Universidad de las Ciencias Informáticas
FACULTAD 6

Título: Módulo de Seguridad para la Solución Integral de Gestión de Datos (SIGDAT)

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor(es): Yuned Rivero Guerra
René Leandro Cruz Laguna
Tutor(es): Ing. Jorge Bedoya Rusenko
Ing.Yanet Rosales Morales

La Habana, mayo de 2013
"Año 54 de la Revolución"

"Para el logro del triunfo siempre ha sido indispensable pasar por la senda de los sacrificios."

Simón Bolívar

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Yuned Rivero Guerra

_____

Firma del Autor

Yanet Rosales Morales

_____

Firma del Tutor

René Leandro Cruz Laguna

_____

Firma del Autor

Jorge Bedoya Rusenko

_____

Firma del Tutor

Jorge Bedoya Rusenko:

Correo electrónico: jbedolla@uci.cu

El tutor Jorge Bedoya Rusenko es graduado de Ingeniero en Ciencias Informáticas. Actualmente forma parte del departamento de Integración de Soluciones del centro DATEC donde desempeña el rol de desarrollador.

Yanet Rosales Morales:

Correo electrónico: yrmorales@uci.cu

La tutora Yanet Rosales Morales es graduada de Ingeniero en Ciencias Informáticas. En estos momentos desempeña el rol de analista en el departamento de Integración de Soluciones.

Yuned Rivero Guerra:

Correo electrónico: yrivero@estudiantes.uci.cu

René Leandro Cruz Laguna:

Correo electrónico: rlcruz@estudiantes.uci.cu

Agradecimientos de Yuned

Hoy se hace realidad mi sueño, la niña de hace 5 años atrás alcanzó su meta y después de tanto esfuerzo y sacrificios se está graduando. Fue un camino largo, con obstáculos, enseñanzas, alegrías e inclusos con lágrimas, pero nunca perdí el paso, pues siempre conté con el apoyo de mis seres queridos y mis amigos. Hoy quiero agradecerles a todas esas personitas que de una forma u otra me apoyaron y ayudaron para estar aquí hoy.

A mi mamá por ser la mejor mami del mundo, por brindarme siempre su apoyo incondicional y darme las fuerzas para vencer cualquier obstáculo durante estos cinco años.

A mi hermana por ser mi amiga, mi confidente, por apoyarme en todas mis decisiones y por ser un ejemplo para mí.

A mis tíos Ana y Guille por ser mis segundos papas y quererme tanto como a una hija.

A Fredy por ser ese papá que me enseñó que con esfuerzo y dedicación todo en la vida se logra.

A mis primitas Deyi y Dayi por ser esas hermanas pequeñas que me llenaban de risas y cariño cuando más triste estaba.

A mi novio Jorge por soportar mis malos días y ser ese que siempre estuvo ahí para mí cuando el día iba mal dándome apoyo y ánimos.

A mi dúo de tesis por esas noches que compartimos de sueño en el laboratorio y por convertirse en un buen amigo.

A Lislien, Yailema, Anita, Sureny, Popi, Migue, Eudel, Darlon, el Potin, Fernan, Ernesto, por ser esos amigos con los que compartí tantos buenos momentos juntos y que nunca olvidaré.

A Grey, Yosbel, Yanet por todos esos días que pasamos en el laboratorio tesiando ayudándonos unos a los otros y por todas las risas y momentos buenos que compartimos este año.

A Garnache por ser mi guía, mi consejero y por todo el apoyo que me brindó.

A las chicas del 93-204 por ser esas tardes y noches de charlas riéndonos y liberando el estrés.

A mis tutores por su ayuda.

En fin a todas esas personas y familiares que de una forma u otra me ayudaron y formaron parte de mi vida en estos 5 años de la carrera.

Agradecimientos de René

Agradezco a toda mi familia por su apoyo y comprensión durante estos 5 años.

A mi mamá por sus consejos y por siempre apoyarme en todas mis decisiones.

A mi abuela por ser mi segunda mamá y por ser esa persona tan especial en mi vida que siempre ha estado para mí cuando la he necesitado.

A mi padre por ser mi amigo y guía inspirador toda la vida.

A mi tío Mariano por todo su apoyo a lo largo de la carrera.

A Irela por ser esa amiga incondicional que se ha convertido en una hermana para mí.

A mi dúo de tesis por ser tan comprensiva y por su apoyo durante todos estos meses.

A Graciela, Marvel, Liandry, Luis Angel, Tomás, Yaisel Alberto Chávez, Jorge Ernesto linares, Ernesto Avilés, Darlon, Alejandro, Osvaldo, Rolando por ser esos amigos con los que compartí buenos momentos durante todos estos años y por convertirse en mí segunda familia.

A Alberto Garnache y Yoandry Pacheco por su ayuda y recomendaciones para lograr un resultado satisfactorio en la tesis.

A Yanet Rosales y Jorge Bedoya por su ayuda incondicional y su tiempo dedicado en estos meses.

A la Revolución y a Fidel por su esfuerzo en construir esta Revolución Socialista y por buscar el sueño de igualdad y de un mundo mejor.

RESUMEN

El departamento de Integración de Soluciones perteneciente al Centro de Tecnologías de Gestión de Datos (DATEC) en la Universidad de las Ciencias Informáticas (UCI), cuenta actualmente con el proyecto Solución Integral de Gestión de Datos (SIGDAT) que realiza un conjunto de procedimientos y acciones tales como la captura, almacenamiento y salida o visualización de la información. En su versión actual la aplicación carece de mecanismos de seguridad que le permitan proteger la información almacenada en la misma. El objetivo del presente trabajo de diploma consiste en desarrollar un módulo que garantice la seguridad de la aplicación SIGDAT. Para guiar el proceso de desarrollo del módulo se caracteriza la metodología, herramientas y tecnologías empleadas. Además se diseñan e implementan las clases del módulo, aplicando buenas prácticas de los patrones de diseño. Se realizaron pruebas funcionales, de rendimiento, integración y seguridad para comprobar el correcto funcionamiento y calidad del módulo de seguridad para la aplicación SIGDAT.

Palabras Claves:
Confidencialidad, disponibilidad, integridad, módulo, seguridad, Sistema de Información.

Índice

Índice de Tablas

Índice de Figuras

INTRODUCCIÓN

Las Tecnologías de la Información y las Comunicaciones (TIC) han tenido un desarrollo acelerado durante las últimas décadas, en este contexto el empleo de Sistemas Informáticos es clave para cualquier esfera de la sociedad. En el ámbito empresarial su uso se ha incrementado considerablemente, debido en gran medida, al hecho de cada vez contar con un mayor volumen de información a gestionar. La importancia de la información administrada en estos sistemas ha despertado gran interés en toda una gama de intrusos cibernéticos, que se han aprovechado de numerosos problemas de seguridad existentes en las aplicaciones, para llevar a cabo toda una serie de ataques como inyecciones SQL, denegación de servicio, autenticación entre otros, es entonces vital disminuir las posibles vulnerabilidades que pudiesen existir en estos sistemas.

El cumplimiento de este objetivo recae fundamentalmente en la seguridad informática la cual se encarga de asegurar la confidencialidad, integridad y disponibilidad de la información. Esta se rige por un conjunto de principios tales como mínimo privilegio, eslabón más débil, proporcionalidad, dinamismo y la participación universal. Para aplicar dichos principios se ponen en práctica diferentes técnicas entre las que se encuentran las copias de respaldo, los antivirus, los cortafuegos, los mecanismos de autenticación y la criptografía. Técnicamente es imposible lograr un sistema informático ciento por ciento seguro, pero buenas medidas de seguridad resultan útiles para evitar ataques que pudieran traer como consecuencia el colapso de estos sistemas o lo que es peor el robo o pérdida de la información gestionada por los mismos. El departamento de Integración de Soluciones perteneciente al Centro de Tecnologías y Gestión de Datos (DATEC) se dedica al desarrollo de sistemas informáticos encaminados a satisfacer las necesidades de gestión de la información. Durante el desarrollo de estos sistemas se tiene presente como requisito indispensable garantizar la seguridad de la información que manipularán. Actualmente se encuentra en desarrollo la Solución Integral de Gestión de Datos (SIGDAT), la cual realiza un conjunto de procedimientos y acciones integradas que responden a algunas etapas del ciclo de vida de la información tales como la captura, almacenamiento y salida o visualización de la misma. En la versión actual de la aplicación no se garantizan elementos claves como la confidencialidad, integridad y disponibilidad de la información gestionada. Lo anteriormente expuesto se evidencia ya que no se logra identificar los usuarios que tienen acceso a la aplicación, además de que no se pueda controlar los permisos de estos sobre los recursos del sistema, trayendo como consecuencia que cualquier persona pueda acceder al módulo que desee y logre crear, eliminar o modificar los recursos existentes sin una previa autorización. Además la aplicación puede verse comprometida ante la realización de ataques de fuerza bruta, inyección o de

secuencia de comandos en sitios cruzados, obteniéndose como resultado que el atacante logre apropiarse de datos sensibles del sistema. Al mismo tiempo la información que se transmite no se encuentra codificada por lo que puede ser monitoreada por personas no autorizadas.

De acuerdo a la situación problemática antes expuesta, en el siguiente trabajo se plantea como **problema de la investigación**: ¿Cómo garantizar la seguridad de la Solución Integral de Gestión de Datos?

Defendiéndose como **objeto de estudio**: la seguridad de la información en los sistemas informáticos, enmarcado en el **campo de acción**: la seguridad de la información en los sistemas web.

Para darle solución al problema antes expuesto se ha propuesto como **objetivo general**: Desarrollar un módulo que garantice la seguridad de la Solución Integral de Gestión de Datos.

En correspondencia con ello, se plantean como **objetivos específicos:**

- Definir los conceptos, metodología, tecnologías y herramientas necesarias para el desarrollo del módulo de seguridad de la Solución Integral de Gestión de Datos.
- Realizar el análisis y diseño del módulo de seguridad de la Solución Integral de Gestión de Datos.
- Realizar la implementación y prueba del módulo de seguridad de la Solución Integral de Gestión de Datos.

Con el fin de resolver el problema de la investigación y darle cumplimiento a los objetivos antes planteados se exponen las siguientes **tareas de la investigación** a realizar:

- Análisis de los conceptos básicos y técnicos implicados en el desarrollo del módulo de seguridad.
- Identificación de los requisitos funcionales y no funcionales del módulo de seguridad.
- Confección del modelo de casos de uso del sistema del módulo de seguridad.
- Elaboración del modelo de diseño del módulo de seguridad.
- Elaboración del modelo de implementación del módulo de seguridad.
- Implementación del módulo de seguridad.
- Integración del módulo de seguridad.
- Diseño de los casos de prueba a partir de los requisitos funcionales para llevar a cabo la realización de las pruebas.
- Realización de pruebas de seguridad, funcionales, rendimiento e integración para comprobar el correcto funcionamiento del módulo.

El presente trabajo de diploma está estructurado en 3 capítulos:

CAPÍTULO I: FUNDAMENTOS TEÓRICOS

En este capítulo se hace un estudio de la seguridad informática en los sistemas y vinculado a ello se realiza un análisis de los principales mecanismos de autenticación y autorización que existen. Además se fundamentan las tecnologías, herramientas y metodología sobre las cuales se asentará el desarrollo posterior del módulo.

CAPÍTULO II: ANÁLISIS Y DISEÑO DEL MÓDULO

En este capítulo se describe el análisis y diseño de las clases del sistema. Se hace un levantamiento de los requisitos funcionales y no funcionales necesarios para lograr que el sistema funcione correctamente. Se construyen los artefactos correspondientes al análisis y diseño, modelando la estructura del módulo.

CAPÍTULO III: IMPLEMENTACIÓN Y PRUEBA DEL MÓDULO

En este capítulo se muestra el modelo de implementación como resultado del diseño anteriormente desarrollado, así como el diagrama de componentes del módulo de seguridad. Se describen las pruebas a realizar, con el objetivo de comprobar el correcto funcionamiento del módulo.

CAPÍTULO I "FUNDAMENTOS TEÓRICOS"

Introducción

En el presente capítulo se abordarán, los elementos principales que justifican y soportan teóricamente la solución propuesta. Se hará un estudio de la Seguridad Informática en los sistemas y vinculado a ello se realizará un análisis de los principales mecanismos de autenticación y autorización que existen. Además se presentarán las tecnologías, herramientas y metodología que serán empleadas en el desarrollo posterior de la solución.

1.1 Seguridad en los sistemas

Se puede entender como seguridad una característica de cualquier sistema (informático o no) que indica que ese sistema está libre de todo peligro, daño o riesgo, y que es, en cierta manera, infalible. A grandes rasgos se entiende que mantener un sistema seguro (o fiable) consiste básicamente en garantizar tres aspectos: confidencialidad, integridad y disponibilidad. (Villalón Huerta, 2009)

La confidencialidad dice que los objetos de un sistema han de ser accedidos únicamente por elementos autorizados a ello y que esos elementos autorizados no van a convertir esa información en disponible para otras entidades. La integridad significa que los objetos solo pueden ser modificados por elementos autorizados, y de una manera controlada. La disponibilidad indica que los objetos del sistema tienen que permanecer accesibles a elementos autorizados; es el contrario de la negación de servicio.

Seguridad en los sistemas web

La seguridad puede ser aplicada tanto a sistemas de escritorio como a sistemas web debido a que ambos están expuestos a recibir cualquier tipo de ataques de intrusos cibernéticos. Debido a la alta demanda de los servicios en internet donde se puede acceder a la información a través de la red, juega un papel importante hoy en día los sistemas para la web. Cuando se refiere al uso de dichos sistemas se puede hablar de múltiples ventajas como: el ahorro de tiempo, la compatibilidad, un consumo bajo de recursos, la portabilidad y por último la posibilidad de contar con herramientas y funcionalidades para interpretar aplicaciones inteligentes de este tipo.

Garantizar la seguridad en los sistemas web se muestra cada vez más complejo debido a que existen muchas razones que pueden poner en peligro la información, por este motivo es de gran importancia el estudio de los riesgos a los que están expuestos frecuentemente. A continuación se realizará una explicación detallada de algunos de los ejemplos existentes**:** (OWASP Top 10, 2010)

1-Inyección: Las fallas de inyección, tales como SQL, OS y LDAP, suceden cuando los datos no confiables son enviados a un intérprete como parte de un comando o consulta; una de las variantes más

extendida y peligrosa son las de inyección SQL. Para explotar una falla de este tipo, el atacante debe de encontrar un parámetro que la aplicación pase hacia la base de datos. Mediante la inserción cuidadosa de comandos SQL la aplicación puede ser engañada para que envíe una solicitud maliciosa a la base de datos. Las consecuencias son muy dañinas debido a que un atacante puede obtener, corromper o destruir la base de datos.

Para evitar este tipo de fallas se debe de utilizar una interfaz de programación de aplicaciones o API (del inglés *Application Programming Interface*) segura que evite el uso del intérprete completamente o provea una interface parametrizada. Otro de los elementos a tener en cuenta es la validación de las entradas de datos, se debe de verificar el tamaño, el tipo, la sintaxis y las reglas del negocio antes de aceptar los datos que se van almacenar. Además se debe de conceder los mínimos privilegios al realizar una conexión a la base de datos y evitar los mensajes de error detallados que son útiles para un atacante.

2- Secuencia de comandos en sitios cruzados (XSS)

Una aplicación es considerada vulnerable ante ataques XSS cuando toma datos no confiables y los envía al navegador web sin una validación y codificación apropiada. Este tipo de ataque se divide en dos categorías persistente y no persistente. Se consideran dentro de la primera categoría aquellos que logran que el código malicioso sobrepase las validaciones y se almacene en alguna base de datos o fichero que el usuario utilice en algún momento. En cambio se clasifica como no persistente cuando no se almacena en el servidor si no que pasa directamente a la víctima desde una fuente externa como un correo, un mensaje de chat u otro sitio web.

Los ataques XSS permiten a los atacantes secuestrar las sesiones de usuario, destruir sitios web, o dirigir al usuario hacia un sitio malicioso. Una opción para resolver el problema es validar la longitud, caracteres, formato de los datos antes de aceptar su entrada.

3- Pérdida de Autenticación y Gestión de Sesiones

Resulta frecuente que las funciones relacionadas con la autenticación y gestión de sesiones son implementadas incorrectamente, permitiendo a los atacantes comprometer contraseñas, llaves, token de sesiones, o explotar diferentes fallas de implementación para asumir la identidad de otros usuarios. Entre los principales tipos de ataques que se pueden producir sobre la gestión de sesiones web se encuentran predicción de sesión, a través de XSS, de fijación de sesión, interceptando la comunicación y mediante errores en el cierre de sesión. A continuación se analizan cada uno de ellos de manera detallada:

Predicción de sesión

Este ataque se caracteriza por la generación de identificadores a partir de los patrones que utilice el servidor. El atacante mediante fuerza bruta prueba todas las posibles variantes hasta encontrar el identificador correcto. La solución para este tipo de riesgo es una suficiente longitud del identificador de sesión.

Captura del identificador a través de ataques XSS

Si una página web presenta una vulnerabilidad XSS un atacante puede aprovecharla para ejecutar código que capture el contenido de la cookie y se lo envíe. En la figura 1 se puede observar un ejemplo donde se hace un ataque XSS para robar el identificador de sesión del usuario.

```
<script type="text/javascript">
new Image().src="http://localhost/?"+encodeURI(document.cookie);
</script>
```

Figura 1 Ejemplo donde se evidencia un ataque XSS

Para eliminar este riesgo se debe de activar la opción httponly en el servidor web de modo que el navegador impida el acceso por medio de scripts a las cookies[1] que tienen este atributo, además se debe de deshabilitar el método TRACE.

Fijación de sesión

Se caracteriza por generar un identificador genuino que no se encuentra asociado a ningún usuario. Luego el atacante espera que algún usuario se autentique y utilice el identificador generado, de esta manera puede realizar acciones en la aplicación afectada con la identidad de la víctima.

Interceptando la comunicación

Este ataque tiene como objetivo capturar el identificador de sesión a partir de la intercepción del tráfico entre el usuario y la aplicación. La solución es hacer uso del protocolo HTTPS para que la comunicación sea cifrada incluyendo la cookie de sesión. Además se debe de activar la opción session.cookie_secure que impide que el navegador pueda enviar la cookie por HTTP y, por tanto, sea posible obtenerla interceptando el tráfico.

Errores en el cierre de sesión

Entre los problemas más comunes relacionados con el cierre de sesión se encuentra la reutilización del identificador de sesión y la no invalidación del mismo ante el cierre de la sesión. Para evitar estos tipos de errores es importante el establecimiento de un tiempo de desconexión y la invalidación del identificador de sesión ante el cierre de la misma, la primera medida permite que ante cierto tiempo de inactividad del usuario se cierre la sesión y se impida que personas no autorizadas puedan utilizarla, por su parte la segunda sirve de apoyo a la anterior y se debe de realizar luego de alcanzar el tiempo establecido para la desconexión o cuando el usuario decide salir del sistema.

4- Referencia Directa Insegura a Objetos

[1] Representan una pequeña porción de información con una fecha de caducidad que se almacena en el fichero o directorio de cookies, referentes a informaciones de las preferencias de los usuarios.

Una referencia directa a objetos ocurre cuando un desarrollador expone una referencia a un objeto de implementación interno, tal como un fichero, directorio, o base de datos. Sin un chequeo de control de acceso u otra protección, los atacantes pueden manipular estas referencias para acceder datos no autorizados. Una posible solución a este tipo de ataque es realizar una comprobación de control de acceso para asegurar que el usuario esté autorizado acceder al objeto solicitado.

5- Falsificación de Peticiones en Sitios Cruzados (CSRF)

Un ataque CSRF obliga al navegador de una víctima autenticada a enviar una petición HTTP falsificada a una aplicación web vulnerable, incluyendo la cookie de sesión del usuario o cualquier otra información incluida automáticamente. Lo anteriormente mencionado permite al atacante alterar los datos o acceder a cualquier funcionalidad que la víctima tenga acceso. Para prevenir dicho ataque se necesita incluir un testigo único y no predecible por cada sesión de usuario en el cuerpo o URL de cada petición HTTP.

6- Defectuosa Configuración de Seguridad

Es recomendable tener definida e implementada una configuración segura para la aplicación, marcos de trabajo, servidor de aplicación, servidor web, base de datos y plataforma. Partiendo de lo anteriormente mencionado se debe de tener en cuenta un conjunto de medidas como por ejemplo la actualización de todo el software, la definición de una arquitectura robusta en las aplicaciones que permitan proveer una buena separación y seguridad entre los componentes, además de la realización de auditorías para ayudar a detectar fallos en la configuración o parches faltantes.

