* s.l. : John Wiley & Sons, 2006.

Socorro Borges, Miguel Angel. 2012. *Integración de los componentes arquitectónicos y tecnologías para el proyecto SIC.* La Habana : UCI, 2012.

SpringSource. 2011. A dynamic language for the Java platform. [En línea] 2011. http://groovy.codehaus.org/.

UCI. 2012. Proyecto Estratégico 2008 - 2012. [En línea] 2012. http://intranet2.uci.cu/sites/default/files/proyeccion_estrategica/pdf/Proyecto_Estrategico_2008%20-%202012.pdf.

UCIPedia. 2007. UCIPedia, Framework. [En línea] 2007. http://ucipedia.uci.cu/index.php/Framework.

Visual Paradigm. [En línea] [Citado el: 5 de Mayo de 2012.] http://www.visual-paradigm.com/product/vpuml/.

Anexos

1.1. Anexo 1. Descripción del CU Gestionar Usuario.

Caso de uso	Gestionar Usuario.
Actores	Usuario Administrador
Resumen	Este caso de uso le permite al Usuario Administrador adicionar nuevos Usuarios, modificar los datos de los existentes y eliminarlos de la aplicación.
Precondición	El Usuario Administrador debe estar autenticado en el sistema.
Flujo normal de eventos	
Acción del Actor	Respuesta del Sistema
1. El Usuario Administrador selecciona la opción Gestionar Seguridad.	2. Muestra la interfaz Gestionar Seguridad.
3. El Usuario Administrador selecciona la opción a realizar.	- Crear nuevo Usuario (Ir a Escenario: Crear nuevo Usuario). - Modificar Usuario (Ir a Escenario: Modificar Usuario). - Eliminar Usuario (Ir a Escenario: Eliminar Usuario). - Asignar Perfil (Ir a Escenario: Asignar Perfil).
Escenario “Crear nuevo Usuario”	
	1. Muestra la interfaz para crear un nuevo Usuario.
2. Introduce los datos del nuevo Usuario.	3. Valida que los datos introducidos están correctos y completos para el registro del Usuario. 4. Verifica que el nombre del Usuario no esté en uso. 5. Registra los datos del Usuario. 6. Muestra un mensaje de “Usuario registrado correctamente”.
Escenario “Modificar Usuario”	

Selecciona el usuario a modificar.	2. El sistema muestra la interfaz para modificar al Usuario seleccionado.
3. Modifica los datos deseados del Usuario.	4. Valida que los datos estén completos y correctos. 5. Modifica los datos del Usuario. 6. Muestra un mensaje de "Usuario actualizado correctamente".
Escenario "Eliminar Usuario"	
1. Selecciona el Usuario a eliminar. Selecciona la opción "Eliminar".	3. Muestra un mensaje "¿Desea realmente eliminar al Usuario seleccionado?"
4. Selecciona Aceptar.	5. Elimina al Usuario de la BD. 6. Muestra el mensaje "Usuario removido con éxito".
Escenario: "Asignar Perfil"	
1. Selecciona el Usuario al que serán asignados los Perfiles.	2. El sistema muestra la interfaz para asignar Perfiles.
3. Selecciona los Perfiles que desea asignarle al usuario y selecciona el botón de asignación.	4. Muestra el mensaje "Perfil asignado".
Flujo alterno de eventos	
Escenario "Crear nuevo Usuario"	
	3.1. Muestra un mensaje especificando el error cometido al introducir los datos.
	4.1. Muestra mensaje indicando que ya existe el Usuario.
Escenario "Modificar Usuario"	
	4.1. Muestra un mensaje especificando el error cometido al introducir los datos.
Escenario "Eliminar Usuario"	
6.1. Selecciona Cancelar.	7.1. Muestra la Interfaz Gestionar Seguridad.
Escenario: "Asignar Perfil"	
3.1. Selecciona los Perfiles que desea	4.1. Muestra el mensaje "Perfil removido".

removerle al usuario y selecciona el botón de asignación.	
Post-condiciones	Se adiciono, modifico o elimino un Usuario. O se le adiciono o elimino un Perfil.
Prototipo de Interfaz	
Véase Anexo6.	

1.2. Anexo 2. Descripción del CU Gestionar Perfil.

Caso de uso	Gestionar Perfil.
Actores	Usuario Administrador
Resumen	Este caso de uso le permite al Usuario Administrador adicionar nuevos Perfiles, modificarlos, así como eliminarlos de la aplicación.
Precondición	El Usuario Administrador debe estar autenticado en el sistema.
Flujo normal de eventos	
Acción del Actor	Respuesta del Sistema
1. El Usuario Administrador selecciona la opción Gestionar Seguridad.	2. Muestra la interfaz Gestionar Seguridad.
3. El Usuario Administrador selecciona la opción a realizar.	- Crear nuevo Perfil (Ir a Escenario: Crear nuevo Perfil). - Modificar Perfil (Ir a Escenario: Modificar Perfil). - Eliminar Perfil (Ir a Escenario: Eliminar Perfil).
Escenario “Crear nuevo Perfil”	
	1. Muestra la interfaz para crear un nuevo Perfil.
2. Introduce los datos del nuevo Perfil.	3. Valida los campos. 4. Verifica que no existe el Perfil. 5. Registra los datos del Perfil. 6. Muestra un mensaje de “Perfil creado correctamente”.