7- Almacenamiento Criptográfico Inseguro

Esta vulnerabilidad se encuentra presente en aplicaciones con una protección débil o nula como resultado de mala implantación de funciones criptográficas o algoritmos de cifrado a nivel del almacenamiento. Al no hacer uso de algoritmos de cifrado se corre el riesgo de comprometer la información sensible como cuentas de usuarios y tarjetas de crédito. Teniendo en cuenta lo anteriormente expuesto se hace necesario el uso adecuado de algoritmos estándares robustos, las claves usadas deben ser fuertes y debe de existir una gestión de adecuada de las mismas. Además se deben almacenar en forma de hash y con salt[2] para protegerlas del acceso no autorizado.

8-Falla de Restricción de Acceso al localizador de recursos uniforme o URL (del inglés *uniform resource locator*)

Frecuentemente las aplicaciones solo garantizan la protección de las funcionalidades delicadas previniendo la visualización de enlaces o URLs a usuarios no autorizados. Esta vulnerabilidad puede ser aprovechada por un atacante para llevar a cabo operaciones no autorizadas accediendo directamente a la

[2] Pequeño dato añadido que hace que los hash sean más difíciles de crackear.

URL. Prevenir el acceso no autorizado a URLs demanda planificar un método que requiera autenticación y autorización adecuadas para cada página.

Con independencia del mecanismo o mecanismos a utilizar es recomendable que la autenticación y autorización deben estar basadas en roles, para minimizar el esfuerzo necesario para mantener estas políticas y la implementación del mecanismo debería negar todo acceso por defecto, requiriendo el establecimiento explícito de permisos a usuarios y roles específicos por cada página.

9-Protección insuficiente en la capa de transporte

Las aplicaciones frecuentemente fallan al autenticar, cifrar y proteger la confidencialidad e integridad de tráfico de red. Cuando esto ocurre, es debido a la utilización de algoritmos débiles, certificados expirados, inválidos, o sencillamente no utilizados correctamente. Los problemas expuestos anteriormente pueden traer como consecuencia el robo de las cuentas de los usuarios y en el peor de los casos si se logra comprometer la cuenta de administración se vería expuesta toda la aplicación.

Entre las medidas para solucionar esta vulnerabilidad se encuentran: requerir capa de conexión segura o SSL (del inglés *Secure Sockets Layer*) para todas las páginas sensibles, establecer el atributo "secure" en todas las cookies sensibles, además las conexiones a sistemas finales (back-end) y otros sistemas también deben utilizar SSL.

10-Redirecciones y reenvíos no validados: Las aplicaciones web frecuentemente redirigen y reenvían a los usuarios hacia otras páginas o sitios web y utilizan datos no confiables para determinar la página de destino. Sin una validación apropiada, los atacantes pueden redirigir a las víctimas hacia sitios de phishing[3] o malware[4], o utilizar reenvíos para acceder páginas no autorizadas. Para evitar este tipo de fallas se recomienda tratar de evitar el uso de redirecciones y reenvíos. En el caso de utilizarlo, no involucrar parámetros manipulables por el usuario para definir el destino. De lo contrario, es importante verificar que el valor facilitado es válido y autorizado para el usuario.

A partir del estudio de los riesgos anteriormente expuestos y con el objetivo de proporcionar un mejor manejo de las sesiones, se implementará un mecanismo que permitirá invalidar la sesión del usuario una vez que este deje de hacer peticiones al sistema en un tiempo configurable en la aplicación. Además se implementarán las funcionalidades necesarias para ofrecer la posibilidad de visualizar los usuarios conectados y cerrar la sesión de los mismos. Con respecto al riesgo "Almacenamiento criptográfico inseguro la medida a tomar será la de utilizar el algoritmo generador de hash SHA1 aplicando una salt

[3] Modalidad de estafa que tiene como objetivo intentar obtener datos de un usuario tales como claves, cuentas bancarias y números de tarjeta de crédito.

[4] Tipo de software que tiene como objetivo infiltrarse o dañar una computadora o sistema de información sin el consentimiento de su propietario.

única para posibilitar una mayor seguridad en las contraseñas almacenadas. Con relación al riego “Referencia Directa Insegura a Objetos”, el sistema verificará que el usuario al llevar a cabo una petición tenga permisos necesarios sobre él o los objetos involucrados en la misma. Otro de los riesgo que se tendrá en cuenta es “Protección insuficiente en la capa de transporte”**,** en la aplicación se requerirá que el usuario que desee conectarse utilice el protocolo HTTPS.

Para evitar posibles inyecciones SQL los datos serán validados en la parte del cliente, y en las entidades, aprovechando las facilidades de los validadores a través de anotaciones que proporciona Symfony. Durante el proceso de autenticación se hará uso de una prueba de Turing más conocida como CAPTCHA para evitar posibles ataques de fuerza bruta. Para fortalecer el proceso de autorización se implementarán funcionalidades que permitan la asignación de direcciones IP desde donde podrán conectarse los usuarios. Se implementará la política para que durante la creación de los usuarios locales se requiera que la contraseña posea más de ocho caracteres de longitud, teniendo una combinación de letras mayúsculas, minúsculas, dígitos y caracteres especiales.

Se propone además realizar la correcta configuración en el servidor web donde se despliegue la aplicación, teniendo en cuenta la configuración de la etiqueta httponly y deshabilitar el método TRACE para evitar posibles ataques XSS. Se activará la opción session.cookie_secure evitando que el navegador pueda enviar la cookie por HTTP y, por tanto, sea imposible obtenerla interceptando el tráfico además de configurar el protocolo HTTPS para fortalecer la seguridad en la capa de transporte. Por último se recomienda desplegar todas las actualizaciones y parches de software de manera oportuna para garantizar una mayor seguridad en el ambiente de despliegue donde se encuentre instalada la aplicación.

1.2 Mecanismos de Autenticación

La seguridad es un proceso de dos etapas, cuyo objetivo es evitar que un usuario acceda a un recurso al cual él/ella no debería tener acceso. En el primer paso del proceso, el sistema de seguridad identifica quién es el usuario obligándolo a presentar algún tipo de identificación. Esto se llama autenticación, y significa que el sistema está tratando de averiguar quién es el usuario, lo más común es utilizar una combinación de identificador de usuario único y contraseña, aunque existen otros. (Pacheco, 2012)

Los métodos de autenticación se suelen dividir en tres grandes categorías en función de lo que utilizan para la verificación de identidad:

Algo que el usuario sabe: el usuario debe demostrar que conoce algún tipo de información secreta, por ejemplo, una palabra clave, un número de identificación personal (PIN), una clave privada.

Algo que el usuario posee: El usuario requiere para identificarse algún tipo de dispositivo, como: una tarjeta magnética, una tarjeta inteligente, un generador de contraseñas o una llave electrónica.
Una característica física del usuario o un acto involuntario del mismo: son conocidos como sistemas de autenticación biométrica basados en características físicas del usuario a identificar.

A partir del estudio anteriormente realizado de los diferentes métodos de autenticación y de las características de la aplicación SIGDAT se decide implementar el método "Algo que el usuario sabe", siendo necesario introducir un nombre de usuario y contraseña para autenticarse en el sistema. Las credenciales suministradas por el usuario serán validadas empleando los datos de los usuarios almacenados en la base de datos local o en un Protocolo Ligero de Acceso a Directorios (LDAP) que será configurable desde la aplicación. Para evitar posibles ataques de fuerza bruta durante el proceso de autenticación se implementará una prueba de Turing para diferenciar máquinas y humanos.

Prueba de Turing para diferenciar máquinas y humanos (CAPTCHA)

Prueba de Turing para Diferenciar Máquinas y Humanos o CAPTCHA (del inglés *Completely Automated Public Turing Tests to Tell Computers and Humans Apart)*, representa uno de los mecanismos más utilizados para que un servidor HTTP pueda identificar si la petición a un determinado servicio ha sido realizada por un humano. Tiene entre sus características que son completamente automatizados, es decir, no es necesario ningún tipo de mantenimiento o intervención humana para su realización. Esto supone grandes beneficios en cuanto a fiabilidad y costo. El CAPTCHA tiene varias aplicaciones para la seguridad práctica, incluyendo: (Carnegie Mellon, 2010)

La prevención de spam en los comentarios en los blogs. La mayoría de los bloggers están familiarizados con los programas que envían comentarios falsos, por lo general con el fin de aumentar las filas del motor de búsqueda de un sitio web. Esto se conoce como spam en los comentarios.
La protección de registro del usuario. El registro de usuarios se encuentra expuesto a script automatizados que son capaces de crear miles de cuentas por minutos por esta razón se hace necesario utilizar letras cifradas para garantizar que sólo los humanos puedan obtener cuentas gratuitas.
Prevención de ataques de diccionario. Durante el proceso de autenticación en sistemas de contraseña resulta importante evitar que un ordenador sea capaz de recorrer todo el espacio de contraseñas. Una vía para lograrlo es obligarle a resolver un CAPTCHA después de un cierto número de intentos fallidos de acceso. Esto es mejor que el enfoque clásico de bloquear una cuenta después de una serie de intentos fallidos de acceso, ya que ello permite a un atacante bloquear las cuentas que desee.

Tipos de Captchas:

Con el fin de brindarle mayor seguridad a las aplicaciones se hace uso de diferentes tipos de Captchas. A continuación se hace mención de los distintos tipos de CAPTCHAs existentes**:**

1. **Establecer la relación entre imágenes y caracteres:** Este es uno de los tipos de CAPTCHAs más comunes encontrados hoy en día en los sitios web y en las aplicaciones. Se trata de reconocer los caracteres alfanuméricos que aparecen en la imagen y reproducirlos en una caja de texto.
2. **Resolver un problema de tipo matemático:** Otro tipo de los más comunes que se ve a diario suelen ser operaciones sencillas, como por ejemplo, sumas con sumandos de un solo dígito.
3. **Resolución de puzles y rompecabezas:** Quizás este sea uno de los tipos menos comunes y más difíciles de realizar, sobre todo si se limitan por tiempo.
4. **Establecer la relación entre sonidos y caracteres. (CAPTCHAs auditivos):** En este tipo de CAPTCHA el usuario para poder pasar la prueba tiene que escribir en la caja de texto los caracteres asociados al sonido que se reproducirá.
5. **Tablas de coordenadas por palabras:** Este tipo de CAPTCHA está compuesto por una tabla que contiene un grupo de palabras que son agrupadas en filas y columnas, el usuario para poder pasar la prueba tiene que escribir en la caja de texto la palabra correspondiente a la intersección de la fila y la columna que el sistema le indique.

Como resultado del estudio de los principales tipos de CAPTCHAs existentes y con el objetivo de brindar una mayor seguridad al proceso de autenticación, se decide implementar el tipo de CAPTCHA denominado *"***Establecer la relación entre imágenes y caracteres***"*.

Protocolo LDAP

Protocolo Ligero de Acceso a Directorios o LDAP *(*del inglés *Lightweight Directory Access Protocol)* está enmarcado en el grupo de protocolos de tipo cliente-servidor para acceder a un servicio de directorio sobre TCP/IP. Generalmente, los datos de un directorio LDAP siguen un modelo de organización de información en árbol. La versión original fue desarrollada por la Universidad de Michigan en el año 1993. Algunos de los mecanismos de autenticación que soporta LDAP son: autenticación anónima y autenticación simple (contraseña en texto claro). (Gómez Oiner, 2012)

Funcionamiento:

LDAP provee diferentes operaciones entre las que se encuentran añadir, borrar y actualizar entradas en el directorio. Las operaciones de búsquedas se hacen sobre la base de algún criterio especificado por un filtro de búsqueda. La búsqueda se realiza de la siguiente manera :primero el cliente se conecta al

servidor y les formula preguntas, luego el servidor responde con una respuesta o con un puntero donde el cliente puede obtener información adicional (normalmente otro servidor LDAP).Un elemento importante a tener en cuenta es que no importa a que servidor LDAP se conecte un cliente este siempre obtendrá la misma visión sobre un directorio ya que un nombre presentado por un servidor LDAP referencia la misma entrada que cualquier otro servidor LDAP.

1.3 Mecanismos de Autorización

El proceso de autorización se nutre de los datos que arroja como salida el proceso de autenticación para distinguir los privilegios de cada usuario. Tiene la responsabilidad de establecer las políticas o privilegios de acceso de los usuarios sobre los recursos en los diferentes dominios. A continuación se analizan las principales soluciones desarrolladas para fortalecer el proceso de autorización:
(Rodríguez, 2010)

Control de Acceso Obligatorio (MAC)

En el modelo de control de acceso obligatorio se asignan las funciones de los usuarios estrictamente de acuerdo a lo establecido por el administrador del sistema. Este es el método de control de acceso más restrictivo**,** porque el usuario final no puede establecer controles de acceso en los archivos**.** El Control de acceso obligatorio es muy popular en ambientes instalaciones altamente secretas, como la industria de defensa donde los archivos “perdidos” pueden afectar a su seguridad nacional.

Control de Acceso Discrecional (DAC):

El control de acceso discrecional está en el otro extremo del espectro de acceso, diferente del modelo de acceso obligatorio, ya que es el menos restrictivo de los tres modelos. En el marco del modelo de acceso discrecional el usuario final tiene total libertad para asignar los derechos a los objetos que desea. Este nivel de control completo sobre los archivos puede ser peligroso porque si un atacante o algún Malware compromete la cuenta a continuación, el usuario malicioso o código tendrá un control completo también.

Control de Acceso Basado en Atributos (ABAC):

En el modelo de Control de Acceso Basado en Atributos o ABAC (del inglés *Attribute Based Access Control*), los privilegios son establecidos en base a la colección de atributos que poseen los sujetos y una política que los determina. En la infraestructura de gestión de políticas se utilizan certificados de atributos para asignar un conjunto de privilegios a cada usuario. La seguridad que proporciona ABAC depende del número de atributos y reglas que se establezcan. Este aspecto influye de forma negativa en el rendimiento de los sistemas informáticos a la hora de realizar las búsquedas para conceder o no el acceso. La ausencia del concepto rol, aumenta la complejidad de mantenimiento de las políticas en entornos heterogéneos y dinámicos.

Controles de Acceso Basado en Roles (RBAC)

La definición básica de RBAC establece que los usuarios son asignados a roles, los permisos son asociados a roles y los usuarios adquieren permisos siendo miembros de roles. Este modelo de control de acceso funciona de manera efectiva en las organizaciones reales, debido a que a los archivos y los recursos se le asignan los permisos de acuerdo a las funciones que lo requieran. Los permisos de control de acceso sólo son asignados por el administrador del sistema. Las asignaciones usuario-rol y permiso-rol pueden ser muchos-a-muchos, por lo que un usuario puede pertenecer a muchos roles y un rol puede poseer muchos usuarios. De manera similar un permiso puede ser asociado a muchos roles y un rol puede tener asociado muchos permisos.

Listas de Control de Acceso (ACL)

En el año 1971, Lampson propuso un modelo que permite establecer el control de acceso a través de una matriz que se enmarca dentro de la estrategia de control de acceso discrecional. A partir de esta solución surgen las Listas de Control de Acceso o ACL (del inglés *Access Control List*). Una ACL puede ser considerada como una estructura de datos jerárquica, cada una de ellas define un conjunto de permisos asociados a un determinado recurso. Los sujetos, objetos y permisos son los conceptos fundamentales que se utilizan para definir las reglas de acceso aplicadas directamente a los usuarios o a los grupos donde pertenecen. (Gómez Oiner, 2012)

Luego de haber analizado cada una de las soluciones existentes en la actualidad para llevar a cabo el proceso de autorización, se decidió para controlar cada uno de los recursos del sistema implementar en la aplicación SIGDAT una lista de control de acceso ya que se adapta correctamente al sistema y además el marco de trabajo Symfony 2.0 proporciona facilidades para el uso de ACL. Además se implementará un control de acceso basado en roles debido a que este funciona de manera efectiva en las organizaciones reales y permite definir los permisos a los usuarios en dependencia del rol que le sea asignado, lo cual proporciona una gran flexibilidad a la hora de llevar a cabo el proceso de autorización.

1.4 Metodología de desarrollo del software

Las metodologías son un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software. La selección de la metodología a emplear, propone los roles que el equipo de desarrollo deberá cubrir, las actividades que debe cumplir cada uno de ellos, los artefactos que serán generados de cada tarea, así como el registro de cada detalle de la información que se irá generando. En la actualidad existe un gran número de metodologías y herramientas capaces de brindar todo tipo de soluciones, entre ellas se destaca OpenUP.

OpenUP

OpenUP es un proceso modelo y extensible, dirigido a la gestión y desarrollo de proyectos de software basados en un desarrollo iterativo, ágil e incremental apropiado para proyectos pequeños y de bajos recursos; es aplicable a un conjunto amplio de plataformas y aplicaciones de desarrollo. El ciclo de vida de un proyecto según esta metodología se divide en 4 fases fundamentales: Concepción, Elaboración, Construcción y Transición. (Sánchez, 2010)

Después de realizar un análisis de la metodología se seleccionó OpenUP por las siguientes características:

- Permite detectar errores tempranos a través de un ciclo iterativo.
- Evita la elaboración de documentación, diagramas e iteraciones innecesarios requeridos en la metodología RUP.
- Por ser una metodología ágil tiene un enfoque centrado al cliente y con iteraciones cortas.
- Se acoge a las necesidades del equipo de desarrollo.
- Es la metodología utilizada en el departamento.

1.5 Herramientas y tecnologías

Las herramientas son programas, aplicaciones o simplemente instrucciones que ofrecen la posibilidad de realizar varias funcionalidades con diferentes propósitos. Por otra parte, las tecnologías son el conjunto de conocimientos técnicos, ordenados científicamente que permiten diseñar y crear bienes y servicios. La correcta selección de las herramientas y tecnologías informáticas a través de un estudio profundo de las características que las identifican, permiten agilizar y facilitar el desarrollo de los sistemas informáticos.

1.5.1 Lenguajes de programación

Un lenguaje de programación es aquel elemento dentro de la informática que nos permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador para que este pueda comunicarse con los dispositivos hardware y software existentes. (Marin, 2008)

PHP

PHP (Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para desarrollo web y que puede ser incrustado en HTML. Además es un lenguaje interpretado de alto nivel embebido (introducido) en páginas HTML y ejecutado en el servidor. El código PHP se incluye entre etiquetas especiales de comienzo y final que nos permitirán entrar y salir del modo PHP. Una de las características más potentes de PHP es su soporte para una gran cantidad de bases de datos. (Bakken, 2011)

Se seleccionó para el desarrollo de la solución la versión 5.3 del lenguaje de programación PHP, por las siguientes características:

- Soporte para espacios de nombres (namespaces).
- Mejor soporte para la Programación Orientada a Objetos, que en versiones anteriores era extremadamente rudimentario, con PHP Data Objects.
- Mejoras de rendimiento.
- Mejor soporte para MySQL con extensión completamente reescrita.
- Mejor soporte a XML (XPath, DOM, etc).
- Soporte nativo para SQLite.
- Soporte integrado para SOAP.
- Iteradores de datos.

Java Script

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. (Pérez, 2012)

Características de JavaScript:

- Es sencillo (su hermano mayor: el Java, es bastante más complejo).
- Es útil (el desarrollo de Internet, se prevé muy rápido en los próximos años).
- Es potente: permite la moderna POO (programación orientada a objetos).
- Es barato: sólo necesitamos un editor de textos y un navegador.
- Es visual: permite la moderna “programación visual” (ventanas, botones, colores, formularios).

1.5.2 Entorno de desarrollo

Un entorno de desarrollo informático IDE (Integrated Development Environment) es un programa informático compuesto por un conjunto de herramientas de programación. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#, Visual Basic, etc. (Maldonado, 2012)

Netbeans

El IDE NetBeans es un entorno galardonado de desarrollo integrado disponible para Windows, Mac, Linux y Solaris. El proyecto NetBeans consiste en un IDE de código abierto y una plataforma de aplicaciones que permiten a los desarrolladores crear rápidamente aplicaciones utilizando la plataforma Java, así como PHP, JavaScript, Ajax, Groovy y Grails, y C / C + +.