Escenario “Modificar Perfil”	
1. Selecciona el Perfil a modificar.	2. El sistema muestra la interfaz para modificar al Perfil seleccionado.
3. Modifica los datos deseados del Perfil.	4. Valida que los datos estén completos y correctos. 5. Modifica los datos del Perfil. 6. Muestra un mensaje de “Perfil actualizado correctamente”.
Escenario “Eliminar Perfil”	
1. Selecciona el Perfil a eliminar. 2. Selecciona la opción “Eliminar”.	3. Muestra un mensaje “¿Desea realmente eliminar el Perfil seleccionado?”
4. Selecciona Aceptar.	5. Elimina al Perfil. 6. Muestra el mensaje “Perfil removido con éxito”.
Flujo alterno de eventos	
Escenario “Crear nuevo Rol”	
	3.1. Muestra el mensaje especificando el error cometido al introducir los datos.
	4.1. Muestra mensaje indicando que ya existe el Perfil.
Escenario “Modificar Perfil”	
	4.1. Muestra el mensaje especificando el error cometido al introducir los datos.
Escenario “Eliminar Permiso”	
4.1. Selecciona Cancelar	5.1. Muestra la Interfaz Gestionar Seguridad.
Post-condiciones	Se adiciono, modifico o elimino un Perfil.
Prototipo de Interfaz	
Véase Anexo7.	

1.3. Anexo 3. Descripción del CU Gestionar Módulo.

Caso de uso	Gestionar Módulo.

Actores	Usuario Administrador
Resumen	Este caso de uso le permite al Usuario Administrador adicionar nuevos Módulos a la aplicación.
Precondición	El Usuario Administrador debe estar autenticado en el sistema.
Flujo normal de eventos	
Acción del Actor	Respuesta del Sistema
1. El Usuario Administrador selecciona la opción Gestionar Seguridad.	2. Muestra la interfaz Gestionar Seguridad.
3. El Usuario Administrador selecciona la opción a realizar.	- Crear nuevo Módulo (Ir a Escenario: Crear nuevo Módulo). - Modificar Módulo (Ir a Escenario: Modificar Módulo). - Eliminar Módulo (Ir a Escenario: Eliminar Módulo).
Escenario “Crear nuevo Módulo”	
	1. Muestra la interfaz para crear un nuevo Módulo.
2. Introduce los datos del nuevo Módulo.	3. Valida los campos. 4. Verifica que no existe el Módulo. 5. Registra los datos del Módulo. 6. Muestra un mensaje de “Módulo creado correctamente”.
Escenario “Modificar Módulo”	
Selecciona el Módulo a modificar.	2. El sistema muestra la interfaz para modificar al Módulo seleccionado.
3. Modifica los datos deseados del Módulo.	4. Valida que los datos estén completos y correctos. 5. Modifica los datos del Módulo. 6. Muestra un mensaje de “Módulo actualizado correctamente”.
Escenario “Eliminar Módulo”	
Selecciona el Módulo a eliminar.	3. Muestra un mensaje “¿Desea realmente eliminar el

2. Selecciona la opción “Eliminar”.	Módulo seleccionado?”
4. Selecciona Aceptar.	5. Elimina el Módulo. 6. Muestra el mensaje “Módulo removido con éxito”.
Flujo alterno de eventos	
Escenario “Crear nuevo Módulo”	
	3.1. Muestra el mensaje especificando el error cometido al introducir los datos.
	4.1. Muestra mensaje indicando que ya existe el Módulo.
Escenario “Modificar Módulo”	
	4.1. Muestra el mensaje especificando el error cometido al introducir los datos.
Escenario “Eliminar Módulo”	
4.1. Selecciona Cancelar	5.1. Muestra la Interfaz Gestionar Seguridad.
Post-condiciones	Se adiciono, modifico o elimino un Módulo.
Prototipo de Interfaz	
Véase Anexo8.	

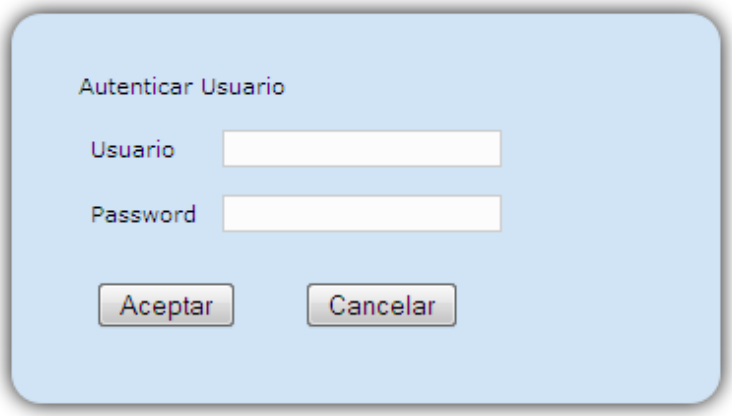
1.4. Anexo 4. Descripción del CU Gestionar Permiso.

Caso de uso	Gestionar Permiso.
Actores	Usuario Administrador
Resumen	Este caso de uso le permite al Usuario Administrador asignar nuevos Permisos a los Usuarios, modificar los existentes o eliminarlos.
Precondición	El Usuario Administrador debe estar autenticado en el sistema.
Flujo normal de eventos	
Acción del Actor	Respuesta del Sistema

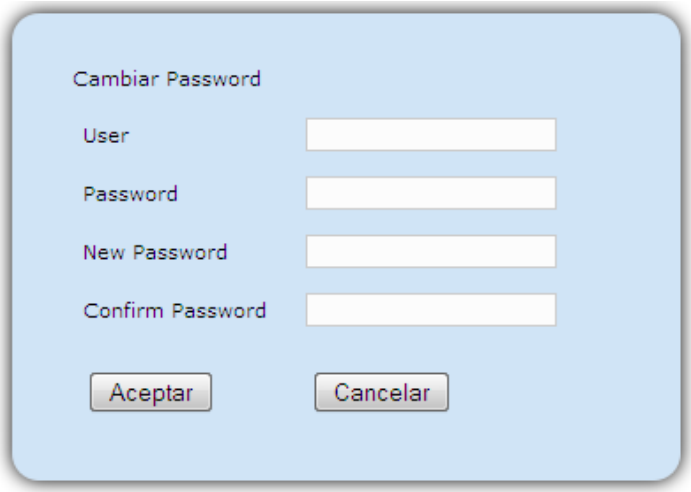
1. El Usuario Administrador selecciona el Módulo a modificar. 2. El Usuario Administrador selecciona la opción a realizar.	- Crear nuevo Permiso (Ir a Escenario: Crear nuevo Permiso). - Modificar Permiso (Ir a Escenario: Modificar Permiso). - Eliminar Permiso (Ir a Escenario: Eliminar Permiso).
Escenario “Crear nuevo Permiso”	
1. Introduce los datos del Permiso. 2. Selecciona la opción “Adicionar”.	3. Valida los campos. 4. Verifica que no existe el Permiso. 5. Registra los datos del Permiso. 6. Muestra un mensaje de “Permiso creado correctamente”.
Escenario “Modificar Permiso”	
1. Selecciona el Permiso que va a modificar.	2. El sistema muestra la interfaz para modificar el Permiso seleccionado.
3. modifica los datos deseados.	4. Valida los datos. 5. Modifica los datos del Permiso. 6. Muestra un mensaje de “Permiso actualizado correctamente”.
Escenario “Eliminar Permiso”	
1. Selecciona el Permiso que va a eliminar.	2. Muestra un mensaje “¿Desea realmente eliminar el Permiso seleccionado?”
3. Selecciona Aceptar.	4. Elimina el Permiso. 5. Muestra el mensaje “Permiso removido con éxito”.
Flujo alterno de eventos	
Escenario “Crear nuevo Permiso”	
	3.1. Muestra un mensaje especificando el error cometido al introducir los datos.
	4.1. Muestra mensaje indicando que ya existe el Permiso.
Escenario “Modificar Permiso”	

	4.1. Muestra un mensaje especificando el error cometido al introducir los datos.
Escenario “Eliminar Permiso”	
3.1. Selecciona Cancelar.	5.1. El sistema muestra la vista Gestionar Permiso.
Post-condiciones	Se adiciono, modifico o elimino un Permiso.
Prototipo de Interfaz	
Véase Anexo9.	

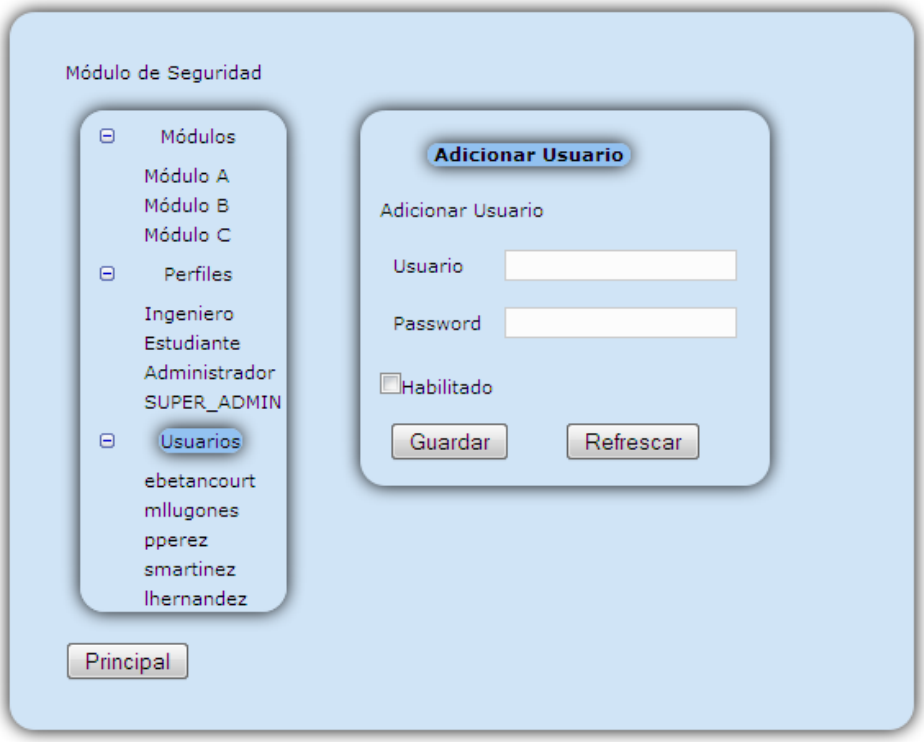
1.5. Anexo 5. Prototipo de Interfaces CU Autenticar Usuario.