La versión que se seleccionó para el desarrollo de la solución del problema de la investigación es NetBeans IDE 7.2. Esta versión ofrece un rendimiento significativamente mejorado y la experiencia de codificación, con las nuevas capacidades de análisis de código estático en el Editor de Java y más inteligente proyecto de exploración. El lanzamiento también incluye características notables, como: (NetBeans, 2011)

- La integración con el Generador de Escena para la creación de formas visualmente JavaFX.
- Mejora de la velocidad de escaneo.
- Mejora del rendimiento de sistemas de archivos remotos.
- Soporte para PHP 5.4.
- Apoyo a marcos de PHP múltiples.
- Mejoras en Java EE, Maven, C / C + + y la plataforma NetBeans.

1.5.3 Gestor de bases de datos

Un Sistema Gestor de Base de Datos (SGBD, en inglés DBMS: DataBase Management System) es un sistema de software que permite la definición de bases de datos; así como la elección de las estructuras de datos necesarios para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación. (BERTINO, E. A. MARTINO,2011)

PostgreSQL 9.1.6

PostgreSQL es un potente motor de bases de datos, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de bases de datos comerciales. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. (Martinez, Rafael, 2010)

Características:

- Es una base de datos 100% ACID.
- Disponible para Linux, UNIX y Windows 32/64bit .
- Completa documentación.
- Licencia BSD.

- Múltiples métodos de autentificación.
- Soporta el almacenamiento de objetos binarios grandes (gráficos, videos, sonido).
- Funciones/procedimientos almacenados (stored procedures) en numerosos lenguajes de programación, entre otros PL/pgSQL (similar al PL/SQL de oracle), PL/Perl, PL/Python y PL/Tcl.

Pgadmin 1.14.0

PgAdmin es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia Open Source .Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. (PGadmin, 2012)

Nuevas características que se han agregado a esta versión:

- **Diálogos para netbooks:** todos los diálogos han sido redimensionado para ser utilizable en una pantalla de 800x600.Contienen tres o más fichas para contener toda la información necesaria sobre cada objeto.
- **Índice:** el cuadro de diálogo Propiedades del índice muestra todas las intercalaciones disponibles y le permite establecer una colación para char, varchar y columnas de texto.
- **Configuración mejorada de plugins**: pgAdmin 1.12 utilizó un solo archivo para almacenar la configuración de todos los plugins disponibles. La versión 1.14 busca los archivos en una carpeta en la que cada archivo contiene información acerca de uno o más plugins. Este nuevo sistema permite que otras aplicaciones se registran como plugins en pgAdmin sin tener que modificar y romper tal vez, el viejo archivo plugins.ini.
- **Propiedades del objeto:** el diálogo de propiedades está disponible como de costumbre (clic derecho sobre el nombre del objeto y se selecciona "Propiedades" del menú). Hay un nuevo acceso directo para obtener el mismo resultado: Ctrl-Alt-Enter.

1.5.4 Lenguaje de Modelado

El Lenguaje Unificado de Modelado o UML (del inglés *Unified Modeling Language*) es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema con gran cantidad de software. UML 2.0 proporciona una forma estándar de escribir los planos de un sistema, cubriendo tanto las cosas conceptuales, tales como procesos del negocio y funciones del sistema, como las cosas concretas, tales como las clases escritas en un lenguaje de programación específico, esquemas de bases de datos y componentes software reutilizables. (Orallo, 2011)

Las funciones de UML 2.0 son las siguientes:

Visualizar: permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.

Especificar: permite especificar cuáles son las características de un sistema antes de su construcción.
Construir: a partir de los modelos especificados se pueden construir los sistemas diseñados.
Documentar: los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

1.5.5 Herramienta CASE

Se puede definir a las herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. CASE es también definido como el conjunto de métodos, utilidades y técnicas que facilitan el mejoramiento del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases. (CASE, 2010)

Visual Paradigm 6.4

Visual Paradigm es una herramienta CASE: Ingeniería de Software Asistida por Computación. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación.

Características:

- Disponibilidad en múltiples plataformas (Windows, Linux).
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Licencia: gratuita y comercial.
- Fácil de instalar y actualizar.
- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- Diagramas de flujo de datos.
- Integración con Visio - Dibujo de diagramas UML con plantillas (stencils) de Microsoft Visio.
- Editor de figuras.

1.5.6 Marcos de trabajo

En el desarrollo de software, un marco de trabajo, es una estructura conceptual y tecnológica de soporte definido con módulos de software concretos, sirviendo de base para que otro proyecto de software pueda ser fácilmente organizado y desarrollado. Puede incluir soporte de programas, bibliotecas y otras herramientas, para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Symfony 2.0

Symfony es un marco de trabajo de PHP basado en la arquitectura MVC (Modelo-Vista-Controlador). Fue diseñado para optimizar el desarrollo de aplicaciones web, proporcionando herramientas para agilizar aplicaciones complejas y guiando al desarrollador a acostumbrarse al orden y buenas prácticas dentro del proyecto. (Pacheco, 2012)

Symfony cuenta con su propia consola de instrucciones o tareas, que permite al desarrollador ejecutar un grupo de comandos desde una terminal en la cual el marco de trabajo genera por si solo el código de la orden dada ganando en velocidad a la hora del desarrollo pues evita que el programador pierda tiempo escribiendo el mismo código varias veces. Otra de las funcionalidades más interesantes, es que contiene un submarco de trabajo para realizar formularios. Lo anteriormente mencionado permite crear una clase orientada a objetos que representa el formulario HTML.
Symfony 2 supone un cambio radical tanto en arquitectura interna como en filosofía de trabajo respecto a sus versiones anteriores. Ha sido ideado para exprimir al límite todas las nuevas características de PHP 5.3 y por eso es uno de los marcos de trabajo de PHP con mejor rendimiento. Cuenta con un gran número de bibliotecas, herramientas y helpers que ayudan a desarrollar una aplicación mucho más rápida que haciéndolo de la manera tradicional. (Pacheco, 2012)

ExtJS 3.4

ExtJS es la elección de los desarrolladores para la construcción de potentes aplicaciones web utilizando JavaScript y estándares web con facilidad. Tanto si es un desarrollador individual o un equipo de desarrollo, el modelo componente de ExtJS mantiene su código bien estructurado por lo que incluso las aplicaciones más grandes se puede mantener fácilmente. Lo mejor de todo, ExtJS ofrece una colección enciclopédica de widgets de interfaz de usuario con un elegante tema de partida. Estas son sólo algunas de las razones por las que ExtJS es la principal elección de los desarrolladores de todo el mundo. (ExtJS, 2012)

1.5.7 Servidor de aplicaciones

El concepto de servidor de aplicaciones está relacionado al de sistema distribuido. Un sistema distribuido, en oposición a un sistema monolítico, permite mejorar tres aspectos fundamentales en una aplicación: la alta disponibilidad, la escalabilidad y el mantenimiento. Proporciona servicios que soportan la ejecución y disponibilidad de las aplicaciones desplegadas, además de preservar la integridad de datos y códigos. (Servidor de Aplicaciones, 2012)

Apache web server

El proyecto Apache HTTP Server es un esfuerzo de desarrollo de software de colaboración destinadas a crear una aplicación robusta, de código fuente de calidad comercial, con muchas características, y libremente disponible de un HTTP (Web) del servidor. El proyecto es administrado conjuntamente por un grupo de voluntarios ubicados en todo el mundo, el uso de Internet y la Web para comunicarse, planear y desarrollar el servidor y su documentación correspondiente. Este proyecto forma parte de la Fundación de Software Apache. Además, cientos de usuarios han contribuido con ideas, código y documentación del proyecto. (Apache, 2011)

La versión que será utilizada en la solución del problema de la investigación es Apache 2.2.22, teniendo en cuenta las siguientes características:

- Es principalmente una liberación de seguridad y corrección de errores.
- Filtro inteligente.
- Almacenamiento en caché mejorado.
- AJP Proxy.
- Equilibrio de carga.
- Soporte de proxy de cierre normal.
- Soporte de archivos grandes.
- Rediseñado y autenticación / autorización.

1.6 Conclusiones parciales del capítulo

Durante el capítulo se realizó un estudio de los principales aspectos relacionados con la seguridad en los sistemas y vinculado a ello los principales mecanismos de autenticación y autorización. Con el objetivo de apoyar dichos mecanismos se hará uso de los CAPTCHAS durante el proceso de autenticación y durante el de autorización el modelo de control de acceso basado en roles y las listas de control de acceso. Para brindar la posibilidad de proveer otros tipos de usuarios se utilizará el protocolo LDAP. Para guiar el proceso de desarrollo del software se utilizará la metodología OpenUP y como herramienta para el modelado Visual Paradigm 6.4 empleando el lenguaje de modelado UML 2.0. Durante la implementación

de la solución se hará uso del IDE de desarrollo Netbeans 7.2, utilizando como lenguajes de programación PHP 5.3 y Java Script soportados por los marcos de trabajos Symfony 2.0 y ExtJS 3.4 respectivamente. Además para la definición de la base de datos se usará como gestor PostgreSQL 9.1, para la administración de la base de datos PgAdmin 1.14 y como servidor web Apache 2.2.22.

CAPÍTULO II "ANÁLISIS Y DISEÑO DEL MÓDULO"

Introducción

En el presente capítulo se describe el análisis y diseño del módulo de seguridad. Se realiza una representación visual de clases conceptuales del entorno real de los objetos del proyecto a través del modelo de dominio. Se hace un levantamiento de los requisitos funcionales y no funcionales necesarios para lograr que el sistema funcione correctamente. Se representan las relaciones existentes entre los actores y casos de uso del sistema mediante el diagrama de casos de uso. Se muestran los diagramas de clases del diseño, por medio de los cuales se representa la estructura estática del sistema. Se especifica la estructura física de la solución que se propone mediante el diagrama de despliegue.

2.1 Modelo de Dominio

Un modelo del dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. También se les denomina modelos conceptuales, ya que ayuda a comprender los conceptos clave de un negocio o un dominio de problema. A continuación se presenta el Modelo de Dominio del módulo de seguridad de la aplicación SIGDAT y se describen sus clases. (Craig Larman, 2012)

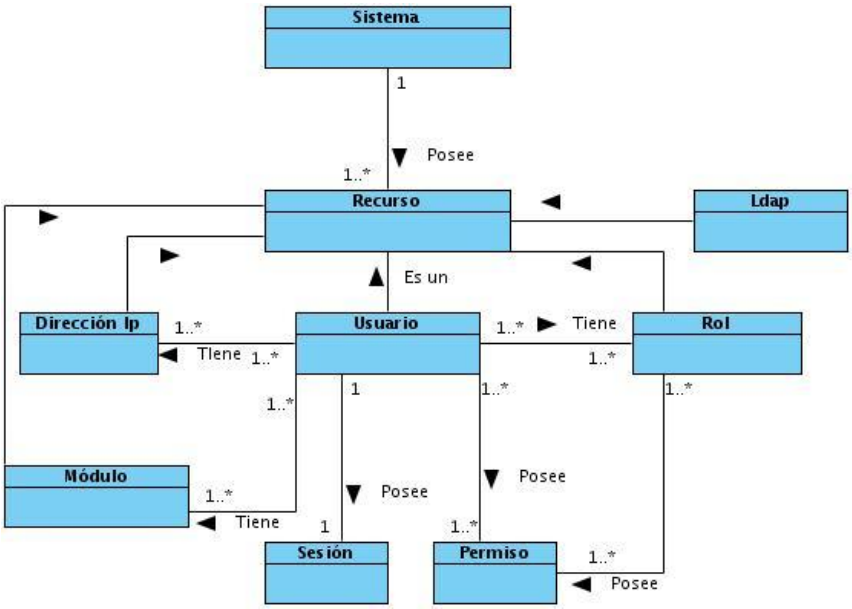


Figura 2 Modelo de Dominio

En el diagrama anterior el sistema representa a la aplicación SIGDAT la cual gestiona un conjunto de recursos representados por la entidad Recurso, de dicha entidad se derivan las clases usuario, dirección IP(Internet Protocol), Rol, Módulo y LDAP(Lightweight Directory Access Protocol) como recursos pertenecientes al módulo de seguridad.

Descripción de las clases del dominio:

Sistema: entidad que representa a la aplicación SIGDAT y está compuesta por los diferentes recursos del sistema.

Recurso: entidad que representa los distintos elementos gestionados dentro del módulo de seguridad.

Usuario: constituye una clase contenedora de información del negocio, forma parte de uno de los recursos gestionables dentro del módulo de seguridad, a la misma le son asignados roles, direcciones ip y módulos. Dicha entidad puede poseer uno o varios permisos sobre los diferentes recursos del sistema y una sesión.

Dirección Ip: representa una clase contenedora de información del negocio, este recurso se le asigna a los usuarios del sistema.

Rol: constituye una clase contenedora de información del negocio, este recurso posee una relación bidireccional con la entidad usuario ya que un usuario se le puede asignar uno o varios roles y viceversa. Además a un rol al igual que a un usuario se le pueden asignar permisos sobre los recursos del sistema.

Módulo: representa una clase contenedora de información del negocio, en este caso constituye los diferentes módulos existentes en la aplicación.

Permiso: constituye los diferentes niveles de acceso que tiene un usuario o rol sobre un recurso gestionado por el módulo.

Sesión: representa la conexión realizada por un usuario al sistema.

Ldap: representa una clase contenedora de información del negocio, específicamente constituye los diferentes recursos de tipo LDAP gestionados en el módulo de seguridad.

2.2 Requisitos del sistema

En la ingeniería del software, los requisitos se utilizan como datos de entrada en la etapa de diseño del producto y establecen qué debe hacer el sistema, pero no cómo hacerlo. Son una condición o capacidad que un usuario necesita para poder resolver un problema o lograr un objetivo. En fin son algo que el sistema debe hacer o una cualidad que el sistema debe poseer. (IAN Somerville, 2012)

2.2.1 Requisitos funcionales

Los requisitos funcionales definen el comportamiento interno de un software, son condiciones que el sistema ha de cumplir. Estos muestran las funcionalidades que deben satisfacerse para cumplir con las especificaciones de software. El módulo de seguridad para la aplicación SIGDAT debe cumplir con los requisitos funcionales que a continuación se describen: (IAN Somerville, 2012)

RF1. Listar usuarios

Descripción: permitirá al usuario conectado ver el listado de los usuarios almacenados en la base de datos sobre los cuales tiene permiso para listar.

Salida: listado de todos los usuarios sobre los que se tiene permiso para ver en la sesión.

RF2. Adicionar usuario
Descripción: se registra en el sistema un nuevo usuario.
Entrada: se registran los detalles del nuevo usuario (usuario, modo de autenticación LDAP o local, nombre y apellidos, correo, estado bloqueado o habilitado, contraseña).
Salida: nuevo usuario.

RF3. Editar usuario
Descripción: permitirá modificar cualquiera de los campos que componen a la entidad usuario.
Entrada: el o los datos de los campos que desean modificar del usuario (usuario, modo de autenticación LDAP o local, nombre y apellidos, correo, estado bloqueado o habilitado, contraseña).
Salida: usuario modificado.

RF4. Buscar usuario
Descripción: permitirá realizar una búsqueda de los usuarios en el sistema a partir de un criterio introducido por el usuario.
Entrada: criterio de búsqueda.
Salida: lista de usuarios que correspondan con el criterio de búsqueda.

RF5. Eliminar usuario
Descripción: se elimina del sistema el usuario seleccionado previamente.
Entrada: se selecciona de la lista de usuarios el que se desea eliminar.
Salida: la lista de usuarios actualizada.

RF6. Asignar rol(es) a usuario
Descripción: permitirá asignar al usuario seleccionado previamente uno o varios roles de los existentes en el sistema a los cuales el usuario conectado tiene permiso para listarlo.
Entrada: rol que se selecciona para asignarle al usuario.
Salida: usuario con nuevo rol asignado.

RF7. Revocar rol(es) a usuario
Descripción: permitirá invalidar al usuario seleccionado previamente uno o varios roles de los existentes en el sistema a los cuales el usuario conectado tiene permiso para listarlo.
Entrada: usuario y la selección del rol que se le desautorizará.
Salida: el usuario sin el rol que tenía asignado.

RF8. Listar roles
Descripción: permitirá mostrar los diferentes roles a los cuales el usuario conectado tiene permiso para listar.
Entrada: rol que se desea listar.
Salida: lista de roles.

RF9. Adicionar rol
Descripción: se registra en el sistema un nuevo rol.
Entrada: detalles del nuevo rol que se desea a adicionar (nombre, descripción).
Salida: nuevo rol.

RF10. Editar rol
Descripción: permitirá modificar cualquiera de los campos que componen a la entidad rol.
Entrada: el cambio en los campos que desean modificar del rol (nombre, descripción).
Salida: rol modificado

RF11. Buscar rol
Descripción: permitirá realizar una búsqueda de los roles en el sistema a partir de un criterio introducido por el usuario.
Entrada: criterio de búsqueda.
Salida: lista de roles que correspondan al criterio de búsqueda previamente introducido.

RF12. Eliminar rol
Descripción: se elimina del sistema el rol seleccionado previamente.
Entrada: se selecciona de la lista de roles el que se desea eliminar.
Salida: la lista de roles actualizada.

RF13. Asignar usuario(s) a un rol
Descripción: permitirá asignar al rol seleccionado previamente uno o varios usuarios de los existentes en el sistema a los cuales el usuario conectado tiene permiso para listarlo.
Entrada: usuario que se selecciona para asignarle al rol.
Salida: nuevos usuarios para el rol seleccionado.

RF14. Revocar usuario a rol(es)
Descripción: permitirá invalidar al rol seleccionado previamente uno o varios usuarios de los existentes en el sistema a los cuales el usuario conectado tiene permiso para listarlo.

Entrada: rol y la selección de los usuarios que se le desautorizarán.
Salida: el rol sin los usuarios que tenía asignados.

RF15. Listar permisos sobre recursos
Descripción: permitirá listar cada uno de los permisos que tienen el usuario o rol seleccionado previamente.
Entrada: seleccionar el usuario o rol del cual se quieren saber los permisos.
Salida: lista de permisos del usuario o rol anteriormente seleccionado.

RF16. Asignar permisos a usuario
Descripción: permitirá asignar a un usuario previamente seleccionado el permiso concedido sobre un determinado recurso.
Entrada: se selecciona el usuario a los que se le desean asignarle los permisos.
Salida: nuevos permisos para el usuario.

RF17. Asignar permisos a rol
Descripción: permitirá asignar a rol previamente seleccionado el permiso concedido sobre un determinado recurso.
Entrada: se selecciona el usuario a los que se le desean asignarle los permisos.
Salida: nuevos permisos para el rol.

RF18. Revocar permisos a usuario
Descripción: permitirá revocar a un usuario previamente seleccionado el permiso concedido sobre un determinado recurso.
Entrada: se selecciona usuario a los que se le desean revocar los permisos.
Salida: usuario sin los permisos que tenía asignados.

RF19. Revocar permisos a rol
Descripción: permitirá revocar a un rol previamente seleccionado el permiso concedido sobre un determinado recurso.
Entrada: se selecciona el rol a los que se le desean revocar los permisos.
Salida: roles sin los permisos que tenían asignados.

RF20. Filtrar permisos
Descripción: permitirá mostrar los permisos de un usuario o rol previamente seleccionado a partir del tipo de entidad que el usuario seleccione para filtrar.

Entrada: el rol o usuario del cual se quieren sus permisos sobre el tipo de entidad seleccionada para filtrar.
Salida: lista de permisos del usuario o rol sobre la entidad.

RF21. Listar LDAP
Descripción: permitirá mostrar los diferentes LDAPs a los cuales el usuario conectado tiene permiso para listar.
Salida: listado de todos los LDAP sobre los que el usuario tiene permiso.

RF22. Adicionar LDAP
Descripción: se registra en el sistema un nuevo LDAP.
Entrada: detalles del nuevo LDAP que se desea a adicionar (nombre, servidor, puerto, cuenta o DN principal, contraseña, DN base, usuario, nombre, apellidos, correo).
Salida: nuevo LDAP.

RF23. Editar LDAP
Descripción: permitirá modificar cualquiera de los campos que componen a la entidad LDAP.
Entrada: el cambio en los campos que desean modificar del LDAP.
Salida: LDAP modificado.

RF24. Buscar LDAP
Descripción: permitirá realizar una búsqueda de los LDAPs en el sistema a partir de un criterio introducido por el usuario.
Entrada: criterio de búsqueda.
Salida: lista de LDAP que correspondan al criterio de búsqueda previamente introducido.