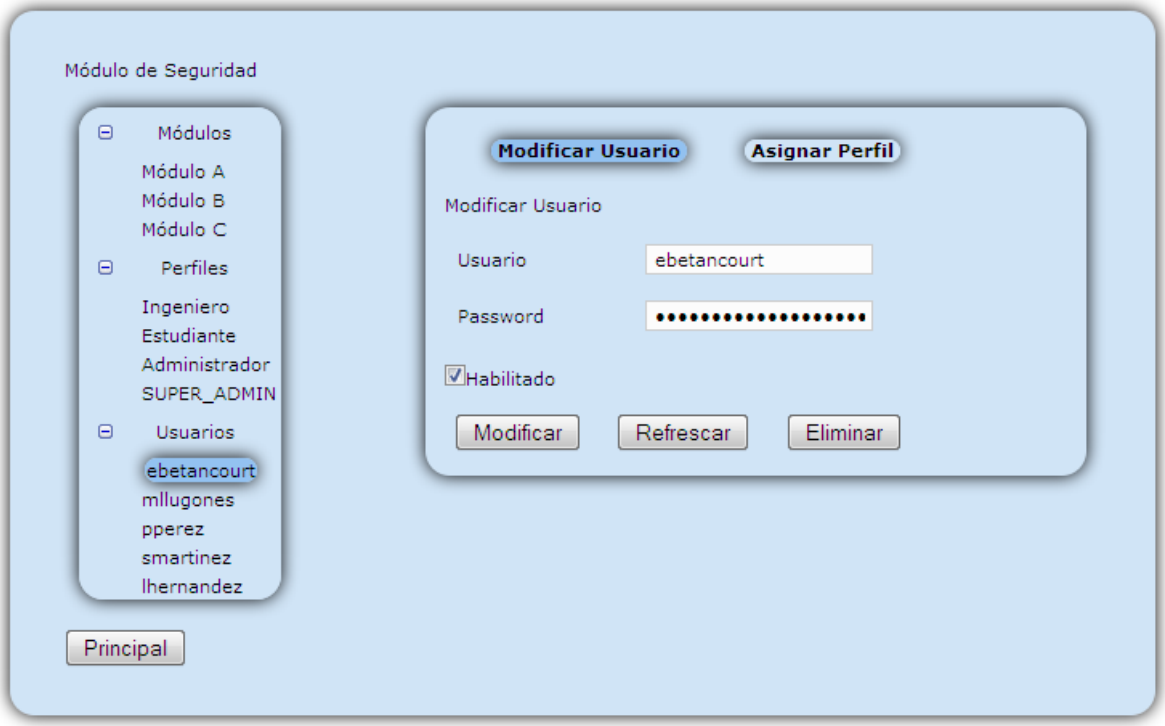


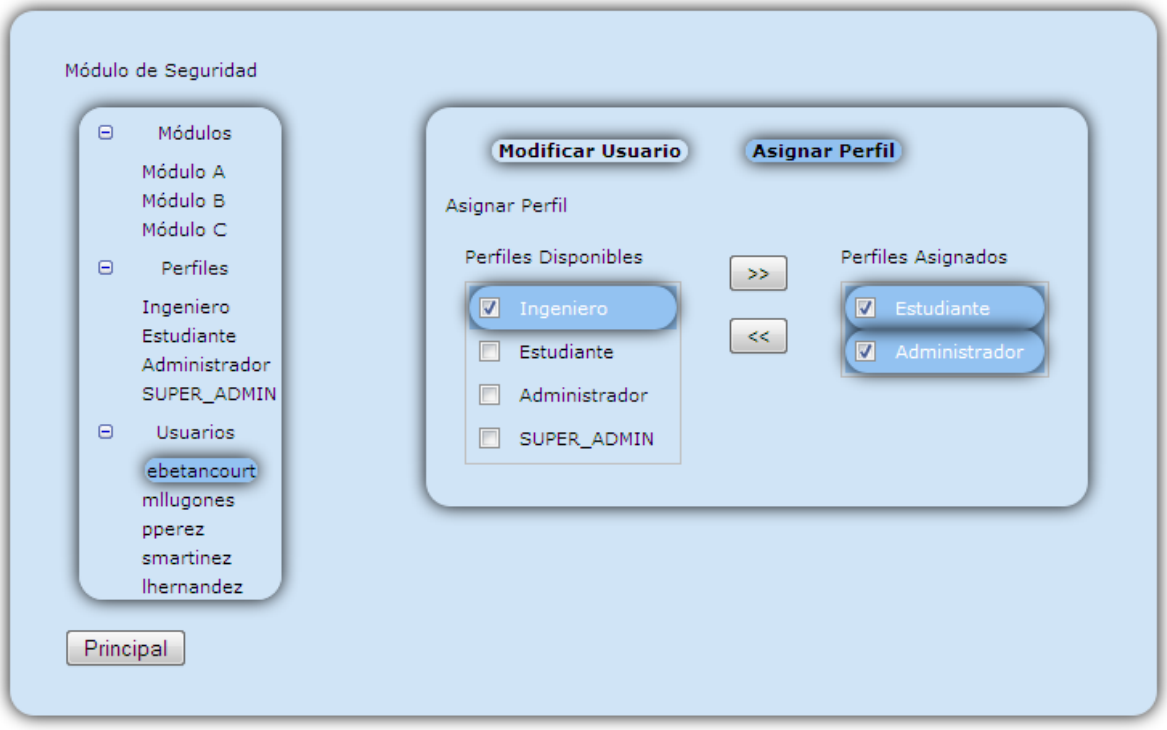
1.6. Anexo 6. Prototipo de Interfaces CU Cambiar Contraseña.



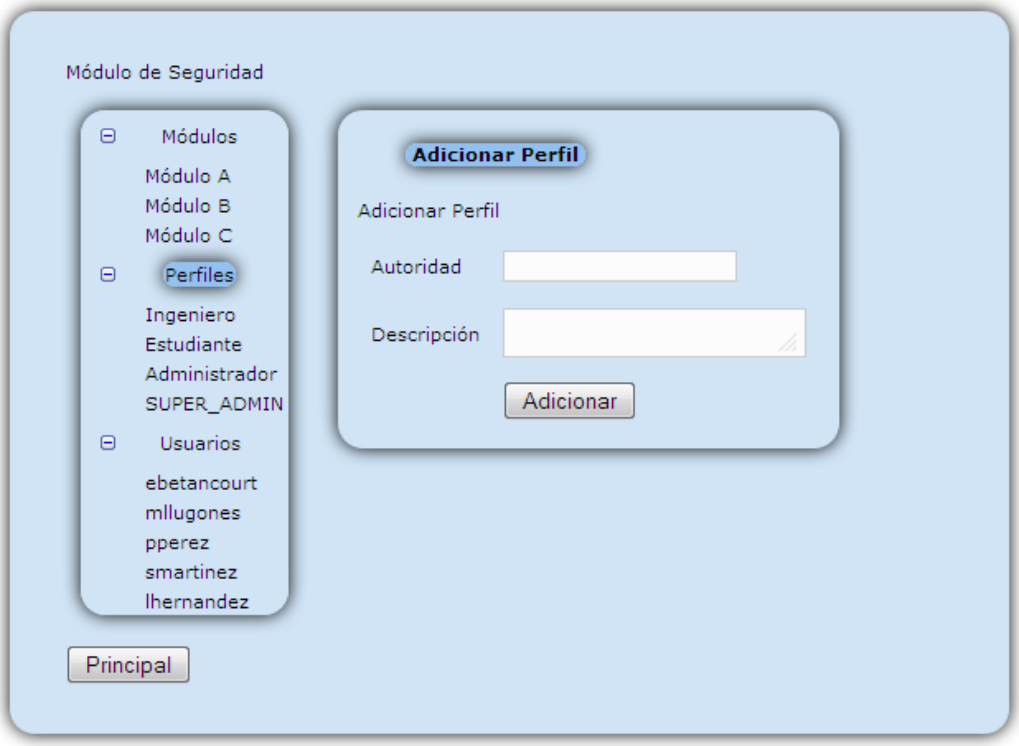
1.7. Anexo 7. Prototipo de Interfaces CU Gestionar Usuario.