RF25. Eliminar LDAP
Descripción: se elimina del sistema el LDAP seleccionado previamente.
Entrada: se selecciona de la lista de LDAP el que se desea eliminar.
Salida: la lista de LDAP actualizada

RF26. Verificar conexión
Descripción: permitirá comprobar si los datos introducidos por el usuario en el momento de insertar un nuevo LDAP son válidos luego de intentar establecer una conexión con el servidor LDAP.
Entrada: datos del nuevo LDAP.
Salida: mensaje de comprobación si se estableció bien o no la conexión.

RF27. Adicionar dirección IP a usuario
Descripción: se le asigna una nueva dirección IP al usuario seleccionado previamente.
Entrada: seleccionar el usuario al que se le desea adicionarle la dirección IP.
Salida: nueva dirección IP para el usuario seleccionado.

RF28. Editar dirección IP asignada
Descripción: permitirá modificar una dirección IP previamente asignada a un usuario.
Entrada: el cambio en los campos que desean modificar de la dirección IP.
Salida: dirección IP modificada.

RF29. Eliminar dirección IP asignada
Descripción: permitirá revocar la asignación de una dirección IP previamente asignada a un usuario.
Entrada: seleccionar el usuario al que se le desea revocar la dirección IP que tiene asignada.
Salida: usuario sin la dirección IP que se le revocó.

RF30. Listar direcciones IP asignadas a usuario
Descripción: permitirá mostrar las direcciones IP que tiene asignada el usuario previamente seleccionado.
Entrada: se selecciona el usuario del que se desean saber las direcciones IP.
Salida: lista de direcciones IP correspondientes al usuario seleccionado.

RF31. Asignar módulo a usuario
Descripción: permitirá asignar uno o varios módulos a un usuario seleccionado previamente.
Entrada: se selecciona el usuario al que se le desea asignar el módulo.
Salida: el usuario con los nuevos módulos asignados.

RF32. Revocar módulo a usuario
Descripción: permitirá invalidar al usuario seleccionado previamente uno o varios módulos de los que tiene asignado.
Entrada: usuario y la selección de los módulos a los que se le revocara el permiso.
Salida: el usuario sin los módulos que tenía asignados.

RF33. Listar usuarios conectados
Descripción: permitirá mostrar un listado de los usuarios que se encuentran conectados al sistema.
Salida: se muestra la lista de usuarios conectados.

RF34. Cerrar sesión de un usuario
Descripción: permitirá invalidar la sesión activa del usuario previamente seleccionado.
Entrada: id del usuario al que se le desea cerrar la sesión.
Salida: se actualiza en la base de datos el campo logged del usuario.

RF35. Cerrar sesión por inactividad
Descripción: permitirá invalidar la sesión activa del usuario una vez que el mismo deje de hacer peticiones al servidor por un tiempo mayor al configurado en el sistema o porque su cuenta se encuentre desactivada.
Entrada: id del usuario que lleva más de diez minutos sin hacer peticiones al servidor.
Salida: se actualiza en la base de datos el campo logged del usuario.

RF36. Autenticar usuario
Descripción: permitirá a los usuarios realizar el proceso de autenticación en la aplicación ofreciendo a la misma los datos correspondientes (usuario y contraseña).
Entrada: contraseña y usuario de la persona que desea entrar al sistema.
Salida: se actualiza en la base de datos el campo logged del usuario.

RF37. Generar CAPTCHA
Descripción: permitirá generar un CAPTCHA luego que un usuario o sistema se equivoque tres veces al proveer los datos necesarios para el acceso.
Entrada: dos intentos de escribir usuario o contraseña incorrecta por el usuario.
Salida: imagen para verificar si las peticiones al sistema son realizadas por un humano o por una máquina.

RF38. Cerrar sesión
Descripción: permitirá al usuario conectado salir del sistema.
Salida: se actualiza en la base de datos el campo logged del usuario.

2.2.2 Requisitos no funcionales

Los requisitos no funcionales son propiedades que hacen al producto atractivo, usable, rápido o confiable. Se refieren a todos los requisitos que ni describen información a guardar, ni funciones a realizar. Además se conocen como un conjunto de características de calidad, que es necesario tener en cuenta al diseñar e implementar el software. (IAN Somerville, 2012)

Requisitos de Usabilidad:

RNF1. El módulo deberá presentar facilidades al usuario para el manejo de la información, se pretende una vista descriptiva, sencilla y fácil de usar para la realización de todas las operaciones, con el objetivo de propiciar un buen entendimiento a los usuarios finales.

Requisitos de Seguridad

RNF2. La información manejada por el sistema deberá estar protegida de acceso no autorizado y encontrarse disponible sólo para los usuarios que pertenezcan al sistema.

RNF3. Requisitos de Software

El servidor donde se instalará la aplicación debe cumplir con los siguientes requisitos:

- Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 10.10 o superior, Debian 4 GNU/Linux o superior.
- Paquetes: apache2, php5, libapache2-mod-php5, php5-cli, php5-pgsql, php5-sqlite, php5-gd, php5-ldap, php-apc.
- Navegador: Mozilla Firefox versión 4.0 ó superior.

El servidor donde se instalará la base de datos debe cumplir con los siguientes requisitos:

- Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 10.10 o superior, Debian 4 GNU/Linux o superior.
- PostgreSQL versión 9.0 o superior.
- PgAdmin III o algún administrador para PostgreSQL.
- PostgreSQL debe estar correctamente configurado para aceptar conexiones vía TCP/IP utilizando el método de autenticación por md5.

RNF4. Requisitos de Hardware

Las PC clientes debe cumplir con los siguientes requisitos de hardware:

- Ordenador Pentium IV o superior, con 1.7 GHz de velocidad de microprocesador.
- Memoria RAM mínimo 256 MB.
- Un mínimo de 1GB de espacio en disco

Las PC Servidor debe cumplir con los siguientes requisitos de hardware:

- Ordenador Pentium IV o superior, con 1.7 GHz de velocidad de microprocesador.
- Memoria RAM mínimo 1Gb.
- Disco Duro con 10Gb de capacidad para instalar el sistema.

RNF5. Tiempo máximo de respuesta para la obtención de los recursos

Los tiempos de respuestas deben ser generalmente rápidos al igual que la velocidad de procesamiento de la información.

Requisitos de Restricciones de diseño e implementación.

RNF6. Lenguaje y marco de trabajo para el desarrollo del sistema del lado del servidor.

El sistema deberá ser implementado en el lenguaje de programación PHP versión 5.3.10 y hacer uso del marco de trabajo Symfony2.

RNF7. Lenguaje y marco de trabajo para el desarrollo del sistema del lado del cliente.

El sistema deberá ser implementado en el lenguaje de programación JavaScript y hacer uso del marco de trabajo ExtJS en su versión 3.4.

Requisitos de Interfaz

RNF8. Interfaces de usuario

Las interfaces de usuario deberán diseñarse a modo de aplicaciones RIA (Rich Internet Application) permitiendo a los usuarios contar con aplicaciones web con una experiencia de usuario similar a la de las aplicaciones de escritorio.

2.3 Diagrama de casos de uso del sistema

Los diagramas de casos de uso documentan el comportamiento de un sistema desde el punto de vista del usuario. Por lo tanto los mismos determinan los requisitos funcionales del sistema, es decir, representan las funciones que un sistema puede ejecutar.

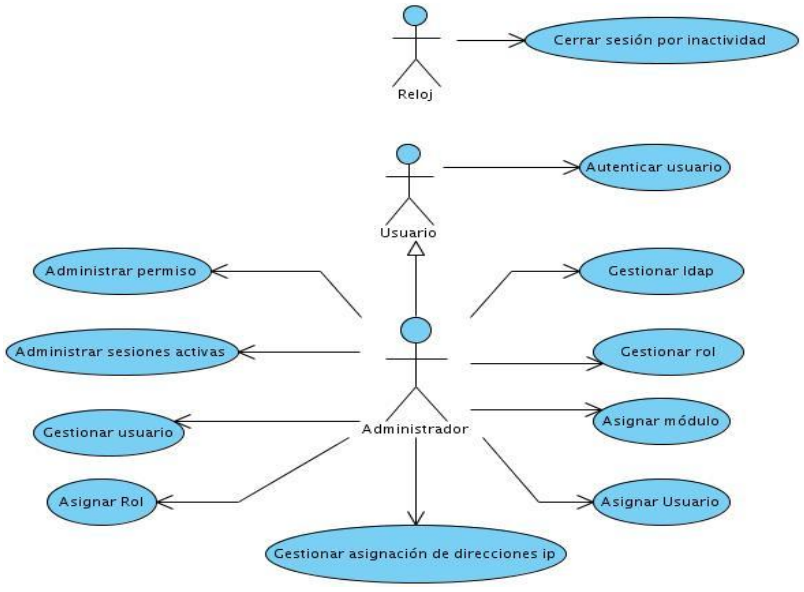


Figura 3 Diagrama de caso de usos del sistema

Descripción de los casos de uso:

Autenticar usuario: permite realizar el proceso de autenticación de los usuarios del sistema además de posibilitarles desconectarse del mismo, en este caso de uso se agrupa además el requisito generar CAPTCHA ya que luego de tres intentos fallidos de autenticación el usuario que intenta acceder al sistema tendrá que completar el CAPTCHA generado por la aplicación.

Administrar permiso: permite listar los permisos de un usuario o rol sobre los recursos gestionados por el sistema, además de brindar la posibilidad al administrador de la aplicación de filtrar, asignar y revocar permisos sobre los usuarios y roles.

Gestionar Ldap: permite adicionar, editar, eliminar, buscar, listar un LDAP además de verificar si se puede establecer la conexión con un servidor de este tipo a partir de los datos suministrados por el usuario.

Administrar sesiones activas: ofrece la posibilidad al administrador de listar los usuarios conectados al sistema, además de poder invalidar su sesión.

Gestionar rol: permite adicionar, editar, eliminar, buscar y listar un rol.

Gestionar usuario: permite adicionar, editar, eliminar, buscar y listar un usuario.

Asignar módulo: permite asignar y revocar permisos a un usuario sobre uno o varios módulos.

Asignar rol: permite asignarle uno o varios roles a un usuario seleccionado.

Asignar Usuario: permite asignarle uno o varios usuarios a un rol seleccionado.

Gestionar asignación de direcciones ip: permite al administrador del sistema adicionar dirección IP a usuario previamente seleccionado, eliminar dirección IP asignada, editar dirección IP asignada y listar direcciones IP asignadas al usuario seleccionado.

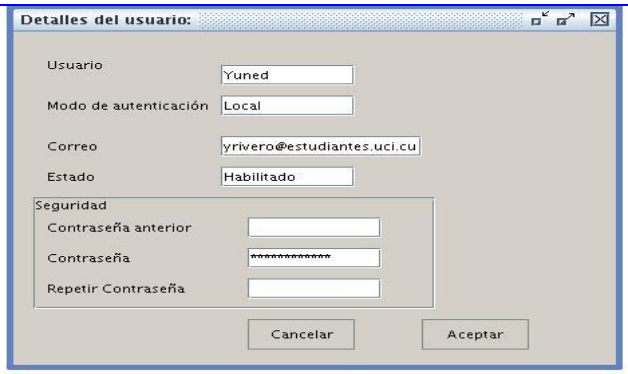
Cerrar sesión por inactividad: permite invalidar la sesión de un usuario luego de que trascurra un tiempo definido sin que el mismo realice alguna petición al sistema o su cuenta sea bloqueada.

2.3.1 Descripción del Caso de Uso arquitectónicamente significativo del sistema.

CU: Gestionar usuario.

Caso de Uso:	Gestionar usuario
Actores:	Administrador
Resumen:	El caso de uso se inicia cuando el actor selecciona una de las opciones asociadas al caso de uso (crear, modificar, eliminar y buscar un usuario). El caso de uso termina una vez finalizada alguna de las opciones mencionadas anteriormente.
Precondiciones	El sistema debe estar instalado y ejecutándose correctamente.

	El actor debe estar autenticado con los permisos necesarios.
Referencias	RF1, RF2, RF3, RF4, RF5
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la pestaña correspondiente al listado de usuarios.	2. El sistema muestra el listado de usuarios a los que tiene permiso el administrador y activa las opciones correspondientes a la gestión de usuarios. El administrador puede: -Adicionar usuario, ver sección **“Adicionar usuario local”.** -Adicionar usuario, ver sección **“Adicionar usuario LDAP”.** -Editar usuario, ver sección **“Editar usuario”.** -Buscar usuario, ver sección “**Buscar usuario”.** -Eliminar usuario, ver sección **“eliminar usuario”.**
Sección “Adicionar usuario local”	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción adicionar usuario.	2.El sistema muestra la interfaz para crear un usuario, solicitando los siguientes datos: (Usuario, Modo de autenticación, Nombre y apellidos, Correo, Estado, Contraseña, Repetir contraseña)
3**.** El administrador introduce los datos correspondientes.	4. El sistema valida los datos introducidos. (El campo usuario debe de comenzar con letras y a continuación letras o dígitos con longitud mínima 4 y máxima 15 caracteres, el campo nombre y apellidos debe de tener el formato Nombre Primero Segundo solo se aceptan letras al comenzar cada palabra en mayúscula, el correo debe de tener el formato usuario@dominio, la contraseña debe ser una combinación de letras mayúsculas minúsculas, números y caracteres especiales con longitud mínima de 8 caracteres),luego de la validación activa el botón aceptar.
5.El administrador selecciona la opción aceptar	6. El sistema adiciona el usuario, recarga el listado de usuarios y muestra el mensaje usuario agregado correctamente, terminando así el caso de uso.
Prototipo de Interfaz	



Flujo Alterno al paso 4 “ Validación”

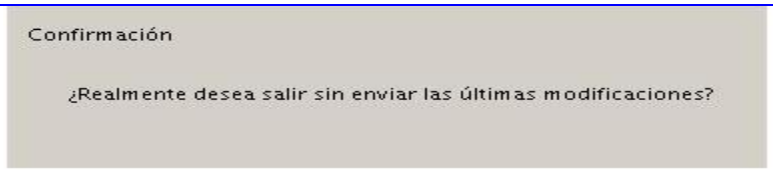
Acción del Actor	Respuesta del Sistema
	4 a. El sistema muestra los campos con errores de validación subrayados en rojo.
4b El administrador se posiciona sobre el campo que tiene errores en los datos.	4c El sistema muestra el mensaje de error de acuerdo al campo donde se posicione el usuario.

Prototipo de Interfaz

Flujo Alterno al paso 1 ,5 “ Cancelar”

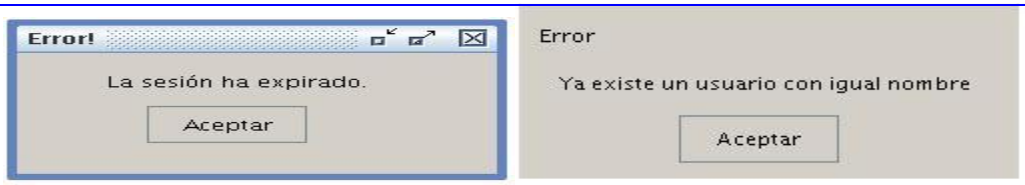
Acción del Actor	Respuesta del Sistema
1a. El administrador decide no adicionar un usuario y cancela la operación.	1b. En caso de que el administrador no haya entrado ningún dato, el sistema cancela la operación y cierra la interfaz correspondiente a la adición de usuario. En caso contrario pregunta ¿Realmente desea salir sin enviar las últimas modificaciones? Si el administrador selecciona si el sistema cancela la operación y cierra la interfaz correspondiente a la adición de usuario, si selecciona no el sistema mantiene la interfaz abierta.

Prototipo de Interfaz



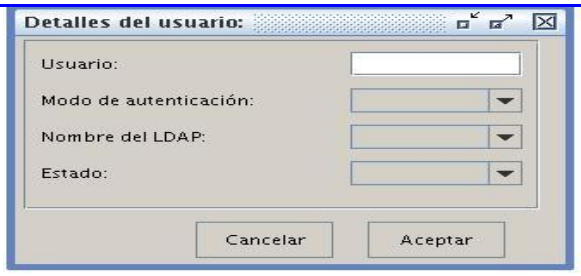
Flujo Alterno al paso 6 " Error al adicionar usuario"	
Acción del Actor	**Respuesta del Sistema**
	1b. En caso de que la sesión expire el sistema muestra el mensaje su sesión ha expirado. Si existe algún otro usuario con el mismo nombre del que se intenta adicionar el sistema muestra el mensaje: **"Ya existe un usuario con igual nombre¨.**

Prototipo de Interfaz



Sección "Adicionar usuario LDAP"	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción adicionar usuario LDAP.	2. El sistema muestra la interfaz para crear un usuario, solicitando los siguientes datos: (Nombre, Servidor, Puerto, Cuenta o DNS principal, Contraseña, DNS base, Usuario, Nombre, Apellidos, Correo electrónico).
3. El administrador introduce los datos correspondientes.	4. El sistema valida los datos introducidos.
5. El administrador selecciona la opción aceptar.	6. El sistema adiciona el usuario LDAP, recarga el listado de usuarios y muestra el mensaje usuario agregado correctamente, terminando así el caso de uso.

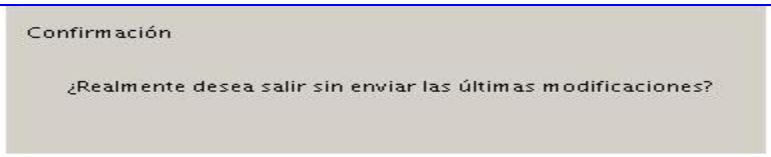
Prototipo de Interfaz



Flujo Alterno al paso 1 ,5 " Cancelar"	
Acción del Actor	**Respuesta del Sistema**

1a. El administrador decide no adicionar un usuario y cancela la operación.	1b. En caso de que el administrador no haya entrado ningún dato, el sistema cancela la operación y cierra la interfaz correspondiente a la adición de usuario. En caso contrario pregunta ¿Realmente desea salir sin enviar las últimas modificaciones? Si el administrador selecciona si el sistema cancela la operación y cierra la interfaz correspondiente a la adición de usuario, si selecciona no el sistema mantiene la interfaz abierta.

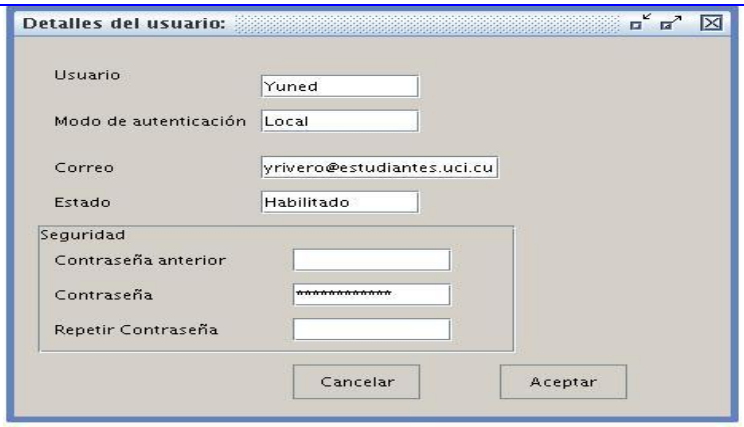
Prototipo de Interfaz



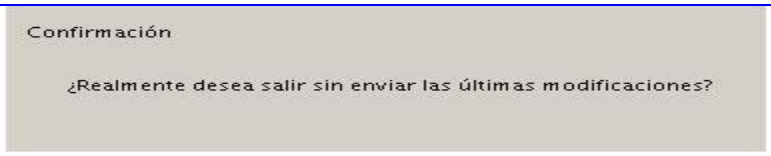
Sección "Editar usuario"

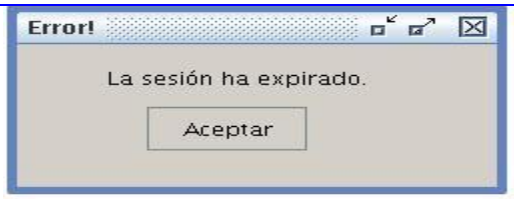
Acción del Actor	Respuesta del Sistema
1. El administrador selecciona el usuario que desea editar del listado de usuarios a los que tiene permisos para ver y selecciona la opción editar.	2. El sistema muestra una interfaz con los datos del usuario seleccionado previamente.
3.El administrador realiza los cambios correspondientes en los campos de la interfaz mostrada previamente por el sistema	4. El sistema valida los campos editados, luego de la validación activa el botón aceptar.
5. El administrador selecciona la opción aceptar	6. El sistema actualiza el usuario seleccionado, recarga el listado para mostrar la actualización realizada y muestra el mensaje usuario actualizado correctamente, terminando así el caso de uso.