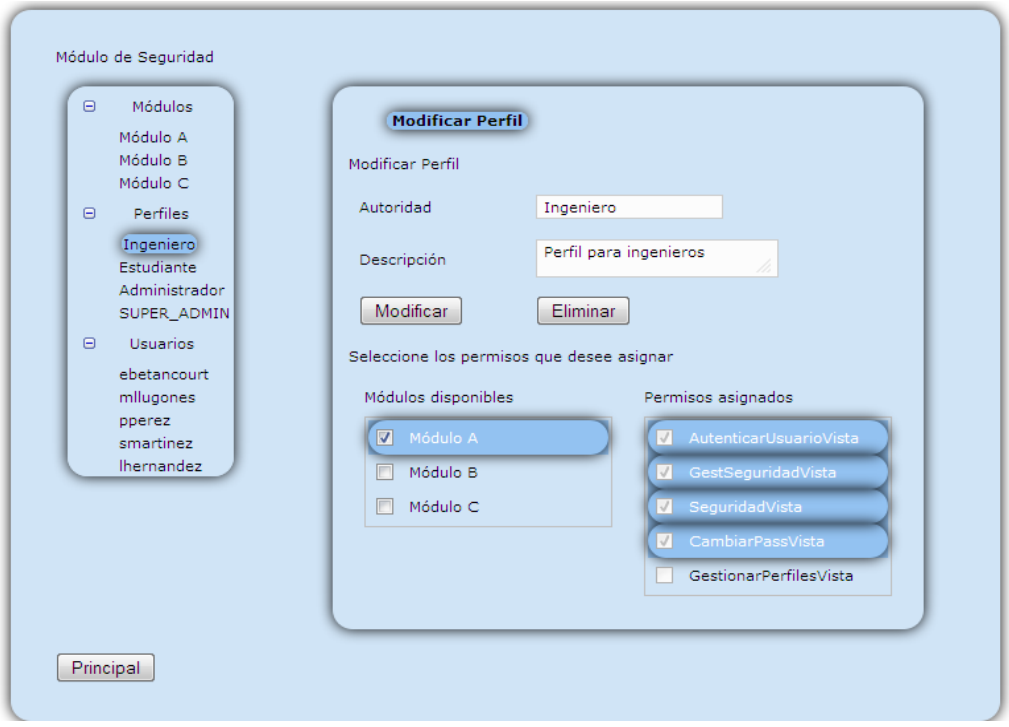



Módulo de Seguridad
Módulos
Módulo A
Módulo B
Módulo C
Perfiles
Ingeniero
Estudiante
Administrador
SUPER_ADMIN
Usuarios
ebetancourt
mllugones
pperez
smartinez
lhernandez
Principal
Modificar Usuario
Asignar Perfil
Modificar Usuario
Usuario
ebetancourt
Password
Habilitado
Modificar
Refrescar
Eliminar

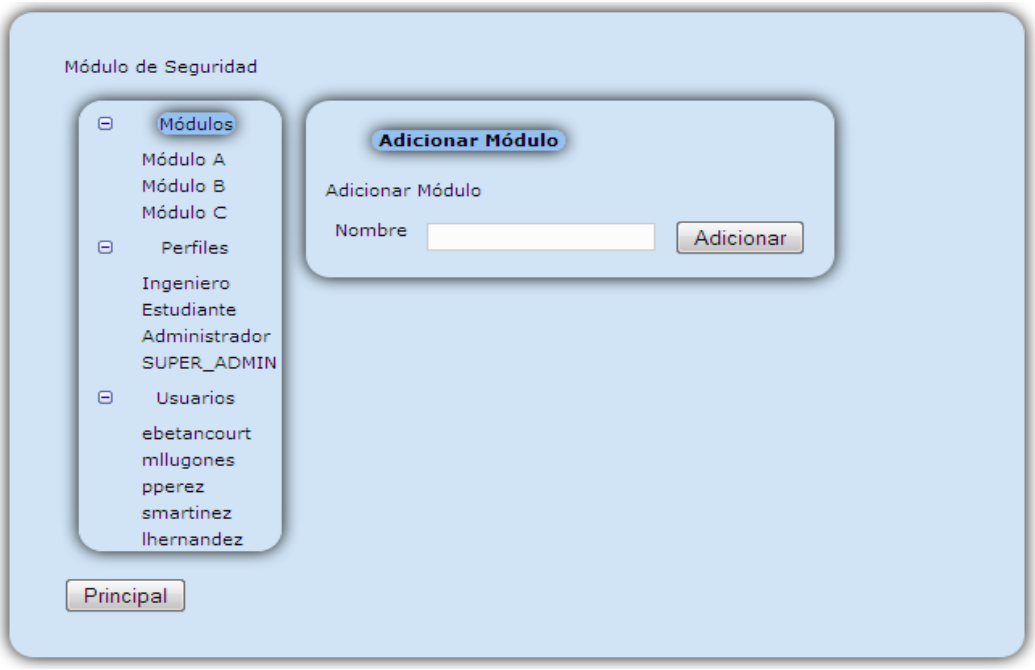

Módulo de Seguridad
Módulos
Módulo A
Módulo B
Módulo C
Perfiles
Ingeniero
Estudiante
Administrador
SUPER_ADMIN
Usuarios
ebetancourt
mllugones
pperez
smartinez
lhernandez
Principal
Modificar Usuario
Asignar Perfil
Asignar Perfil
Perfiles Disponibles
Ingeniero
Estudiante
Administrador
SUPER_ADMIN
>>
<<
Perfiles Asignados
Estudiante
Administrador