Prototipo de Interfaz



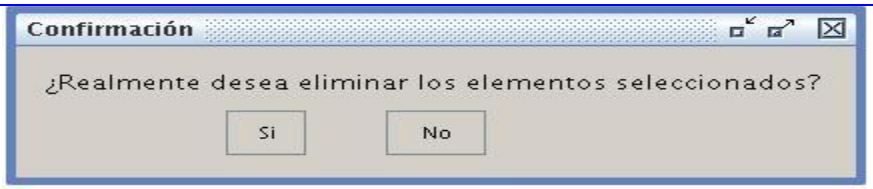
Flujo Alterno al paso 4 " Validación"

Acción del Actor	Respuesta del Sistema
	4 a. El sistema muestra los campos con errores de validación subrayados en rojo.
4b El administrador se posiciona sobre el campo que tiene errores en los datos.	4c El sistema muestra el mensaje de error de acuerdo al campo donde se posicione el usuario.
Prototipo de Interfaz	
	
Flujo Alterno al paso 1 ,5 " Cancelar"	
Acción del Actor	**Respuesta del Sistema**
1a. El administrador decide no editar el usuario y cancela la operación.	1b. En caso de que el administrador no haya cambiado ningún dato, el sistema cancela la operación y cierra la interfaz correspondiente a la edición de usuario. En caso contrario pregunta ¿Realmente desea salir sin enviar las últimas modificaciones? Si el administrador selecciona si el sistema cancela la operación y cierra la interfaz correspondiente a la edición de usuario, si selecciona no el sistema mantiene la interfaz abierta.
Prototipo de Interfaz	
	
Flujo Alterno al paso 6 " Error al actualizar usuario"	
Acción del Actor	**Respuesta del Sistema**
	6a. En caso de que la sesión expire el sistema muestra el mensaje su sesión ha expirado. Si existe algún otro usuario con el mismo nombre del que se intenta actualizar, el sistema muestra el mensaje de error ¨Ya existe un usuario con igual nombre¨, terminando así el caso de uso.
Prototipo de Interfaz	



Sección "Eliminar usuario"	
Acción del Actor	**Respuesta del Sistema**
1-El administrador selecciona el usuario o usuarios que desea eliminar del listado de usuario a los que tiene permisos para ver y selecciona la opción eliminar	2-El sistema muestra una interfaz preguntando ¿Realmente desea eliminar los elementos seleccionados?
3-El administrador selecciona la opción si del cuadro de dialogo mostrado por el sistema.	4-El sistema elimina el o los registros seleccionados, recarga el listado de usuarios y muestra el mensaje la operación se realizó correctamente, terminando así el caso de uso.

Prototipo de Interfaz

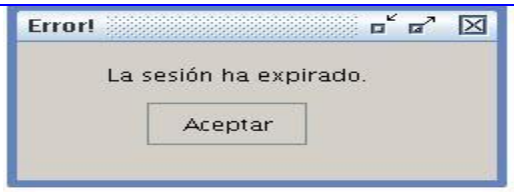


Flujo Alterno al paso 3 " Selección de la opción no"	
Acción del Actor	**Respuesta del Sistema**
3a. El administrador decide no realizar la operación de eliminación y selecciona la opción no.	3 b. El sistema cierra la interfaz mostrada en el paso 2, terminando así el caso de uso.

Flujo Alterno al paso 4 " Error al eliminar"	
Acción del Actor	**Respuesta del Sistema**
	4a. En caso de que la sesión expire el sistema muestra el mensaje "su sesión ha expirado", terminando así el caso de uso.

Prototipo de Interfaz

Sección "Buscar usuario"

Acción del Actor	Respuesta del Sistema
1. El administrador selecciona la opción buscar	2. El sistema muestra una interfaz para introducir el criterio de búsqueda.
3. El administrador escribe un criterio de búsqueda y selecciona la opción buscar	4. El sistema muestra el o los usuarios que se relacionan con el criterio introducido por el administrador.
Prototipo de Interfaz	
	
Flujo Alterno al paso 1,3 " Cancelar"	
Acción del Actor	**Respuesta del Sistema**
1a. El administrador decide no realizar la operación de búsquedas y selecciona la opción buscar.	1b. El sistema cierra la interfaz de búsqueda y recarga el listado de usuarios.
Flujo Alterno al paso 4 " Error al buscar"	
Acción del Actor	**Respuesta del Sistema**
	4a. En caso de que la sesión expire el sistema muestra el mensaje "su sesión ha expirado.
Prototipo de Interfaz	
	
Pos-condiciones	El sistema queda con un nuevo usuario creado, modificado o sin uno usuario de los existentes.

2.4 Modelo de Diseño

El modelo de diseño describe la realización de casos de uso, y sirve como una abstracción del modelo de aplicación y su código fuente. Se utiliza como parte esencial para las actividades en ejecución y prueba, que se basa en el análisis y los requisitos de la arquitectura del sistema. Representa los componentes de aplicación y determina su colocación adecuada y el uso dentro de la arquitectura en general del sistema. (Rodríguez, 2011)

2.4.1 Diagrama de Paquetes

La organización física de los elementos que conforman el sistema se realizará teniendo en cuenta los principios establecidos por Symfony2 que estructuran el proyecto en paquetes o bundles. En la figura # 4 se muestra detalladamente la estructura del bundle del módulo de seguridad.

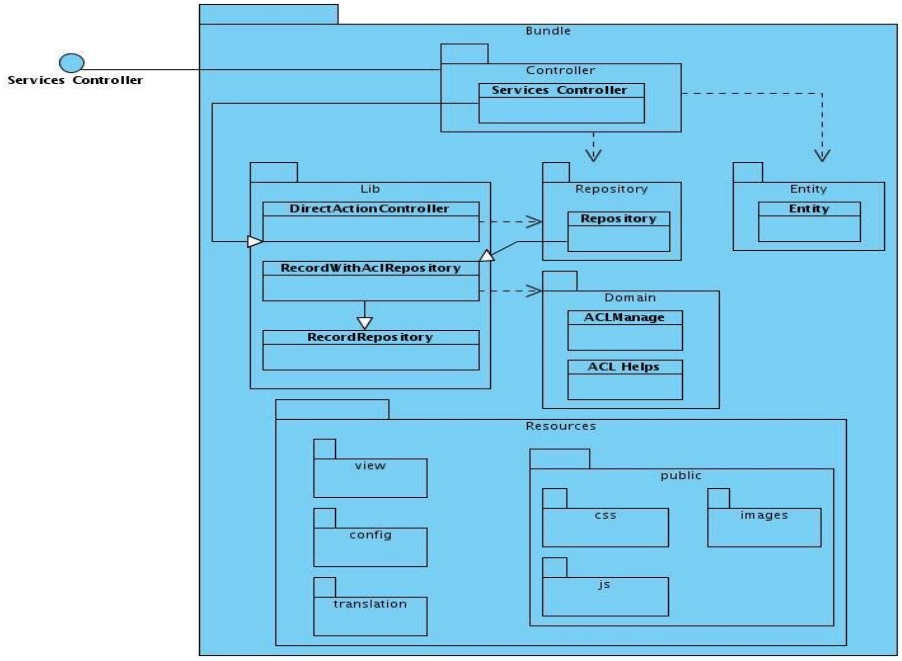


Figura 4 Diagrama de Paquetes

El bundle de la aplicación estará compuesto por una carpeta denominada Controller donde se almacenarán todos los controladores del sistema. Estos harán uso de los repositorios y entidades correspondientes en dependencia de las acciones que realicen. Los repositorios estarán contenidos dentro de la carpeta Repository, estos archivos se encargarán de organizar las sentencias del lenguaje de consulta Doctrine o DQL (del inglés *Doctrine Query Language*) de una entidad en cuestión y heredarán de la clase RecordWithAclRepository posibilitando garantizar la inclusión de permisos en los diferentes métodos de búsquedas que se implementen en dichos repositorios. Las entidades se almacenarán dentro de la carpeta Entity y se encargarán de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes del tipo de gestor de bases de datos. La carpeta Lib contiene las clases **DirectActionController**, **RecordRepository RecordWithAclRepository y LdapConnection**. La clase DirectActionControlller será una implementación genérica de una clase controladora que contiene las funcionalidades necesarias para listar y eliminar en dependencia de la entidad que sea manejada por el controlador. Dicha clase permitirá que un controlador herede sus funcionalidades e implemente las acciones más específicas como crear y actualizar. La clase RecordRepository poseerá las funcionalidades

para el manejo de las transacciones con la base de datos además de contener distintos métodos para realizar búsquedas. RecordWithAclRepository heredará de dicha clase e implementará los métodos de búsquedas teniendo en cuenta los diferentes permisos como propietario, permiso de edición, eliminación o de solo lectura. Por su parte la clase LdapConnection facilitará la verificación de conexión y búsqueda sobre un determinado LDAP. Para facilitar un mejor manejo de los permisos en las búsquedas se definirán las clases AclManage y AclHelper que estarán contenidas dentro de la carpeta Domain. La clase AclManage permitirá adicionar permisos, eliminar una ACL para un objeto de dominio determinado, revocar uno o varios permisos, obtener los permisos de un usuario o rol sobre un objeto de dominio y obtener todas las clases ACL registradas. Por su parte AclHelper ofrecerá la posibilidad de clonar y aplicar los permisos sobre una determinada consulta a realizar.

Otra de las carpetas dentro del bundle es Resources la cual contendrá las carpetas view, config, translation y public. Dentro de la carpeta view se encontrarán las plantillas Twig, encargadas de mostrar los diferentes componentes del sistema. En la carpeta public se ubicarán diferentes recursos como hojas de estilos, imágenes y archivos JavaScript, cada uno en su carpeta correspondiente. Para configurar los diferentes servicios que se serán utilizados en el sistema se contará con el archivo services.yml el cual se ubicará dentro de la carpeta config. Para el manejo de la traducción de las diferentes excepciones se utilizará el archivo messages.es.yml almacenado en la carpeta translation.

Es importante destacar que independientemente de la riqueza de Symfony para la elaboración de la presentación, estas posibilidades no se utilizarán dado que la presentación se implementará utilizando Ext.JS 3.4, para ello se hace necesario definir una organización (Fig.5) para los componentes de presentación.

Las acciones a realizar en el sistema se encontrarán almacenadas en la carpeta actions. Los stores y record encargados de realizar las diferentes peticiones a los controladores en la parte del servidor se encontrarán en la carpeta model, las vistas del sistema se almacenarán en la carpeta view cada una en su carpeta correspondiente en dependencia de la función que cumpla dentro del sistema por ejemplo las correspondientes a la administración de permisos, asignación de módulos y direcciones IP a los usuarios se almacenarán en grant, las involucradas en la adición, edición, eliminación, búsqueda de usuarios y asignación de roles estarán en user, las encargadas de las gestión de roles y asignación de usuarios a dichos roles se encontrarán en role y las relacionadas con la gestión de LDAP en la carpeta del mismo nombre.

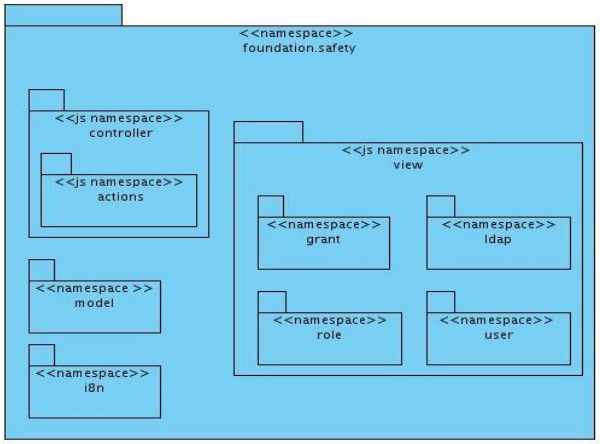


Figura 5 Diagrama de Paquetes del lado del cliente

2.4.2 Diagrama de clases del diseño

Los diagramas de clases del diseño son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargarán del funcionamiento y la relación entre uno y otro. Describen gráficamente las especificaciones de las clases del software y de las interfaces en una aplicación. En su diseño contiene como información (clases, métodos, información sobre los tipos de los atributos, dependencias). (Larman, 2012)

En la figura#6 se muestra la representación del diagrama de clases del diseño genérico que servirá de soporte para el posterior modelo de diseño de la solución, el cual incluye el marco reutilizable de Lycan[5], la especificación para un CRUD genérico y la conexión del mismo con la parte del servidor.

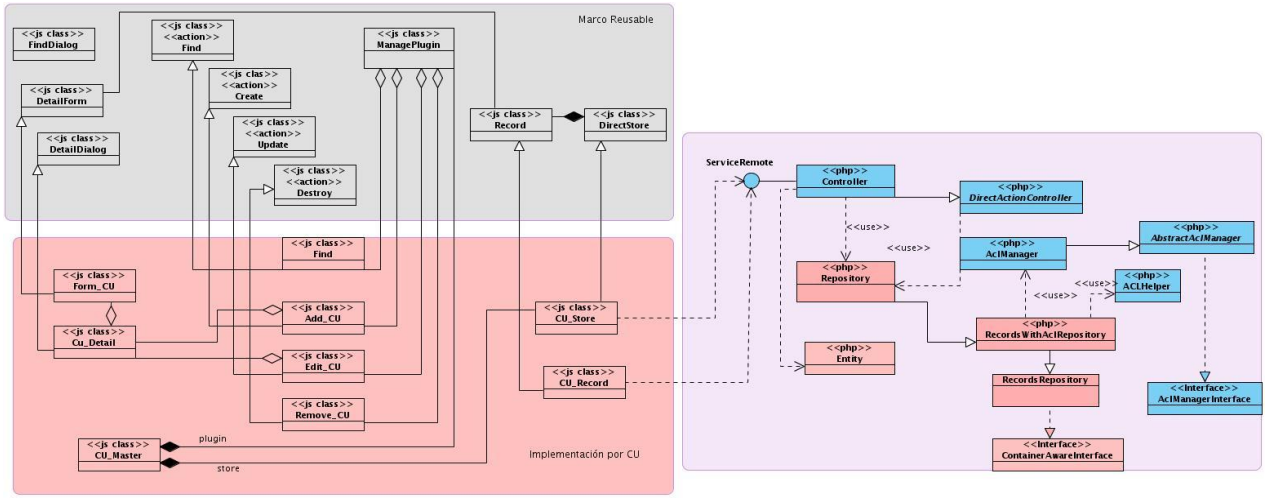


Figura 6 Diagrama de clases del diseño

[5] Marco reutilizable que está compuesto por un conjunto de clases entre las que se destacan Create.js, Destroy.js, Record.js, Update.js entre otras, dichas clases permiten realizar un CRUD completo de manera más sencilla.

A continuación se realiza un breve descripción de las clases del de diagrama de clases del diseño genérico que se representado en la imagen anterior. Para mayor información de las clases y las relaciones representadas en el mismo dirigirse al artefacto Modelo de Diseño del módulo de seguridad de SIGDAT.

Nombre: Controller
Descripción: clase de PHP que es creada y ejecutada por Symfony como realización del CU, los métodos que se implementan permiten generar el API y direccionar las llamadas de ExtDirect. El producto final del controlador es: un objeto de respuesta Symfony2.

Nombre: DirectActionController
Descripción: ofrece las funcionalidades necesarias para listar y eliminar los registros existentes de la entidad definida en el atributo defaultEntityname el cual obtendrá valor en el controlador que extienda de dicha clase. Además hereda de la clase Controller de Symfony la cual le proporciona acceso a los diferentes servicios disponibles para los controladores.

Nombre: AclManager
Descripción: permite adicionar permisos, eliminar una ACL para un objeto de dominio determinado, revocar uno o varios permisos, obtener los permisos de un usuario o rol sobre un objeto de dominio y obtener todas las clases ACL registradas.

Nombre: ACLHelper
Descripción: permite clonar y aplicar los permisos sobre una determinada consulta a realizar.

Nombre: Entity
Descripción: es la clase de PHP ejecutada por Symfony como realización del CU, los métodos y los atributos que se implementan son generados por el ORM Doctrine para el acceso a datos. Se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes del tipo de gestor de bases de datos.

Nombre: RecordRepository
Descripción: posee las funcionalidades para el manejo de las transacciones con la base de datos además de contener distintos métodos para realizar búsquedas.

Nombre: Clase RecordWitchAclRepository
Descripción: RecordWithAclRepository hereda de la clase RecordRepository y redefine los métodos de búsquedas teniendo en cuenta los diferentes permisos como propietario, permiso de edición, eliminación o de

solo lectura.

Nombre: Repositoy
Descripción: es la clase de PHP ejecutada por Symfony como realización del CU definida como CU_Repository.php, donde los métodos que se implementan por un criterio determinado realizan la búsqueda de la entidad en la Base de Datos. Cada entidad (entity) tiene asociado un repositorio (repository).

Nombre: DetailForm.js
Descripción: es una clase en JavaScript correspondiente a una interfaz visual que se encarga de mostrar los detalles o campos correspondientes al objeto que se gestione. Implementa un conjunto de funciones que serán reutilizadas en la realización del caso de uso que se implemente, logrando especificar las funcionalidades necesarias para la gestión de los campos descriptivos del objeto en cuestión.

Nombre: DetailDialog.js
Descripción: es una clase en JavaScript correspondiente a una interfaz visual que se encarga de mostrar el formulario o DetailForm correspondientes al objeto que se gestione. Implementa un conjunto de funciones que serán reutilizadas en la realización del caso de uso que se implemente, logrando especificar las funcionalidades necesarias para la integración de dicho diálogo con el formulario correspondiente.

Nombre: ManagePlugin.js
Descripción: La clase ManagePlugin es una clase en JavaScript que permite agregar comportamientos específicos a la clase maestro (CU_Master.js) con la que interactúe; tales como la incorporación de un menú contextual y la barra de tarea inferior de las acciones que en ese contexto se pueden acometer (Adicionar, Actualizar, Buscar, Eliminar, entre otras). Implementa el comportamiento que serán reutilizados en la realización del caso de uso que se implemente, permitiendo eliminar el objeto que se gestiona en el formulario correspondiente.

Nombre: Record.js
Descripción: La clase Record es una clase en JavaScript que representa los atributos propios del objeto que se gestiona. Un record (Record.js) se corresponde a un formulario (DetailForm.js), donde el formulario gestiona todos los atributos o elementos que describen al objeto que se gestiona y el record es la instancia de dicho objeto.

Nombre: DirectStore.js

Descripción: La clase DirectStore es una clase en JavaScript que representa el listado de objetos o record que se gestionan, donde la clase CU_Store.js hereda de la misma para la realización del CU que se implementa. Un store (CU_Strore) está compuesto por varios objetos o record (CU_Record), donde se adjuntan y se listan en la clase maestro para ser mostrados visualmente, permitiendo el manejo de los comportamientos o acciones que se implementan en la realización del CU en cuestión a través de la clase ManagePlugin.js.

Nombre: CU_Master.js
Descripción: El Maestro (Master.js) es una clase en JavaScript correspondiente a una interfaz visual (siguiendo el patrón para el desarrollo de interfaces 'Maestro-Detalle') que se encarga de mostrar en una tabla todos los elementos almacenados en la base de datos del objeto o entidad que se esté gestionando. Implementa un conjunto de funciones utilizadas en la realización del Caso de Uso que se implemente, está relacionada con la clase Store de JavaScript puesto que la información que visualiza está asociada a dicho Store.

2.4.3 Diagrama de clases del diseño del caso de uso Gestionar usuario.

El siguiente diagrama de clase del diseño (ver figura 7) contiene las principales clases con sus respectivos métodos y atributos para el caso de uso gestionar usuario. En este diagrama las clases representadas en color amarillo conforman la parte de la vista, incluyendo componentes como ventana, formulario, grillas de datos pertenecientes al marco de trabajo ExtJS, además se hace uso de otros componentes del marco reutilizable de Lycan como por ejemplo las acciones, el ManagePlugin, DirectStore, DetailDialog, DetailForm y Record.

Las acciones implementan las funcionalidades que permite el caso de uso, en este caso buscar, editar, adicionar y eliminar usuario, cada una representada a través de las clases JavaScript Find, EditUser, AddUser y RemoveUser respectivamente. Las acciones de edición y adición controlan una vista representada por la clase UserDetail la cual contiene una clase denominada FormUser que permite insertar o editar un objeto de tipo usuario que puede ser local o LDAP. Luego que se realiza la edición o adición, la clase UserDetail entrega a la acción correspondiente ya sea EditUser o AddUser el record asociado a los datos capturados en el formulario, la acción utiliza la capacidad del record para comunicarse con el servidor y ejecutar una petición de adición o edición según la acción que se realice en ese momento.

La comunicación del record con el controlador en la parte del servidor se logra mediante la utilización de DirectBundle, el cual es una implementación de ExtDirect específicamente para Symfony 2.x permitiendo

la comunicación remota entre el lado del cliente y la lógica del negocio. Ext.direct es una arquitectura de comunicación remota de JavaScript que a través de las especificaciones que implementa le permite reflejar los métodos del lado del servidor en el lado del cliente. Estas especificaciones utilizan como mecanismo de comunicación RPC en el marco ExtJS. Los servicios a consumir en el lado del cliente serán definidos en un API perteneciente a las clases UserRecord y UsersStore, en el API mencionado anteriormente se especificarán cada uno de las funciones a llamar en el controlador del lado del servidor dependiendo de las acciones que se ejecuten en el lado del cliente. Otro de los elementos fundamentales en la vista es la clase UserMaster la cual se encarga de mostrar los usuarios registrados en la base de datos a los que se tiene permiso de lectura. Esta clase contiene un ManagePlugin y un UsersStore que es el encargado de representar la lista de usuario.

Las clases ManageUserController, ManageLdapController y DirectActionController representan los controladores, ellos son los encargados de dar respuesta a las peticiones que se hacen desde el lado del cliente. Para ofrecer respuesta, hacen uso de diferentes clases como User, Ldap, UserRepository y LdapRepository. Las clases User y Ldap son entidades que forman parte del modelo, se encargan de la abstracción de la lógica relacionada con los datos de los usuarios y los LDAP. Las clases UserRepository y LdapRepository tienen la responsabilidad de facilitar las búsquedas sobre la base de datos así como el manejo de la persistencia de las entidades, para realizar las operaciones anteriormente mencionadas los repositorios hacen uso de las clases AclManager y ACLHelper las cuales facilitan el manejo de las listas de control de acceso para el control de los permisos sobre las entidades.

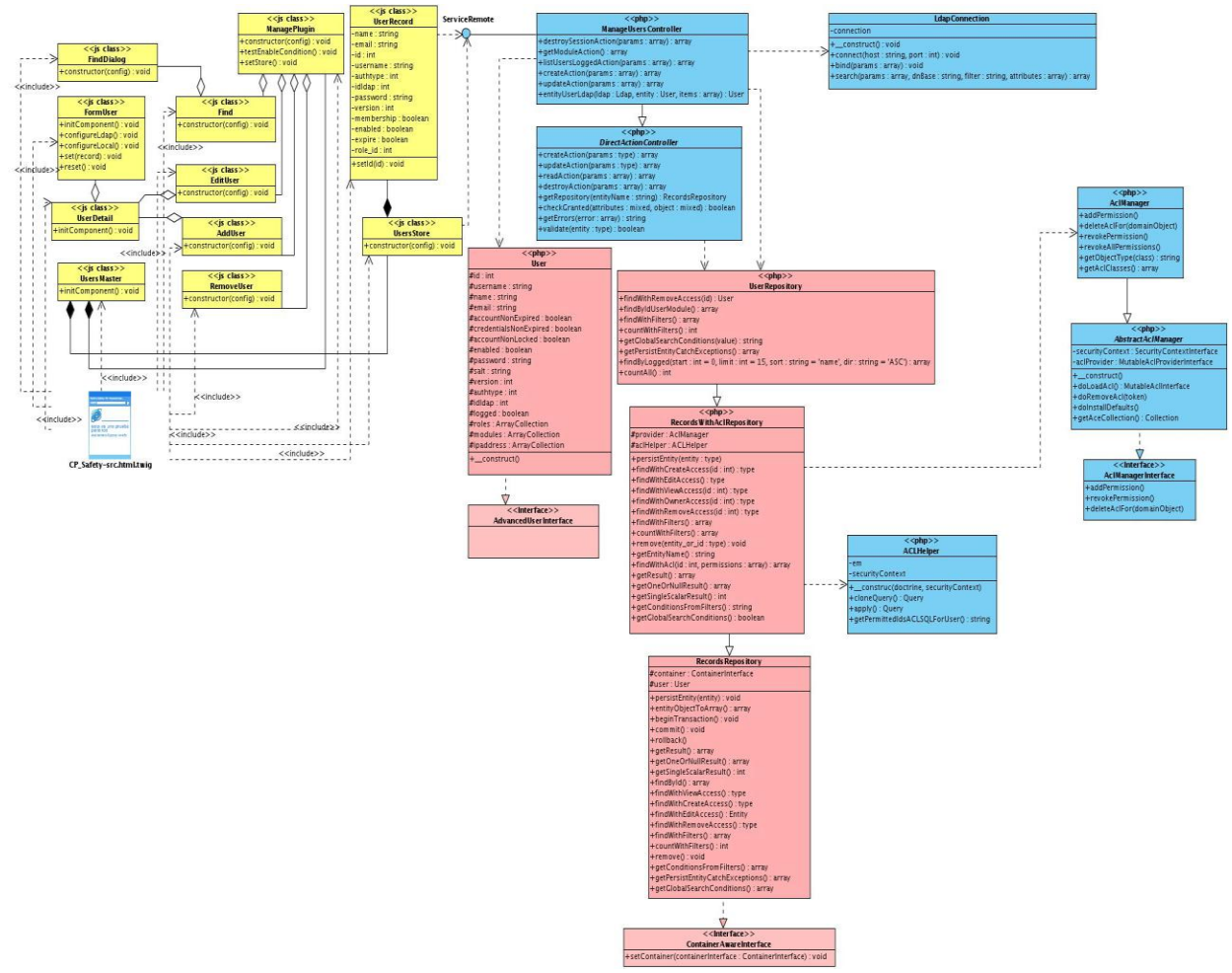


Figura 7 Diagrama de clases del diseño del Caso de uso Gestionar usuario.

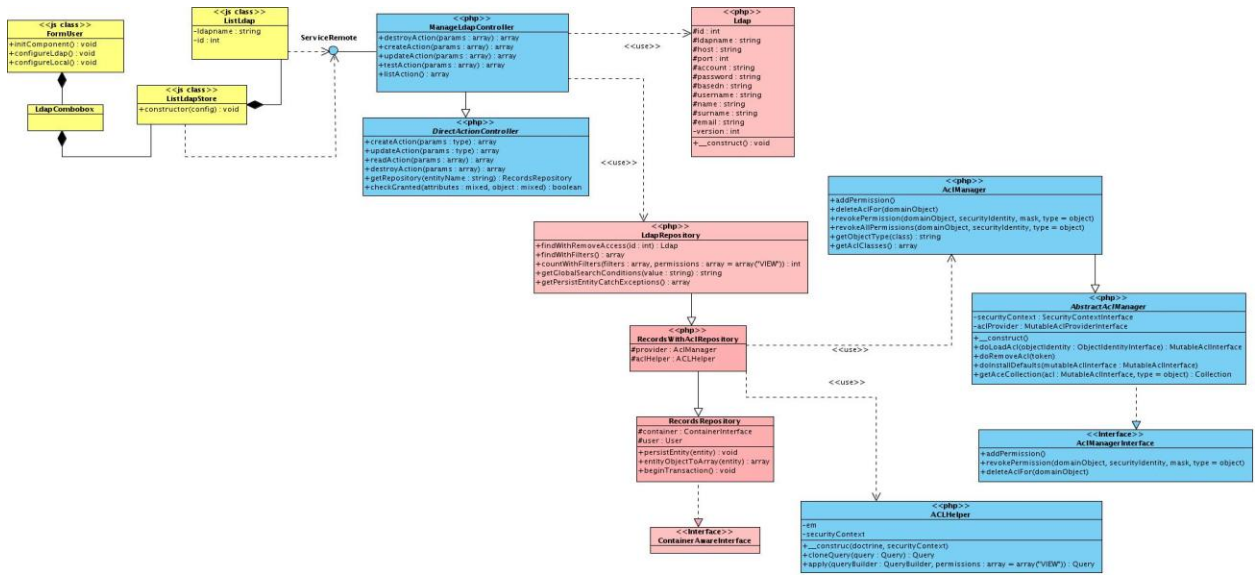


Figura 8 Diagrama de clases del diseño del Caso de uso Gestionar usuario (segunda parte)

2.5 Patrones utilizados en la solución

Los patrones son formas de describir las mejores prácticas, buenos diseños, y encapsulan la experiencia de forma tal que es posible para otros reutilizar dicha experiencia. Constituyen mecanismos cuyo objetivo es la solución de problemas que ocurren repetidamente dentro de un contexto muy bien definido. Las soluciones que son propuestas a través de patrones involucran algunas clases de estructuras que permiten contemplar los requisitos no funcionales. (Garnache, 2009)

2.5.1 Patrón de Caso de Uso

Los patrones de caso de uso son comportamientos que deben existir en el sistema, ayudan a describir qué es lo que el sistema debe hacer, es decir, describen el uso del sistema y cómo este interactúa con los usuarios.

CRUD Completo

En el diseño del sistema se utilizó el patrón de caso de uso CRUD Completo para modelar todas las operaciones que pueden ser realizadas sobre una parte de la información, tales como la creación, lectura, actualización y eliminación. Ejemplo de ello se evidencia en los caso de usos de Gestionar usuario, rol, LDAP representados en el diagrama de casos de uso del sistema.

2.5.2 Patrón Arquitectónico

La selección de patrón arquitectónico modelo vista controlador en lo adelante MVC se basa en la utilización del marco de trabajo Symfony2 para el desarrollo de la solución, a continuación se explica de manera breve en qué consiste dicho patrón.

MVC

Es un patrón de diseño de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página.

Modelo: esta es la representación específica del dominio de la información sobre la cual funciona la aplicación. El modelo es otra forma de llamar a la capa de dominio. Es el responsable de acceder a la capa de almacenamiento de datos.

Vista: esta presenta el modelo en un formato adecuado para interactuar, usualmente un elemento de la interfaz de usuario.

Controlador: este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. (CraigLarman, 2010)

2.5.3 Patrones de Diseño

Los patrones de diseño de software son soluciones reutilizables de problemas recurrentes que aparecen durante el proceso de diseño de software orientado a objetos. Estos surgen por la necesidad de transmitir la experiencia a los demás desarrolladores.

Con el conocimiento de estos patrones, los programadores son capaces de identificar las situaciones en las que éstos tienen aplicación, y utilizarlos sin tener que detenerse para analizar el problema y vislumbrar diferentes estrategias de resolución. A continuación se realiza una descripción de algunos de los patrones de diseño utilizados en la solución.

Patrones GRASP

Controlador

El patrón controlador permite facilitar la centralización de actividades, delegar las actividades en otras clases con las que mantiene un modelo de alta cohesión.

El uso de este patrón se pone de manifiesto en la utilización de diversos controladores en el sistema como por ejemplo ManageUsersController, ManageRoleController, ManageLdapController, SecuredController, ManageUserModulesController y ManageAclController. Los controladores mencionados anteriormente son los encargados de interactuar con el modelo y enviar una respuesta a la vista.

Experto

Permite asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Se hace uso del mismo en la clase AsignIpAction (ver figura 9), indicando que la responsabilidad de la creación de la vista representada por la clase AsignIpWindow debe recaer sobre la misma, ya que posee la información necesaria para crearla.

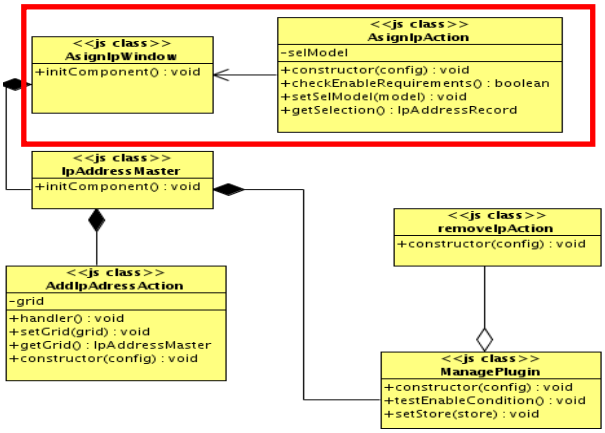


Figura 9 Patrón Experto

Alta cohesión

Facilita la solución al problema ¿Cómo mantener manejable la complejidad? mediante la asignación de responsabilidades de manera que la información que almacena una clase sea coherente y esté relacionada con la clase.

En la solución se observa el uso de este patrón en las clases del modelo, dígase User, Ldap, Role, IpAddress y DydamsiModule, las cuales poseen la estructura necesaria para almacenar la información de una manera coherente y de acuerdo a la responsabilidad que tenga asignada cada clase.

Bajo acoplamiento

Facilita la solución al problema ¿Cómo dar soporte a una dependencia escasa y a un aumento de la reutilización?

En la solución se observa el uso de este patrón en las clases controladoras del módulo, tales como ManageUsersController, ManageRoleController, ManageLdapController, SecuredController, ManageUserModulesController y ManageAclController las cuales heredan de la clase DirectActionController logrando un bajo acoplamiento y reutilización de los métodos existentes en la clase anteriormente mencionada.

Patrones GOF

Plantilla de Método (Template Method)

Permite definir el esqueleto básico de un algoritmo. Se trata de una clase que define parte de un algoritmo mediante clases abstractas. Posteriormente, las subclases modifican los métodos abstractos para implementar las acciones concretas.

En la solución se utiliza este patrón en la clase AclManager (ver figura 10) la cual extiende de AbstractAclManager que a su vez tiene una realización de la interfaz AclManagerInterface donde se encuentra el esqueleto de los métodos addPermission, revokePermission y deleteAclFor los cuales se implementan en la subclase AclManager.

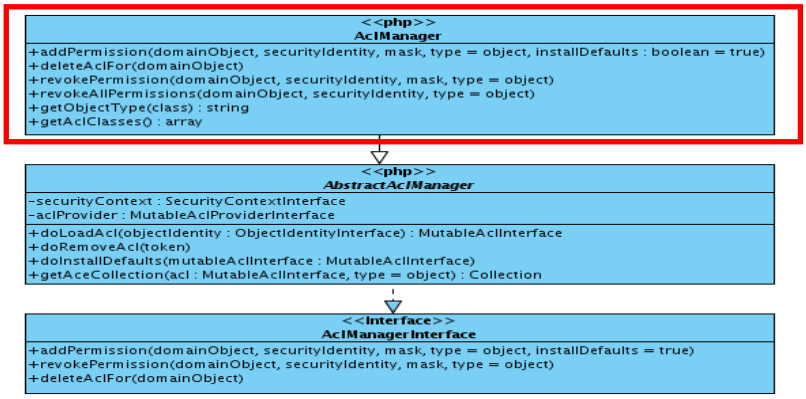


Figura 10 Plantilla de Método (Template Method)

Observador (Observer)

Define una dependencia de uno-a-muchos entre objetos de forma que cuando un objeto cambia estado, todos sus dependientes son notificados y actualizados automáticamente. Symfony2 utiliza el patrón de diseño Observer para la gestión de sus eventos. Symfony2 notifica diversos eventos en los momentos más importantes que se producen durante la ejecución de la petición. Para ejecutar código cuando se produzca uno de esos eventos, se crea una clase llamada listener con el código a ejecutar y se indica al central event dispatcher que interesa un determinado evento. Cada vez que se produzca ese evento, Symfony2 lo notifica al dispatcher. El dispatcher ejecuta el código de todas las clases que antes expresaron su interés por ese evento. (Fabien Potencier, 2012).

En la solución se hace se le informa al event dispatcher la necesidad de escuchar el evento kernel.request para conocer cuando un usuario ejecuta una petición al sistema y poder validar el tiempo entre una petición y otra. En la figura# 11 se muestra como es escuchado el evento kernel.request por la clase SessionHandler.

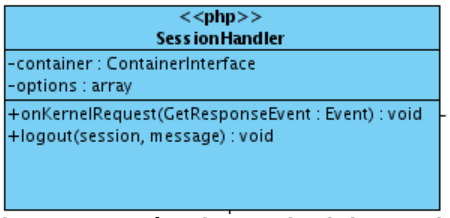


Figura 11 Patrón Observador (Observer)

2.6 Diagrama de Secuencia

Los diagramas de secuencias muestran la forma en que un grupo de objetos se comunican (interactúan) entre sí a lo largo del tiempo. Un diagrama de secuencia consta de objetos, mensajes entre estos objetos y una línea de vida del objeto representada por una línea vertical. (Pressman, 2011)

2.6.1 Diagrama de secuencia de la sección “Adicionar usuario local”

El diagrama de la figura#12 el actor administrador selecciona la opción adicionar, luego la página cliente se encarga de mandar a ejecutar el método handler en la clase AddUser, esta llamada permite mostrar la vista UserDetail mediante la ejecución del método show. Una vez mostrada la vista el usuario introduce los datos sobre el formulario, luego se validan los datos introducidos y el usuario selecciona la opción aceptar. La vista lanza el evento accept, la acción AddUser captura el evento lanzado y envía una petición al controlador ManageUserController para crear el usuario utilizando el API definido en el record que le entrega la clase UserDetail. El controlador construye un nuevo usuario a partir de los datos de la petición, persiste el usuario construido y envía una respuesta a la acción. La acción captura la respuesta del controlador y en caso de que se haya realizado la adición correctamente se realiza una nueva petición

para obtener los datos de los usuarios y actualizar el listado existente en la vista. La acción utiliza una instancia de la clase UserStore que le ofrece el pluginManager, aprovechando la existencia de esta instancia la acción utiliza el API definido en el UserStore para ejecutar la acción readAction del controlador, este le pide al repositorio el listado de los usuarios mediante la ejecución de la función findWithFilters existente en la clase UserRepository. El controlador envía el listado de usuario y se construye la vista UsersMaster. Luego se actualiza la vista UserDetail mediante la ejecución del método reset, como último paso la acción AddUser muestra un mensaje de éxito en la página cliente.