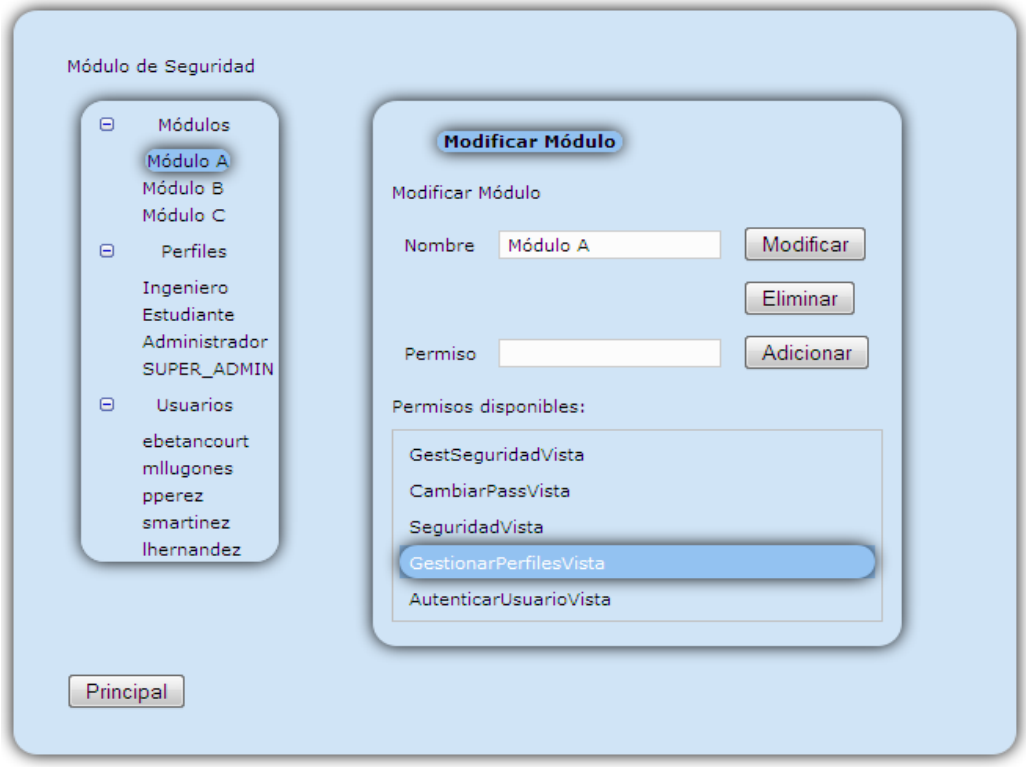
1.8. Anexo 8. Prototipo de Interfaces CU Gestionar Perfil.



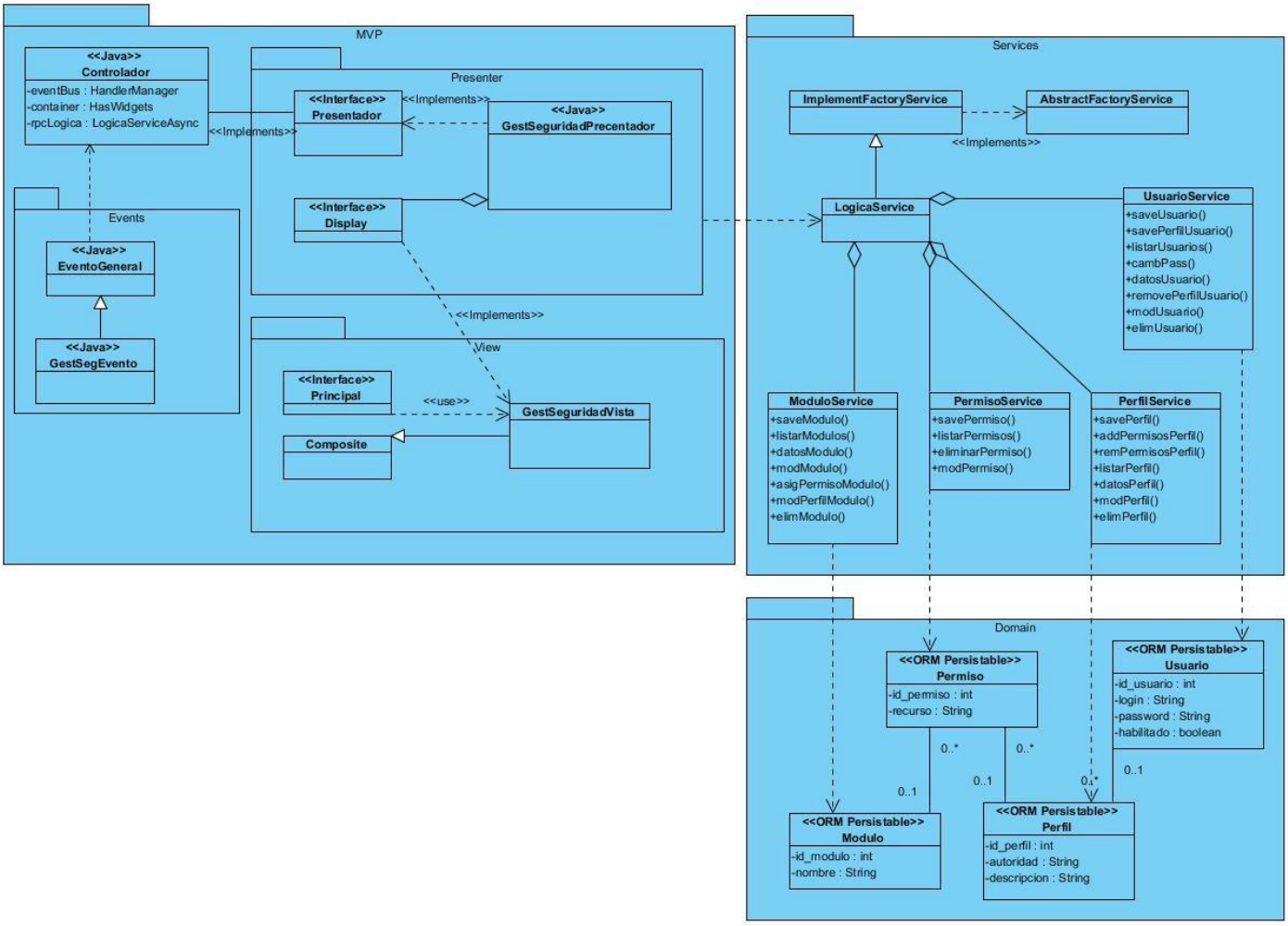


1.9. Anexo 9. Prototipo de Interfaces CU Gestionar Permiso.





1.10. Anexo 10. Diagrama de Clases para los Casos de Uso: Gestionar Usuario, Gestionar Perfil, Gestionar Módulo y Gestionar Permiso.



Glosario de términos

Activo: Recurso del sistema de información o relacionado con éste, necesario para que la organización funcione correctamente y alcance los objetivos propuestos.

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

Riesgo: Posibilidad de que se produzca un Impacto determinado en un Activo, en un Dominio o en toda la Organización.

Vulnerabilidad: posibilidad de ocurrencia de la materialización de una amenaza sobre un Activo.

Ataque: Evento, exitoso o no, que atenta sobre el buen funcionamiento de un sistema.

TCP/IP: Es el estándar de protocolo de comunicaciones requerido por las computadoras que acceden a Internet.

HTML: (Hyper Text Markup Language, Lenguaje de Marcas de Hipertexto) Lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas Web.

HTTP: (Hyper Text Transfer Protocol) Protocolo de Trasferencia de Hipertexto. Protocolo usado para la transferencia de documentos WWW. Estas transferencias requieren un programa cliente http en un extremo de la comunicación y un servidor http en el otro.

SMTP: (Simple Mail Transfer Protocol) Protocolo Simple de Transferencia de Correo-electrónico. Protocolo de red basado en texto utilizado para el intercambio de mensajes de correo electrónico entre computadoras y/o distintos dispositivos.

XML: (Extensible Markup Language) Lenguaje Extensible de Etiquetas. Es un meta-lenguaje que permite definir lenguajes de marcado adecuado a usos determinados desarrollado por el World Wide Web Consortium (W3C).

DML: (Data Manipulation Lenguaje) Lenguaje de manipulación de datos. Las sentencias DML son aquellas utilizadas para insertar, borrar, modificar y consultar los datos de una base de datos.

URL: (Uniform Resource Locutor) Localizador Uniforme de Recursos. URL una cadena de caracteres con la cual se asigna dirección única a cada uno de los recursos de información disponibles en internet.

JSP: (Java Server Pages) Páginas Servidoras Java. Es la tecnología para generar páginas Web de forma dinámica en el servidor. Permite a los programadores generar dinámicamente páginas HTML.

CRUD: (Create, Read, Update, Delete) Crear, Leer, Actualizar, Borrar.

ORM: Mapeado Objeto Relacional. Consiste en un traductor entre dos paradigmas, el orientado a objetos y el modelo relacional, también se conoce como motor de persistencia.

XSS: (Cross Site Scripting) Tipo de vulnerabilidad surgida como consecuencia de errores de filtrado de las entradas del usuario en aplicaciones Web.

XFS: (Cross Frame Scripting), un tipo de ataque de phishing que envenena a un solo cuadro en la página.

Hacker: Generalmente referido a los piratas de las redes, engloba términos como “White hat”, “black hat” entre otros.

SQL: (Structured Query Language) Lenguaje estándar de comunicación con bases de datos.

Trigger: Un trigger o un “disparador” es un evento que se ejecuta en una base de datos cuando se cumple una condición establecida al realizar una operación de inserción, actualización o borrado.

Phishing: El phishing o “Pesca” haciendo alusión al acto de pescar usuarios mediante señuelos cada vez más sofisticados es un tipo de delito informático encuadrado dentro del ámbito de las estafas cibernéticas.

Logs: Se refiere al registro de eventos de un usuario dentro de la aplicación.

SOAP: (Simple Object Access Protocol) es un protocolo estándar creado por Microsoft IBM, define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML, es decir que permite la comunicación entre aplicaciones a través de mensajes por medio de Internet. Es independiente de la plataforma, y del lenguaje, además de ser un protocolo abierto, ya que es una especificación abierta, construido sobre tecnologías también abiertas como XML y HTTP.

Plugin: (plug-in del inglés "enchufable") El **“**complemento” o “enchufe”, es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica.

LDAP: (Lightweight Directory Access Protocol) es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

AJAX: (Asynchronous Javascript and XML) Javascript y XML asíncrono, es una técnica de desarrollo Web para crear aplicaciones.

RIA: (Rich Internet Application) Aplicaciones de Internet Ricas.

CASE: (Computer Aided Software Engineering) Ingeniería de Software Asistida por Computación. Son aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software y reducir el coste de las mismas.

Inyección de comandos OS: (Operative System) Es un tipo de ataque de inyección de comandos donde la entrada del usuario es pasada a una función que ejecuta comandos del sistema operativo.