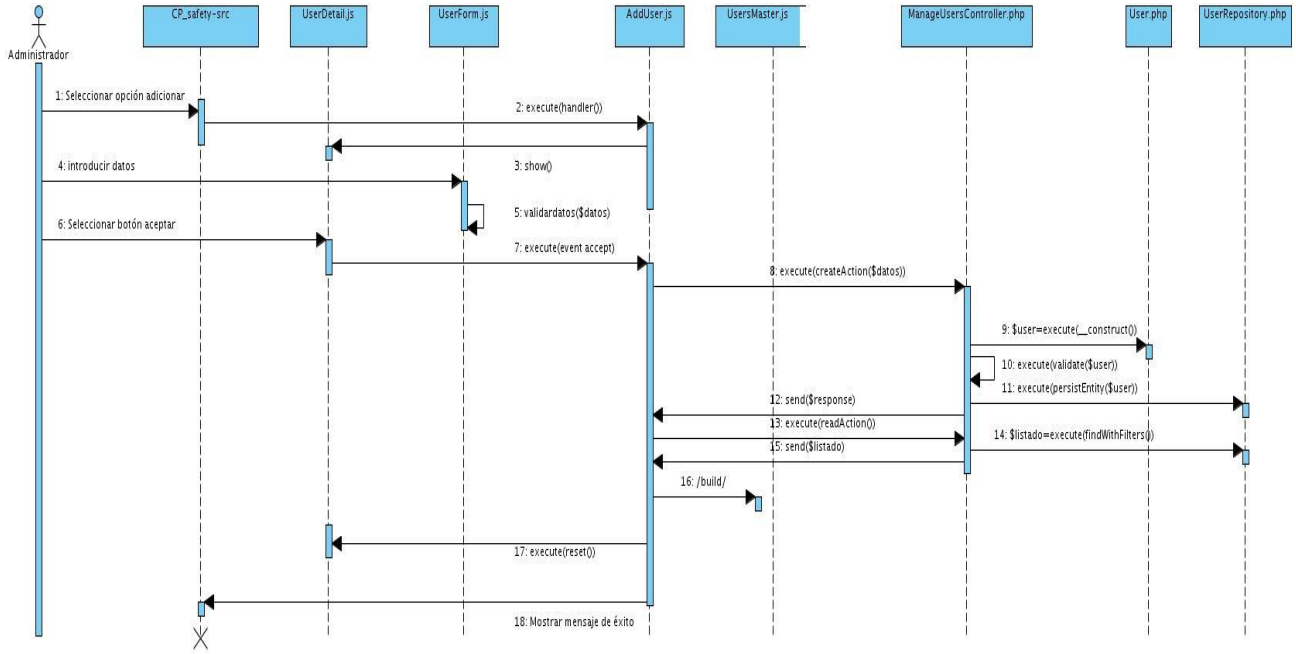


Figura 12 Diagrama de secuencia de la sección “Adicionar usuario local”

2.7 Modelo de Datos

Un modelo de datos es la combinación de una colección de estructuras de datos, operadores o reglas de inferencia y de reglas de integridad, las cuales definen un conjunto de estados consistentes. El cual puede ser usado como una herramienta para especificar los tipos de datos y la organización de los mismos. Es un sistema formal y abstracto que permite describir los datos de acuerdo con reglas y convenios predefinidos. A continuación se muestra el modelo de datos del módulo de seguridad. (Codd, 2012)

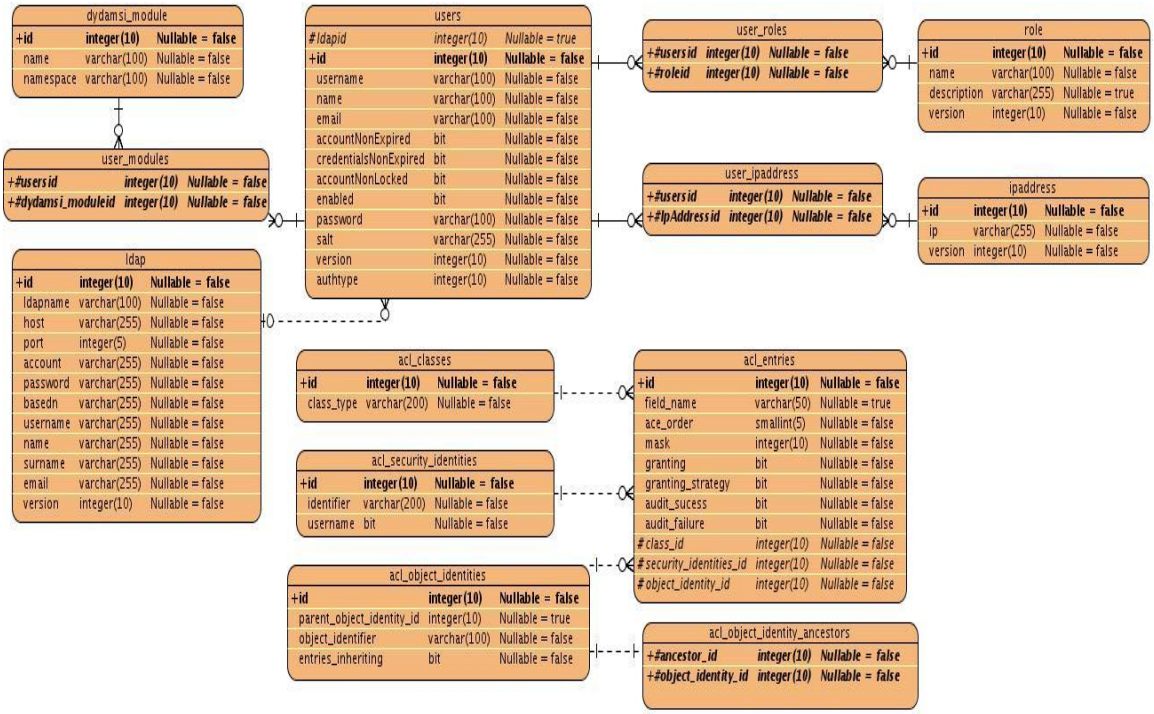


Figura 13 Modelo de Datos

Descripción de las principales tablas del modelo de datos

Nombre: users		
Descripción: Almacena los datos de los usuarios.		
Atributo	**Tipo**	**Descripción**
idapid	integer(10)	Identificador auto-incremental
id	Integer(10)	Identificador auto-incremental.
username	varchar(100)	Usuario por el cual se registrará
name	varchar(100)	Nombre del usuario
email	varchar(100)	Correo electrónico del usuario
accountNonExpired	bit	Maneja si la cuenta expira
accountNonLocked	bit	Maneja si la cuenta está bloqueada o no.
enable	bit	Indica la cuenta está activa o no.
password	varchar(100)	Contraseña de usuario
salt	varchar(100)	Valor que utilizara para codificar la contraseña
version	Integer(10)	Versión del registro del usuario.
authtype	Integer(10)	Tipo de autenticación (local, Ldap).

Tabla 1 Descripción de la tabla users de la base de datos.

Nombre: ldap
Descripción: Almacena los datos del ldap.

Atributo	Tipo	Descripción
id	integer(10)	Identificador auto-incremental.
idname	varchar(100)	Identificador auto-incremental.
host	varchar(100)	Servidor de la base de datos.
port	integer(10)	Puerto por donde se conecta el servidor.
account	varchar(100)	Cuenta o DNS principal
password	varchar(100)	Contraseña de usuario
basedn	varchar(100)	Parámetro para la búsqueda del ldap.
username	varchar(100)	Usuario por el cual se registrará
name	varchar(100)	Nombre del usuario
surname	varchar(100)	Apellidos del usuario
email	varchar(100)	Correo electrónico del usuario
versión	Integer(10)	Versión del registro del LDAP.

Tabla 2 Descripción de la tabla ldap.

Nombre: role		
Descripción: Almacena los datos de los roles		
Atributo	**Tipo**	**Descripción**
id	integer(10)	Identificador auto-incremental.
name	varchar(100)	Nombre del rol.
description	varchar(100)	Función del rol
versión	integer(10)	Versión del registro del rol.

Tabla 3 Descripción de la tabla role.

Nombre: ipaddress		
Descripción: Almacena los datos de las direcciones ip		
Atributo	**Tipo**	**Descripción**
id	integer(10)	Identificador auto-incremental.
ip	varchar(100)	Dirección ip.
versión	integer(10)	Versión del registro de la dirección ip.

Tabla 4 Descripción de la tabla ipaddress de la base de datos.

2.8 Modelo de Despliegue

Los diagramas de despliegue muestran las relaciones físicas entre los componentes de hardware y software en el sistema final. Se compone por nodos, dispositivos y conectores; donde los nodos son elementos de procesamiento con al menos un procesador, los dispositivos son nodos estereotipados sin capacidad de procesamiento en el nivel de abstracción que se modela y los conectores expresan el tipo de conector o protocolo utilizado entre el resto de los elementos del modelo. A continuación la figura#14 muestra el diagrama de despliegue del módulo de seguridad.

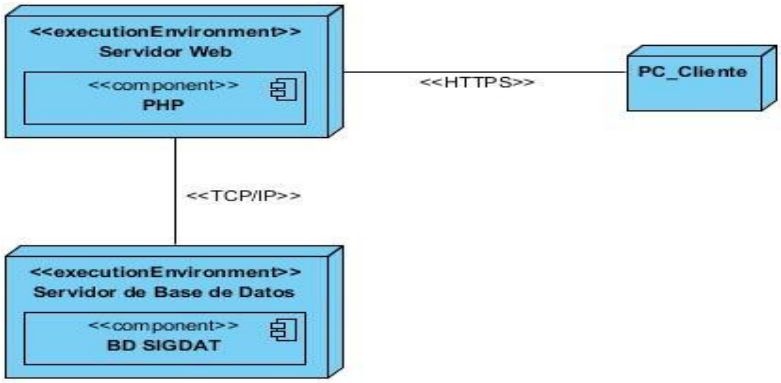


Figura 14 Modelo de Despliegue

El diagrama realizado representa 3 nodos principales. El nodo PC_Cliente que requiere de un navegador que soporte JavaScript, el nodo Servidor BD en el cual debe estar instalado el Sistema Gestor de Base de Datos PostgreSQL y el nodo Servidor Web, en el cual debe estar instalado el servidor web.

El nodo PC Cliente estará conectado al nodo Servidor de aplicación por el protocolo de transferencia de hipertexto [Hypertext Transfer Protocol Secure] (HTTPS) como protocolo de comunicación. La conexión entre el Servidor Web y el Servidor de Base de Datos se realizará mediante el protocolo de comunicación TCP/IP.

2.9 Conclusiones parciales del capítulo

En el desarrollo del presente capítulo se realizó una representación visual de clases conceptuales del entorno real de los objetos del proyecto a través del modelo de dominio y se describieron los 38 requisitos funcionales y 8 no funcionales que el sistema debe cumplir para su correcto funcionamiento. Además se identificaron los actores del sistema, 11 casos de uso y la relación existente entre ellos reflejada en el diagrama de casos de uso del sistema, apoyado de la descripción detallada de los casos de uso del sistema para obtener así un mayor entendimiento de los requisitos funcionales previamente establecidos. De igual manera, se realizó el modelo de diseño con el propósito de describir cómo se debe implementar el sistema. Se utilizaron los diagramas de secuencia para inspeccionar los aspectos dinámicos del módulo. Se identificaron los patrones de diseño a utilizar.

CAPÍTULO III “IMPLEMENTACIÓN Y PRUEBA”

Introducción

En el presente capítulo se muestra el modelo de implementación como resultado del diseño anteriormente desarrollado, así como el diagrama de componentes del módulo de seguridad. Se describen las pruebas a realizar, con el objetivo de comprobar las funcionalidades del módulo en los diferentes escenarios, para de esta forma verificar en todos los casos que los resultados de las pruebas sean los esperados y el correcto funcionamiento del módulo.

3.1 Modelo de Implementación

La implementación brinda la posibilidad de probar y desarrollar componentes como unidades, que finalmente serán integrados como un sistema ejecutable. En este flujo de trabajo se organizan y realizan las pruebas unitarias y se integran los componentes implementados, basándose en las especificaciones de diseño.

3.1.1 Diagrama de componentes

Un diagrama de componentes representa la separación de un sistema de software en componentes físicos y muestra las dependencias entre estos. Cada subsistema corresponde a un paquete físico y cada componente a un módulo, fichero o biblioteca existente en la memoria de almacenado. (Ian Sommerville, 2010).

Seguidamente se expone el diagrama de componente del CU “Gestionar usuario” el cual cuenta con 3 paquetes de implementación básicos. Estos paquetes son:

- El Paquete de Clases Vista, que agrupa los componentes que permiten la interacción directa con el usuario final del sistema, mostrando y recogiendo información.
- El Paquete de Clases Controlador, que contendrá las clases que manipulan los eventos del usuario y se apoya en subsistema modelo para dar respuesta a las peticiones de las vistas.
- El Paquete de Clases Modelo, que agrupa las clases que interactúan con la base de datos y velan por el cumplimento de las reglas del negocio. (ver figura 15)

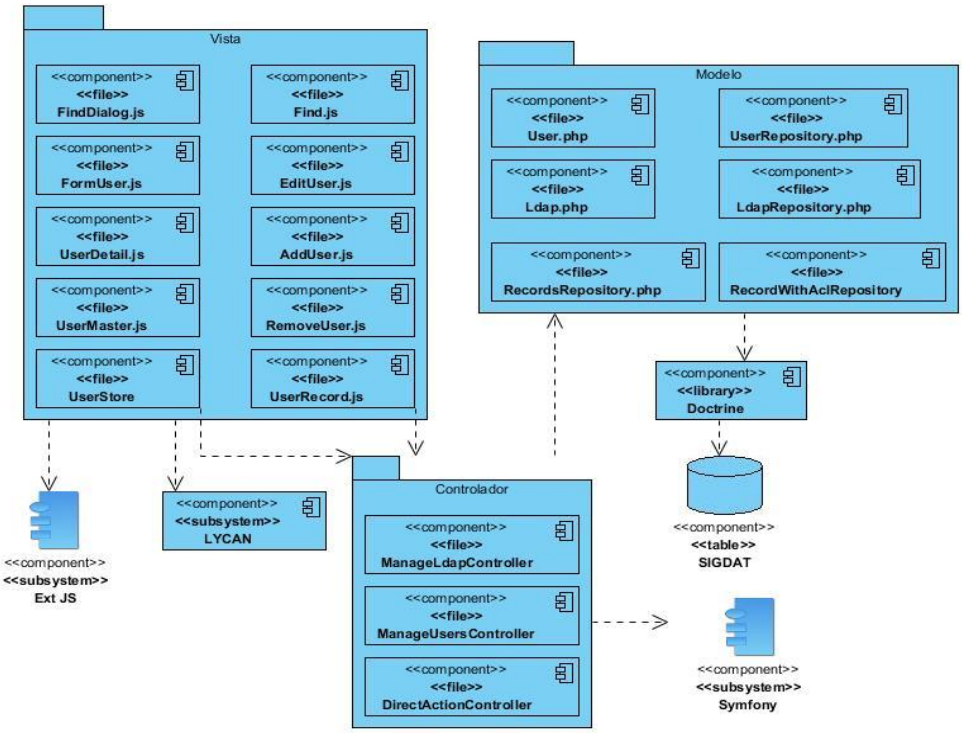


Figura 15 Diagrama de componentes del CU Gestionar usuario

3.2 Código Fuente

El código fuente de un sistema informático es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para así poder ejecutar las funcionalidades requeridas por el software. El código fuente varía según el lenguaje de programación, y debe ser traducido al lenguaje máquina para que sea ejecutado.

3.2.1 Estándares de codificación

Un estándar de codificación son reglas que se siguen para la escritura del código fuente y comprende todos los aspectos de la generación de código. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Es importante establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. (Fabien Potencier, 2009)

Estilo de codificación utilizado:

- Los nombres de las variables y funciones estarán escritos con el estándar "lowerCamelCase".
- Se utilizarán llaves para indicar la estructura del cuerpo de control, independientemente del número de declaraciones que contenga.

- ❖ Todas las funciones y clases estarán comentadas, para explicar el flujo del código y el propósito de las funciones o variables.
- ❖ Todas las etiquetas PHP deben ser completas (<?php ?>)... no reducidas (<? ?>).
- ❖ Las cadenas tienen que ser definidas utilizando comillas simples siempre que sea posible, para obtener un mejor rendimiento.

Ejemplo de código fuente

A continuación la figura #16 muestra el fragmento de código del método: "Adicionar usuario" el cual le permite al administrador adicionar un nuevo usuario al sistema.

```
    public function createAction($params) {
        $items = $params['items'];
        $repo = $this->getRepository();
        $entity = new User();
//si es uno es de tipo ldap si no es local
        if ($items['authtype'] == 1) {
            $idldap = $items['idldap'];
            $ldap = $this->getRepository('SafetyBundle:Ldap')->findWithViewAccess($idldap);
            $entity = $this->entityUserLdap($ldap, $entity, $items);
        } else {
            $entity->setUsername($items['username']);
            $entity->setAuthType(0);
            $entity->setSalt($entity->getUsername());
            $entity->setPassword($items['password']);
            $entity->setName($items['name']);
            $entity->setEmail($items['email']);
            $entity->setEnabled($items['enabled']);
            $entity->setAccountNonExpired(true);
            $entity->setAccountNonLocked(true);
            $entity->setCredentialsNonExpired(true);
        }
        $this->validate($entity);
        $repo->persistEntity($entity);
        return array('success' => true);
    }
```

Figura 16 Ejemplo de código fuente del Método "Adicionar usuario"

3.3 Pruebas del software

El proceso de pruebas se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en los procesos externos funcionales, es decir, la realización de las prueba para la detección de errores. Además son utilizadas para identificar posibles fallos de implementación, calidad o usabilidad de un programa. (Pressman, 2010).

3.3.1 Niveles de Prueba

La prueba es aplicada para diferentes tipos de objetivos, en determinados momentos del ciclo de vida del software. Existen diferentes niveles de pruebas como: pruebas de desarrollador, independiente, integración, sistema y aceptación. En el desarrollo de la fase de pruebas del módulo de seguridad se

aplicarán las pruebas a nivel de desarrollador e integración para probar que el sistema cumpla con los requisitos funcionales previamente definidos en la fase de análisis.

Tipos de Prueba

Existen diferentes tipos de pruebas que se pueden aplicar para comprobar el correcto funcionamiento del módulo de seguridad, dentro de los que se seleccionaron los que a continuación se muestran:

Pruebas funcionales: prueban una funcionalidad completa, donde pueden estar implicadas una o varias clases y la propia interfaz de usuario.

Pruebas de Seguridad: tiene como principal objetivo verificar antes de la liberación del sistema, la aplicación de los mecanismos de protección incorporados y se realizan para detectar la consecuente existencia de vulnerabilidades y/o defectos de seguridad para eliminarlos.

Pruebas de rendimiento (Carga y Stress): pruebas de rendimiento que se realizan para comprobar la velocidad con la que un sistema ejecuta una tarea. Además sirven para verificar el comportamiento de una aplicación bajo una demanda excesiva.

Pruebas de Integración: conjunto de pruebas unitarias, funcionales, regresión y aceptación que se realizan para probar el software.

Método de Prueba

Pruebas de Caja Negra: son las pruebas que se llevan a cabo sobre la interfaz del software. Las mismas se centran en los requisitos funcionales del software, es decir, permiten obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. (Pressman, 2010).

3.3.2 Diseño de Caso de Prueba

Los casos de pruebas se realizan con el objetivo de determinar que una funcionalidad ha sido implementada satisfaciendo las necesidades del cliente. Para cada caso de uso debe estar asociado un caso de prueba que recoja la especificación de ese caso de uso, dividido en secciones y escenarios, detallando las funcionalidades descritas en él y describiendo cada variable que recoge el caso de uso en cuestión.

En la siguiente tabla se detallan las variables que se encuentran asociadas al caso de uso Gestionar usuario

No	Nombre del Campo	Clasificación	Valor Nulo	Descripción

1	Usuario	Campo texto	No	Cadena de texto con longitud mayor que 3 que inicia con letras, se pueden entrar hasta 10 letras ya sea mayúscula o minúscula y puede terminar en número, la longitud máxima es de 15.
2	Modo de autenticación	Campo lista desplegable	No	Campo lista desplegable. Se selecciona Local o LDAP.
3	Nombres y apellidos	Campo texto	No	Campo alfanumérico que permite una cadena de texto que contiene al iniciar las palabras letras mayúsculas, van separados por espacio y tiene como longitud máxima 100 caracteres.
4	Correo	Campo texto	No	Campo alfanumérico que permite una cadena de texto cumpliendo el estándar de correo: usuario@dominio.dominio.
5	Estado	Campo lista desplegable	No	Campo lista desplegable. Se selecciona Habilitado o Bloqueado.
6	Contraseña anterior	Campo texto	No	Campo alfanumérico que indica la contraseña anterior. Se habilita cuando el usuario autenticado indica que desea editar su información.
7	Contraseña	Campo texto	No	Campo alfanumérico con una combinación de mayúscula, minúsculas, números y caracteres especiales. Longitud mínima de 8 y máxima hasta 10.
8	Repetir Contraseña	Campo texto	No	Campo alfanumérico con una combinación de mayúscula, minúsculas, números y caracteres especiales. Longitud mínima de 8 y máxima hasta 100 caracteres. Debe coincidir con la contraseña indicada.

Tabla 5 Tabla de variables para el caso de prueba

Esta descripción posibilitó que se realizara una matriz de datos, donde se evaluó y probó la validez de cada uno de los datos introducidos en el sistema, específicamente en la sección que se estuvo probando. Utilizando un juego de datos válidos e inválidos se identificó el empleo de la técnica de partición de equivalencia.

Matriz de Datos

Escenario Adicionar usuario

Escenario	Variables (Enumeradas según la descripción anterior)								Descripción	Respuesta del sistema	Flujo Central
	1	2	3	4	5	6	7	8			
EC 1.1 Adicionar usuario	V	V	V	V	V	V	V	V	En este escenario se realiza la adición de un nuevo usuario al sistema correctamente.	Se adiciona correctamente el usuario nuevo y el sistema permite la inserción de un nuevo usuario.	1. Módulo Seguridad/Gestionar Usuarios/Adicionar (El botón Adicionar se encuentra: Menú superior dentro del panel Gestionar Usuarios, Clic derecho sobre el grid Usuario o en el menú inferior). 2. Se llenan los campos correctamente. 3. Se presiona el botón Aceptar.
EC 1.2 Adicionar usuario con datos incorrectos	V	V	V	V	V	I	V	V	En este escenario se realiza la adición de un nuevo usuario al sistema con datos incorrectos.	El sistema valida que los campos estén llenos correctamente y es cuando se activa el botón Aceptar.	1. Módulo Seguridad/Gestionar Usuarios/Adicionar (El botón Adicionar se encuentra: Menú superior dentro del panel Gestionar Usuarios, Clic derecho sobre el grid Usuario o en el menú inferior). 2. Se llenan los campos con datos no válidos.

3.3.3 Resultados de las pruebas

Caja Negra

Después de realizar las pruebas de caja negra mediante los casos de prueba asociados a cada caso de uso, se comprobó el correcto funcionamiento del módulo y la correcta validación de los campos, verificando que solo se acepten los caracteres válidos para los mismos. Fueron detectadas 5 no conformidades a las que se les dio seguimiento y fueron eliminadas a medida que se fue avanzando en el proceso de pruebas.

Pruebas de rendimiento (Carga y Stress)

Se realizaron pruebas de rendimiento (carga y estrés) para comprobar la velocidad con la que el sistema ejecuta una tarea. Para ejecutar este tipo de prueba se usó la herramienta JMeter. Los resultados de esta prueba se muestran en la siguiente tabla, donde los tiempos de respuestas obtenidos como resultado son expresados en milisegundos (ms), donde 1 segundo equivale a 1000 milisegundos. Como resultado se observa que son tiempos de espera admisibles para cualquier persona que interactué con una computadora, tiempos que no llegan a 2 segundos.

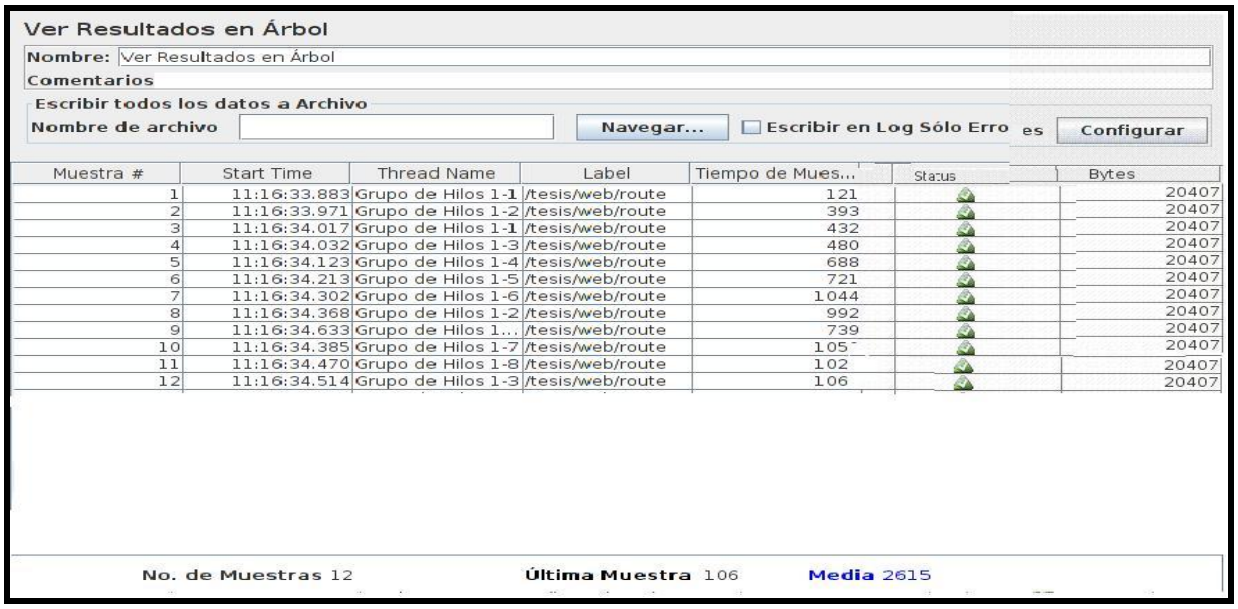


Figura 17 Prueba de Carga y estrés

Pruebas de Seguridad

Para la realización de las pruebas de seguridad al módulo, primeramente, se identificaron los distintos tipos usuarios que están definidos en el sistema y sus respectivos niveles de acceso o privilegios definidos; luego se creó un usuario de cada tipo y se procedió con las pruebas.

Casos de pruebas realizadas	
Descripción	**Resultado**
Las primeras pruebas fueron intentos de violación de la seguridad del sistema, tratando de acceder a éste con usuarios no existentes o utilizando usuarios reales con contraseñas erróneas.	Estas pruebas de acceso arrojaron resultados favorables al sistema, ya que no se logró acceder con usuarios no registrados previamente ni con falsas contraseñas, demostrando que el módulo cuenta con un mecanismo de autenticación seguro, garantizando que solamente trabajen con el programa personas autorizadas previamente.
Se accede al sistema con un usuario específico y se intenta acceder a funciones no permitidas para ese determinado usuario, como son: creación o eliminación de usuarios, modificación de los permisos de usuarios existentes, asignar módulos, asignar roles, entre otras funciones de las que depende el correcto	Este mismo procedimiento se ejecutó una y otra vez con los tres tipos de usuarios con que cuenta la aplicación y los resultados de las pruebas fueron los esperados, los usuarios registrados en el sistema tienen acceso solamente a las funcionalidades predefinidas por los administradores del sistema, impidiendo en todo

funcionamiento del sistema	momento que se lleven a cabo funcionalidades por personas no autorizadas.

Prueba de Integración (incremental ascendente)

Luego de realizar las pruebas de caja negra se integró el módulo al sistema SIGDAT y se pudo comprobar el correcto funcionamiento con los requisitos definidos. Constando actualmente con una aval (ver Anexo 3) que corrobora el éxito de esta integración. Para realizar el proceso de integración se siguieron los siguientes pasos:

1. Se garantizó que las clases Repository de los restantes módulos de la aplicación aplicaran la seguridad heredando de la clase RecordWithAclRespository perteneciente al módulo de seguridad, la cual permite controlar los diferentes permisos sobre los recursos que se gestionan en el sistema.
2. Las clases controladoras de los restantes módulos extendieron de la clase DirectActionController lo cual facilitó el acceso a los repositorios según la entidad que se maneje en cada caso.
3. Se incluyó en el fichero routing.yml las diferentes rutas del módulo de seguridad tales como el punto de acceso para la autenticación y cierre de sesión.
4. Se configuró en el archivo Security.yml como proveedor de usuarios la entidad User.
5. Se configuraron los codificadores utilizando el algoritmo Sha1 y la ruta del cortafuegos de la aplicación configurando las posibles aéreas donde no era necesario la autenticación
6. Se configuró el control de acceso donde se definió el protocolo requerido para interactuar con el sistema.

3.4 Conclusiones parciales del capítulo

En el desarrollo del presente capítulo se realizó el modelo de implementación del módulo de seguridad de SIGDAT con el propósito de mostrar los componentes del sistema y sus relaciones, a través del diagrama de componentes. Se especificó el uso de los estándares de codificación para lograr un estilo claro y organizado del código durante la fase de implementación. Se realizaron pruebas de seguridad, funcionales e integración para validar el correcto funcionamiento del módulo, utilizando el método de caja negra basado en la técnica de partición de equivalencia. Se midió el rendimiento del sistema a través de las pruebas de carga y estrés automatizándolas con la herramienta JMeter 2.3. Se identificaron 5 no conformidades que fueron resultas posteriormente.

Conclusiones

Con la realización de este trabajo de diploma se obtuvo una propuesta que da cumplimiento al objetivo general planteado, al lograr desarrollar un módulo de seguridad para proteger al sistema SIGDAT. Para llegar a este resultado se puede concluir lo siguiente:

- Se realizó un estudio de los principales aspectos relacionados con la seguridad en los sistemas, lo cual posibilitó identificar los riesgos a los que están expuestos frecuentemente, sentando las bases para identificar las vulnerabilidades de la aplicación SIGDAT.
- Se realizó el análisis y diseño del módulo de Seguridad para SIGDAT obteniéndose como resultado los artefactos y diagramas necesarios para guiar el desarrollo del mismo.
- Se realizó la implementación del módulo Seguridad para SIGDAT cumpliendo con los 38 requisitos funcionales identificados en las fases de análisis y diseño.
- El diseño y ejecución de pruebas del sistema permitieron comprobar el correcto funcionamiento del Módulo de Seguridad para el sistema SIGDAT.

Recomendaciones

Luego de haber analizado los resultados del presente trabajo de diploma, se puede llegar a las siguientes recomendaciones:

- Desplegar todas las actualizaciones y parches de software de manera oportuna para garantizar una mayor seguridad en el ambiente de despliegue donde se encuentre instalada la aplicación.

- Realizar el estudio de otros mecanismos para evitar ataques de fuerza bruta durante el proceso de autenticación tales como PlayThru, para una futura implementación al sistema.

REFERENCIAS BIBLIOGRÁFICAS

1. **Rivas López, José Luis.** Protección de la información. [En línea] http://www.dtic.co.cu/?q=content/protecci%C3%B3n-de-la-informaci%C3%B3n.

2. **Villalón Huerta, Antonio.** SEGURIDAD EN UNIX Y REDES. [En línea] http://www.rediris.es/cert/doc/unixsec/.

3. **OWASP TOP 10**. Los diez Riesgos más importantes en aplicaciones web. [En línea] https://www.owasp.org/images/2/2d/OWASP_Top_10_-_2010_FINAL_%28spanish%29.pdf

4. **University, Carnegie Mellon.** CAPTCHA. [En línea] http://www.captcha.net/&ei=NGWoUOWqKsWG0QHT44DYBA.

5. **Marin, Marvin David Arias.** [En línea] http://catedraprogramacion.foroactivos.net/t83-definicion-de-lenguaje-de-programacion-tipos-ejemplos.

6. **Bakken, Stig Sæther.** Manual_de_PHP. [En línea] http://php.uci.cu/search.php.

7. **Maldonado, Daniel.** El CoDiGo K. Qué son los IDE de programación. [En línea] http://www.ecured.cu/index.php/IDE_de_Programaci%C3%B3n.

8. **BERTINO, E. A. MARTINO.** Sistemas de bases de datos orientadas a objetos. [En línea] http://www.ecured.cu/index.php/Sistema_Gestor_de_Base_de_Datos.

9. **Martinez, Rafael.** Sobre Postgresql. [En línea] http://www.postgresql.org.es/sobre_postgresql.

10. **PGadmin.** [En línea] http://www.pgadmin.org/visualtour14.php .

11. **Orallo, Enrique Hernández.** El Lenguaje Unificado de Modelado (UML). [En línea] www.disca.upv.es/enheror/pdf/ActaUML.PDF.

12. **CASE Herramientas**. [En línea] http://www.ecured.cu/index.php/CASE.

13.**Servidor de Aplicaciones.** [En línea] http://www.ecured.cu/index.php/Servidor_de_Aplicaciones.

14. **Apache.** [En línea] http://httpd.apache.org/.

15. **Mario Rivas Sánchez**. *OpenUP como alternativa metodológica para proyectos pequeños de software.* [En línea] http://www.ecured.cu/index.php/OpenUp openup.html.

16. **Gómez Oiner**, Tésis de Doctorado: Modelo de Control de Acceso para Sistemas de Información en entornos multidominios.

17. **Craig Larman**. UML y Patrones. [En línea] *www.publidisa.com/preview-libro-9788483229279.pdf*

18. **IAN Somerville**. Ingeniería del Software. [En línea]http://www.filecrop.com/ian-sommerville-ingenieria-de-software-septima-edicion.pdf.html.

BIBLIOGRAFÍA

1. **Desarrollo web**. [En línea] http://www.desarrolloweb.com/articulos/2454.php.
2. **Tipos de Captchas**. [En línea] http://www.scribd.com/doc/10581961/Clasificacion-de-Los-Captcha Clasificacion-de-Los-Captcha.html
3. **ExtJS**. [En línea] http://www.sencha.com/products/extjs3/.
4. **Pérez, Javier Eguíluz.** Manual de JavaScript. [En línea] www.librosweb.es.
5. **Rodriguez, Juan.** Guias y Tips Seguridad. [En línea] http://www.subinet.es/guias-y-tips/guias-y-tips-seguridad/¿cuales-son-los-modelos-de-control-de-acceso/.
6. **Craig Larman**. UML y Patrones. [En línea] *www.publidisa.com/preview-libro-9788483229279.pdf*
7. **IAN Somerville**. Ingeniería del Software. [En línea]http://www.filecrop.com/ian-sommerville-ingenieria-de-software-septima-edicion.pdf.html.
8**. Seguridad en aplicaciones web.** [En línea] http://books.google.es/books?id=2rYqygAACAAJ&dq=seguridad+en+aplicaciones&hl=es&sa=X&ei=HleMUfDbF-nT0wHG0IGQAw&ved=0CEEQ6AEwAw
9. **Fabien Potencier.** Symfony la Guía Definitiva**.** [En linea] www.jesusda.com/docs/ebooks/symfony_guia_definitiva.pdf.
10. **Pressman. Ingeniería del Software 6ta edición.** [En linea] http://ebookbrowse.com/roger-s-pressman-ingenieria-del-software-v-ed-pdf-d47183684
11. **OWASP TOP 10**. Los diez Riesgos más importantes en aplicaciones web. [En línea] https://www.owasp.org/images/2/2d/OWASP_Top_10_-_2010_FINAL_%28spanish%29.pdf
12. **Pacheco, Nacho.** Desarrollo Agil de Symfony2. [En línea] ftp://ftp.prod.uci.cu/PHP/Documentacion/Symfony2/Desarrollo_agil_symfony2.pdf.
13. **SCHNEIER, B.** Vulnerabilidades. [En línea] http://www.schneier.com/crypto-gram-0009.htm
14. **MONTALVO, C**. Ventajas de las aplicaciones web. [En línea] http://www.calinsoft.com/2008/08/aplicaciones-web-ventajas-y-desventajas.html
15. **USAID, ADS Chapter** 545 Information Systems Security. U. S. A. I. Development. 545_102611, pp: 1-62. [En línea] http://www.usaid.gov
16. **Suárez, D.** ISO 27000: Garantizar la Seguridad de la Información.[En línea] http://www.revista-ays.com

17. **Microsoft.Msdn.** MSDN. Diagramas de componentes de UML [En línea] http://msdn.microsoft.com/es-es/library/dd409390.aspx

18. **Herramientas CASE**. [En línea] http://www.ecured.cu/index.php/CASE

19. **Pruebas de software**.[En línea] http://www.ecured.cu/index.php/Pruebas_de_software

20. **Guillermo Lemus**. Tipos de pruebas del software. [En línea] http://www.slideshare.net/GuillermoLemus/tipos-de-pruebas-de-software

21. **Reyna, Rafael**. 2010. Herramientas Case. [En línea] http://es.scribd.com/doc/36908565/Herramientas-Case

22. **Arevalo, Maria.** 2010. Introducción al Patrón de Arquitectura por Capas. [En línea] http://arevalomaria.wordpress.com/2010/12/02/introduccion-al-patron-de-arquitectura-por-capas/

23. **INGENIERIA DE SOFTWARE II**. 2010. Desarrollo de Software. Metodologias,Frameworks y Modelos de Desarrollo De Software. [En línea] http://softwareiimarfrednarvaez.blogspot.com/2012_02_01_archive.html

24. **LDAP**. [En línea] http://www.ecured.cu/index.php/LDAP

ANEXOS

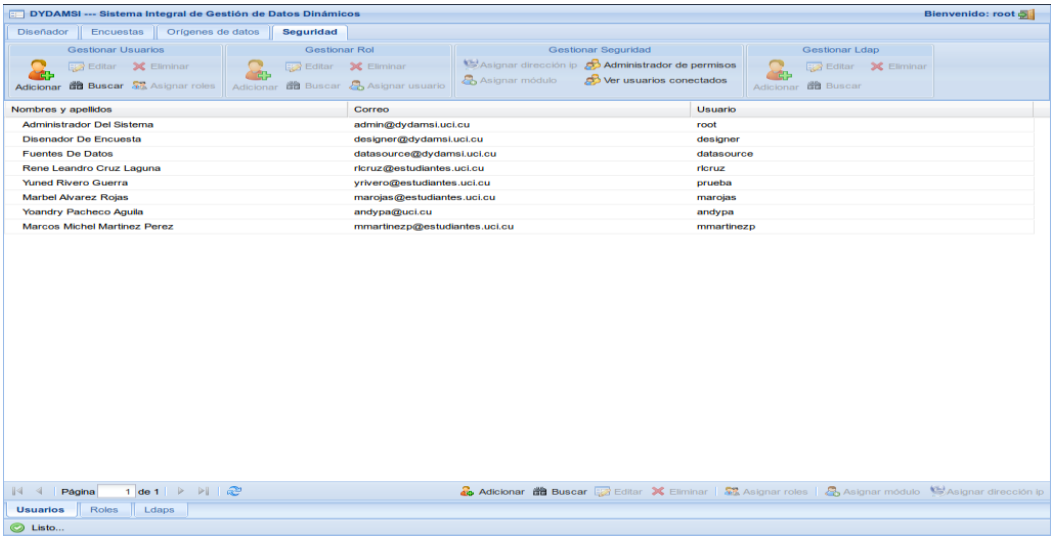
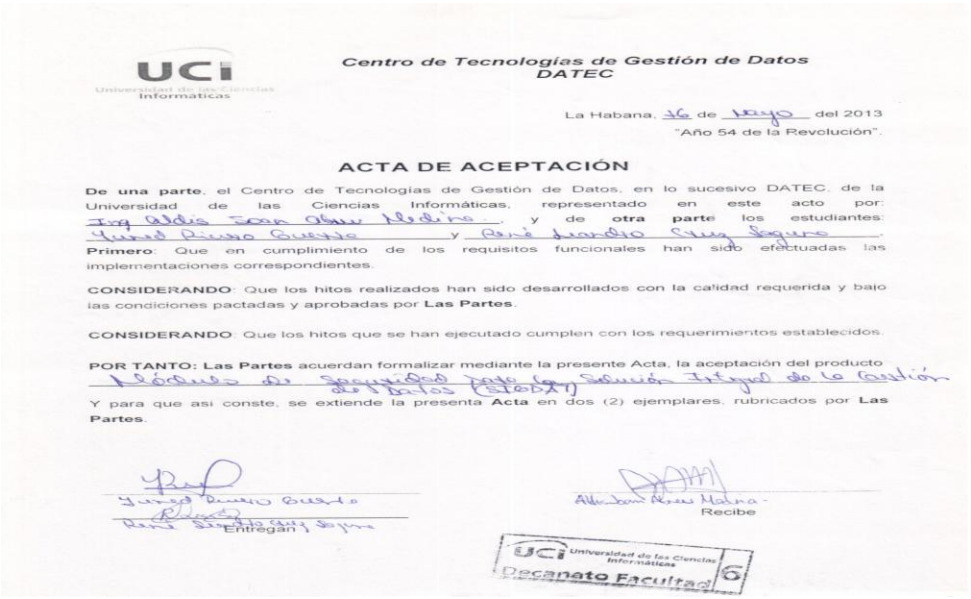


Figura 18 Anexo 1: Interfaz del módulo de seguridad

Figura 19 Anexo 2: Tipo de CAPTCHA establecer relación entre caracteres



UCI
Universidad de las Ciencias Informáticas

Centro de Tecnologías de Gestión de Datos
DATEC

La Habana, 16 de Mayo del 2013
"Año 54 de la Revolución".

ACTA DE ACEPTACIÓN

De una parte, el Centro de Tecnologías de Gestión de Datos, en lo sucesivo DATEC, de la Universidad de las Ciencias Informáticas, representado en este acto por: Ing Aldis Jean Abreu Medina, y de **otra parte** los estudiantes: Yuned Rivero Guerra y René Leandro Cruz Laguna.

Primero: Que en cumplimiento de los requisitos funcionales han sido efectuadas las implementaciones correspondientes.

CONSIDERANDO: Que los hitos realizados han sido desarrollados con la calidad requerida y bajo las condiciones pactadas y aprobadas por **Las Partes**.

CONSIDERANDO: Que los hitos que se han ejecutado cumplen con los requerimientos establecidos.

POR TANTO: Las Partes acuerdan formalizar mediante la presente Acta, la aceptación del producto Módulo de Seguridad para la Solución Integral de la Gestión de Datos (SIGDAT).

Y para que así conste, se extiende la presente **Acta** en dos (2) ejemplares, rubricados por **Las Partes**.

Yuned Rivero Guerra
René Leandro Cruz Laguna
Entregan

Alden Jean Abreu Medina
Recibe

UCI Universidad de las Ciencias Informáticas
Decanato Facultad 6

Figura 20 Anexo 3 Carta de aceptación del módulo de seguridad para SIGDAT

GLOSARIO DE TÉRMINOS

CU: Caso de uso.

CAPTCHA: Prueba de Turing para Diferenciar Máquinas y Humanos.

IP: Internet Protocol.

LDAP: Protocolo Ligero de Acceso a Directorios.

MVC: Modelo-Vista-Controlador.